Jasmine Leblond-Chartrand

COMP 499 Final Project Report

Answering public health questions with National Health Surveys

A case study of primary hypertension with NHANES 2013-2014

The NHANES (National Health and Nutrition Examination Survey) is a survey conducted across United-States to assess the health and nutritional status of individuals living in that country. The dataset includes extensive information about the participants (i.e. demographics, labs, examinations, and questionnaires)

10,175 people were interviewed in the 2013-2014 edition.

This project consists of a case study in a data exploratory perspective of hypertension, the condition for which medication is the most often prescribed according to the NHANES medications.csv dataset (n = 1595). This is also the most common primary diagnosis in the United States. QUESTIONS Q1: What are the variables associated with hypertension? (correlation) Q2: Can we identify individuals with hypertension? (classification) Q3: What is the age at which people are told they have hypertension? (regression)

Data Acquisition (via Kaggle API)

In [1]:
```
# Install the Kaggle API to be allowed download the file

# pip install kaggle
```

In [2]:
```
# Need to download a kaggle.json file from the user account page on kaggle
.com
# put it in ~/.kaggle/kaggle.json
```

In [3]:
```
import kaggle

# dataset from "https://www.kaggle.com/cdc/national-health-and-nutrition-e
xamination-survey/"

!kaggle datasets download -d cdc/national-health-and-nutrition-examination
-survey -p './data/' --unzip

# -p denotes download path
    # download the dataset as a single zip file to './data'
# --unzip
    # extract the files ['demographic.csv', 'diet.csv', 'examination.csv',
  'labs.csv', 'medications.csv', 'questionnaire.csv']
```

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /
Users/macbookpro/.kaggle/kaggle.json'
```

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /
Users/macbookpro/.kaggle/kaggle.json'
Downloading national-health-and-nutrition-examination-survey.zip to ./data
100%|██████████████████████████████████████| 6.74M/6.74M [00:02<00:00, 3.14MB/s]
```

Data Profiling and Cleaning

Step 2: Loading the data

In [4]:
```python
import pandas as pd
import numpy as np


data_demographic = pd.read_csv('./data/demographic.csv', index_col='SEQN',
 usecols=['SEQN', 'RIAGENDR', 'RIDAGEYR', 'INDHHIN2', 'INDFMIN2'])
# data_diet = pd.read_csv('./data/diet.csv')
# data_examination = pd.read_csv('./data/examination.csv')
# data_labs = pd.read_csv('./data/labs.csv')
data_medications = pd.read_csv('./data/medications.csv',  encoding = "ISO-
8859-1", usecols= ['SEQN', 'RXDRSC1', 'RXDRSD1'])
data_questionnaire = pd.read_csv('./data/questionnaire.csv', index_col='SE
QN')

data_questionnaire.head(10)
```

Out[4]:

	ACD011A	ACD011B	ACD011C	ACD040	ACD110	ALQ101	ALQ110	ALQ120Q	ALQ120U	AL
SEQN										
73557	1.0	NaN	NaN	NaN	NaN	1.0	NaN	1.0	3.0	
73558	1.0	NaN	NaN	NaN	NaN	1.0	NaN	7.0	1.0	
73559	1.0	NaN	NaN	NaN	NaN	1.0	NaN	0.0	NaN	
73560	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
73561	1.0	NaN	NaN	NaN	NaN	1.0	NaN	0.0	NaN	
73562	NaN	NaN	NaN	4.0	NaN	1.0	NaN	5.0	3.0	
73563	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
73564	1.0	NaN	NaN	NaN	NaN	2.0	1.0	2.0	3.0	
73565	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	
73566	1.0	NaN	NaN	NaN	NaN	1.0	NaN	1.0	1.0	

10 rows × 952 columns

In [5]:
```python
data_medications.iloc[:10]
```

Out[5]:

	SEQN	RXDRSC1	RXDRSD1
0	73557	NaN	NaN
1	73557	E11	Type 2 diabetes mellitus
2	73558	G25.81	Restless legs syndrome
3	73558	E11	Type 2 diabetes mellitus
4	73558	E11.2	Type 2 diabetes mellitus with kidney complicat...
5	73558	E78.0	Pure hypercholesterolemia
6	73559	E11	Type 2 diabetes mellitus
7	73559	E11	Type 2 diabetes mellitus
8	73559	K86.9	Disease of pancreas, unspecified
9	73559	E78.0	Pure hypercholesterolemia

In [6]:
```
data_questionnaire.describe()

# Note: descriptive analysis on data_questionnaire will be done at a later
 step, after merging the 2 datasets.
```

Out[6]:

	ACD011A	ACD011B	ACD011C	ACD040	ACD110	ALQ101	ALQ110	ALQ12
count	5759.0	16.0	171.0	2374.000000	1007.000000	5421.000000	1631.000000	4479.000
mean	1.0	8.0	9.0	3.101095	2.956306	1.311197	1.594727	4.709
std	0.0	0.0	0.0	1.511821	1.733794	0.545023	0.615303	34.428
min	1.0	8.0	9.0	1.000000	1.000000	1.000000	1.000000	0.000
25%	1.0	8.0	9.0	2.000000	1.000000	1.000000	1.000000	1.000
50%	1.0	8.0	9.0	3.000000	3.000000	1.000000	2.000000	2.000
75%	1.0	8.0	9.0	4.000000	5.000000	2.000000	2.000000	4.000
max	1.0	8.0	9.0	9.000000	5.000000	9.000000	9.000000	999.000

8 rows × 950 columns

Descriptive statistical analysis is irrelevant for data_medications, since we are only dealing with 'SEQN', which is the respondant sequence number.

In [7]:
```
data_medications.describe()
```

Out[7]:

	SEQN
count	20194.000000
mean	78545.350946
std	2933.000334

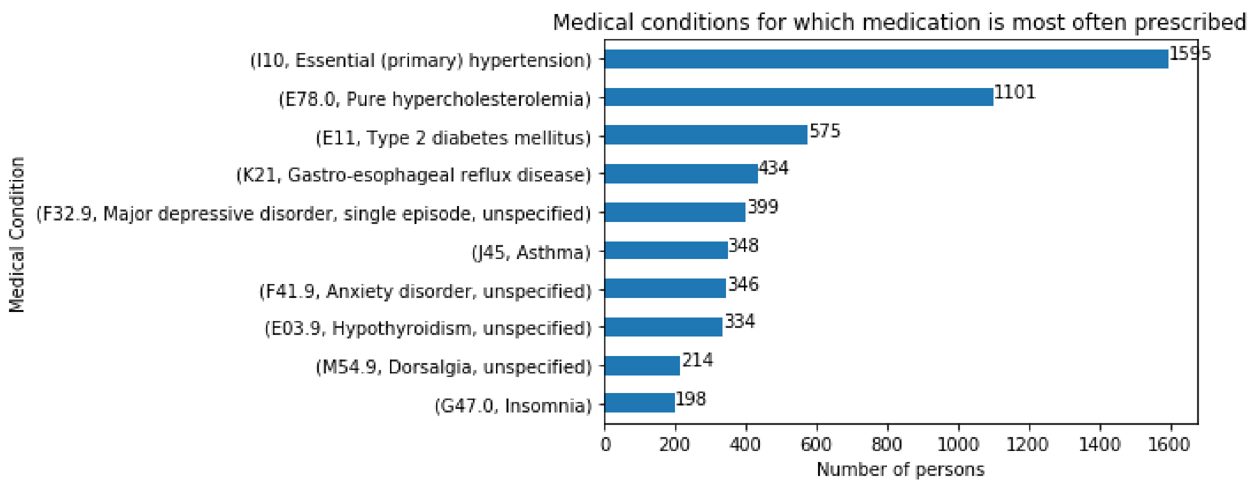
min	73557.000000
25%	75984.250000
50%	78506.000000
75%	81063.000000
max	83731.000000

Defining Questions

In [8]:
```
pd.set_option('display.max_rows',500)
pd.set_option('display.max_columns',500)


# List of the most frequent medical diagnoses
# Hypertension is the most frequently seen condition in this dataset.
# Therefore, this project will revolve around this topic

patients_conditions = data_medications.drop_duplicates(subset=['SEQN', 'RX
DRSC1'], keep='first').copy()

grouped_conditions = patients_conditions.groupby(['RXDRSC1', 'RXDRSD1'])
df_grouped_conditions = grouped_conditions['SEQN'].count().sort_values(asc
ending=False).head(10)
print(df_grouped_conditions)
```

```
RXDRSC1  RXDRSD1
I10      Essential (primary) hypertension                          1595
E78.0    Pure hypercholesterolemia                                 1101
E11      Type 2 diabetes mellitus                                   575
K21      Gastro-esophageal reflux disease                           434
F32.9    Major depressive disorder, single episode, unspecified     399
J45      Asthma                                                     348
F41.9    Anxiety disorder, unspecified                              346
E03.9    Hypothyroidism, unspecified                                334
M54.9    Dorsalgia, unspecified                                     214
G47.0    Insomnia                                                   198
Name: SEQN, dtype: int64
```

In [9]:
```
import matplotlib.pyplot as plt
%matplotlib inline


df_grouped_conditions.plot.barh().invert_yaxis()
plt.title('Medical conditions for which medication is most often prescribe
d')
plt.xlabel('Number of persons', fontsize=10)
plt.ylabel('Medical Condition', fontsize=10)
for index,data in enumerate(df_grouped_conditions):
    plt.text(x=data, y =index , s=f"{data}" , fontdict=dict(fontsize=10))
#plt.show()
import os
outdir = './figs/'
if not os.path.exists(outdir):
    os.mkdir(outdir)

plt.savefig(outdir+'medical_conditions.png', bbox_inches = 'tight')
```



Datasets and indicators that will be used in this project:

data_medications

```
* SEQN: Respondent sequence number
* RXDRSC1:  ICD-10-CM code 1
* RXDRSD1: ICD-10-CM code 1 description
```

data_questionnaire

```
* All indicators
```

Step 2: Basic Statistics (Profiling)

In [10]:
```
print ('Number of participants in data_questionnaire: {}'.format(data_ques
tionnaire.shape[0]))
```

Number of participants in data_questionnaire: 10175

In [11]:
```
print ('Number of rows in data_medications: {}'.format(data_medications.sh
ape[0]))
```

Number of rows in data_medications: 20194

In [12]:
```
print ('Number of variables in data_questionnaire: {}'.format(data_questio
nnaire.shape[1]))
```

Number of variables in data_questionnaire: 952

Step 3: Dealing with duplicates (Cleaning)

data_questionnaire

In [13]:
```
# No duplicate participants in data_questionnaire

nb_duplicates = sum(data_questionnaire.duplicated(keep=False))
print('Number of duplicate participants in data_questionnaire: ' + str(nb_
duplicates))
```

```
Number of duplicate participants in data_questionnaire: 0
```

data_medications

This is a long-format table with duplicate participant rows, where each rows represents a unique combination of ['SEQN', 'RXDDRUG'] columns (=> a prescribed drug to a patient)

There will certainly be duplicates 'SEQN', or duplicates 'RXDRSC1'(if a patient takes more than one medication for the same diagnosed condition)

In [14]:
```
print ('Number of rows in data_medications: {}'.format(data_medications.sh
ape[0]))

nb_duplicates = sum(data_medications.duplicated(subset=['SEQN'], keep=Fals
e))
print('Number of duplicate participants in data_medications: ' + str(nb_du
plicates))
```

```
Number of rows in data_medications: 20194
Number of duplicate participants in data_medications: 12777
```

- We have to filter participants based on hypertension diagnosis (ICD10 code: 'I10')

In [15]:
```
print ('Number of rows in data_medications: {}'.format(data_medications.sh
ape[0]))

unique_participants = sum(data_medications.duplicated(subset=['SEQN'], kee
p=False))
print('Number of unique participants in data_medications: ' + str(unique_p
articipants))
```

```
Number of rows in data_medications: 20194
Number of unique participants in data_medications: 12777
```

In [16]:
```
rows_with_hypertension = data_medications[(data_medications.RXDRSC1=='I10'
)]

nb_rows_with_hypertension = len(rows_with_hypertension)
print('Number of rows with hypertension in data_medications: ' + str(nb_ro
ws_with_hypertension))


nb_duplicate_patients_hypertension = sum(rows_with_hypertension.duplicated
(subset=['SEQN'], keep=False))
print('Number of duplicate patients with hypertension in data_medications:
```

```
' + str(nb_duplicate_patients_hypertension))
```

```
Number of rows with hypertension in data_medications: 2421
Number of duplicate patients with hypertension in data_medications: 1431
```

In [17]:
```
# Remove duplicate patient entries
unique_patients = rows_with_hypertension.drop_duplicates(subset=['SEQN'],
keep='first').copy()
# set 'SEQN' as table index
unique_patients = unique_patients.set_index('SEQN')

nb_unique_patients = len(unique_patients)
print('Number of unique patients with hypertension: ' + str(nb_unique_pati
ents))
```

```
Number of unique patients with hypertension: 1595
```

Data Matching & Merging

In [18]:
```
data = pd.merge(data_demographic, data_questionnaire, how='outer', left_in
dex=True, right_index=True)
data = pd.merge(unique_patients, data, how='outer', left_index=True, right
_index=True)

# Preview of data
data.head(10)
```

Out[18]:

	RXDRSC1	RXDRSD1	RIAGENDR	RIDAGEYR	INDHHIN2	INDFMIN2	ACD011A	ACD011B	AC
SEQN									
73557	NaN	NaN	1	69	4.0	4.0	1.0	NaN	
73558	NaN	NaN	1	54	7.0	7.0	1.0	NaN	
73559	I10	Essential (primary) hypertension	1	72	10.0	10.0	1.0	NaN	
73560	NaN	NaN	1	9	9.0	9.0	1.0	NaN	
73561	I10	Essential (primary) hypertension	2	73	15.0	15.0	1.0	NaN	
73562	I10	Essential (primary) hypertension	1	56	9.0	9.0	NaN	NaN	
73563	NaN	NaN	1	0	15.0	15.0	NaN	NaN	
73564	NaN	NaN	2	61	10.0	10.0	1.0	NaN	
73565	NaN	NaN	1	42	15.0	15.0	NaN	NaN	
73566	NaN	NaN	2	56	4.0	4.0	1.0	NaN	

10 rows × 958 columns

Basic Stats

In [19]:
```
rows_count = data.shape[0] # Number of rows

columns_count = data.shape[1] # Number of columns

rows_count
```

Out[19]:
```
10175
```

Cleaning

There are no rows with all NaN column values.

In [20]:
```
# Drops rows where all column values are NaN
data.dropna(axis=0, how='all')
data.shape
```

Out[20]:
```
(10175, 958)
```

Replacing missing values

In [21]:
```
dont_know_values = [99.0, 999.0]
refused_to_answer_values = [77.0, 777.0]

values_to_replace = dont_know_values + refused_to_answer_values

for value in values_to_replace:
    data.replace(float(value), np.nan, inplace=True)

data.replace('', np.nan, inplace=True)

data.head(10)
```

Out[21]:

	RXDRSC1	RXDRSD1	RIAGENDR	RIDAGEYR	INDHHIN2	INDFMIN2	ACD011A	ACD011B	AC
SEQN									
73557	NaN	NaN	1	69.0	4.0	4.0	1.0	NaN	
73558	NaN	NaN	1	54.0	7.0	7.0	1.0	NaN	
73559	I10	Essential (primary) hypertension	1	72.0	10.0	10.0	1.0	NaN	
73560	NaN	NaN	1	9.0	9.0	9.0	1.0	NaN	
73561	I10	Essential (primary) hypertension	2	73.0	15.0	15.0	1.0	NaN	

73562	I10	Essential (primary) hypertension	1	56.0	9.0	9.0	NaN	NaN
73563	NaN	NaN	1	0.0	15.0	15.0	NaN	NaN
73564	NaN	NaN	2	61.0	10.0	10.0	1.0	NaN
73565	NaN	NaN	1	42.0	15.0	15.0	NaN	NaN
73566	NaN	NaN	2	56.0	4.0	4.0	1.0	NaN

10 rows × 958 columns

For each column containing 7.0/9.0 numeric values, we need to verify whether we have continuous values (i.e. age, quantity) or categorical values (i.e. 'refused' or 'don't know' value)

In [22]:
```python
arr_values_to_check = [7.0, 9.0]

for value in arr_values_to_check:
    print('Indexes where value = '+str(value))
    print(data.columns[(data == value).iloc[0]])
```

```
Indexes where value = 7.0
Index(['DUQ240', 'SLD010H', 'SMD480'], dtype='object')
Indexes where value = 9.0
Index(['INQ244', 'MCQ195', 'OSQ200'], dtype='object')
```

DUQ240 - Ever used cocaine/heroin/methamphetamine

Description: Have you ever used cocaine, crack cocaine, heroin, or methamphetamine?

Code or Value	Value Description	Count	Cumulative	Skip to Item
1	Yes	723	723	
2	No	3800	4523	DUQ370
7	Refused	10	4533	DUQ370
9	Don't know	7	4540	DUQ370
.	Missing	517	5057	

SLD010H - How much sleep do you get (hours)?

Description: The next set of questions is about your sleeping habits. How much sleep)do you/does SP) usually get at night on weekdays or workdays?

Code or Value	Value Description	Count	Cumulative	Skip to Item
2 to 11	Range of Values	6835	6835	
12	12 hours or more	39	6874	
77	Refused	0	6874	

99	Don't know	8	6882	
.	Missing	7	6889	

SMD480 - In past week # days person smoked inside

Description: (Not counting decks, porches, or detached garages) During the past 7 days, that is since last [TODAY'S DAY OF WEEK], on how many days did {anyone who lives here/you}, smoke tobacco inside this home?

Instructions: ENTER NUMBER OF DAYS FROM 0 TO 7. CAPI INSTRUCTION: IF ONLY ONE PERSON LIVING IN HOUSEHOLD DISPLAY "you.." IF MORE THAN ONE PERSON LIVING IN HOUSEHOLD, DISPLAY "anyone who lives here.."

Code or Value	Value Description	Count	Cumulative	Skip to Item
0	0	98	98	
1	1	179	277	
2	2	230	507	
3	3	49	556	
4	4	35	591	
5	5	27	618	
6	6	7	625	
7	7	679	1304	
77	Refused	0	1304	
99	Don't know	3	1307	
.	Missing	8868	10175	

INQ244 - Family has savings more than $5000

Description: Do you/NAMES OF OTHER FAMILY/you and NAMES OF FAMILY MEMBERS have more than $5,000 in savings at this time? Please include money in your checking accounts.

Code or Value	Value Description	Count	Cumulative	Skip to Item
1	Yes	449	449	End of Section
2	No	4803	5252	
7	Refused	107	5359	End of Section
9	Don't know	141	5500	End of Section
.	Missing	4256	9756	

MCQ195 - Which type of arthritis was it?

Description: Which type of arthritis was it?

Code or Value	Value Description	Count	Cumulative	Skip to Item
1	Osteoarthritis or degenerative arthritis	576	576	
2	Rheumatoid arthritis	249	825	
3	Psoriatic arthritis	17	842	
4	Other	104	946	
7	Refused	0	946	
9	Don't know	419	1365	
.	Missing	7999	9364	

OSQ200 - Did father ever fracture hip?

Description: Did (your/SP's) biological father ever fracture his hip?

Code or Value	Value Description	Count	Cumulative	Skip to Item
1	Yes	109	109	
2	No	5766	5875	
7	Refused	4	5879	
9	Don't know	339	6218	
.	Missing	0	6218	

Conclusion:

Categorical columns (where 7.0 or 9.0 represents 'refused' or 'don't know'): DUQ240, INQ244, MCQ195, OSQ200

In [23]:
```
columns = ['DUQ240', 'INQ244', 'MCQ195', 'OSQ200', 'BPQ020']
values_to_replace = [7.0, 9.0]

for value in values_to_replace:
    data[columns] = data[columns].replace(float(value), np.nan, inplace=Fa
lse)


data[columns].head(10)
```

Out[23]:

	DUQ240	INQ244	MCQ195	OSQ200	BPQ020
SEQN					
73557	NaN	NaN	NaN	NaN	1.0
73558	1.0	1.0	NaN	2.0	1.0
73559	NaN	NaN	NaN	2.0	1.0

73560	NaN	NaN	NaN	NaN	NaN
73561	NaN	NaN	NaN	2.0	1.0
73562	2.0	NaN	2.0	2.0	1.0
73563	NaN	NaN	NaN	NaN	NaN
73564	2.0	NaN	2.0	2.0	1.0
73565	NaN	NaN	NaN	2.0	2.0
73566	2.0	2.0	2.0	2.0	2.0

Replace hypertension condition strings to bool (0.0 or 1.0)

In [24]:
```
hypertension_values = ['I10', 'Essential (primary) hypertension']
# Replace all values 'I10' or 'Essential (primary) hypertension' with 1.0
for value in hypertension_values:
    data.replace(value, 1, inplace=True)

# Replace all values NaN in these columns with 0.0
data['RXDRSC1'].replace(np.nan, 0, inplace=True)
data['RXDRSD1'].replace(np.nan, 0, inplace=True)

data.head(5)
```

Out[24]:

	RXDRSC1	RXDRSD1	RIAGENDR	RIDAGEYR	INDHHIN2	INDFMIN2	ACD011A	ACD011B	AC
SEQN									
73557	0.0	0.0	1	69.0	4.0	4.0	1.0	NaN	
73558	0.0	0.0	1	54.0	7.0	7.0	1.0	NaN	
73559	1.0	1.0	1	72.0	10.0	10.0	1.0	NaN	
73560	0.0	0.0	1	9.0	9.0	9.0	1.0	NaN	
73561	1.0	1.0	2	73.0	15.0	15.0	1.0	NaN	

5 rows × 958 columns

Removing columns with string dtype

In [25]:
```
df_subset = data.select_dtypes(exclude=[np.number])

# SMDUPCA - Cig 12-digit Universal Product Code-UPC
# Description: Cigarette 12-digit Universal Product Code (UPC)
# Link: https://wwwn.cdc.gov/Nchs/Nhanes/2013-2014/SMQ_H.htm#SMDUPCA
data['SMDUPCA']

# SMD100BR - Cigarette Brand/sub-brand
# Description: BRAND OF CIGARETTES SMOKED BY SP (SUB-BRAND INCLUDED IF APP
LICABLE AND AVAILABLE)
```

```
# Link: https://wwwn.cdc.gov/Nchs/Nhanes/2013-2014/SMQ_H.htm#SMD100BR
data['SMD100BR']


data = data.drop('SMDUPCA', axis=1)
data = data.drop('SMD100BR', axis=1)
```

Descriptive Analysis

The count, for each column, denotes how many patients have answered this column.

There are many columns for which the number of patients (n) is too small to draw conclusions.

In [26]:
```
data.describe()
```

Out[26]:

	RXDRSC1	RXDRSD1	RIAGENDR	RIDAGEYR	INDHHIN2	INDFMIN2	ACD011A
count	10175.000000	10175.000000	10175.000000	10137.000000	9715.000000	9738.000000	5759.0
mean	0.156757	0.156757	1.508305	31.313505	8.489758	8.208256	1.0
std	0.363589	0.363589	0.499956	24.307553	4.475009	4.523212	0.0
min	0.000000	0.000000	1.000000	0.000000	1.000000	1.000000	1.0
25%	0.000000	0.000000	1.000000	10.000000	5.000000	5.000000	1.0
50%	0.000000	0.000000	2.000000	26.000000	7.000000	7.000000	1.0
75%	0.000000	0.000000	2.000000	52.000000	14.000000	14.000000	1.0
max	1.000000	1.000000	2.000000	80.000000	15.000000	15.000000	1.0

8 rows × 956 columns

Deleting columns where std() = 0

Those columns are removed to simplify the analysis of the datasets. These are often variables split into multiple components. i.e. 'ACD011A' (Speak English at home) 1 or NaN 'ACD011B' (Speak Spanish at home) 8 or NaN 'ACD011C' (Speak other language at home) 9 or NaN

In [27]:
```
arr_columns_to_remove = data.std()[data.std() == 0.0].index.values
print('Columns to remove (std() = 0): ')
print(arr_columns_to_remove)

print('The number of attributes went')
print('from ' + str(columns_count))
data = data.drop(arr_columns_to_remove, axis=1)
print('to ' + str(data.shape[1]))
```

```
Columns to remove (std() = 0):
['ACD011A' 'ACD011B' 'ACD011C' 'CSQ120A' 'CSQ120B' 'CSQ120C' 'CSQ120D'
 'CSQ120E' 'CSQ120F' 'CSQ120G' 'CSQ120H' 'CDQ009A' 'CDQ009B' 'CDQ009C'
 'CDQ009D' 'CDQ009E' 'CDQ009F' 'CDQ009G' 'CDQ009H' 'DIQ175A' 'DIQ175B'
 'DIQ175C' 'DIQ175D' 'DIQ175E' 'DIQ175F' 'DIQ175G' 'DIQ175H' 'DIQ175I'
 'DIQ175J' 'DIQ175K' 'DIQ175L' 'DIQ175M' 'DIQ175N' 'DIQ175O' 'DIQ175P'
 'DIQ175Q' 'DIQ175R' 'DIQ175S' 'DIQ175T' 'DIQ175U' 'DIQ175V' 'DIQ175W'
 'DIQ175X' 'DBQ073A' 'DBQ073B' 'DBQ073C' 'DBQ073D' 'DBQ073E' 'DBQ073U'
 'DBQ223A' 'DBQ223B' 'DBQ223C' 'DBQ223D' 'DBQ223E' 'DBQ223U' 'DUQ380B'
 'DUQ380C' 'DUQ380D' 'DUQ380E' 'HIQ031A' 'HIQ031B' 'HIQ031C' 'HIQ031D'
 'HIQ031E' 'HIQ031F' 'HIQ031G' 'HIQ031H' 'HIQ031I' 'HIQ031J' 'HIQ031AA'
 'MCQ230D' 'OHQ780A' 'OHQ780B' 'OHQ780C' 'OHQ780D' 'OHQ780E' 'OHQ780F'
 'OHQ780G' 'OHQ780H' 'OHQ780I' 'OHQ780J' 'OHQ780K' 'OSQ040AC' 'OSQ040BE'
 'OSQ040BF' 'OSQ040CC' 'OSQ090F' 'OSQ090H' 'OSQ120H' 'OSQ160A' 'OSQ160B'
 'PAQ724A' 'PAQ724B' 'PAQ724C' 'PAQ724D' 'PAQ724E' 'PAQ724F' 'PAQ724G'
 'PAQ724H' 'PAQ724I' 'PAQ724J' 'PAQ724K' 'PAQ724L' 'PAQ724M' 'PAQ724N'
 'PAQ724O' 'PAQ724P' 'PAQ724Q' 'PAQ724R' 'PAQ724S' 'PAQ724T' 'PAQ724U'
 'PAQ724V' 'PAQ724W' 'PAQ724X' 'PAQ724Y' 'PAQ724Z' 'PAQ724AA' 'PAQ724AB'
 'PAQ724AC' 'PAQ724AD' 'PAQ724AE' 'PAQ724AF' 'PAQ724CM' 'PAQ759A'
 'PAQ759B' 'PAQ759C' 'PAQ759D' 'PAQ759E' 'PAQ759F' 'PAQ759G' 'PAQ759H'
 'PAQ759I' 'PAQ759J' 'PAQ759K' 'PAQ759L' 'PAQ759M' 'PAQ759N' 'PAQ759O'
 'PAQ759P' 'PAQ759Q' 'PAQ759R' 'PAQ759S' 'PAQ759U' 'PAQ759V' 'PAQ772B'
 'PAQ772C' 'RHQ542A' 'RHQ542B' 'RHQ542C' 'RHQ542D' 'SMQ665C' 'SMQ690A'
 'SMQ690B' 'SMQ690C' 'SMQ690G' 'SMQ690H' 'SMQ690D' 'SMQ690E' 'SMQ690I'
 'SMQ690F' 'WHD080A' 'WHD080B' 'WHD080C' 'WHD080D' 'WHD080E' 'WHD080F'
 'WHD080G' 'WHD080H' 'WHD080I' 'WHD080J' 'WHD080K' 'WHD080M' 'WHD080N'
 'WHD080O' 'WHD080P' 'WHD080Q' 'WHD080R' 'WHD080S' 'WHD080T' 'WHD080U'
 'WHD080L']
The number of attributes went
from 958
to 774
```

Count

STATISTICAL SIGNIFICANCE

In [28]:
```
nb_samples_required = 310
# based on
# https://select-statistics.co.uk/calculators/sample-size-calculator-popul
ation-proportion/
# confidence=0.95, n=1595

print('The number of attributes went')
print('from ' + str(columns_count))
data = data.loc[:, data.count() >= nb_samples_required]

print('to ' + str(data.shape[1]))
```

```
The number of attributes went
from 958
to 571
```

In [29]:
```
# Preview of data table with only statistically significant columns
data.head(10)
```

Out[29]:

	RXDRSC1	RXDRSD1	RIAGENDR	RIDAGEYR	INDHHIN2	INDFMIN2	ACD040	ACD110	ALQ1
SEQN									
73557	0.0	0.0	1	69.0	4.0	4.0	NaN	NaN	1

73558	0.0	0.0	1	54.0	7.0	7.0	NaN	NaN	1
73559	1.0	1.0	1	72.0	10.0	10.0	NaN	NaN	1
73560	0.0	0.0	1	9.0	9.0	9.0	NaN	NaN	N
73561	1.0	1.0	2	73.0	15.0	15.0	NaN	NaN	1
73562	1.0	1.0	1	56.0	9.0	9.0	4.0	NaN	1
73563	0.0	0.0	1	0.0	15.0	15.0	NaN	NaN	N
73564	0.0	0.0	2	61.0	10.0	10.0	NaN	NaN	2
73565	0.0	0.0	1	42.0	15.0	15.0	5.0	NaN	N
73566	0.0	0.0	2	56.0	4.0	4.0	NaN	NaN	1

10 rows × 571 columns

Exploratory Data Analysis

Correlation

In [30]:
```
corr = data.corr()
# corr
```

In [31]:
```
df = data.loc[:, abs(data.corr()['RXDRSC1']) > 0.2]
df.head()
```

Out[31]:

	RXDRSC1	RXDRSD1	RIDAGEYR	BPQ020	BPD035	BPQ040A	BPQ050A	BPQ056	BPQ059
SEQN									
73557	0.0	0.0	69.0	1.0	62.0	1.0	2.0	2.0	2.0
73558	0.0	0.0	54.0	1.0	53.0	2.0	NaN	1.0	2.0
73559	1.0	1.0	72.0	1.0	40.0	1.0	1.0	2.0	2.0
73560	0.0	0.0	9.0	NaN	NaN	NaN	NaN	NaN	NaN
73561	1.0	1.0	73.0	1.0	55.0	1.0	1.0	1.0	2.0

In [32]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10175 entries, 73557 to 83731
Data columns (total 39 columns):
RXDRSC1      10175 non-null float64
RXDRSD1      10175 non-null float64
RIDAGEYR     10137 non-null float64
BPQ020       6459 non-null float64
BPD035       2127 non-null float64
BPQ040A      2174 non-null float64
BPQ050A      1815 non-null float64
```

```
BPQ056       6464 non-null float64
BPQ059       6464 non-null float64
BPQ080       6464 non-null float64
BPQ070       4620 non-null float64
BPQ090D      4620 non-null float64
HSD010       6467 non-null float64
HSAQUEX      9422 non-null float64
DIQ010       9769 non-null float64
DLQ050       8780 non-null float64
HUQ010       10175 non-null int64
HUQ051       10164 non-null float64
IMQ090       796 non-null float64
INQ030       10052 non-null float64
INQ080       10052 non-null float64
MCQ160A      5769 non-null float64
MCQ365A      6464 non-null float64
MCQ365B      6464 non-null float64
MCQ365C      6464 non-null float64
MCQ365D      6464 non-null float64
MCQ370C      6464 non-null float64
OCD150       6459 non-null float64
PFQ051       5769 non-null float64
PFQ054       5769 non-null float64
PFQ090       5769 non-null float64
PAQ650       7147 non-null float64
RHQ031       3256 non-null float64
RHD280       2620 non-null float64
RXQ510       3815 non-null float64
SMD055       1203 non-null float64
SMAQUEX2     7168 non-null float64
SMQ856       6113 non-null float64
SMAQUEX.y    6979 non-null float64
dtypes: float64(38), int64(1)
memory usage: 3.1 MB
```

Remove irrelevant columns

In [33]:
```
# Remove questionnaire indexes columns (i.e. 'source of questionnaire admi
nistration')
df = df.drop(['HSAQUEX', 'SMAQUEX.y', 'SMAQUEX2'], axis=1)

# Remove duplicate hypertension diagnosis column
df = df.drop(['RXDRSD1'], axis=1)
df.head()
```

Out[33]:

	RXDRSC1	RIDAGEYR	BPQ020	BPD035	BPQ040A	BPQ050A	BPQ056	BPQ059	BPQ080	BF
SEQN										
73557	0.0	69.0	1.0	62.0	1.0	2.0	2.0	2.0	1.0	
73558	0.0	54.0	1.0	53.0	2.0	NaN	1.0	2.0	1.0	
73559	1.0	72.0	1.0	40.0	1.0	1.0	2.0	2.0	1.0	
73560	0.0	9.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

73561	1.0	73.0	1.0	55.0	1.0	1.0	1.0	2.0	2.0

Export cleaned data to csv

In [34]:
```
import os

outputted_file_name = 'project_data.csv'

outdir = './data/cleaned/'
if not os.path.exists(outdir):
    os.mkdir(outdir)

full_path = os.path.join(outdir, outputted_file_name)

df.to_csv(full_path)
```

Create a dir where figures are saved

In [1]:
```
# Make dir for figures

import os
outdir = './figs/'
if not os.path.exists(outdir):
    os.mkdir(outdir)
```

Loading pre-cleaned data

In [2]:
```
import pandas as pd
import numpy as np


data = pd.read_csv('./data/cleaned/project_data.csv', index_col='SEQN')

data.columns
```

Out[2]:
```
Index(['RXDRSC1', 'RIDAGEYR', 'BPQ020', 'BPD035', 'BPQ040A', 'BPQ050A',
       'BPQ056', 'BPQ059', 'BPQ080', 'BPQ070', 'BPQ090D', 'HSD010', 'DIQ01
0',
       'DLQ050', 'HUQ010', 'HUQ051', 'IMQ090', 'INQ030', 'INQ080', 'MCQ160
A',
       'MCQ365A', 'MCQ365B', 'MCQ365C', 'MCQ365D', 'MCQ370C', 'OCD150',
       'PFQ051', 'PFQ054', 'PFQ090', 'PAQ650', 'RHQ031', 'RHD280', 'RXQ510
',
       'SMD055', 'SMQ856'],
      dtype='object')
```

Removing participants with missing value age

In [3]:
```
data = data[data.RIDAGEYR.notnull()]
# data = data[data.RIDAGEYR != 0]

data.shape
```

Out[3]:
```
(10137, 35)
```

Replacing BPQ020 missing values

BPQ020 - Ever told you had high blood pressure

(Have you/Has SP) ever been told by a doctor or other health professional that (you/s/he) had hypertension, also called high blood pressure?

Code or Value	Value Description	Count	Cumulative	Skip to Item

1	Yes	2174	2174	
2	No	4285	6459	BPQ056
7	Refused	0	6459	BPQ056
9	Don't know	5	6464	BPQ056
.	Missing	0	6464	

In [4]:
```
data[['BPQ020']] = data[['BPQ020']].fillna(value=2)
data.head()
```

Out[4]:

	RXDRSC1	RIDAGEYR	BPQ020	BPD035	BPQ040A	BPQ050A	BPQ056	BPQ059	BPQ080
SEQN									
73557	0.0	69.0	1.0	62.0	1.0	2.0	2.0	2.0	1.0
73558	0.0	54.0	1.0	53.0	2.0	NaN	1.0	2.0	1.0
73559	1.0	72.0	1.0	40.0	1.0	1.0	2.0	2.0	1.0
73560	0.0	9.0	2.0	NaN	NaN	NaN	NaN	NaN	NaN
73561	1.0	73.0	1.0	55.0	1.0	1.0	1.0	2.0	2.0

5 rows × 35 columns

Correlation

In [5]:
```
corr = data.corr()
corr
```

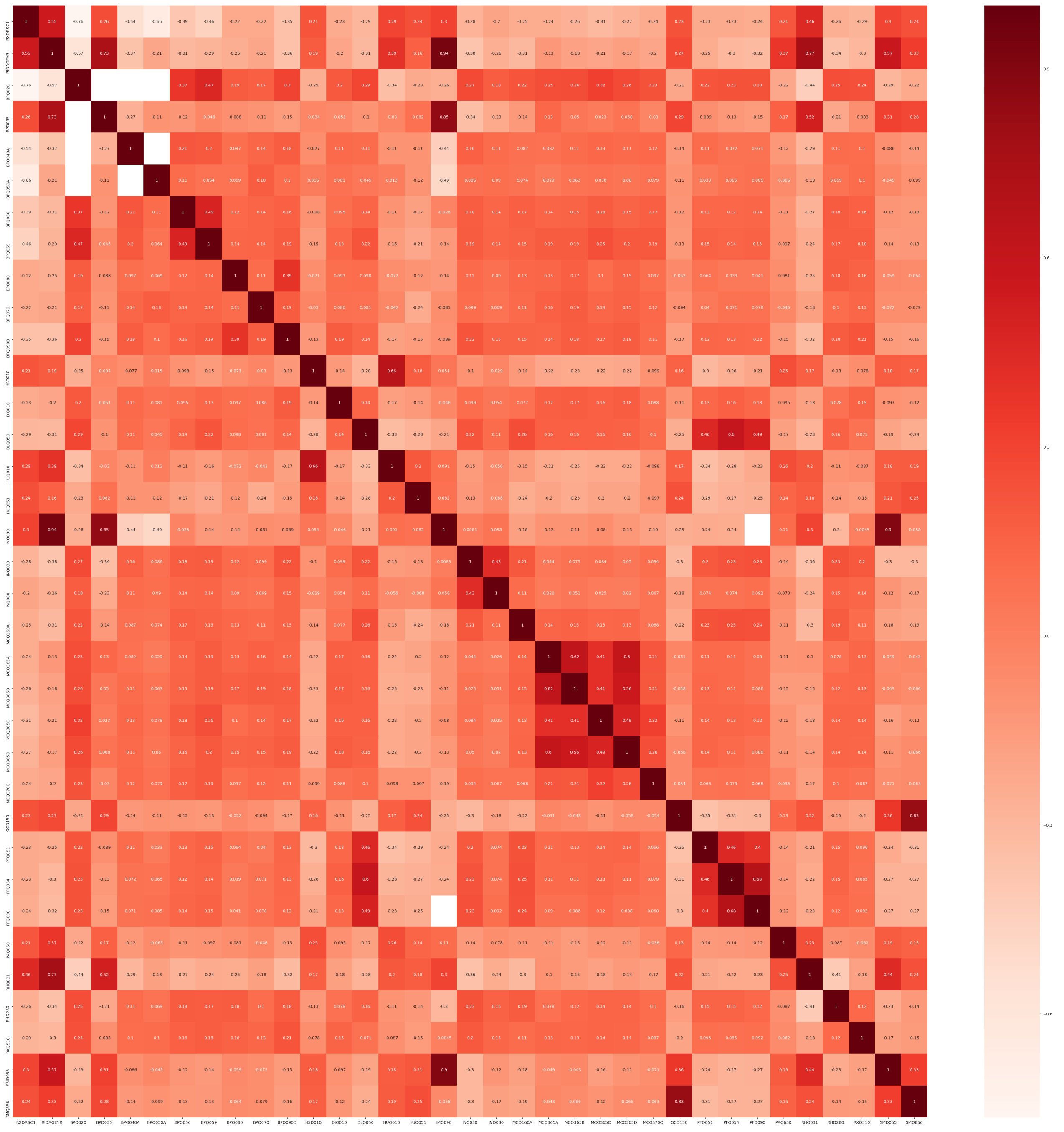
Out[5]:

	RXDRSC1	RIDAGEYR	BPQ020	BPD035	BPQ040A	BPQ050A	BPQ056	BPQ059
RXDRSC1	1.000000	0.548234	-0.764652	0.258779	-0.541644	-0.659324	-0.388907	-0.459707
RIDAGEYR	0.548234	1.000000	-0.573809	0.731509	-0.367382	-0.212364	-0.308514	-0.287059
BPQ020	-0.764652	-0.573809	1.000000	NaN	NaN	NaN	0.372050	0.467402
BPD035	0.258779	0.731509	NaN	1.000000	-0.268173	-0.114092	-0.119344	-0.045514
BPQ040A	-0.541644	-0.367382	NaN	-0.268173	1.000000	NaN	0.212280	0.200222
BPQ050A	-0.659324	-0.212364	NaN	-0.114092	NaN	1.000000	0.105112	0.064371
BPQ056	-0.388907	-0.308514	0.372050	-0.119344	0.212280	0.105112	1.000000	0.487027
BPQ059	-0.459707	-0.287059	0.467402	-0.045514	0.200222	0.064371	0.487027	1.000000
BPQ080	-0.221391	-0.249293	0.193353	-0.088076	0.096848	0.069305	0.123898	0.136410
BPQ070	-0.220526	-0.212413	0.172147	-0.109748	0.136918	0.177920	0.140420	0.138624
BPQ090D	-0.352857	-0.355333	0.300413	-0.153237	0.175279	0.102327	0.163820	0.190952
	0.214602	0.188328	-0.249317	-0.034392	-0.076721	0.015389	-0.097913	-0.147231

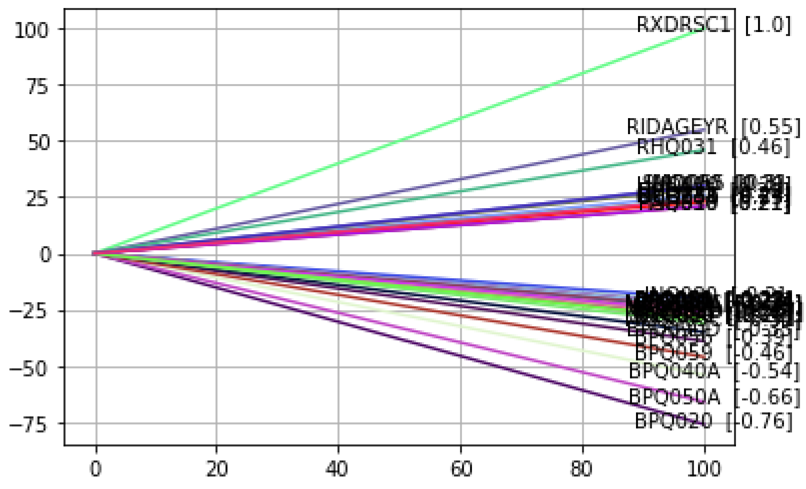
HSD010									
DIQ010	-0.234419	-0.204519	0.198002	-0.050881	0.114979	0.080958	0.095300	0.129000	(
DLQ050	-0.289263	-0.306293	0.288384	-0.104513	0.108370	0.045380	0.141450	0.215793	(
HUQ010	0.291399	0.393897	-0.339184	-0.030326	-0.110811	0.013248	-0.113556	-0.156377	-(
HUQ051	0.243855	0.161066	-0.232548	0.082026	-0.111656	-0.115218	-0.171996	-0.213573	-(
IMQ090	0.303978	0.940118	-0.257963	0.846697	-0.435781	-0.494604	-0.025743	-0.143930	-(
INQ030	-0.277743	-0.377502	0.265605	-0.337989	0.164286	0.086168	0.178113	0.190915	(
INQ080	-0.198916	-0.256106	0.175114	-0.225886	0.106064	0.090112	0.136876	0.140798	(
MCQ160A	-0.247166	-0.307936	0.217340	-0.143343	0.087009	0.073738	0.171419	0.153525	(
MCQ365A	-0.241523	-0.131983	0.253276	0.129514	0.081514	0.028963	0.141906	0.189059	(
MCQ365B	-0.263439	-0.180286	0.259141	0.049681	0.105788	0.063029	0.145668	0.189494	(
MCQ365C	-0.314929	-0.206061	0.319278	0.022698	0.130023	0.077687	0.180608	0.245871	(
MCQ365D	-0.269040	-0.166360	0.262832	0.067769	0.114314	0.059575	0.148608	0.195724	(
MCQ370C	-0.238794	-0.204849	0.232194	-0.030355	0.117986	0.079385	0.168798	0.193931	(
OCD150	0.233824	0.265806	-0.210165	0.293402	-0.135945	-0.107070	-0.121445	-0.128473	-(
PFQ051	-0.228599	-0.245380	0.218799	-0.088874	0.105717	0.032577	0.129583	0.152600	(
PFQ054	-0.229541	-0.302580	0.231657	-0.129526	0.071975	0.065440	0.124580	0.140557	(
PFQ090	-0.241789	-0.321320	0.231135	-0.153580	0.071256	0.085391	0.135180	0.152684	(
PAQ650	0.212924	0.369479	-0.218103	0.165588	-0.124524	-0.064538	-0.105330	-0.096715	-(
RHQ031	0.455096	0.770754	-0.435377	0.524250	-0.293751	-0.176466	-0.269522	-0.240671	-(
RHD280	-0.256650	-0.344301	0.251060	-0.213262	0.108394	0.068943	0.184066	0.173318	(
RXQ510	-0.287994	-0.297186	0.235579	-0.083011	0.099858	0.104193	0.159995	0.175067	(
SMD055	0.296148	0.566025	-0.293121	0.310689	-0.085655	-0.044890	-0.121126	-0.141457	-(
SMQ856	0.237187	0.334164	-0.221936	0.277148	-0.136316	-0.098860	-0.127133	-0.133366	-(

35 rows × 35 columns

```
In [6]: import matplotlib.pyplot as plt
        import seaborn as sns
        import numpy as np
        %matplotlib inline

        plt.figure(figsize=(50, 50), dpi=100)
        sns.heatmap(corr, annot=True, cmap=plt.cm.Reds)
        plt.show()
```

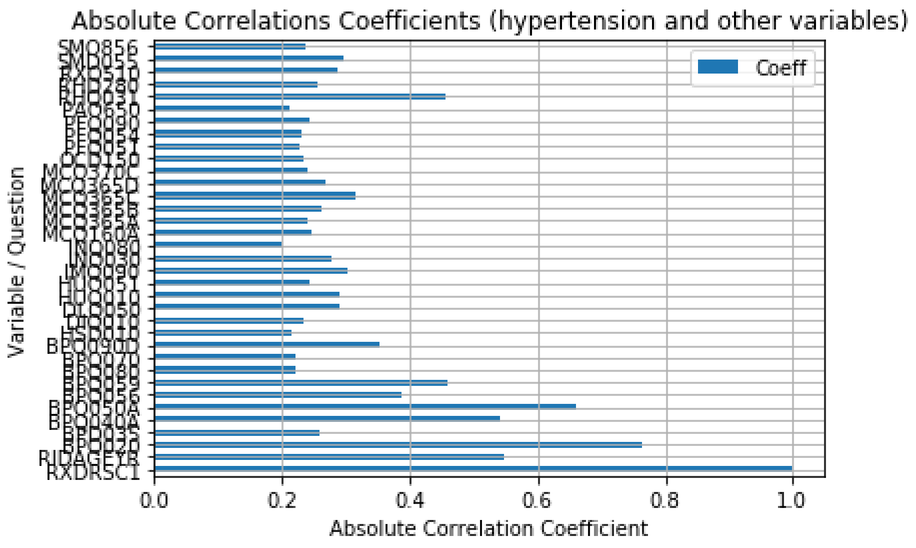


```
In [7]: corr = np.round(corr, decimals=2)

        fig, ax = plt.subplots()
        idx = 0
        for corr_coeff in corr['RXDRSC1']:
            x = np.linspace(0, 100, 150)
            y = corr_coeff*x
            ax.plot(x, y, color=np.random.random(3))
            ax.grid()
            ax.annotate(s=corr['RXDRSC1'].index[idx]+'  ['+str(corr_coeff)+']', xy
        =(x[-1],y[-1]), xytext=(5,0), textcoords='offset points', ha='center')
            idx +=1


        fig.savefig(outdir+'linear_corr_with_hypertension.png', bbox_inches = 'tig
        ht', dpi=300)
```



In [8]:
```
plt.figure(figsize=(50, 100))

ax = pd.DataFrame(abs(data.corrwith(abs(data.RXDRSC1)))).plot.barh(width=0
.5)
ax.grid(zorder=0)

plt.title('Absolute Correlations Coefficients (hypertension and other vari
ables)')
plt.xlabel('Absolute Correlation Coefficient', fontsize=10)
plt.ylabel('Variable / Question', fontsize=10)
plt.legend(['Coeff'])


plt.savefig(outdir+'corr_with_hypertension.png', bbox_inches = 'tight', dp
i=300)
```

```
<Figure size 3600x7200 with 0 Axes>
```



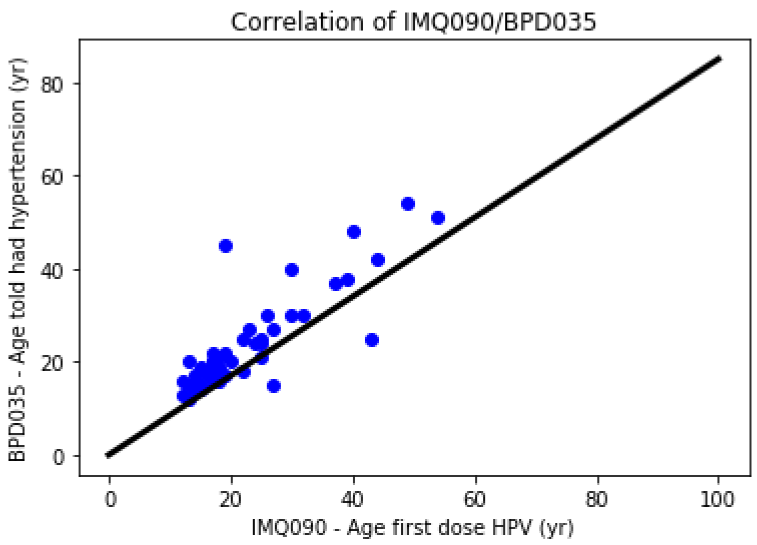
Exploring variables and correlations

In [9]:
```
df_bivar= data[['BPD035', 'IMQ090']]
df_bivar = df_bivar.dropna()
```

```
x = np.linspace(0, 100, len(df_bivar))
plt.scatter(df_bivar['IMQ090'], df_bivar['BPD035'],  color='blue')

y = corr['BPD035']['IMQ090']*x
plt.plot(x, y, color='black', linewidth=3)
plt.title('Correlation of IMQ090/BPD035')
plt.xlabel('IMQ090 - Age first dose HPV (yr)', fontsize=10)
plt.ylabel('BPD035 - Age told had hypertension (yr)', fontsize=10)


plt.savefig(outdir+'corr_IMQ090_BPD035.png', bbox_inches = 'tight', dpi=20
0)

print('Correlation coefficient: '+str(corr['BPD035']['IMQ090']))
len(df_bivar)

# This is a good correlation coefficient, however the number of samples N
is very small...
# However there exist some research about the link between HPV and hyperte
nsion!
```

```
Correlation coefficient: 0.85
```

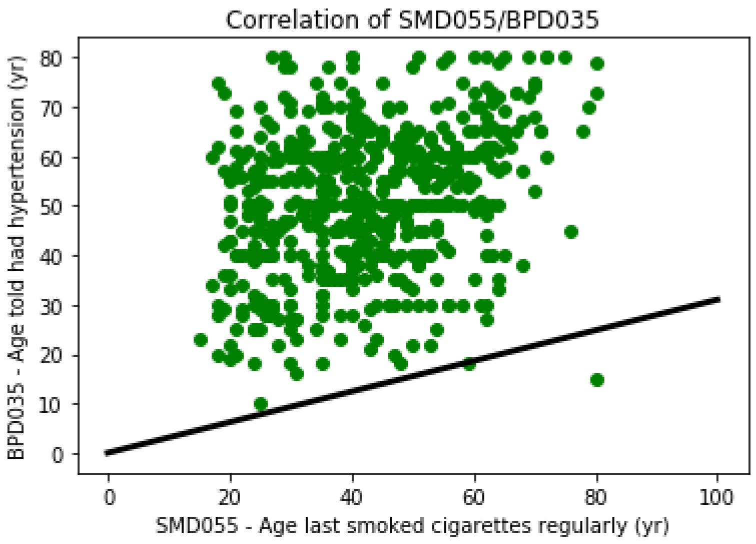
Out[9]:
```
46
```



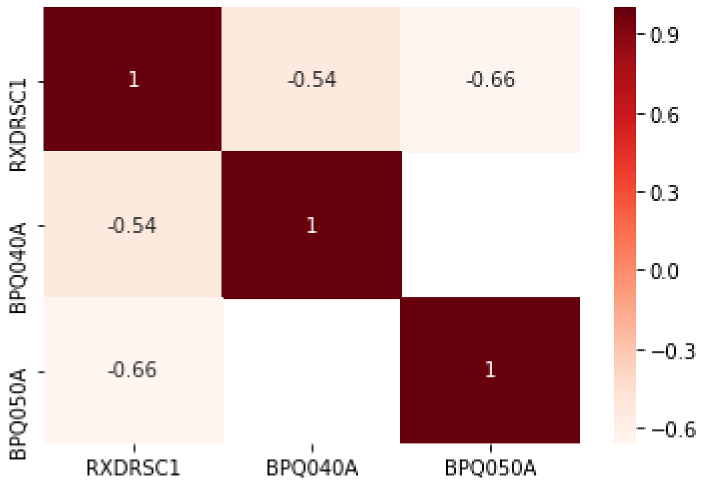
In [10]:
```
df_bivar= data[['BPD035', 'SMD055']]
df_bivar = df_bivar.dropna()

x = np.linspace(0, 100, len(df_bivar))
plt.scatter(df_bivar['SMD055'], df_bivar['BPD035'],  color='green')

y = corr['BPD035']['SMD055']*x
plt.plot(x, y, color='black', linewidth=3)
plt.title('Correlation of SMD055/BPD035')
plt.xlabel('SMD055 - Age last smoked cigarettes regularly (yr)', fontsize=
10)
plt.ylabel('BPD035 - Age told had hypertension (yr)', fontsize=10)
plt.savefig(outdir+'corr_SMD055_BPD035.png', bbox_inches = 'tight', dpi=20
0)

print('Correlation coefficient: '+str(corr['BPD035']['SMD055']))
len(df_bivar)

# The data here may not illustrate well the correlation...
# The sample size is larger, but the correlation seem to be weak.
```

```
Correlation coefficient: 0.31
```

Out[10]:
```
564
```



In [11]:
```
df_bp_subset = data[['RXDRSC1', 'BPQ040A', 'BPQ050A']]
bp_subset_corr = df_bp_subset.corr()

plt.figure()
sns.heatmap(bp_subset_corr, annot=True, cmap=plt.cm.Reds)
plt.show()
```



In [12]:
```
asf = data[['BPQ040A', 'BPQ050A']]
asf = asf.dropna()
asf.head(10)
```

Out[12]:

	BPQ040A	BPQ050A
SEQN		
73557	1.0	2.0
73559	1.0	1.0
73561	1.0	1.0
73562	1.0	1.0
73564	1.0	2.0
73571	1.0	1.0
73600	1.0	2.0
73613	1.0	1.0
73615	1.0	1.0
73626	1.0	1.0

% of missing values per column

In [13]:
```
data.isnull().sum(axis=0)/data.shape[0]
```

Out[13]:
```
RXDRSC1     0.000000
RIDAGEYR    0.000000
BPQ020      0.000000
BPD035      0.792345
BPQ040A     0.788004
BPQ050A     0.823419
BPQ056      0.366085
BPQ059      0.366085
BPQ080      0.366085
BPQ070      0.547697
BPQ090D     0.547697
HSD010      0.365197
DIQ010      0.040051
DLQ050      0.137615
HUQ010      0.000000
HUQ051      0.000986
IMQ090      0.921476
INQ030      0.012134
INQ080      0.012134
MCQ160A     0.434645
MCQ365A     0.366085
MCQ365B     0.366085
MCQ365C     0.366085
MCQ365D     0.366085
MCQ370C     0.366085
OCD150      0.366578
PFQ051      0.434645
PFQ054      0.434645
PFQ090      0.434645
PAQ650      0.298708
```

```
RHQ031      0.680182
RHD280      0.742922
RXQ510      0.627405
SMD055      0.883200
SMQ856      0.400710
dtype: float64
```

Classification

Q: Can we identify participants with hypertension based on some selected variable?

In [14]:
```
df = data[['RXDRSC1', 'BPQ080', 'PAQ650', 'HSD010', 'INQ030', 'INQ080']]
df = df.dropna()

target_column = 'RXDRSC1'

df.shape
```

Out[14]:
```
(5692, 6)
```

In [15]:
```
target = df[target_column]
data_ML = df.loc[:, df.columns != target_column]
```

In [16]:
```
from sklearn.model_selection import train_test_split

#split data set into train and test sets
data_train, data_test, target_train, target_test = train_test_split(data_M
L,target, test_size = 0.30, random_state = 10)
```

In [17]:
```
model_comparison = []
model_names = []
mean_comparison = []
```

In [18]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score

model = DecisionTreeClassifier().fit(data_train, target_train)
pred = model.predict(data_test)

print(accuracy_score(target_test, pred, normalize = True))

results = cross_val_score(model, data_train, target_train, scoring='accura
cy')
model_comparison.append(results)
model_names.append("DecisionTree")
mean_comparison.append(results.mean())
output_message = "%s| Mean=%f STD=%f" % ("model_name", results.mean(), res
ults.std())
print(output_message)
```

```
0.7804449648711944
```

```
model_name| Mean=0.800703 STD=0.000864
```

```
/Users/macbookpro/anaconda3/lib/python3.6/site-packages/sklearn/model_sele
ction/_split.py:1978: FutureWarning: The default value of cv will change f
rom 3 to 5 in version 0.22. Specify it explicitly to silence this warning.
  warnings.warn(CV_WARNING, FutureWarning)
```

In [19]:
```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

model = GaussianNB().fit(data_train, target_train)
pred = model.predict(data_test)

print("Naive-Bayes accuracy : ", accuracy_score(target_test, pred, normali
ze = True))

results = cross_val_score(model, data_train, target_train, scoring='accura
cy')
model_comparison.append(results)
model_names.append("NaiveBayes")
mean_comparison.append(results.mean())
output_message = "%s| Mean=%f STD=%f" % ("model_name", results.mean(), res
ults.std())
print(output_message)
```

```
Naive-Bayes accuracy :  0.772248243559719
model_name| Mean=0.788655 STD=0.007101
```

```
/Users/macbookpro/anaconda3/lib/python3.6/site-packages/sklearn/model_sele
ction/_split.py:1978: FutureWarning: The default value of cv will change f
rom 3 to 5 in version 0.22. Specify it explicitly to silence this warning.
  warnings.warn(CV_WARNING, FutureWarning)
```

In [20]:
```
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score

model = LinearSVC(random_state=0).fit(data_train, target_train)
pred = model.predict(data_test)

print("LinearSVC accuracy : ", accuracy_score(target_test, pred, normalize
 = True))

results = cross_val_score(model, data_train, target_train, scoring='accura
cy')
model_comparison.append(results)
model_names.append("Linear SVC")
mean_comparison.append(results.mean())
output_message = "%s| Mean=%f STD=%f" % ("model_name", results.mean(), res
ults.std())
print(output_message)
```

```
/Users/macbookpro/anaconda3/lib/python3.6/site-packages/sklearn/svm/base.p
y:929: ConvergenceWarning: Liblinear failed to converge, increase the numb
er of iterations.
  "the number of iterations.", ConvergenceWarning)
/Users/macbookpro/anaconda3/lib/python3.6/site-packages/sklearn/model_sele
ction/_split.py:1978: FutureWarning: The default value of cv will change f
```

```
rom 3 to 5 in version 0.22. Specify it explicitly to silence this warning.
  warnings.warn(CV_WARNING, FutureWarning)
/Users/macbookpro/anaconda3/lib/python3.6/site-packages/sklearn/svm/base.p
y:929: ConvergenceWarning: Liblinear failed to converge, increase the numb
er of iterations.
  "the number of iterations.", ConvergenceWarning)
```

```
LinearSVC accuracy :  0.7769320843091335
model_name| Mean=0.791664 STD=0.003882
```

```
/Users/macbookpro/anaconda3/lib/python3.6/site-packages/sklearn/svm/base.p
y:929: ConvergenceWarning: Liblinear failed to converge, increase the numb
er of iterations.
  "the number of iterations.", ConvergenceWarning)
/Users/macbookpro/anaconda3/lib/python3.6/site-packages/sklearn/svm/base.p
y:929: ConvergenceWarning: Liblinear failed to converge, increase the numb
er of iterations.
  "the number of iterations.", ConvergenceWarning)
```

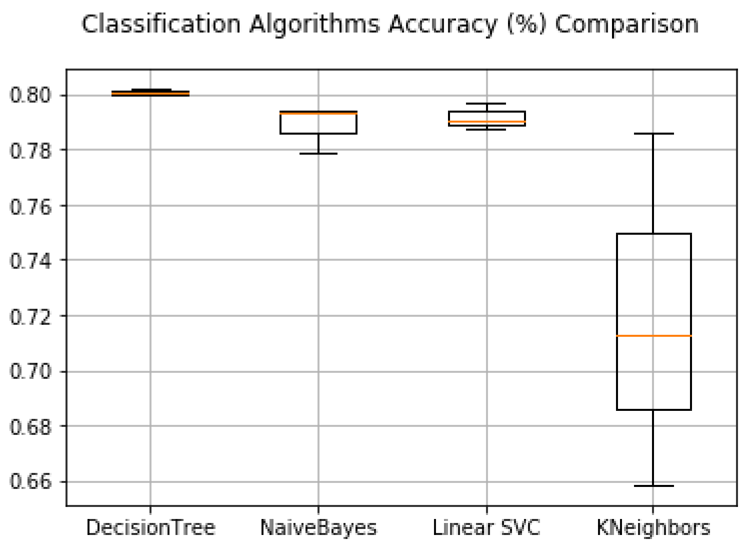
In [21]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

model = KNeighborsClassifier(n_neighbors=3).fit(data_train, target_train)
pred = model.predict(data_test)

print ("KNeighbors accuracy score : ", accuracy_score(target_test, pred))

results = cross_val_score(model, data_train, target_train, scoring='accura
cy')
model_comparison.append(results)
model_names.append("KNeighbors")
mean_comparison.append(results.mean())
output_message = "%s| Mean=%f STD=%f" % ("model_name", results.mean(), res
ults.std())
print(output_message)
```

```
KNeighbors accuracy score :  0.7558548009367682
```

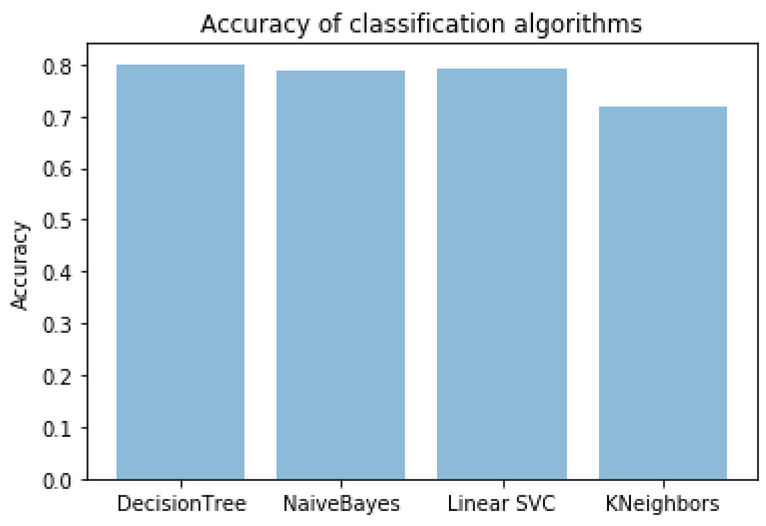
```
/Users/macbookpro/anaconda3/lib/python3.6/site-packages/sklearn/model_sele
ction/_split.py:1978: FutureWarning: The default value of cv will change f
rom 3 to 5 in version 0.22. Specify it explicitly to silence this warning.
  warnings.warn(CV_WARNING, FutureWarning)
```

```
model_name| Mean=0.719159 STD=0.052266
```

In [22]:
```
fig = plt.figure()
fig.suptitle('Classification Algorithms Accuracy (%) Comparison')

ax = fig.add_subplot(111)
plt.boxplot(model_comparison)
ax.set_xticklabels(model_names)

ax.grid()
# plt.show()

fig.savefig(outdir+'classification_algorithms_comparison.png', bbox_inches
 = 'tight', dpi=200)
```



In [23]:
```
y_pos = np.arange(len(mean_comparison))

plt.bar(y_pos, mean_comparison, align='center', alpha=0.5)
plt.xticks(y_pos, model_names)
plt.ylabel('Accuracy')
plt.title('Accuracy of classification algorithms')

plt.savefig(outdir+'accuracy_classification_algorithms_barchart.png', bbox
_inches = 'tight', dpi=200)
```



In [24]:
```
mean_arr = np.round(mean_comparison, decimals=4)

idx = 0
for m_name in model_names:
    print(m_name + ' accuracy: ')
    print('    '+str(mean_arr[idx]))
    idx += 1
```

```
DecisionTree accuracy:
   0.8007
NaiveBayes accuracy:
   0.7887
Linear SVC accuracy:
   0.7917
KNeighbors accuracy:
   0.7192
```

Regression

Q: Can we predict the age of diagnosis of hypertension?

# non-null values BPD035 2127 Age told had hypertension # non-null values RXDRSC1 10175 Reason for use of medication RIDAGEYR 10175 Age in years, at the time of the screening interview BPQ020 6464 Ever told you had high blood pressure SMD055 1203 Age last smoked cigarettes regularly HUQ010 10175 General health condition

In [25]:
```
df2 = data[['RXDRSC1', 'BPD035', 'RIDAGEYR', 'BPQ020', 'SMD055', 'HUQ010']
]
df2 = df2.dropna()

target_column2 = 'BPD035'
```

In [26]:
```
target_2 = df2[target_column2]
data_ML_2 = df2.loc[:, df2.columns != target_column2]
```

In [27]:
```
from sklearn.model_selection import train_test_split

#split data set into train and test sets
data_train_2, data_test_2, target_train_2, target_test_2 = train_test_spli
t(data_ML_2,target_2, test_size = 0.30, random_state = 10)
```

In [28]:
```
from sklearn.linear_model import LinearRegression
from math import sqrt
from sklearn.metrics import mean_squared_error

model = LinearRegression().fit(data_train_2, target_train_2)
pred = model.predict(data_test_2)

mse = mean_squared_error(target_test_2, pred)
print("LinearRegression")
print("Mean Squared Error: "+ str(mse))
print("Root Mean Squared Error: "+ str(sqrt(mse)))
```

```
LinearRegression
Mean Squared Error: 143.28934352822856
Root Mean Squared Error: 11.970352690218803
```

In [29]:
```
from sklearn.linear_model import Ridge, RidgeCV, Lasso, LassoCV
from sklearn.metrics import mean_squared_error
from sklearn.metrics import accuracy_score

model = LassoCV(max_iter = 10000, normalize = True).fit(data_train_2, targ
et_train_2)
pred = model.predict(data_test_2)

mse = mean_squared_error(target_test_2, pred)
print("LassoCV")
print("Mean Squared Error: "+ str(mse))
print("Root Mean Squared Error: "+ str(sqrt(mse)))
```

```
LassoCV
Mean Squared Error: 143.62685524369564
Root Mean Squared Error: 11.984442216628008
```

```
/Users/macbookpro/anaconda3/lib/python3.6/site-packages/sklearn/model_sele
ction/_split.py:1978: FutureWarning: The default value of cv will change f
rom 3 to 5 in version 0.22. Specify it explicitly to silence this warning.
  warnings.warn(CV_WARNING, FutureWarning)
```

In [30]:
```
df_reg = pd.DataFrame({'Actual': target_test_2, 'Predicted': pred})
df_reg.head(15)
```

Out[30]:

	Actual	Predicted
SEQN		
80601	71.0	56.734433
78074	51.0	43.331944
81709	80.0	61.901351
78443	50.0	49.403016
74734	56.0	46.005869
77976	65.0	57.951873
78515	80.0	63.227164
74570	53.0	53.218088
77979	66.0	55.575117
78967	36.0	38.261942
78131	36.0	60.865363
74254	60.0	55.346885
79377	45.0	56.800156
83639	50.0	62.330354
76576	74.0	63.354877

Conclusion and Implications

For such a precise study question (hypertension), it is difficult to use a National Health Survey dataset which contains a lot of missing data, inconsistent data, too many variables, etc. Much more data cleaning would have been required in order to select features adequately.

It would have been preferrable to use a dataset that is really specific to hypertension.

It seems that the National Health Surveys purpose is to reflect the population's current health status, habits, etc.

The conclusions from this project are too imprecise to mirror the current epidemiological studies conclusions.

In [ ]:
```

```