Coursework 2: Data Anonymisation and Privacy

Group 8

December 15, 2022

This 19-page document describes annotated code used to anonymise data from iInsure123.

Contents

Section 1: Import Packages

Import the required packages for data accessing and anonymisation

[1]:
```
## the following lines are to be uncommented and run if they have yet to be␣
 ↪installed
#pip install pycountry-convert ## for identifying continents
#pip install progressbar ## to visualise loop completion
#pip install pyminizip ## for file zipping
```

[2]:
```
import pandas as pd ## for dataframe manipulation
import numpy as np ## for statistical averaging, and finding max and min
import datetime ## for date calculations
import os ## for file saving
import stat ## for making file readonly
import pycountry_convert as pc ## for identifying continents
import re ## for regex manipulations
import progressbar ## to visualise loop completion
import requests ## to communicate with api
import pyminizip ## for file zipping
import shutil ## for directory zipping
import secrets ## for generating password
import string ## for string manipulation
```

Section 2: Data Import and Exploratory Analysis

Data is imported and visualised, with the column names extracted before data manipulation occurs in next section.

[3]:
```
# import csv
data = pd.read_csv("Data/customer_information.csv")
```

[4]:
```
# dataframe found to have 1000 rows, 18 columns
data.shape
```

[4]:
```
(1000, 18)
```

[5]:
```
# preview of what the last 5 rows of data look like
data.tail(5)
```

[5]:
```
    given_name   surname gender   birthdate country_of_birth current_country  \
995      Allan   Hammond      M  1964-01-26            Nepal  United Kingdom
996      Robin    Morris      M  2002-06-19          Estonia  United Kingdom
997     Stacey   Barnett      F  1956-04-26         Botswana  United Kingdom
998      Jayne  Harrison      F  1962-08-16         Guernsey  United Kingdom
999     Oliver    Holmes      M  1957-01-10           Canada  United Kingdom

       phone_number  postcode national_insurance_number  bank_account_number  \
995   +447700900869  SA92 1SJ               ZZ 648472 T             72521708
996  (07700) 900743  TS27 2FD               ZZ 851919 T             14900523
997  +447700 900776   G89 7HN                 ZZ783809T             28276780
998   (07700)900596  CT5B 5BN                 ZZ793814T             62820464
999   07700 900 536  SR56 7HG           ZZ 09 94 67 T             88029663

     cc_status  weight  height blood_group  avg_n_drinks_per_week  \
995          0    92.7    1.98          A+                    1.8
996          0    56.1    1.85          B+                    7.7
997          0    94.9    2.00          O+                    0.9
998          0    75.6    1.50          A+                    4.7
999          0    95.6    1.65          B-                    0.7

     avg_n_cigret_per_week education_level  n_countries_visited
995                  262.4       secondary                   21
996                  336.2           other                   35
997                   55.7       secondary                   35
998                  430.5        bachelor                   35
999                   34.6         masters                   47
```

Section 3: Feature Preprocessing

Removing non required information and anonymising remaining information.

3.1 Type of Data Present

From Section 2, the dataframe was found to contain the following columns, as demonstrated by the row values, that represented the following information, as demonstrated by the column headers:

Surveyee Info	Demographic Info	Financial Info	Body Info	Vices Info	Other Info
given_name	gender	national_insurance_number	cc_status	avg_n_drinks_per_week	education_level
surname	birthdate	bank_account_number	weight	avg_n_cigret_per_week	n_countries_visited
-	country_of_birth	-	height	-	-
-	current_country	-	blood_group	-	-
-	phone_number	-	-	-	-
-	postcode	-	-	-	-

3.2 Sensitive Columns to Drop

Due to its sensitive nature, we intend to drop the columns of: * given_name * surname * phone_number * national_insurance_number * bank_account_number

3.3 Non-informative Column to Drop

We intend to drop the following column as every individual in the dataset was a resident of the UK at the point of data collection due to the scope of the data collection methodology: * current_country

3.4 Columns to Keep Unmanipulated

We have decided to leave the following columns unmanipulated. blood_group is left unmanipulated as it contains confidential information that is not accessible in public datasets. cc_status is left unmanipulated as it encodes data in a binary format that would lose readability should further manipulations be conducted. Additionally, as both of these columns are not quasi-identifiers, we find no additional reason to propose anonymisation of these variables: * blood_group * cc_status

3.5 Confidential Columns to undergo Manipulation

We intend to retain the information provided in the remaining columns below while improving data privacy through the method of banding. * weight * height * avg_n_drinks_per_week * avg_n_cigret_per_week * n_countries_visited

3.5.1 Weight

- Banded into 10kg intervals

[6]:
```python
# get range of weight values
print(data['weight'].min())
print(data['weight'].max())

# bins for weight
bins = [30, 40, 50, 60, 70, 80, 90, 100]
data['banded_weight']= pd.cut(data['weight'], bins)
```

```
35.0
100.0
```

3.5.2 Height

- Banded into 10cm intervals

[7]:
```python
# get range of height values
print(data['height'].min())
print(data['height'].max())

# bins for height
bins = [1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0]
data['banded_height']= pd.cut(data['height'], bins)
```

```
1.4
2.0
```

3.5.3 Drinks Per Week

- Split into intervals of low, medium and high based on UK Health Recommendations for ease of modeling analysis
  - Drinking "in moderation" is usually taken to mean consuming seven to 14 units of alcohol a week, equivalent to six pints of average-strength beer or seven glasses of wine. The UK guidelines say that drinking no more than 14 units a week (6 drinks) on a regular basis will keep health risks to a low level.
  - Anything > 6 as not great
  - Anything > 12 as very heavy
  - Source: https://www.nhs.uk/live-well/alcohol-advice/the-risks-of-drinking-too-much/

[8]:
```python
# avg_n_drinks per week
np.max(data['avg_n_drinks_per_week']) #10
np.min(data['avg_n_drinks_per_week']) #0
np.mean(data['avg_n_drinks_per_week']) # 4.7658

# split into bins
bins = [0, 6, 12, 100]
data['banded_weekly_avg_drinks']= pd.cut(data['avg_n_drinks_per_week'], bins,␣
↪labels=['low', 'medium', 'high'])
```

3.5.4 Cigarettes Per Week

- Split into intervals of light, medium and heavy smokers based on UK Health Recommendations for ease of modeling analysis
  - As of 2019, average cigarettes smoked per uk citizen was 9.1 per day; 63.7 per week. Source: https://ash.org.uk/resources/view/smoking-statistics
  - Light smokers: Light smokers have been classified as smoking less than 1 pack/day, less than 15 cig/day, less than 10 cig/day, and smoking 1–39 cig/week. Source: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2865193/
  - Heavy smokers: Heavy smokers (those who smoke greater than or equal to 25 or more cigarettes a day); >=175 per week. Source: https://pubmed.ncbi.nlm.nih.gov/1614993/

[9]:
```
# view range
np.max(data['avg_n_cigret_per_week']) #500
np.min(data['avg_n_cigret_per_week']) #0.3
np.mean(data['avg_n_cigret_per_week']) # 243.83140000000017

# split into bins
bins = [0, 40, 175, 500]
data['banded_weekly_avg_cigret']= pd.cut(data['avg_n_cigret_per_week'], bins,␣
↪labels=['light', 'medium', 'heavy'])
```

3.5.5 Countries Visited

- Banded into 5 count intervals so that distance between groups is maintained

[10]:
```
bins = [0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55]
data['n_countries_visited_grouped'] = pd.cut(data.n_countries_visited, bins)
```

3.6 Quasi-Identifier Columns to undergo Manipulation

We intend to manipulate the following columns as they contain information that makes the individuals identifiable. The methods used include banding and using placeholder strings that map to specific values, creating separate key-value tables that map these relationships. * birthdate * country_of_birth * postcode * education_level

3.6.0 Create directory to store key tables Since placeholder strings are mapped to specific values, a separate directory is required to store these key-value tables that map these relationships between placeholder strings and specific values.

[11]:
```
!mkdir keys
```

3.6.1 Birthdate

- Present as age instead of birthdate
- For further anonymisation, present age in 20-year bands

[12]:
```
# extract birth year
data['birthYear'] = data.apply(lambda x: datetime.datetime.strptime(x.birthdate,␣
↪"%Y-%m-%d").year,axis = 1)

# calculate age this year
data['birthAge'] = data.apply(lambda x: datetime.datetime.now().year - x.
↪birthYear, axis = 1)

# get a perspective to see if all are two digit
data.birthAge.unique()

# band age data
bins = [0, 20, 40, 60, 80]
data['banded_age']= pd.cut(data['birthAge'], bins)
```

3.6.2 Country of Birth

- Scale up to cross-continent level to maintain geographic information
- Match cross-continent regions to color to ensure data privacy; thus no longer a quasi-identifier

[13]:
```
# this list is for all the countries that were not registered in the library
data.loc[data.country_of_birth == 'Korea', 'country_of_birth'] = 'Asia'
data.loc[data.country_of_birth == 'Palestinian Territory', 'country_of_birth'] =␣
↪'Asia'
data.loc[data.country_of_birth == 'Saint Barthelemy', 'country_of_birth'] =␣
↪'North America'
data.loc[data.country_of_birth == 'Saint Helena', 'country_of_birth'] = 'Africa'
data.loc[data.country_of_birth == 'Reunion', 'country_of_birth'] = 'Africa'
data.loc[data.country_of_birth == 'United States Minor Outlying Islands',␣
↪'country_of_birth'] = 'North America'
data.loc[data.country_of_birth == 'Antarctica (the territory South of 60 deg␣
↪S)', 'country_of_birth'] = 'Antarctica'
data.loc[data.country_of_birth == 'Western Sahara', 'country_of_birth'] =␣
↪'Africa'
data.loc[data.country_of_birth == 'Svalbard & Jan Mayen Islands',␣
↪'country_of_birth'] = 'Europe'
data.loc[data.country_of_birth == 'Libyan Arab Jamahiriya', 'country_of_birth']␣
↪= 'Africa'
data.loc[data.country_of_birth == 'Pitcairn Islands', 'country_of_birth'] =␣
↪'Oceania'
data.loc[data.country_of_birth == 'Slovakia (Slovak Republic)',␣
↪'country_of_birth'] = 'Europe'
data.loc[data.country_of_birth == 'Bouvet Island (Bouvetoya)',␣
↪'country_of_birth'] = 'Antarctica'
```

```
data.loc[data.country_of_birth == 'Holy See (Vatican City State)',␣
↪'country_of_birth'] = 'Europe'
data.loc[data.country_of_birth == 'Timor-Leste', 'country_of_birth'] = 'Asia'
data.loc[data.country_of_birth == 'British Indian Ocean Territory (Chagos␣
↪Archipelago)', 'country_of_birth'] = 'Asia'
data.loc[data.country_of_birth == "Cote d'Ivoire", 'country_of_birth'] = 'Africa'
data.loc[data.country_of_birth == "Netherlands Antilles", 'country_of_birth'] =␣
↪'North America'

# function to convert country to continent
list_of_continents = ['Africa', 'North America', 'South America', 'Antarctica',␣
↪'Oceania', 'Asia', 'Europe']
def country_to_continent(country_name):
    if country_name in list_of_continents:
        return country_name
    country_alpha2 = pc.country_name_to_country_alpha2(country_name)
    country_continent_code = pc.country_alpha2_to_continent_code(country_alpha2)
    country_continent_name = pc.
↪convert_continent_code_to_continent_name(country_continent_code)
    return country_continent_name

# apply function and assign to dataframe
country_name = data['country_of_birth']
continent_of_birth_list = [country_to_continent(country) for country in␣
↪country_name if country is not None]
continent_of_birth = pd.Series(continent_of_birth_list)
continent_of_df = pd.DataFrame(continent_of_birth)
continent_of_df_named = continent_of_df.rename(columns={0: 'continent_of_birth'})
data['continent_of_birth'] = continent_of_df_named.loc[:,'continent_of_birth']

# replace continents with cross-continent regions
APAC_list = ['Asia', 'Oceania', 'Antarctica']
Americas_list = ['South America', 'North America']
data.loc[data.continent_of_birth == 'South America', 'continent_of_birth'] =␣
↪'The Americas'
data.loc[data.continent_of_birth == 'North America', 'continent_of_birth'] =␣
↪'The Americas'
data.loc[data.continent_of_birth == 'Asia', 'continent_of_birth'] = 'APAC'
data.loc[data.continent_of_birth == 'Oceania', 'continent_of_birth'] = 'APAC'
data.loc[data.continent_of_birth == 'Antarctica', 'continent_of_birth'] = 'APAC'
```

[14]:
```
# examine counts within each cross-continent region
data['continent_of_birth'].value_counts()
```

[14]:
```
APAC          334
Europe        228
```

```
Africa           225
The Americas     213
Name: continent_of_birth, dtype: int64
```

[15]:
```
# for further privacy, replace region with colours so it is more difficult to␣
 ↪identify individuals
data.loc[data.continent_of_birth == 'APAC', 'continent_of_birth'] = 'Red'
data.loc[data.continent_of_birth == 'Europe', 'continent_of_birth'] = 'Blue'
data.loc[data.continent_of_birth == 'Africa', 'continent_of_birth'] = 'Green'
data.loc[data.continent_of_birth == 'The Americas', 'continent_of_birth'] =␣
 ↪'Yellow'
```

[16]:
```
# store key-value dataframe that maps colors to true cross-continent regions

# initialize list of lists
keytable = [['Red', 'APAC'], ['Blue', 'Europe'], ['Green', 'Africa'], ['Yellow',␣
 ↪'The Americas']]
# create the DataFrame
keytable = pd.DataFrame(keytable, columns=['Color', 'Continent'])
# print dataframe
keytable
# export the dataframe
keytable.to_csv('keys/keytable.csv', index=True)
# make csv readonly to ensure data is protected
os.chmod('keys/keytable.csv', stat.S_IREAD|stat.S_IRGRP|stat.S_IROTH)
```

3.6.3 Post Code

- Split into regions of the UK using postcodes api

[17]:
```
data
```

[17]:
```
    given_name   surname gender   birthdate          country_of_birth  \
0     Lorraine      Reed      F  1984-07-05                   Armenia
1       Edward  Williams      M  1997-06-17  Northern Mariana Islands
2       Hannah    Turner      F  1990-06-15                 Venezuela
3    Christine   Osborne      F  2000-07-29                   Eritrea
4    Francesca     Yates      F  1968-11-04                   Ecuador
..         ...       ...    ...         ...                       ...
995      Allan   Hammond      M  1964-01-26                     Nepal
996      Robin    Morris      M  2002-06-19                   Estonia
997     Stacey   Barnett      F  1956-04-26                  Botswana
998      Jayne  Harrison      F  1962-08-16                  Guernsey
999     Oliver    Holmes      M  1957-01-10                    Canada

    current_country     phone_number  postcode national_insurance_number  \
0    United Kingdom    (07700) 900876   LS5 8FN             ZZ 19 48 92 T
```

```
1    United Kingdom  (07700) 900 877   MOU 1RA             ZZ 753513 T
2    United Kingdom   +447700 900148   SO1 8HZ             ZZ 947196 T
3    United Kingdom   +447700 900112   B18 8LW          ZZ 39 69 47 T
4    United Kingdom    07700 900 413   TQ2 6BE          ZZ 30 98 91 T
..              ...              ...       ...                    ...
995  United Kingdom    +447700900869  SA92 1SJ             ZZ 648472 T
996  United Kingdom   (07700) 900743  TS27 2FD             ZZ 851919 T
997  United Kingdom   +447700 900776   G89 7HN               ZZ783809T
998  United Kingdom    (07700)900596  CT5B 5BN               ZZ793814T
999  United Kingdom    07700 900 536  SR56 7HG          ZZ 09 94 67 T

     bank_account_number  ...  n_countries_visited  banded_weight  \
0               51157818  ...                   48       (70, 80]
1              103328715  ...                   42       (60, 70]
2               69342327  ...                    9      (90, 100]
3               85159170  ...                   32       (60, 70]
4               11399166  ...                   34      (90, 100]
..                   ...  ...                  ...            ...
995             72521708  ...                   21      (90, 100]
996             14900523  ...                   35       (50, 60]
997             28276780  ...                   35      (90, 100]
998             62820464  ...                   35       (70, 80]
999             88029663  ...                   47      (90, 100]

     banded_height banded_weekly_avg_drinks  banded_weekly_avg_cigret  \
0       (1.7, 1.8]                   medium                     heavy
1       (1.7, 1.8]                      low                    medium
2       (1.8, 1.9]                   medium                    medium
3       (1.5, 1.6]                      low                     heavy
4       (1.8, 1.9]                      low                     heavy
..             ...                      ...                       ...
995     (1.9, 2.0]                      low                     heavy
996     (1.8, 1.9]                   medium                     heavy
997     (1.9, 2.0]                      low                    medium
998     (1.4, 1.5]                      low                     heavy
999     (1.6, 1.7]                      low                     light

     n_countries_visited_grouped birthYear  birthAge banded_age  \
0                       (45, 50]      1984        38   (20, 40]
1                       (40, 45]      1997        25   (20, 40]
2                        (5, 10]      1990        32   (20, 40]
3                       (30, 35]      2000        22   (20, 40]
4                       (30, 35]      1968        54   (40, 60]
..                           ...       ...       ...        ...
995                     (20, 25]      1964        58   (40, 60]
996                     (30, 35]      2002        20    (0, 20]
997                     (30, 35]      1956        66   (60, 80]
```

```
998                       (30, 35]      1962        60   (40, 60]
999                       (45, 50]      1957        65   (60, 80]

    continent_of_birth
0                  Red
1                  Red
2               Yellow
3                Green
4               Yellow
..                 ...
995                Red
996               Blue
997              Green
998               Blue
999             Yellow

[1000 rows x 27 columns]
```

[18]:

```
# generate region data from postcodes

from progressbar import ProgressBar
pbar = ProgressBar()

invalid_count = 0 # checking the number of post code entries that are not␣
↪computable
# store anonymised postcodes in list
anon_postcode = ['NA']*1000 # non-computable post code entries to be stored as NA
count = 0 # for indexing anonymised postcode list

# looping through all postcodes
for i in pbar(data['postcode']):
    # isolate outcode from postcode, since regional information is stored in␣
↪outcode
    each_postcode = i.split(' ', 1)[0]
    # find nearest existing postcode given outcode to ensure that
    # as many non-computable postcodes as possible are salvaged
    # via autocomplete function of postcodes api
    resp = requests.get('https://api.postcodes.io/postcodes/'+each_postcode+'/
↪autocomplete')

    # Catch 1: check that outcode exists
    if resp.json()['result']!= None:
        # extract full autocompleted postcode
        valid_postcode = resp.json()['result'][0]
        # get region given postcode
        resp = requests.get('https://api.postcodes.io/postcodes/
↪'+str(valid_postcode))
```

```
        region = resp.json()['result']['region']
        # store in list
        # Catch 2: check that this is not None
        if region != None:
            anon_postcode[count] = region
    else: # if outcode is not computable
        invalid_count += 1
    count += 1

print(invalid_count) # to check output
```

```
100% |########################################################################|
```

```
554
```

[19]:

```
# store region data in dataframe

data['postcode_region'] = anon_postcode
```

[20]:

```
# further banding of regions together to ensure ease of understanding

data.loc[data.postcode_region == 'North West', 'postcode_region'] = 'North'
data.loc[data.postcode_region == 'Yorkshire and The Humber', 'postcode_region']␣
 ↪= 'North'
data.loc[data.postcode_region == 'North East', 'postcode_region'] = 'North'

data.loc[data.postcode_region == 'South East', 'postcode_region'] = 'South'
data.loc[data.postcode_region == 'South West', 'postcode_region'] = 'South'

data.loc[data.postcode_region == 'East Midlands', 'postcode_region'] = 'Midlands'
data.loc[data.postcode_region == 'West Midlands', 'postcode_region'] = 'Midlands'
```

[21]:

```
# for further privacy, replace education level with colours so it is more␣
 ↪difficult to identify individuals
data.loc[data.postcode_region == 'NA', 'postcode_region'] = '0'
data.loc[data.postcode_region == 'North', 'postcode_region'] = '1'
data.loc[data.postcode_region == 'London', 'postcode_region'] = '2'
data.loc[data.postcode_region == 'Midlands', 'postcode_region'] = '3'
data.loc[data.postcode_region == 'South', 'postcode_region'] = '4'
data.loc[data.postcode_region == 'East of England', 'postcode_region'] = '5'


# store key-value dataframe that maps colors to true cross-continent regions

# initialize list of lists
```

```
postkeytable = [['0', 'NA'], ['1', 'North'], ['2', 'London'], ['3', 'Midlands'],␣
 ↪['4', 'South'], ['5', 'East of England']]
# create the DataFrame
postkeytable = pd.DataFrame(postkeytable, columns=['ID', 'UK Region'])
# print dataframe
print(postkeytable)
# export the dataframe
postkeytable.to_csv('keys/postkeytable.csv', index=True)
# make csv readonly to ensure data is protected
os.chmod('keys/postkeytable.csv', stat.S_IREAD|stat.S_IRGRP|stat.S_IROTH)
```

```
  ID        UK Region
0  0               NA
1  1            North
2  2           London
3  3         Midlands
4  4            South
5  5  East of England
```

[22]:
```
# checking of counts for each region type to ensure there is enough variety for␣
 ↪data to remain unidentifiable

data['postcode_region'].value_counts()
```

[22]:
```
0    629
1    144
2    105
3     56
4     45
5     21
Name: postcode_region, dtype: int64
```

3.6.4 Education Level

- banding into compulsory, undergraduate and postgraduate tiers since deviation is greatest between groups rather than within groups
  - compulsory representing primary and secondary
  - undergraduate representing bachelors
  - postgraduate representing masters and phD

[23]:
```
# iterate through the different education levels
    # compulsory
data.loc[data.education_level == 'primary', 'education_level'] = 'compulsory'
data.loc[data.education_level == 'secondary', 'education_level'] = 'compulsory'
    # undergraduate
data.loc[data.education_level == 'bachelor', 'education_level'] = 'undergraduate'
    # postgraduate
data.loc[data.education_level == 'masters', 'education_level'] = 'postgraduate'
```

```
data.loc[data.education_level == 'phD', 'education_level'] = 'postgraduate'

# for further privacy, replace education level with colours so it is more␣
 ↪difficult to identify individuals
data.loc[data.education_level == 'compulsory', 'education_level'] = 'Grey'
data.loc[data.education_level == 'undergraduate', 'education_level'] = 'White'
data.loc[data.education_level == 'postgraduate', 'education_level'] = 'Brown'
data.loc[data.education_level == 'other', 'education_level'] = 'Black'

# store key-value dataframe that maps colors to true cross-continent regions

# initialize list of lists
edukeytable = [['Grey', 'compulsory'], ['White', 'undergraduate'], ['Brown',␣
 ↪'postgraduate'], ['Black', 'other']]
# create the DataFrame
edukeytable = pd.DataFrame(edukeytable, columns=['Color', 'Education Level'])
# print dataframe
edukeytable
# export the dataframe
edukeytable.to_csv('keys/edukeytable.csv', index=True)
# make csv readonly to ensure data is protected
os.chmod('keys/edukeytable.csv', stat.S_IREAD|stat.S_IRGRP|stat.S_IROTH)

# check resulting counts to ensure sufficient counts within groups for data␣
 ↪privacy to be maintained
data['education_level'].value_counts()
```

[23]:
```
Grey     519
White    209
Brown    164
Black    108
Name: education_level, dtype: int64
```

Section 4: Prepare Data for Export and Calculate K-Anonymity

- Drop columns as mentioned in Section 2
- Keep only the manipulated columns, while dropping their source column
- Find k-anonymity for anonymised dataset

[24]:
```
# preview manipulated dataframe
data
```

[24]:
```
      given_name   surname gender   birthdate          country_of_birth  \
0      Lorraine      Reed      F  1984-07-05                   Armenia
1        Edward  Williams      M  1997-06-17  Northern Mariana Islands
2        Hannah    Turner      F  1990-06-15                 Venezuela
3     Christine   Osborne      F  2000-07-29                   Eritrea
4     Francesca     Yates      F  1968-11-04                   Ecuador
..          ...       ...    ...         ...                       ...
995       Allan   Hammond      M  1964-01-26                     Nepal
996       Robin    Morris      M  2002-06-19                   Estonia
997      Stacey   Barnett      F  1956-04-26                  Botswana
998       Jayne  Harrison      F  1962-08-16                  Guernsey
999      Oliver    Holmes      M  1957-01-10                    Canada

      current_country     phone_number  postcode national_insurance_number  \
0      United Kingdom   (07700) 900876   LS5 8FN             ZZ 19 48 92 T
1      United Kingdom  (07700) 900 877   MOU 1RA               ZZ 753513 T
2      United Kingdom   +447700 900148   SO1 8HZ               ZZ 947196 T
3      United Kingdom   +447700 900112   B18 8LW             ZZ 39 69 47 T
4      United Kingdom    07700 900 413   TQ2 6BE             ZZ 30 98 91 T
..                ...              ...       ...                       ...
995    United Kingdom    +447700900869  SA92 1SJ               ZZ 648472 T
996    United Kingdom   (07700) 900743  TS27 2FD               ZZ 851919 T
997    United Kingdom   +447700 900776   G89 7HN                 ZZ783809T
998    United Kingdom    (07700)900596  CT5B 5BN                 ZZ793814T
999    United Kingdom    07700 900 536  SR56 7HG             ZZ 09 94 67 T

      bank_account_number  ...  banded_weight  banded_height  \
0                51157818  ...       (70, 80]     (1.7, 1.8]
1               103328715  ...       (60, 70]     (1.7, 1.8]
2                69342327  ...      (90, 100]     (1.8, 1.9]
3                85159170  ...       (60, 70]     (1.5, 1.6]
4                11399166  ...      (90, 100]     (1.8, 1.9]
..                    ...  ...            ...            ...
995              72521708  ...      (90, 100]     (1.9, 2.0]
996              14900523  ...       (50, 60]     (1.8, 1.9]
997              28276780  ...      (90, 100]     (1.9, 2.0]
998              62820464  ...       (70, 80]     (1.4, 1.5]
999              88029663  ...      (90, 100]     (1.6, 1.7]
```

```
     banded_weekly_avg_drinks banded_weekly_avg_cigret  \
0                      medium                    heavy
1                         low                   medium
2                      medium                   medium
3                         low                    heavy
4                         low                    heavy
..                        ...                      ...
995                       low                    heavy
996                    medium                    heavy
997                       low                   medium
998                       low                    heavy
999                       low                    light

     n_countries_visited_grouped  birthYear birthAge  banded_age  \
0                       (45, 50]       1984       38    (20, 40]
1                       (40, 45]       1997       25    (20, 40]
2                        (5, 10]       1990       32    (20, 40]
3                       (30, 35]       2000       22    (20, 40]
4                       (30, 35]       1968       54    (40, 60]
..                           ...        ...      ...         ...
995                     (20, 25]       1964       58    (40, 60]
996                     (30, 35]       2002       20     (0, 20]
997                     (30, 35]       1956       66    (60, 80]
998                     (30, 35]       1962       60    (40, 60]
999                     (45, 50]       1957       65    (60, 80]

    continent_of_birth postcode_region
0                  Red               1
1                  Red               0
2               Yellow               4
3                Green               3
4               Yellow               4
..                 ...             ...
995                Red               0
996               Blue               1
997              Green               0
998               Blue               0
999             Yellow               0

[1000 rows x 28 columns]
```

[25]:

```
# remove columns that contain sensitive info or non-informative info

cleaned_data = data[['gender', 'banded_age', 'continent_of_birth',␣
 ↪'postcode_region', 'cc_status', 'banded_weight', 'banded_height',␣
 ↪'blood_group', 'banded_weekly_avg_drinks', 'banded_weekly_avg_cigret',␣
 ↪'education_level', 'n_countries_visited_grouped']]
```

```
cleaned_data
```

[25]:

```
    gender banded_age continent_of_birth postcode_region  cc_status  \
0        F   (20, 40]                Red               1          0
1        M   (20, 40]                Red               0          0
2        F   (20, 40]             Yellow               4          0
3        F   (20, 40]              Green               3          0
4        F   (40, 60]             Yellow               4          0
..     ...        ...                ...             ...        ...
995      M   (40, 60]                Red               0          0
996      M    (0, 20]               Blue               1          0
997      F   (60, 80]              Green               0          0
998      F   (40, 60]               Blue               0          0
999      M   (60, 80]             Yellow               0          0

    banded_weight banded_height blood_group banded_weekly_avg_drinks  \
0        (70, 80]    (1.7, 1.8]          B+                   medium
1        (60, 70]    (1.7, 1.8]          O-                      low
2       (90, 100]    (1.8, 1.9]          B+                   medium
3        (60, 70]    (1.5, 1.6]          O+                      low
4       (90, 100]    (1.8, 1.9]          A-                      low
..            ...           ...         ...                      ...
995     (90, 100]    (1.9, 2.0]          A+                      low
996      (50, 60]    (1.8, 1.9]          B+                   medium
997     (90, 100]    (1.9, 2.0]          O+                      low
998      (70, 80]    (1.4, 1.5]          A+                      low
999     (90, 100]    (1.6, 1.7]          B-                      low

    banded_weekly_avg_cigret education_level n_countries_visited_grouped
0                      heavy           Brown                    (45, 50]
1                     medium            Grey                    (40, 45]
2                     medium           White                     (5, 10]
3                      heavy            Grey                    (30, 35]
4                      heavy            Grey                    (30, 35]
..                       ...             ...                         ...
995                    heavy            Grey                    (20, 25]
996                    heavy           Black                    (30, 35]
997                   medium            Grey                    (30, 35]
998                    heavy           White                    (30, 35]
999                    light           Brown                    (45, 50]

[1000 rows x 12 columns]
```

[26]:

```
# calculate k anonymity
quasi_identifiers = ['gender', 'banded_age'] # can be used to identify unique␣
↪individuals
data_k_anon = cleaned_data.groupby(quasi_identifiers,
```

```
                                            as_index=False,
                                            observed=True).size()

print(data_k_anon['size'].min()) # 15
```

```
15
```

Section 5: Export Data

- Ensure data is zipped and password protected
- Store data as read-only csv stored in a password-protected zip file, with the encrypted hash password stored in a separate textfile

5.1 Zip Keys Directory

- Zip folder containing the three key-value tables referencing pseudoanonymised data

[27]:
```
shutil.make_archive('keys', 'zip', 'keys')
```

[27]:
```
'/Users/divyashridar/Documents/Imperial College London/(1) term 1/(4) clinical
data management/assignments/Coursework 2/keys.zip'
```

5.2 Zip Anonymised Data

[28]:
```
# generate password to protect .zip
import secrets
import string
alphabet = string.ascii_letters + string.digits
password = ''.join(secrets.choice(alphabet) for i in range(20))  # for a␣
↪20-character password
password

# save generated passsword
    #open text file
text_file = open("wanderlust.txt", "w")
    #write string to file
text_file.write(password)
    #close file
text_file.close()

# make csv read-only
cleaned_data.to_csv('cleaned_data.csv', index=True)
os.chmod('cleaned_data.csv', stat.S_IREAD|stat.S_IRGRP|stat.S_IROTH)

# zip csv file
pyminizip.compress('cleaned_data.csv', None, 'data.zip', password, 5)

# remove csv so only zip remains
os.remove('cleaned_data.csv')
```