# TMA 01, question 1

**Name**: [Pablo Toledo]

**PI**: [C4451553]

This TMA question will get you applying some of the ideas that you've seen in the module so far.

## Part a)

At:

https://data.gov.uk/dataset/manholes-in-devon

is a list of the manhole covers in the English county of Devon. The entire dataset is available for download in different formats, including as CSV and as JSON.

Visit the website via the above link, read the description of the data and then answer the following questions.

**i)** Look at the website descriptions and metadata.

1. Is the provenance of the data made clear?
2. Is it clear when the data was last updated?
3. What can you infer from your answers to these questions about the completeness and reliability of the data? (Remember to describe any evidence from the website you use in your answer.)
4. What are the rights granted to you to make use of the content of the website? How is this presented?

### 5 marks

1. The provenance of the manhole dataset is not made explicit on the data.gov.uk website, or in the data itself. The website states the publisher as Devon County Council, and the publication date as 14/07/2015. However, we are given no further details. For instance, was the data collected by Devon County Council, or by a third party? What methods/instruments were used in collecting this data? Is this dataset a combination of other datasets?
2. We are given a publication date of 14/07/2015, but cannot tell if the dataset has been updated since. In the absence of any further information, it would be reasonable to assume it hasn't.
3. Given we're unsure of the exact provenance of the dataset, it's hard to be sure of the completeness and reliability of the data. On quick inspection of the dataset there do seem to be a number of missing values (street names, material types etc.). Also, as we don't believe the data has been updated since publication, we have to be cautious about any conclusions we might draw from this dataset as there may be changes in trends since July 2015 which we are unable to see.
4. This dataset is published under the Open Government license, to which a link is provided on the website. This license grants permission for anyone to 'copy, publish, distribute and transmit the Information; adapt the Information; exploit the Information commercially and non-commercially for example, by combining it with other Information, or by including it in your own product or application.'. A key condition of the license is about protecting personal data. So long as this dataset is used by itself, or in combination with data which has already been published (under the FOI2000 or otherwise), it is covered under the Open Government license. However, the license license makes clear that it does not cover the publication of personal data (i.e. any data covered by the DPA98) or data which has not been previously published. Another condition that the information source (in this case Devon County Council) is always acknowledged.

## Part b)

The National Renal Data Set is intended to provide a resource for improving kidney care in patients. You can find the website at:

http://www.hscic.gov.uk/article/2117/National-Renal-Data-Set

Visit the website, and explore what the site tells you this data.

Answer the following questions, giving brief explanations of your answers, supported by evidence from the website. As well as the particular page linked above, you should also look at other pages on the same site to help answer these questions. Some questions may have more than one possible answer.

1. Who do you think is the intended audience for this website?
2. What purpose does this website serve? Is it dissemination, archival or curatorial or something else?
3. How have the site designers handled issues of data provenance, authority and trust with respect to the data and information it hosts?
4. How have the site designers encouraged exploratory engagement with the content, and is this tailored to producing author-driven or reader-driven data stories?
5. What requirements would be placed on you if you were to use data extracted from this site in your research or as the basis of a publication or news article?

### 10 marks

1. The NHS Digital website's target audience is not members of the public, but Health and Social Care professionals such as those working for the NHS, but also health care commissioners, medical or public health researchers, or those who run other health-related public services. The NRD webpage is targeted at professionals working in kidney-related fields. These might be doctors, clinicians, organ transplant services or suppliers of dialysis machines.
2. NHS Digital's main purpose is the dissemination of data relating to healthcare and public health to support decision-making, service delivery and medical research. The website provides public access to a vast array of datasets relating to all aspects of medicine, healthcare and public health (see http://content.digital.nhs.uk/searchcatalogue). Interested parties can also make a request for bespoke data-extracts from their datasets (http://content.digital.nhs.uk/dars).

The National Renal Data Set webpage does not actually make any data available for download. Its purpose is to explain the rationale behind the NRD, and document and publish a standard approach to data collection, collation and storage across different organisations who generate relevant data. This facilicates data sharing and research. As stated on the webpage, the NRD is designed to 'be used by kidney care services to assess their achievement of the quality standards and to improve kidney care for patients'. Therefore, the primary role of the NRD project (and of this webpage) is one of data curation through documenting the type and manner in which renal data is collected, what the data means, how it is stored and how it can be accessed and used.

1. The NRD webpage provides a high level of documentation regarding the NRD project and the 'business requirements' for the projct, as well as the 'NRD Human Behavioural, Organisation and Technical Guidance' document which outlines what data is collected in the NRD, what it means, and how it can be accessed and used. Additionally the webpage details the organisations with whom the NRD was developed in partnership with, and that the NRD 'extends the existing collections of the UK Renal Registry, UK Transplant and the British Association of Paediatric Nephrologists'. Thus we have a strong provenance for the data, although because the NRD consists of data from various sources, we may not be able to tell exactly which organisation supplied which data.

In terms of trust as to the accuracy of the data, the webpage provides a 'Data set Conformance Checklist' document, which adds confidence as to the accuracy and completeness of the data which is collected in the NRD, though of course we have to trust that the data was collected and input accurately in the first instance.

Other webpages on NHS Digital which make datasets available for download (rather than just documenting and curating a dataset) also appear to have a good level of documentation which allows us to establish data provenance, scope, accuracy and completeness. E.g.http://content.digital.nhs.uk/media/10048/FAQs-Practice-Level-Prescribingpdf/pdf/FAQs\_Practice\_Level\_Prescribing.pdf).

1. NHS Digital is very much a provider of data for research and decision-making, rather than a provider of data analyses and interpretations (author-driven data-stories). They do also provide some consultancy and training services, but otherwise appear to carry no 'agenda' with regards the data they provide. Based on the sample of datasets I saw available on their site, the data appears to be trustworthy, of good provenance, well documented and easily accessible in well-recognized formats (such as CSV and json). The data appears adequately transformed and cleansed, but has undergone no further processing or analysis, leaving the reader free explore the data and arrive at their own conclusinos ('reader-driven data stories').
2. The requirements placed on someone making use of data for research or publication would depend very much on the data used in question. Some data is available under the Open Data initiative, and covered under the FOI2000 act (see http://content.digital.nhs.uk/transparency). Other data sets have to be requested through the DAR service, and have more restrictive licensing and conditions of use (see 'How to access the data' and 'How does DARS protect data?', http://content.digital.nhs.uk/dars). Ultimately, once data is acquired, processed, or combined with other datasets, the onus is then on you as the 'data controller' to ensure that the resulting data does not any personal data to be revealed, thereby breaching the DPA98.

# TMA 01, question 2

**Name**: [Pablo Toledo]

**PI**: [C4451553]

This TMA question gives you the opportunity to demonstrate your mastery of the techniques in carrying out a small-scale data analysis. In the TMA, you will apply some of the ideas that you've seen in the module so far.

Specifically, this question requires you to obtain and clean two datasets, combine and reshape them, and graphically present the cleaned data. All the techniques required to answer this question can be found in Parts 2-5, and are illustrated in the associated notebooks.

In question 1, you started looking at the Devon manholes dataset. In this question, you are required to combine some of the information from this dataset with some data from the Devon County Council Property Assets dataset, which you can download from this site:

https://data.gov.uk/dataset/dcc-property-assets

For this question, you are asked to produce a graphical representation of the number of manhole covers, and the number of schools, in each parish in Devon.

To do this, you must produce a pandas dataframe, so that for each parish in Devon, the number of manhole covers is listed, and the number of schools in the parish. The final dataframe should look something like this:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Parish | Number\_of\_manhole\_covers | Number\_of\_schools |
| **0** | Sourton CP | 45 | 12 |
| **1** | Northam CP | 23 | 1 |
|  |  |  |  |

(although note that the figures 45, 12, 23 and 1 are just for illustration; they are not necessarily the correct values for the question).

You should then construct a plot showing the number of manhole covers and the number of schools for each parish, and give an explanation of what you believe the plot shows.

This question requires that you complete a number of tasks:

1. You must obtain the datasets from the two sites. This task uses the techniques described in Part 2.
2. You need to examine the datasets. You should consider questions such as how missing data is handled, whether there is any dirtiness or ambiguity in the data, and any differences in how data is represented in the two datasets. This task uses the techniques described in Part 3, section 2.
3. You will need to capture the data in a dataframe in the form described above. This task uses the techniques described in Part 3, section 3 and Part 4.
4. Finally, you should select a visualisation method for the data in the dataset, and present a plot of the data, with a description of how you think it should be interpreted. This task uses the techniques described in Part 5.

It is crucial for this question to bear in mind that at each stage, you must describe what you have done in sufficient detail that someone could replicate your work. This means that you must:

* explain what any code that you have written does, and execute it in the body of your submitted notebook,
* where you have used tools that are not accessed via python or the Notebooks (such as OpenRefine), you should include some screenshots to show what you did, and to help the marker understand your thinking,
* clearly explain any assumptions or simplifications that you have made about the data, and
* interpret your final results in the context of these assumptions and simplifications.

Some guidance on presentation:

* You must present your answer in this notebook.
* Do not put too much text or code into each notebook cell. Each cell should contain one or two paragraphs at most, or around ten lines of python.
* Ensure that in your code, you use meaningful variable names.
* You should have a specific cell whose return value is the dataframe described above.
* You should have a specific cell which plots the data in the dataframe.

### 40 marks

PART 1 - ACQUIRING THE DATA To begin with, I downloaded the manhole cover data as a CSV file from the following site: https://data.gov.uk/dataset/manholes-in-devon. Both CSV, JSON and XML formats were available, but given the simple tabular nature of the manhole cover data, there seemed little benefit in working with JSON or XML data (both of which are better suited structured document data).

We've also been provided with property assets data (again, in CSV format), previously downloaded from https://data.gov.uk/dataset/dcc-property-assets

PART 2 - On inspecting the manhole cover data, we can see that we have a simple table, with one row per manhole cover. Each row has the following attributes: ITEM\_TYPE\_NAME, ITEM\_UID, STREET\_DESCRIPTOR, ITEM\_IDENTITY\_CODE, START\_DATE, PARISH\_NAME, MATERIAL\_TYPE\_NAME, EASTING, NORTHING.

Immediately see that there are a number of missing values for MATERIAL\_TYPE\_NAME and the STREET\_DESCRIPTOR set to 'Dummy Street'. Of most concern to us however are missing values for PARISH\_NAME (represented by question marks), as we will certainly need these to calculate the number of manhole covers per parish.

We will therefore have to do some data cleansing here, and either fix these rows (by extrapolating the parish from the STREET\_DESCRIPTOR, EASTING or NORTHING) or remove them altogether. Once this is done we can remove all columns apart from ITEM\_TYPE\_NAME and PARISH\_NAME, and then create a pivot table to show the number of manhole covers per parish name.

Now looking at the buildings data, we can again see we have a simple table, this time with each row representing a building in the Devon area. Each row has the following attributes: SITE\_UPRN, SITE\_NAME, PRIMARY\_ADDRESS, STREET, LOCALITY, TOWN, COUNTY, POSTCODE, GIA, EASTING, NORTHING, PARISH, DISTRICT, SUBCLIENT, SITESTATUS, SITEFUNCTIONDETAIL.

Of these columns, we are only interested in PARISH, SUBLCIENT (which holds the class of building e.g. 'Education schools', 'youth services') and SITEFUNCTIONDETAIL (which holds the type of building e.g. 'pre-school', 'primary school', 'secondary school' etc.).

On looking briefly at the buildings data, it is apparent that there are entries for a number of different types of schools, such as pre-school, primary school, secondary school, special schools etc. This throws up an interesting question regarding the task. Should we include all types of school in the final count of schools per parish, or only mainstream primary and secondary schools? I decided that we should only include schools that form mandatory education, i.e. primary school, secondary school, as this is what is most commonly understood by the term 'school'.

This data appears complete from a cursory glance, but for some reason every other row is blank. These will have to be removed before we can use the data to calculate the school count per parish.

In order to create the final dataframe, with the count of manhole covers and schools for each parish, we will have to join the above 2 tables using the parish name. In order to do this we must first ensure the data is harmonised, so that parish names (and boundaries) are consistent across both datasets, otherwise the resulting table will be incorrect.

PART 2 - CLEANSING THE DATA USING OPEN REFINE To begin with, I'll use Open Refine to clean the manhole cover data, and create a new CSV file (called manhole\_cleaned.csv) which can be imported into a Panda Dataframe.

I'll first import the downloaded CSV file, using the default CSV encodings. I uncheck the 'store blank rows' option, as we have no need to do so. We have 77964 rows, and the preview of the first 10 looks reasonable, giving me confidence the import was successful.

Working left to right, the first thing I do is check that each row is of type MANHOLE. Using the text facet option from the ITEM\_TYPE\_NAME, we can see that this is indeed the case.

Next we'll deal with rows where the PARISH\_NAME column is populated with a '?'. Using the text facet feature we can see there are 324 of those. These rows have to be either fixed, or removed. Initially, I thought that each row with a missing parish could by fixed by either by using the STREET\_DESCRIPTOR, or a combination of EASTING and NORTHING to look up the location on a map and then work out the parish. However, I wondered if there might be other reasons as to why the parish wasn't populated. Perhaps some manhole covers didn't fall within a specific parish? Some quick research into parishes indeed reveal that some areas are 'unparished', which might explain some of these rows (see https://en.wikipedia.org/wiki/List\_of\_civil\_parishes\_in\_Devon). There may be other reasons for these rows - missing data for instance. But 328 rows out of 77,964 represents just 0.4% of the size of the dataset, and so the effect of removing them should be negligible on the final analysis. I therefore remove them using the 'Remove matching rows' option. Open refine now shows no rows with a '?' for a parish name.

Next, we can remove the unwanted columns and just retain ITEM\_TYPE\_NAME and PARISH\_NAME. Again, we can select these columns in Open Refine and select Edit Column > Remove this column.

Now I export the data as a CSV, saving it to a file called manhole\_cleaned.csv.

Next, I'll clean the buildings data in a similar fashion. On importing the data into Open Refine I uncheck the 'store blank rows' option, which removes the blank rows. The preview looks good, and we have 1276 rows of data.

I notice the last row has a value 'road' for its SITE\_UPRN, with a site name of 'ENTRY TO HANDLE ROAD SCHEMES LEASES and PTR's' . Clearly this row is attempting to capture a road instead of a building, so I remove it.

Next I select only those rows with a SITEFUNCTIONDETAIL of 'Primary School', 'Secondary School' and 'Secondard School - Foundation'. There are 509 matching rows. I notice something interesting. For each school more than 1 building may be listed. For example, CLYST HYDON PRIMARY SCHOOL has 3 sites: MAIN SCHOOL SITE, DETACHED PLAYING FIELD and VILLAGE HALL SITE. Clearly we don't want to count individual buildings as this will distort the figures. In order to get a count of unique schools we will therefore have to group individual school buildings together. The postcode field seems like our best bet for doing this (the above 3 buildings all share the same postcode). If the postcode represents the school main site, rather than the postcode of the individual building, then this grouping will work. If on the other hand it represents the postcode of the building, then we have have an issue if some schools which had various buildings spread out geographically. We can check this after we've applied the grouping, and apply a further grouping as required.

Next I invert the selection and remove all the non-school rows. Then I remove unwanted columns - SITE\_UPRN, PRIMARY\_ADDRESS, GIA, EASTING, NORTHING, DISTRICT, SUBCLIENT, SITESTATUS, SITEFUNCTIONDETAIL. I retain PARISH, and STREET, LOCALITY, TOWN, COUNTY, POSTCODE in order to group buildings by school. I then export the cleaned data and save it to buildings\_cleaned.csv.

PART 3 - CAPTURING THE DATA IN A DATAFRAME. First, I'll import the manhole cover data into a dataframe:

#import panads and dataframe library  
import pandas as pd  
from pandas import DataFrame  
  
#read manhole\_cleaned.csv file into dataframe  
manholes\_df = pd.read\_csv('data/Manhole\_cleaned.csv')  
  
#check the dataframe looks ok  
manholes\_df.head(5)

ITEM\_TYPE\_NAME

PARISH\_NAME

0

MANHOLE

Okehampton Hamlets CP

1

MANHOLE

Okehampton Hamlets CP

2

MANHOLE

Sourton CP

3

MANHOLE

Sourton CP

4

MANHOLE

Okehampton Hamlets CP

#next, we want a count of manholes for each parish.   
manholes\_df = manholes\_df.groupby('PARISH\_NAME').count() #gives count of manholes for each parish  
#we need to give the dataframe a new index, so that parish is just another normal column.  
manholes\_df = manholes\_df.reset\_index()  
#lets rename ITEM\_TYPE\_NAME to TOTAL\_MANHOLES as this is more meaningful. Also lets rename PARISH\_NAME to PARISH as  
#this is how its referred to in the buildings data, and we'll eventually need to join on that column.  
manholes\_df.rename(columns={'PARISH\_NAME': 'PARISH', 'ITEM\_TYPE\_NAME': 'TOTAL\_MANHOLES'}, inplace=True)  
manholes\_df.head(5)

PARISH

TOTAL\_MANHOLES

0

Abbotsham CP

120

1

Abbotskerswell CP

124

2

All Saints CP

12

3

Alverdiscott CP

43

4

Alwington CP

90

#lets check the number of rows..  
manholes\_df.shape[0]

411

#we have 411 rows, with each row representing a parish and the TOTAL\_MANHOLES field representing the total number of   
#manholes in that parish

#now lets import the buildings data  
#read buildings\_cleaned.csv file into dataframe  
buildings\_df = pd.read\_csv('data/buildings\_cleaned.csv')  
  
#check the dataframe looks ok  
buildings\_df.head(6)

SITE\_NAME

STREET

LOCALITY

TOWN

COUNTY

POSTCODE

PARISH

0

AXMINSTER COMMUNITY PRIMARY SCHOOL

STONEY LANE

NaN

AXMINSTER

DEVON

EX135BU

Axminster CP

1

BROADCLYST PRIMARY SCHOOL

SCHOOL LANE

BROADCLYST

EXETER

DEVON

EX53JG

Broad Clyst CP

2

CHERITON BISHOP COMMUNITY PRIMARY SCHOOL-MAIN ...

CHURCH LANE

CHERITON BISHOP

EXETER

DEVON

EX66HY

Cheriton Bishop CP

3

CHERITON BISHOP COMMUNITY PRIMARY SCHOOL-ASHES...

CHURCH LANE

CHERITON BISHOP

EXETER

DEVON

EX66HY

Cheriton Bishop CP

4

CLYST HYDON PRIMARY SCHOOL-MAIN SCHOOL SITE

NaN

CLYST HYDON

CULLOMPTON

DEVON

EX152ND

Clyst Hydon CP

5

CLYST HYDON PRIMARY SCHOOL-DETACHED PLAYING FIELD

NaN

CLYST HYDON

CULLOMPTON

DEVON

EX152ND

Clyst Hydon CP

#next, we want to ignore any rows where the postcode has already been used, so that we get one row per school,   
#instead of one row per building  
buildings\_df.drop\_duplicates(subset='POSTCODE', inplace='true')  
buildings\_df.shape[0]  
#this results in 345 rows, but I realise that this de-duplication relies on the postcode format being exactly the   
#same in each case. Any additional whitespaces would prevent de-duplication. Therefore I reclean the buildings data,   
#removing any white spaces, and retry the operation. This now shows 343 rows.

343

#previewing the buildings dataframe we can now see there are no longer any duplicate postcodes:  
buildings\_df.head(6)

SITE\_NAME

STREET

LOCALITY

TOWN

COUNTY

POSTCODE

PARISH

0

AXMINSTER COMMUNITY PRIMARY SCHOOL

STONEY LANE

NaN

AXMINSTER

DEVON

EX135BU

Axminster CP

1

BROADCLYST PRIMARY SCHOOL

SCHOOL LANE

BROADCLYST

EXETER

DEVON

EX53JG

Broad Clyst CP

2

CHERITON BISHOP COMMUNITY PRIMARY SCHOOL-MAIN ...

CHURCH LANE

CHERITON BISHOP

EXETER

DEVON

EX66HY

Cheriton Bishop CP

4

CLYST HYDON PRIMARY SCHOOL-MAIN SCHOOL SITE

NaN

CLYST HYDON

CULLOMPTON

DEVON

EX152ND

Clyst Hydon CP

7

CLYST ST MARY PRIMARY SCHOOL-MAIN SCHOOL SITE

NaN

CLYST ST MARY

EXETER

DEVON

EX51BG

Clyst St. Mary CP

9

COLYTON PRIMARY SCHOOL

THE BUTTS

WEST STREET

COLYTON

DEVON

EX246NU

Colyton CP

#next, we can discard all columns apart from PARISH and POSTCODE. We'll use these 2 columns to get a count of postcodes   
#(i.e. schools) per parish.   
schools\_df = DataFrame(buildings\_df['PARISH'])  
schools\_df['POSTCODE'] = buildings\_df['POSTCODE']  
schools\_df.head(6)

PARISH

POSTCODE

0

Axminster CP

EX135BU

1

Broad Clyst CP

EX53JG

2

Cheriton Bishop CP

EX66HY

4

Clyst Hydon CP

EX152ND

7

Clyst St. Mary CP

EX51BG

9

Colyton CP

EX246NU

#next, we count the number of postcodes (schools) per parish:  
schools\_df = DataFrame(schools\_df.pivot\_table(index=['PARISH'], aggfunc='count'))   
#Needed to use dataframe constructor to get static dataframe  
  
#we need to give the dataframe a new index, so that parish is just another normal column.  
schools\_df = schools\_df.reset\_index()  
#lets rename the POSTCODE column to TOTAL\_SCHOOLS as this is more meaningful  
schools\_df.rename(columns={'POSTCODE': 'TOTAL\_SCHOOLS'}, inplace=True)  
schools\_df.head(5)

PARISH

TOTAL\_SCHOOLS

0

Abbotsham CP

1

1

Abbotskerswell CP

1

2

All Saints CP

1

3

Ashburton CP

2

4

Ashwater CP

1

#finally, we want to join the schools\_df table with the manholes\_df table, on PARISH\_NAME  
#we want an outer join to retain both columns  
schools\_manholes\_df = pd.merge(schools\_df, manholes\_df, on=['PARISH'], how='outer')  
schools\_manholes\_df.head(5)

PARISH

TOTAL\_SCHOOLS

TOTAL\_MANHOLES

0

Abbotsham CP

1

120

1

Abbotskerswell CP

1

124

2

All Saints CP

1

12

3

Ashburton CP

2

620

4

Ashwater CP

1

14

#lets check the number of rows...  
schools\_manholes\_df.shape[0]  
#the number of rows is 415. This compares to 411 rows in the manholes\_df table. This indicates that we have at least   
#4 rows in this table with no corresponding TOTAL\_MANHOLE value

415

#lets check for Nan values, as a result of parishes with no schools, or manholes:  
schools\_manholes\_df[pd.isnull(schools\_manholes\_df).any(axis=1)].head(5)

PARISH

TOTAL\_SCHOOLS

TOTAL\_MANHOLES

15

Bickleigh (Mid Devon) CP

1

NaN

16

Bickleigh (South Hams) CP

1

NaN

107

Horwood, Lovacott and Newton Tracey CP

1

NaN

219

Woolfardisworthy (Torridge) CP

1

NaN

221

Alverdiscott CP

NaN

43

#lets populate any Nan values as 0. Note, 0 in this context will mean either there is no data available,  
#or that there is genuinely a zero value for TOTAL\_SCHOOLS or TOTAL\_MANHOLES  
schools\_manholes\_df.fillna(0, inplace=True)

#lets re-run the null check  
schools\_manholes\_df[pd.isnull(schools\_manholes\_df).any(axis=1)]

PARISH

TOTAL\_SCHOOLS

TOTAL\_MANHOLES

Part 4 - Visualising the data Now that we've cleaned and transformed the data, we want to select an appropriate visualisation. As the data is discrete as opposed to continuous, I don't believe there would be any value in using a line plot. A bar plot would seem the natural choice, as this would allow us to compare the number of schools and manhole covers between parishes. To see if there were any relationship between number of schools and number of manhole covers in a parish, we could plot both bars on the same graph.

#to start with, lets sort the table by number of schools, and then by manholes  
schools\_manholes\_df.sort\_values(['TOTAL\_SCHOOLS', 'TOTAL\_MANHOLES'], ascending=[1, 1], inplace=True)  
schools\_manholes\_df.head(5)

PARISH

TOTAL\_SCHOOLS

TOTAL\_MANHOLES

228

Ashton CP

0

1

251

Bulkworthy CP

0

1

273

Coryton CP

0

1

300

Harford CP

0

1

317

Kennerleigh CP

0

1

#While the sort has worked, it does appear that there are a large number of parished with zero schools and 1 manhole.   
#I think there is a case for removing these any row with small numbers of schools, as they will just add noise to the final plot.   
#lets see how many rows with only 1 or more schools  
schools\_manholes\_df[(schools\_manholes\_df['TOTAL\_SCHOOLS'] >=1)].shape[0]

221

#this number is still very high (221), and will be hard to interpret the plot. Lets increase the threshold to 2 schools  
schools\_manholes\_df[(schools\_manholes\_df['TOTAL\_SCHOOLS'] >=2)].shape[0]

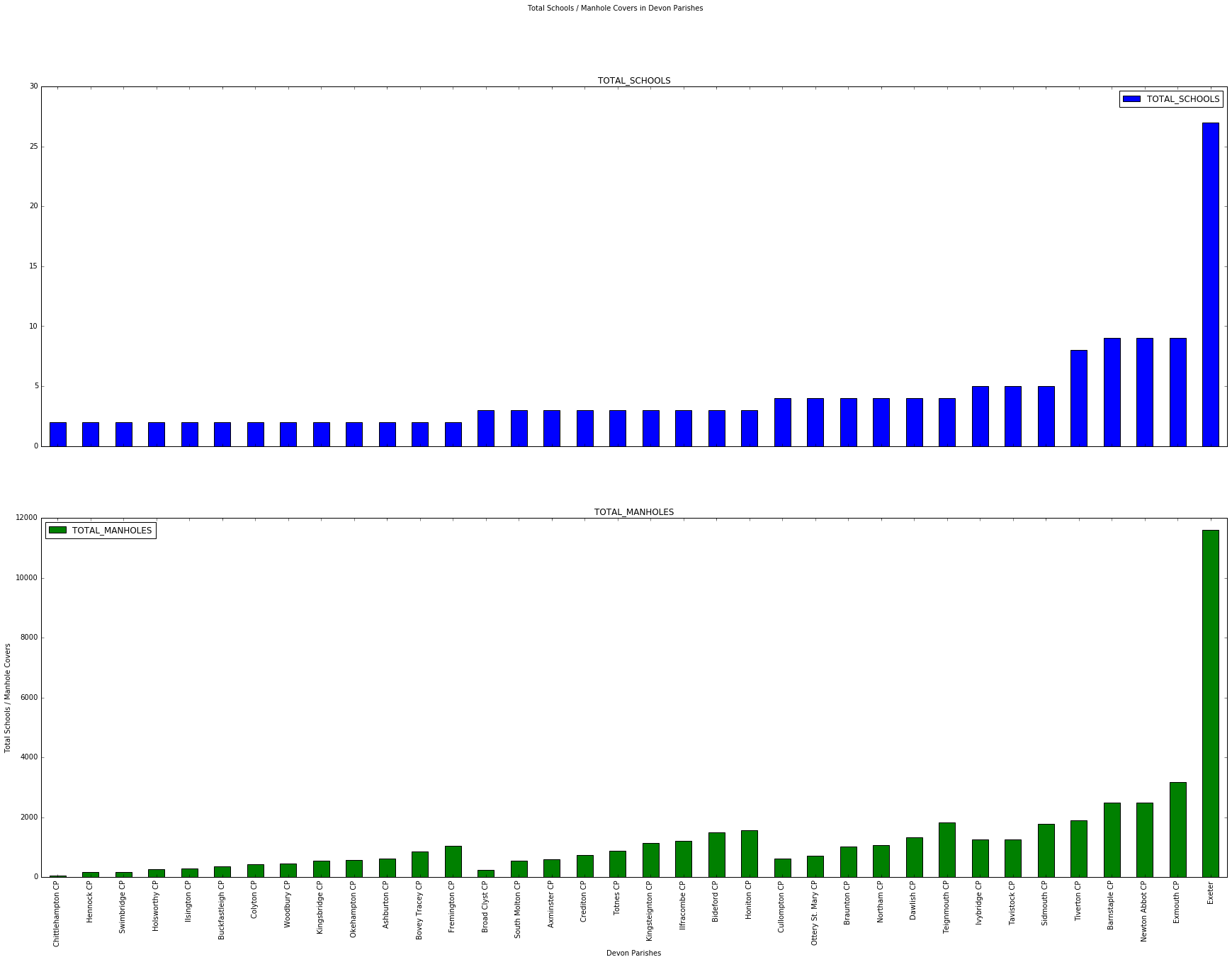
36

#36 is much more reasonable number of bars to plot, lets do that  
schools\_manholes\_df = schools\_manholes\_df[(schools\_manholes\_df['TOTAL\_SCHOOLS'] >=2)] #delete unwanted rows  
schools\_manholes\_df.shape[0]

36

schools\_manholes\_df.plot.bar(x='PARISH', title="Total Schools / Manhole Covers in Devon Parishes", subplots=True,figsize=(30,20))  
plt.xlabel('Devon Parishes')  
plt.ylabel('Total Schools / Manhole Covers')

<matplotlib.text.Text at 0xafe3c24c>



png

#This plot seems to show a possible correlation between number of schools, and number of manhole covers. Being able to   
#plot both variables as separate plots is useful. It would be nice to combine them onto the same plot, but the scale   
#of manhole covers (which run in the thousands) and schools (which run in single digits, apart from Exeter) mean we're   
#going to have to adjust the manhole numbers. Dividing by 100 should do the trick of bringing the numbers down sufficiently  
#that we can plot these variables on the same chart. Also, I am going to up the threshold of bars to only include those with  
#4 or more schools as the chart still looks too crowded, even at this size

#exclude rows with less than 3 schools  
schools\_manholes\_df = schools\_manholes\_df[(schools\_manholes\_df['TOTAL\_SCHOOLS'] >=3)]   
schools\_manholes\_df.shape[0]

23

#next, add another column to schools\_manholes\_df. We'll call it TOTAL\_MANHOLES\_HUNDREDS  
schools\_manholes\_df['TOTAL\_MANHOLES\_HUNDREDS'] = (schools\_manholes\_df['TOTAL\_MANHOLES'] / 100)

schools\_manholes\_df.head(5)

PARISH

TOTAL\_SCHOOLS

TOTAL\_MANHOLES

TOTAL\_MANHOLES\_HUNDREDS

36

Broad Clyst CP

3

236

2.36

181

South Molton CP

3

552

5.52

7

Axminster CP

3

594

5.94

62

Crediton CP

3

747

7.47

202

Totnes CP

3

879

8.79

#The school and manhole totals are now of the same magnitude, so should plot well. We can now drop the TOTAL\_MANHOLES   
#column:  
schools\_manholes\_df.drop('TOTAL\_MANHOLES', axis=1, inplace=True)

schools\_manholes\_df.head(5)

PARISH

TOTAL\_SCHOOLS

TOTAL\_MANHOLES\_HUNDREDS

36

Broad Clyst CP

3

2.36

181

South Molton CP

3

5.52

7

Axminster CP

3

5.94

62

Crediton CP

3

7.47

202

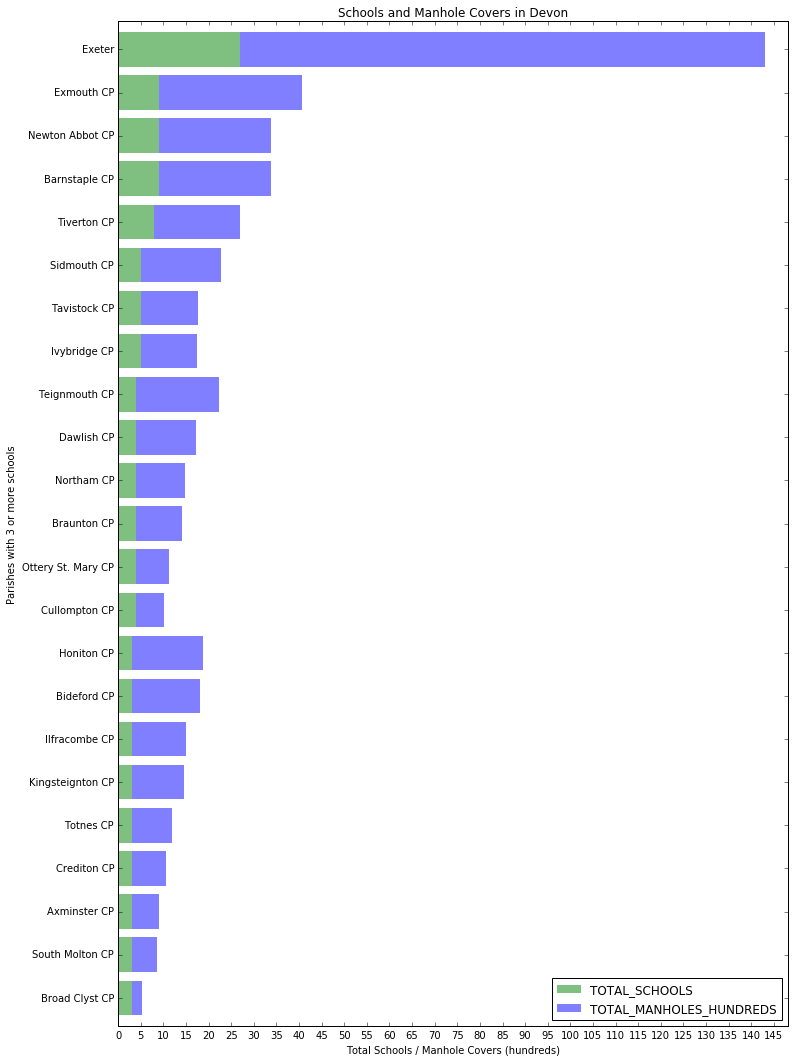
Totnes CP

3

8.79

#now lets try to plot again. A horizonal, stacked chart seems to bring out the pattern better. I've also adjusted the  
#xticks to make it easier to read the values (as most are small), and with bar and line width to fit the bars closer   
#together and bring out the pattern.  
schools\_manholes\_df.plot.barh(x='PARISH', y=['TOTAL\_SCHOOLS','TOTAL\_MANHOLES\_HUNDREDS'],  
 title="Schools and Manhole Covers in Devon",   
 figsize=(12,18),  
 xlim=(0, max(schools\_manholes\_df['TOTAL\_MANHOLES\_HUNDREDS']) +   
 max(schools\_manholes\_df['TOTAL\_SCHOOLS']) + 5),  
 stacked=True,  
 color=['green','blue'],  
 alpha=0.5,  
 width=0.8,  
 linewidth=0,  
 xticks=range(0, 150, 5)  
 )  
plt.ylabel('Parishes with 3 or more schools')  
plt.xlabel('Total Schools / Manhole Covers (hundreds)')

<matplotlib.text.Text at 0xafe6e04c>



png

This chart does seem to reveal a pattern in the data which in hindsight seems obvious: the higher the number of schools in a parish, the higher the number of manhole covers. This is not a precise linear relationship - looking at the 9 smallest parishes in this chart (with only 3 schools each), the number of manholes range from 200 to about 1200. There is a similar spread of values for total manholes for those parishes with 4 and 5 schools. However, the chart still shows a strong relationship with a step-change in the number of manholes as the number of schools increases. This could be explained by the fact that a higher number of schools in a parish indicates a higher population density, resulting in a need for a larger network of drains, which is therefore broadly reflected in a higher number of manhole covers.

# TMA 01, question 3

**Name**: [Enter your name here]

**PI**: [Enter your student id here]

In this question, you will examine a set of results of an Open University online quiz (iCMA). These are genuine results from a level 1 module, though the data has been anonymised by using cryptographic hashes to obscure personal information.

The rubric for the iCMA was similar to that for TM351. Students are allowed to take the iCMA as many times as they like, with only their highest score counting. Students have to achieve a threshold score of 40% to pass this iCMA. The iCMA remained open until the module end date.

The question has several parts, taking you through the data analysis pipeline. Most of the question parts concern with various analyses of the data. The final part of the question looks at some issues with anonymisation of the data.

Record all your activity and observations in this notebook. Generate additional notebook cells as required.

Ensure that you have made of copy of the TMA01\_Question5 Notebook and renamed it so that it has your personal identifier (PI) at the front of the Notebook filename (i.e. YourPI\_TMA01\_Question5.ipynb). You must submit this notebook as part of your TMA submission.

# Load the necessary libraries here:  
import pandas as pd  
import matplotlib.pyplot as plt  
import numpy as np  
  
# If you require additional libraries to answer any questions   
# then import them as necessary.

## Contents

* 1. [Import and cleaning](#a) (8 marks)
  2. [Number of attempts](#b) (4 marks)
  3. [Timescales](#c) (3 marks)
  4. [Time and day of quiz](#d) (8 marks)
  5. [Difficulty of questions](#e) (7 marks)
  6. [Data investigation summary](#f) (5 marks)
  7. [Anonymisation and privacy](#g) (10 marks)

# a) Import and cleaning (8 marks)

([Contents](#contents))

In this part of the question you will load and prepare the data file. You will also quickly examine the data with summaries and visualisations.

Read the csv file data/icma.csv into a DataFrame called icma\_df and display the first 3 rows.

Note that the 'Started on' and 'Completed' columns contain datetimes and should be converted on import. Also not that a single hyphen is used in the file to denote missing numerical data; hyphens should be converted to 'NA' on import.

Also note that it can be easier to see the data if you transpose the dataframe you display: append .T to your display function call.

# Enter your code here.

Check to see how the columns are typed

# Enter your code here.

# Recalculate the 'Time taken' values.  
icma\_df['Time taken'] = icma\_df['Completed'] - icma\_df['Started on']  
icma\_df.head(3).T

describe() the icma\_df DataFrame. (Again, you may find the results easier to see if you transpose .T the description.)

# Enter your code here.

All iCMA attempts are recorded in the data, whether or not they are finished. The State column shows the completion state. What are the different values for State, and how many are in each state? How many questions were answered in each state?

# Enter your code here

Use Pandas hist() to generate a histogram of number of tests for each Grade. As the test is out of 20, use 21 bins (0--20 inclusive). Add a suitable title (use plt.title()).

# Enter your code here

The above shows only completed tests. What are the marks awarded for 'In progress' tests?

Hint: Sum the question scores. You need to tell .sum() to add by row, not column. Use fillna() to include rows with no answered questions.

# Enter your code here

**Question:** What do these tables and charts tell you? Comment on:

1. the proportion of complete and incomplete quizzes, and which data is present or missing in which state.
2. the range of values for the time taken.
3. the ranges of the overall grade (Grade/20.00) and the marks for individual questions (Q. 1/1.00 to Q. 20/1.00).
4. the distribution of marks of complete and incomplete quizzes.
5. the number of questions answered in quizzes.

**Write your answer here** *(200 words)*

# For convenience, hold the selectors for the completed and incomplete attempts  
finished = icma\_df['State'] == 'Finished'  
in\_progress = ~finished  
  
# e.g. icma\_df[finished] is just the rows corresponding to finished attempts.

# For convenience, hold a list of column names that store question marks.  
question\_columns = ['Q. 1 /1.00', 'Q. 2 /1.00',  
 'Q. 3 /1.00', 'Q. 4 /1.00', 'Q. 5 /1.00', 'Q. 6 /1.00', 'Q. 7 /1.00',  
 'Q. 8 /1.00', 'Q. 9 /1.00', 'Q. 10 /1.00', 'Q. 11 /1.00', 'Q. 12 /1.00',  
 'Q. 13 /1.00', 'Q. 14 /1.00', 'Q. 15 /1.00', 'Q. 16 /1.00',  
 'Q. 17 /1.00', 'Q. 18 /1.00', 'Q. 19 /1.00', 'Q. 20 /1.00']  
  
# e.g. icma\_df[question\_columns] is just the columns for the individual questions

Aspects of the data you will explore further are:

1. how many attempts each student has made
2. how the time taken for the iCMAs affects the score
3. times and dates of starting and finishing iCMAs
4. which questions are harder than others
5. summarising these results and indicating interesting aspects
6. anonymity and privacy of the data

# b) Number of attempts (4 marks)

([Contents](#contents))

Students can make multiple attempts at a quiz. In this part of the question, you will see how many attempts different students made.

Count the number of attempts by each student.

Store the results in a new data frame, attempt\_counts\_df. This new dataframe should have one row for each student, with the columns showing the number of quiz attempts for that student.

# Enter your code here

Group attempt\_counts\_df by Completed to show the number of students who completed each number of quizzes. (For example, how many students completed no quiz attempts, how many completed 3 attempts, and so on.) Give both the numerical results and a bar chart visualisation.

# Enter your code here

Find the PIs of students that completed the quiz more than five times, and how many times they completed the quiz. Display the results in order of number of completed quizzes.

# Enter your code here

Find the PI of the student who completed the most number of quizzes. Show the datetime and overall grade of their attempts, in datetime order.

# Enter your code here

Many students started a quiz but did not complete any. What would their scores have been if they had pressed the "submit" button on the quiz? Plot the results as a histogram (with 21 bins).

First, find the PIs of students with zero completed quizzes.

*Hint:* The Personal Identifier is the index of the attempt\_counts DataFrame. Store the relevant Personal Identifiers in a variable called zero\_attempted.

# Enter your code here

The code below will select the rows of the icma\_df DataFrame where the Personal Identifier is in the zero\_attempted set you just identified. Use it to find the scores each of these quiz attempts would have been awarded had the student simply pressed 'submit'. Plot those results as a histogram (with 21 bins).

Ensure you find the total per student, not the total per question.

unsubmitted\_attempts = icma\_df[icma\_df['Personal Identifier'].isin(zero\_attempted)]  
unsubmitted\_attempts

# Enter your code here

## Analysis

What do these graphs and data extracts tell you about how students take quizzes?

**Write your answer here** *(100 words)*

# c) Timescales (3 marks)

([Contents](#contents))

Does the time taken to complete a quiz have any bearing on the score for that quiz attempt? In other words, does taking your time over a quiz lead to a higher mark, or vice versa?

Add a new column to the icma\_df DataFrame that holds the number of hours for that quiz attempt.

The 'Time taken' data are timedelta objects. The .total\_seconds() method of timedelta gives the number of seconds in that time interval. You will need to .apply() that function to every row of the icma\_df DataFrame.

describe the distribution of times. Plot a histogram of number of quiz attempts against time taken.

# Enter your code here

# Enter your code here

Many quizzes are completed quickly. How many are completed in less than two hours? Plot a histogram of number of quizzes completed by time. Include only the quizzes completed in less than two hours, and use 40 bins to show the detail.

# Enter your code here

# Enter your code here

Plot scatter plots of time taken against grade, for all attempts and for attempts completed within two hours.

# Enter your code here

# Enter your code here

## Analysis

What does these plots and summaries of the data tell you about how the time to complete a quiz affects the score of that quiz?

**Write your answer here** *(100 words)*

# d) Time and day of quiz (8 marks)

([Contents](#contents))

Does when a quiz is started or finished have any effect on its grade?

*Pandas* doesn't like making scatter plots with time as one axis. This code will convert the start time of a quiz into an integer (number of seconds), add it to the icma\_df DataFrame, then draw a scatter plot.

icma\_df['Start'] = icma\_df['Started on'].astype(np.int64)  
icma\_df.plot.scatter(x='Start', y='Grade/20.00')

Make a scatter plot of completion time against score.

# Enter your code here

# The following line of code maps dates to day of the week with Monday=0, Sunday=6  
icma\_df['DoW'] = icma\_df['Started on'].dt.weekday  
icma\_df[:3].T

How many quiz attempts were started on each day of the week? What were the average scores of those completed quiz attempts?

Show the numeric results of each of these questions, and produce a bar chart of the results.

Make a scatter plot of day of week against score.

# Show the number of quizzes started on each day of the week.  
# Enter your code here

# Plot the results above as a bar chart.  
# Enter your code here

# Show the mean score of quizzes started on each day of the week.  
# Enter your code here

# Plot the results above as a bar chart.  
# Enter your code here

# Plot the results above as a scatter chart, of grade against day of week.  
# Enter your code here

Perform the same analysis for time of day. Find the hour that each quiz was started, and give numerical and graphical representations of the number of quizzes completed each hour and their average score. Use intermediate steps as needed.

# Enter your code here

Do the complete and incomplete quiz attempts have a different distribution of starting times?

Generate two plots in one figure, with the finished quiz times above the in-progress quiz times. Give each plot a title and perhaps a different colour. (Keeping calls of

plt.tight\_layout()  
fig.subplots\_adjust(top=0.90)

to the very end of your code cell can improve the layout.)

Use reindex() and fillna() to fill in any missing groups so that both graphs have the same number of data points.

fig = plt.figure(figsize=(8, 8))  
fig.suptitle("Attempts by hour", fontsize='x-large')  
  
# Enter your code here  
  
# Keep these lines at the end  
plt.tight\_layout()  
fig.subplots\_adjust(top=0.90)

## Analysis

What does these plots and summaries of the data tell you about how the time of day quiz affects the score of that quiz?

**Write your answer here** *(150 words)*

# e) Difficulty of questions (7 marks)

([Contents](#contents))

Which questions are harder?

Note that we can judge difficulty in two ways: lower grades for particular questions, or fewer questions answered (students tend to avoid questions they perceive as difficult).

Generate graphs to show the number of scores for each question. Generate three graphs, for all attempts, completed quizzes, and in-progress quizzes.

Plot the three graphs, one above the other, in one figure. Give each subplot a title and use different colours for each plot. Again, plt.tight\_layout() and fig.subplots\_adjust() may improve the appearance.

fig = plt.figure(figsize=(8, 10)) # Make the whole figure big enough to see the individual graphs.  
  
fig.suptitle("Number of attempts", fontsize='x-large')  
  
# Enter your code here

Create a scatter plot that shows the number of questions answered (*x* axis) against the grade for that quiz attempt (*y* axis). You may find it easier to add a column to the icma\_df DataFrame to store the number of attempted questions.

# Enter your code here

Similar to the counts above, produce three charts that show the mean scores for each question. You should have one graph for all attempts, one for completed attemtps, and one for in-progress attempts. Ensure you are taking the average per question, not the score per attempt.

fig = plt.figure(figsize=(8, 10)) # Make the whole figure big enough to see the individual graphs.  
  
fig.suptitle("Mean scores", fontsize='x-large')  
  
# Enter your code here

Generate a bar chart that shows two columns for each question. One column should show the number of attempts at that question, the other column should show the mean score. You should rescale the number of question attempts to fit the range 0–1 so that the two types of data a visible on the same graph.

Use only data from completed quiz attempts.

You may need to generate intermediate DataFrames as you go.

# Enter your code here

## Analysis

What does these plots and summaries of the data tell you about which questions are harder?

**Write your answer here** *(150 words)*

# f) Data investigation summary (5 marks)

([Contents](#contents))

Look back over the analysis you have produced. Summarise the main conclusions you have drawn from this work. Highlight how different results can be combined to deduce more about the dataset. Take note of these questions:

* What conclusions can you draw about how and when students answer the iCMAs, and any effects that may have on the quiz grade?
* What are the differences between the completed and in progress quiz attempts?

If you were extending this exploration, how might you proceed?

**Write your answer here** *(400 words)*

# g) Anonymisation and privacy (10 marks)

([Contents](#contents))

The raw data for this question contained students' full names, personal email addresses, OU computer usernames, and personal identifiers. Open University students, as part of their registration agreement, give permission to The Open University for their personal data to be used internally for analysis and research.

## i) (4 marks)

What issues would there be in releasing this data for analysis by TM351 students? What laws would govern the disclosure of this data, and would unredacted data release be legal? In what way are full names, email addresses, OU computer usernames, and personal identifiers "personal data"? What other information in the dataset could be used to identify individuals?

Justify your answers.

**Write your answer here** *(300 words)*

## ii) (4 marks)

One approach to obscuring the data is to use a cryptographic hash function, such as MD5. Such a hash function is deterministic (each value always results in the same hash), one-way (the original value cannot be recovered from the hashed value), and collision-free (two different values will not generate the same hashed value). See the examples below:

import hashlib  
  
print('TM351', hashlib.md5('TM351'.encode('ascii','ignore')).hexdigest())  
print('TM352', hashlib.md5('TM352'.encode('ascii','ignore')).hexdigest())  
print('tm351', hashlib.md5('tm351'.encode('ascii','ignore')).hexdigest())

This technique could be used easily with the iCMA data by applying such a hash function to some of the data (name, email address, personal identifier, OU username) and only releasing the hashed values.

What form of anonymisation is this?

What are the advantages and disadvantages of this approach? How could data, obscured in this way, be used to deanonyimise some or all of the data?

(For information, the data in this released dataset had the personal identifiers replaced (masked) by randomly-generated keys. Nonce values for other personal information were generated deterministically from the new personal identifiers. These values were then hashed and stored.)

**Write your answer here** *(250 words)*

## iii) (2 marks)

Give two other approaches that could be used to anonymise the iCMA data so it could be published for analysis? What are their disadvantages?

**Write your answer here** *(200 words)*

### 45 marks in total