# 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.

Actually most comments in the Notebooks, as few needed for Qu1, but more for Qu2 and 3.

## 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? √ (1)
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. √ (1)
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. √ (1)
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. √ (1)

## 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)
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. √ (2)

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). √ (2)

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').√ (1) Note that the actual data referred to is NOT available as it is patient confidential.
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. √ (1) But the data given on the website itself IS available for use under OGL.

# 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.

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.

Of the columns available to us, 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' when looking at the results.

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. The parish column is also differently named in both tables (PARISH\_NAME in the manhole table, and PARISH in the bulidings data), so we'll have to rename one of the columns in order to join the tables successfully.

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  
pd.set\_option('notebook\_repr\_html', False)  
pd.set\_option('max\_rows', 20)  
  
  
#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 \  
0 AXMINSTER COMMUNITY PRIMARY SCHOOL STONEY LANE   
1 BROADCLYST PRIMARY SCHOOL SCHOOL LANE   
2 CHERITON BISHOP COMMUNITY PRIMARY SCHOOL-MAIN ... CHURCH LANE   
3 CHERITON BISHOP COMMUNITY PRIMARY SCHOOL-ASHES... CHURCH LANE   
4 CLYST HYDON PRIMARY SCHOOL-MAIN SCHOOL SITE NaN   
5 CLYST HYDON PRIMARY SCHOOL-DETACHED PLAYING FIELD NaN   
  
 LOCALITY TOWN COUNTY POSTCODE PARISH   
0 NaN AXMINSTER DEVON EX135BU Axminster CP   
1 BROADCLYST EXETER DEVON EX53JG Broad Clyst CP   
2 CHERITON BISHOP EXETER DEVON EX66HY Cheriton Bishop CP   
3 CHERITON BISHOP EXETER DEVON EX66HY Cheriton Bishop CP   
4 CLYST HYDON CULLOMPTON DEVON EX152ND Clyst Hydon CP   
5 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 resulted 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 \  
0 AXMINSTER COMMUNITY PRIMARY SCHOOL STONEY LANE   
1 BROADCLYST PRIMARY SCHOOL SCHOOL LANE   
2 CHERITON BISHOP COMMUNITY PRIMARY SCHOOL-MAIN ... CHURCH LANE   
4 CLYST HYDON PRIMARY SCHOOL-MAIN SCHOOL SITE NaN   
7 CLYST ST MARY PRIMARY SCHOOL-MAIN SCHOOL SITE NaN   
9 COLYTON PRIMARY SCHOOL THE BUTTS   
  
 LOCALITY TOWN COUNTY POSTCODE PARISH   
0 NaN AXMINSTER DEVON EX135BU Axminster CP   
1 BROADCLYST EXETER DEVON EX53JG Broad Clyst CP   
2 CHERITON BISHOP EXETER DEVON EX66HY Cheriton Bishop CP   
4 CLYST HYDON CULLOMPTON DEVON EX152ND Clyst Hydon CP   
7 CLYST ST MARY EXETER DEVON EX51BG Clyst St. Mary CP   
9 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)]

Empty DataFrame  
Columns: [PARISH, TOTAL\_SCHOOLS, TOTAL\_MANHOLES]  
Index: []

I have added comments in the Python Notebooks, as it was more appropriate (I needed to add my own lines of code to check things out first)

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]

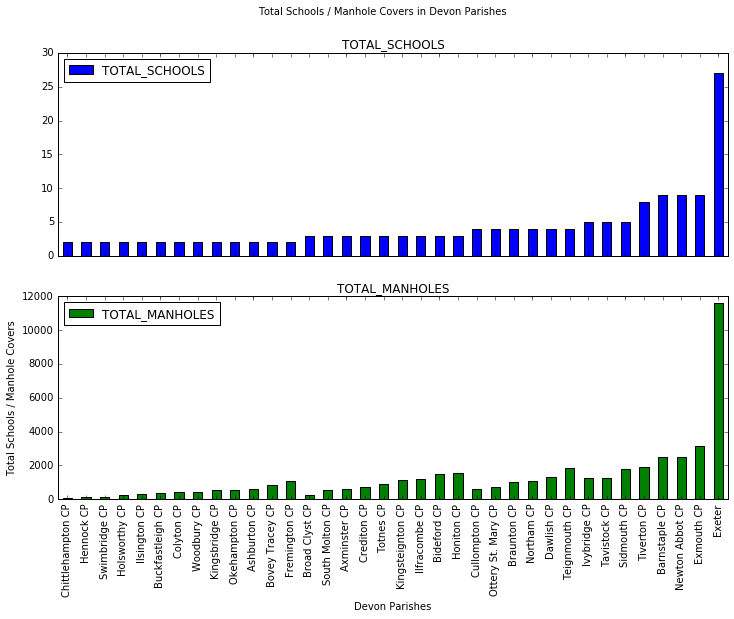
36

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

36

schools\_manholes\_df.plot.bar(x='PARISH',   
 title="Total Schools / Manhole Covers in Devon Parishes",   
 subplots=True,  
 figsize=(12,8))  
plt.xlabel('Devon Parishes')  
plt.ylabel('Total Schools / Manhole Covers')

<matplotlib.text.Text at 0xafd7c32c>



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 with4 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

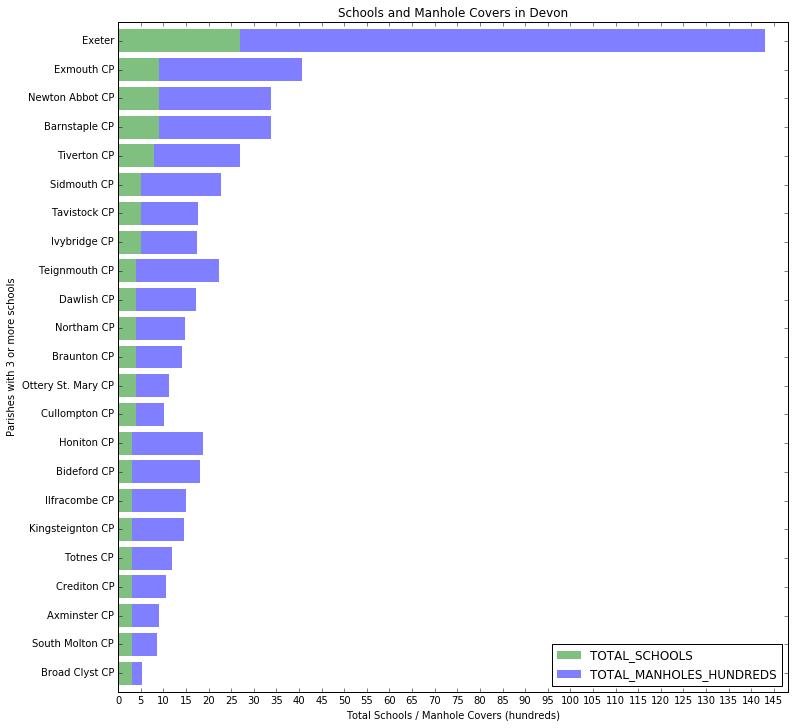
#The school and manhole totals are now of the same magnitude, so should plot well.   
#We can now drop the TOTAL\_MANHOLES column:  
schools\_manholes\_df.drop('TOTAL\_MANHOLES', axis=1, inplace=True)

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=(12,12),  
 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 0xafd9640c>



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. The chart 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.

However, 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. These variations could be caused by a number of reasons - some schools may be bigger than others, and hence serve more families in an area (which would show on the chart as parishes with unusually large number of manholes). As some geographical areas are 'unparished', it may be that some parishes make use of schools outside their parish, which again could be distorting the figures. Further, as most Devon parishes have a very small number of schools each (with only Exeter having more than 10 schools) this makes the results very sensitive to small variations in data, which might be eliminated if we had larger parishes in our dataset where such variations would be less prominent.

Even with these caveats however, there does seem to be good evidence here to support a relationship between total schools and total manholes covers in a parish.

# TMA 01, question 3

**Name**: [Pablo Toledo]

**PI**: [C4451553]

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  
pd.set\_option('notebook\_repr\_html', False)  
pd.set\_option('max\_rows', 20)  
  
  
# 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.

#read icma.csv data into dataframe  
icma\_df = pd.read\_csv('data/icma.csv', parse\_dates=['Started on','Completed'], na\_values=['-'])  
#display first 3 rows  
icma\_df.head(3).T

0 \  
Surname b7a1c60935d72dd330a48021b25ec0c4   
First name 34f9e4c9be637c99f9b2cfb3f5d1994c   
OU Computer Username b675bd4a9a46ec179a894866ad80b71d   
Personal Identifier ec0b15df629ce1fc564462e81ba6b267   
Institution NaN   
Department NaN   
Email address b3d15cf86ba4dbbafa458124c09ca816   
State Finished   
Started on 2016-01-29 06:56:00   
Completed 2016-02-08 10:31:00   
... ...   
Q. 11 /1.00 0.89   
Q. 12 /1.00 1   
Q. 13 /1.00 1   
Q. 14 /1.00 1   
Q. 15 /1.00 0.89   
Q. 16 /1.00 1   
Q. 17 /1.00 1   
Q. 18 /1.00 1   
Q. 19 /1.00 0.89   
Q. 20 /1.00 0   
  
 1 \  
Surname 8933e77b83eee991c3d3aae3c88b8176   
First name d3d2792b4f002c766a99bb7152ccf3ee   
OU Computer Username 5759a8752ba4984049c0be67c7a6270f   
Personal Identifier 7f30f4954957eb71909d7ec30ccb0e0a   
Institution NaN   
Department NaN   
Email address ff9114a689c8e5141f46cf1386b5ba9d   
State Finished   
Started on 2016-01-29 09:14:00   
Completed 2016-01-30 22:10:00   
... ...   
Q. 11 /1.00 1   
Q. 12 /1.00 1   
Q. 13 /1.00 1   
Q. 14 /1.00 1   
Q. 15 /1.00 0.89   
Q. 16 /1.00 1   
Q. 17 /1.00 1   
Q. 18 /1.00 1   
Q. 19 /1.00 1   
Q. 20 /1.00 1   
  
 2   
Surname aa8af604652b6b631f11061fe8c69809   
First name ec83707bbca209e51a65e9864ac8baeb   
OU Computer Username b8fe6559c9ba7bd27819a5460c9359c8   
Personal Identifier 4e74dca9c74ad198eeb162dac5193d9e   
Institution NaN   
Department NaN   
Email address ec01fe85557bab32113d8fbb47b0685d   
State Finished   
Started on 2016-01-29 11:16:00   
Completed 2016-01-29 11:57:00   
... ...   
Q. 11 /1.00 1   
Q. 12 /1.00 1   
Q. 13 /1.00 1   
Q. 14 /1.00 1   
Q. 15 /1.00 1   
Q. 16 /1.00 1   
Q. 17 /1.00 1   
Q. 18 /1.00 1   
Q. 19 /1.00 1   
Q. 20 /1.00 1   
  
[32 rows x 3 columns]

Check to see how the columns are typed

# Enter your code here.  
icma\_df.dtypes

Surname object  
First name object  
OU Computer Username object  
Personal Identifier object  
Institution float64  
Department float64  
Email address object  
State object  
Started on datetime64[ns]  
Completed datetime64[ns]  
 ...   
Q. 11 /1.00 float64  
Q. 12 /1.00 float64  
Q. 13 /1.00 float64  
Q. 14 /1.00 float64  
Q. 15 /1.00 float64  
Q. 16 /1.00 float64  
Q. 17 /1.00 float64  
Q. 18 /1.00 float64  
Q. 19 /1.00 float64  
Q. 20 /1.00 float64  
dtype: object

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

0 \  
Surname b7a1c60935d72dd330a48021b25ec0c4   
First name 34f9e4c9be637c99f9b2cfb3f5d1994c   
OU Computer Username b675bd4a9a46ec179a894866ad80b71d   
Personal Identifier ec0b15df629ce1fc564462e81ba6b267   
Institution NaN   
Department NaN   
Email address b3d15cf86ba4dbbafa458124c09ca816   
State Finished   
Started on 2016-01-29 06:56:00   
Completed 2016-02-08 10:31:00   
... ...   
Q. 11 /1.00 0.89   
Q. 12 /1.00 1   
Q. 13 /1.00 1   
Q. 14 /1.00 1   
Q. 15 /1.00 0.89   
Q. 16 /1.00 1   
Q. 17 /1.00 1   
Q. 18 /1.00 1   
Q. 19 /1.00 0.89   
Q. 20 /1.00 0   
  
 1 \  
Surname 8933e77b83eee991c3d3aae3c88b8176   
First name d3d2792b4f002c766a99bb7152ccf3ee   
OU Computer Username 5759a8752ba4984049c0be67c7a6270f   
Personal Identifier 7f30f4954957eb71909d7ec30ccb0e0a   
Institution NaN   
Department NaN   
Email address ff9114a689c8e5141f46cf1386b5ba9d   
State Finished   
Started on 2016-01-29 09:14:00   
Completed 2016-01-30 22:10:00   
... ...   
Q. 11 /1.00 1   
Q. 12 /1.00 1   
Q. 13 /1.00 1   
Q. 14 /1.00 1   
Q. 15 /1.00 0.89   
Q. 16 /1.00 1   
Q. 17 /1.00 1   
Q. 18 /1.00 1   
Q. 19 /1.00 1   
Q. 20 /1.00 1   
  
 2   
Surname aa8af604652b6b631f11061fe8c69809   
First name ec83707bbca209e51a65e9864ac8baeb   
OU Computer Username b8fe6559c9ba7bd27819a5460c9359c8   
Personal Identifier 4e74dca9c74ad198eeb162dac5193d9e   
Institution NaN   
Department NaN   
Email address ec01fe85557bab32113d8fbb47b0685d   
State Finished   
Started on 2016-01-29 11:16:00   
Completed 2016-01-29 11:57:00   
... ...   
Q. 11 /1.00 1   
Q. 12 /1.00 1   
Q. 13 /1.00 1   
Q. 14 /1.00 1   
Q. 15 /1.00 1   
Q. 16 /1.00 1   
Q. 17 /1.00 1   
Q. 18 /1.00 1   
Q. 19 /1.00 1   
Q. 20 /1.00 1   
  
[32 rows x 3 columns]

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

# Enter your code here.  
icma\_df.describe().T

count mean std \  
Institution 0 NaN NaN   
Department 0 NaN NaN   
Time taken 726 16 days 09:30:07.768595 29 days 07:00:03.466198   
Grade/20.00 726 17.1574 2.75742   
Q. 1 /1.00 821 0.994348 0.0589023   
Q. 2 /1.00 809 0.949184 0.149324   
Q. 3 /1.00 796 0.895415 0.211043   
Q. 4 /1.00 808 0.884814 0.24743   
Q. 5 /1.00 782 0.903619 0.12584   
Q. 6 /1.00 761 0.771498 0.247875   
... ... ... ...   
Q. 11 /1.00 755 0.926702 0.169854   
Q. 12 /1.00 740 0.892689 0.207733   
Q. 13 /1.00 757 0.985575 0.0739072   
Q. 14 /1.00 748 0.875428 0.267047   
Q. 15 /1.00 737 0.864274 0.20345   
Q. 16 /1.00 754 0.993859 0.0478942   
Q. 17 /1.00 749 0.934927 0.159529   
Q. 18 /1.00 709 0.668491 0.3768   
Q. 19 /1.00 716 0.838031 0.250563   
Q. 20 /1.00 726 0.712617 0.343511   
  
 min 25% 50% \  
Institution NaN NaN NaN   
Department NaN NaN NaN   
Time taken 0 days 00:01:00 0 days 00:35:00 0 days 01:46:00   
Grade/20.00 1 16.2875 17.765   
Q. 1 /1.00 0 1 1   
Q. 2 /1.00 0 1 1   
Q. 3 /1.00 0 1 1   
Q. 4 /1.00 0 1 1   
Q. 5 /1.00 0 0.89 0.89   
Q. 6 /1.00 0 0.67 0.78   
... ... ... ...   
Q. 11 /1.00 0 0.89 1   
Q. 12 /1.00 0 0.89 1   
Q. 13 /1.00 0.33 1 1   
Q. 14 /1.00 0 1 1   
Q. 15 /1.00 0 0.89 1   
Q. 16 /1.00 0.33 1 1   
Q. 17 /1.00 0 1 1   
Q. 18 /1.00 0 0.33 0.67   
Q. 19 /1.00 0 0.67 1   
Q. 20 /1.00 0 0.5 0.83   
  
 75% max   
Institution NaN NaN   
Department NaN NaN   
Time taken 21 days 18:49:00 154 days 00:58:00   
Grade/20.00 18.935 20   
Q. 1 /1.00 1 1   
Q. 2 /1.00 1 1   
Q. 3 /1.00 1 1   
Q. 4 /1.00 1 1   
Q. 5 /1.00 1 1   
Q. 6 /1.00 1 1   
... ... ...   
Q. 11 /1.00 1 1   
Q. 12 /1.00 1 1   
Q. 13 /1.00 1 1   
Q. 14 /1.00 1 1   
Q. 15 /1.00 1 1   
Q. 16 /1.00 1 1   
Q. 17 /1.00 1 1   
Q. 18 /1.00 1 1   
Q. 19 /1.00 1 1   
Q. 20 /1.00 1 1   
  
[24 rows x 8 columns]

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?

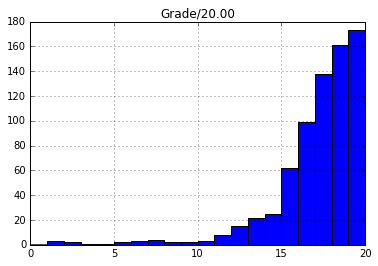
# First we group the table by state, and then apply the count function (we could also use a pivot table  
# but the results are the same).   
# If we then transpose the table we can easily see we have 2 kinds of states:'Finished' and 'In progress'  
# There are 726 'finished' ones (see Personal Identifier row), and 108 'In progress'  
# For each question we can now also see how many questions were answered (e.g. 726 of those students who   
# finished answered question 1, but only 95 who are still in progress have answered question 1)  
  
icma\_df.groupby('State').count().T

State Finished In progress  
Surname 726 108  
First name 726 108  
OU Computer Username 726 108  
Personal Identifier 726 108  
Institution 0 0  
Department 0 0  
Email address 726 108  
Started on 726 108  
Completed 726 0  
Time taken 726 0  
... ... ...  
Q. 11 /1.00 706 49  
Q. 12 /1.00 701 39  
Q. 13 /1.00 705 52  
Q. 14 /1.00 704 44  
Q. 15 /1.00 702 35  
Q. 16 /1.00 705 49  
Q. 17 /1.00 705 44  
Q. 18 /1.00 689 20  
Q. 19 /1.00 696 20  
Q. 20 /1.00 696 30  
  
[31 rows x 2 columns]

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()).

icma\_df.hist(column='Grade/20.00', bins=range(21))  
#NB plt.title, plt.xlabel and plt.ylabel don't work here. Unable to find solution online.   
#plt.xlabel('Grade/20')  
#plt.ylabel('Total Students')  
#plt.title("Total number of tests per grade/20")

array([[<matplotlib.axes.\_subplots.AxesSubplot object at 0xafeb1a6c>]], dtype=object)



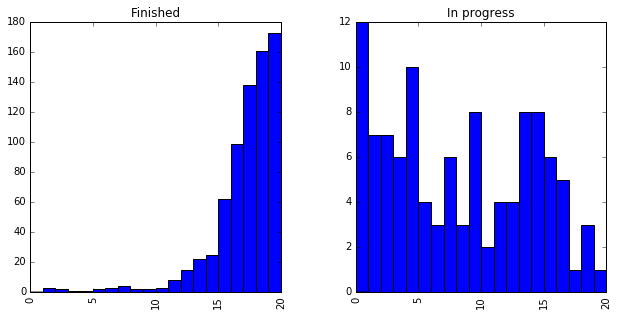
png

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.

#get list of columns, and discard all non-answer columns  
answer\_columns = list(icma\_df)  
answer\_columns = answer\_columns[12:] #discard first 12 columns   
  
# now replace all NaNs with 0s for the answer columns only  
icma\_df[answer\_columns] = icma\_df[answer\_columns].fillna(0)  
  
# Next, recalculate all the students' scores by summing answer columns for each row,   
# and update grade column with result   
icma\_df['Grade/20.00'] = icma\_df[answer\_columns].sum(axis=1)   
  
#now we can replot the histogram, and also group by state to compare..  
icma\_df.hist(column='Grade/20.00', bins=range(21), by=['State'], figsize=(10,5))

array([<matplotlib.axes.\_subplots.AxesSubplot object at 0xafe3adec>,  
 <matplotlib.axes.\_subplots.AxesSubplot object at 0xafd184cc>], dtype=object)



png

**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)* Looking at the table which shows the number of questions answered in each state, we can see there are 726 finished attempts (87%), and 108 unfinished attempts (13%).

The table of summary statistics shows us that the average time taken to complete a quiz is 16 days, 9 hours. However, the maximum time taken is an enormous 154 days, which is probably skewing the average. This is supported by a large standard deviation (29 days!), suggesting a highly spread out distribution. This suggests a wide range of study patterns for different students.

While there is a large range in scores for each question (0 to 1 in most cases), the average score for each question is very high, with only 2 questions (Q18 and Q20) having a mean score of less than 0.75. Interestingly, one would expect the later questions to have lower scores due to being more difficult, but the data doesn't show this, with some extremely high average marks towards the end of the quiz (such as question 16 with an average of 99%!).

In terms of the distribution of marks, we can see from the histogram that the majority of students scored 15 or above, with the average score for all students being 17.16. For students who didn't complete, there is an interesting peak at around 15/20, suggesting that had students clicked submit they would have performed better than they expected to.

We can also see differences in how many students attempted a question, and how many gave up. Q1 for instance, was answered by 821 students (98%), compared to Q18 which was only answered by 709 students (83%).

# 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.

# group the icma data by personal identifier - this will be unique for each student  
# this gives us a count of attempts by each student  
attempt\_counts\_df = pd.pivot\_table(icma\_df, index=['Personal Identifier'],aggfunc='count')  
attempt\_counts\_df.head(5)

Completed Department Email address \  
Personal Identifier   
003047356d0f4b451008051bc7580a61 1 0 1   
00a0827e06178d3cc66ca00614da7224 1 0 1   
00a3ecab1901a4df63f7bbe9305c9273 2 0 2   
00b0749a276fadc59c67593679cdd352 2 0 2   
00ba4c0d58a2a240bfb77191c790b8a8 1 0 1   
  
 First name Grade/20.00 Institution \  
Personal Identifier   
003047356d0f4b451008051bc7580a61 1 1 0   
00a0827e06178d3cc66ca00614da7224 1 1 0   
00a3ecab1901a4df63f7bbe9305c9273 2 2 0   
00b0749a276fadc59c67593679cdd352 2 2 0   
00ba4c0d58a2a240bfb77191c790b8a8 1 1 0   
  
 OU Computer Username Q. 1 /1.00 \  
Personal Identifier   
003047356d0f4b451008051bc7580a61 1 1   
00a0827e06178d3cc66ca00614da7224 1 1   
00a3ecab1901a4df63f7bbe9305c9273 2 2   
00b0749a276fadc59c67593679cdd352 2 2   
00ba4c0d58a2a240bfb77191c790b8a8 1 1   
  
 Q. 10 /1.00 Q. 11 /1.00 ... \  
Personal Identifier ...   
003047356d0f4b451008051bc7580a61 1 1 ...   
00a0827e06178d3cc66ca00614da7224 1 1 ...   
00a3ecab1901a4df63f7bbe9305c9273 2 2 ...   
00b0749a276fadc59c67593679cdd352 2 2 ...   
00ba4c0d58a2a240bfb77191c790b8a8 1 1 ...   
  
 Q. 4 /1.00 Q. 5 /1.00 Q. 6 /1.00 \  
Personal Identifier   
003047356d0f4b451008051bc7580a61 1 1 1   
00a0827e06178d3cc66ca00614da7224 1 1 1   
00a3ecab1901a4df63f7bbe9305c9273 2 2 2   
00b0749a276fadc59c67593679cdd352 2 2 2   
00ba4c0d58a2a240bfb77191c790b8a8 1 1 1   
  
 Q. 7 /1.00 Q. 8 /1.00 Q. 9 /1.00 \  
Personal Identifier   
003047356d0f4b451008051bc7580a61 1 1 1   
00a0827e06178d3cc66ca00614da7224 1 1 1   
00a3ecab1901a4df63f7bbe9305c9273 2 2 2   
00b0749a276fadc59c67593679cdd352 2 2 2   
00ba4c0d58a2a240bfb77191c790b8a8 1 1 1   
  
 Started on State Surname Time taken   
Personal Identifier   
003047356d0f4b451008051bc7580a61 1 1 1 1   
00a0827e06178d3cc66ca00614da7224 1 1 1 1   
00a3ecab1901a4df63f7bbe9305c9273 2 2 2 2   
00b0749a276fadc59c67593679cdd352 2 2 2 2   
00ba4c0d58a2a240bfb77191c790b8a8 1 1 1 1   
  
[5 rows x 31 columns]

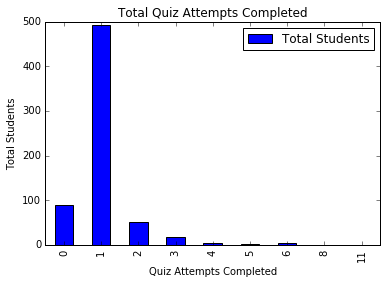
Group attempt\_counts\_df by Completed to show the number of students who completed each number of quizzes. (For example, how many students completed no quiz attempts, how many completed 3 attempts, and so on.) Give both the numerical results and a bar chart visualisation.

# pivot again, this time on completed column, count completed and discard all columns apart from one  
attempts\_summary = pd.pivot\_table(attempt\_counts\_df, index=['Completed'],values=['Email address'],aggfunc='count')  
attempts\_summary.reset\_index(inplace=True) #turn Completed into a regular column, and rename columns  
attempts\_summary.rename(columns={'Completed':'Quiz Attempts Completed', 'Email address':'Total Students'}, inplace=True)  
attempts\_summary

Quiz Attempts Completed Total Students  
0 0 90  
1 1 493  
2 2 51  
3 3 19  
4 4 4  
5 5 3  
6 6 4  
7 8 1  
8 11 1

#now plot the above table  
attempts\_summary.plot.bar(x='Quiz Attempts Completed', title="Total Quiz Attempts Completed")  
plt.xlabel('Quiz Attempts Completed')  
plt.ylabel('Total Students')

<matplotlib.text.Text at 0xafc9470c>



png

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.

# Select rows from pivot table where Completed count > 5  
# apply a sort by completed  
over\_5\_quiz\_attempts = attempt\_counts\_df[attempt\_counts\_df['Completed']>5].sort\_values(by=['Completed'])  
over\_5\_quiz\_attempts['Completed']

Personal Identifier  
5355523788afb249175bdb8b1d12380e 6  
86a729acd802df7d50607198c2185979 6  
c93061b5be7add3dde1d71cf78664b69 6  
f09d619892b5f1c63015683b6217fe54 6  
8db4ad5565daec67476734944a321ee7 8  
de9b3258c50941ba6ff048c203194c8b 11  
Name: Completed, dtype: int64

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.

#go back to original attempt\_counts\_df dataframe, and call idxmax method to return the index of the row with the max count  
#i.e. the PI of the student with the max number of attempts  
most\_quiz\_attempts\_pi = attempt\_counts\_df.idxmax()  
most\_quiz\_attempts\_pi = most\_quiz\_attempts\_pi['Completed']  
  
#now use PI of student with most attempts to select all his / her rows from original dataframe  
attempts = icma\_df[icma\_df['Personal Identifier'] == most\_quiz\_attempts\_pi]  
#sort attempts by completed date  
attempts.sort\_values(by=['Completed'])  
#select just the columns we want  
attempts[['Completed','Grade/20.00']]

Completed Grade/20.00  
34 2016-02-04 14:20:00 12.07  
88 2016-02-11 15:14:00 12.33  
97 2016-02-13 13:21:00 13.11  
103 2016-02-14 13:13:00 14.00  
105 2016-02-14 14:24:00 17.50  
109 2016-02-15 14:47:00 15.45  
111 2016-02-15 15:19:00 16.73  
112 2016-02-15 15:42:00 16.34  
121 2016-02-16 14:43:00 18.62  
123 2016-02-16 15:20:00 18.34  
125 2016-02-16 15:56:00 19.12

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.

# first, select rows with completed count of 0  
zero\_attempted = attempt\_counts\_df[attempt\_counts\_df['Completed']==0]  
# next, grab the index column (which is the student PI)  
zero\_attempted = zero\_attempted.index  
zero\_attempted

Index(['01679542bebfb044ca81f73848bc322c', '01c05a8732e2091d2aceab53f6bcc87d',  
 '065f17ebad6c6258dd55293d7efbf4b5', '0826a6d1da7c62e82c22d7ff4bc6c697',  
 '0d5b2caddd3153ba426efecc6bdd9f05', '0fac3c1704877efe52fddc6e1162e5c3',  
 '1198ef963362eb127f888e263a012daf', '1352e5ad87b7171858b955d02cf4b5f4',  
 '13a5cd6605cb7f8c09fa812bdf575184', '17a3d70bad7c6a0355a9c65e6f694e05',  
 '181b6b7816deae79627a1a4942ec1189', '184995ce7ac197b6752aac38c4072ebd',  
 '1fa06ea1b35d1b360f9a8c1395e07f6a', '228cb5a4d543bfa880d7d90bdb4d15e2',  
 '245ec7644f5159c297eb2b1b82965be1', '256dce1fd1523049b0da537a408909b3',  
 '25c8e17fe87b1b47294e65ac8755ea09', '29011958b5531db313faa15ac46018b2',  
 '2ea13f5044a777bb718bd38879d90344', '31e9d5931045f0eb995e89e2ca4e5bfb',  
 '35635ff8e5cc20633c4ffa11380a4af6', '3739ccdfa542a82f0522506d7aa115c7',  
 '3830174918e12839f30698416a4af9a3', '38d51786877591180a4f1446f09642fb',  
 '3d32c15b73037fde7c93ca7ddba88206', '3dddb096209a5a20834800e068a9ed80',  
 '3e370fa3d04b666fd232c9dafccdd25c', '40c5345dc5e700cd3538dfb2c23e2df7',  
 '4c2086a037a747c98b8449526e6b93c4', '4eb19d9e5bdcc21392dde742073e5449',  
 '4ff2ad90ec52ef36fc2572479fba4d7c', '5055e33d802c696a34a00988d683781a',  
 '56dde1a725f52e314b908305bcad0819', '575f4680c64354158219c277d6532c94',  
 '57ce0da18422bd5a77fc62b38615f033', '59050e91344327353d026b732a380c54',  
 '59eb0934528ea196a7843f6e2d476481', '5aefc74c9157da19d8de9b7e313e9ee6',  
 '5f93fcfa26b649f062b339368ca08787', '5ffcd3541f37b1c699f57376bfef49e6',  
 '6168849326223c6fc0b3a9caa6f2c67f', '64c09c422447f1397d0146815d10f0bc',  
 '6a6286a7fcfb4df85e3beda2da32dba4', '6ab577e527fcd4cf81c6f35bce32833a',  
 '6bb00a38772bd769540d67e578f710f9', '72905c9e40f141402f4f7e71436d9e5e',  
 '764a7ed54722cc17343ce5ef34cbb0b7', '766e7c5d9d5ea29cec799d8f26652c12',  
 '78d6cbc3eea347b7bee98ef62c8ebe17', '7b9a664e8d596e994d37a8abf6882b4c',  
 '7e8041aa5dbdf0d6f74532bbdfc8c5e8', '822aee71af1c6776ae47afa001ef9606',  
 '84d6a6a2350aebcaf00f924c731ec309', '856fc63844a0aab4f44ae018c53b70ae',  
 '8752daaf04700d0ae15e8b8daf9fca48', '8ab034b8586e7fff18ec07c6b57829c5',  
 '8b9cdb41623d86e49ab42ebe03705ee7', '953d2e941bad277652d5f0c41a3e33fa',  
 '97e247705aaa8595fa1a02f0e04e38f6', 'a4e8d6b3b9173c7e2b1f32edb258a2f4',  
 'a61cc8e799fe840b6a2cb6a4b0087a28', 'a655111f433d5b2644ed7b00163481ea',  
 'a6b8c7cf17638a340731dd2e55bceace', 'b7fe7e5f30c78dda9e27fee2446096bb',  
 'bc5232d92d1d4c297a146ed813a01a5a', 'c00565f0aecfe4b63d042a95d87486ca',  
 'c1c640f41bedf921604fecf52fde35a3', 'c68ec2d5d9a28cba88f1acc30935ce3e',  
 'c959702bebc661561e4d98cd3404a5a1', 'caaf871cdf80be7cd2824711986f5353',  
 'cd30162988d4200afe1cc047654d026a', 'cf3ae2d4882471630795545a86ad9944',  
 'd383cad8d552206ef886abc24db73167', 'd73aa4f1e1f4d82e5e587ddb0d866e91',  
 'd7491082fac40b7f00ca1fdd1bb1b0c3', 'd7fecdcd4b5ae855406e42bbd4638777',  
 'dd530a1b6eaf968adf37250441be2681', 'df535950bf4b94e4bafc3e3a1185a3da',  
 'e1afc97c387751b8c72030ced7953018', 'e69b9f4756e7335b4bdc6db6946107d8',  
 'e79e0a902eb53e6088506ffd97cf4aba', 'e9be09553c4df94c6e5317a320308400',  
 'eae436699194e44c31f1a82819838568', 'ed01798c6f9bab7e608f00c67f3e8f3e',  
 'f459e265f2e02e4f4442aa6ef6d00806', 'f8b632943dbee103b4a2668c26049ac7',  
 'fcb697800666346fa164073b40d04e43', 'fd3c373fd7a20be445c384147b4bc8e4',  
 'fde48e7b9fc630052b63bc106941711b', 'ffde1db48e99481474e60caf2c002aa9'],  
 dtype='object', name='Personal Identifier')

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).

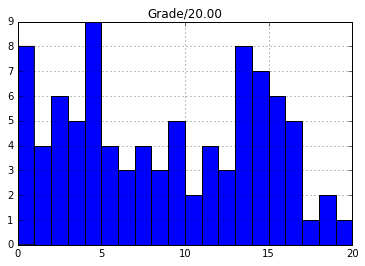
Ensure you find the total per student, not the total per question.

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

Surname First name \  
6 1b4a4ba779ce44b1d2ce658bbd199c14 7130f665bca85623df6944f349e4080f   
27 c574d948293e2fce20f1d2c3ad2e7c52 6225db62f77fbf8cd896ab214c1b183d   
57 a551255aba280834b0f4800d42ef1bda ac0592b0dcb141b624b6512c3a5a207e   
70 6627906f306950b8d77614a198c6ebd7 3949260500a1a05c4cd3a97a4b35f9f4   
90 ff99f61214c921e5a2ccc1acdadacc78 88c23c38cef5a19b543da3c08d7636bb   
92 966bf06d585f2a4c0680f3dec8da7bca 3900772a0dc83c545ff5e17f08cebf9e   
93 75ecbcc2ee1ecdf832183cdb019fa07c 10b36dc315e8669cb5d6a31ef431be67   
99 a0df726b4cab6f7c18bb05c0e697b865 10aab2ffe5e17184ab06b2c388811e11   
110 adf1a5fdec02079b63e310a4236c0479 9e2f6a17030dd80c1c4267e17e494835   
115 fea016e501612657bebc4d48d0abd55f be9f54fd90c1265489eed9c23d485487   
.. ... ...   
747 a5450848d8ec407c9fae1fe0c6ef0440 3cf7c9f8a5a7d6fe3be50d0f8409fc20   
748 4de5cabc03a123aa92436d9864c4c28a d012a1bf23470a71e503a8fcd48a4fa6   
749 40ab070f01e67b5ed143ef6dd28475ec 38b4b2adde282eacba901ebbe1b3b6f0   
758 b0d1c0b27a4d32c197229136cffbd612 0fe54ec440a95be0b43fd69b56ebc836   
766 4e3b087ab03bdc40efef842e155fb433 88327d48e420a64e68bd78d3e07595b6   
773 70461322ab07c6c0e8685804a5d87113 766650f5b60f8b9d44d3c9b2bf114ad7   
806 79f9ed63e4b08579ecd527b66b4e1e15 24ddd2a223507534363584a6cc6694d7   
828 b751fd84bdef447b9f9cba6fd84e4003 5033b1d9e4d71c945a068e856fbffc5c   
831 c0188bada1cbf321ca33be8a6103d3cb 8a18ee2d38798a30132e80cb01c73d64   
833 dee9b185fdc3772dd80669a5d3d3f6f9 75ffc9017bf01f43660b2e29a74d2d54   
  
 OU Computer Username Personal Identifier \  
6 b17eb52dd4e580b58e100385d233f43f 764a7ed54722cc17343ce5ef34cbb0b7   
27 23c673e072ed1924fba62f4a7232517d a655111f433d5b2644ed7b00163481ea   
57 06a4b1369c11daa16485aa3f730dabd8 e69b9f4756e7335b4bdc6db6946107d8   
70 b671ea725339c123fad85e9d047db3cd f8b632943dbee103b4a2668c26049ac7   
90 2772042445231475461d597c693ffb43 29011958b5531db313faa15ac46018b2   
92 c811c39c572bd96cd763bcc1f558ab00 e1afc97c387751b8c72030ced7953018   
93 3c46c0327185cbf559cab29cb549fed2 d383cad8d552206ef886abc24db73167   
99 5539a407ac9357730a9ef187aeef6818 3830174918e12839f30698416a4af9a3   
110 ba4e4c20bce7ee0d33475363a485f4d6 13a5cd6605cb7f8c09fa812bdf575184   
115 2a2786b0981c2c32c3b5398256f970c9 0826a6d1da7c62e82c22d7ff4bc6c697   
.. ... ...   
747 475cf3768019458e6216fc5da67e4494 df535950bf4b94e4bafc3e3a1185a3da   
748 36b00e6c327d2d70b89e5a0d89b5dd17 a4e8d6b3b9173c7e2b1f32edb258a2f4   
749 642f073529edeb4994c0bc64a6809ac4 c1c640f41bedf921604fecf52fde35a3   
758 84a4649688de3bbbec3ee631e64216cb fde48e7b9fc630052b63bc106941711b   
766 1d48f7cf09f1424cbc1e84c7ae4d18ea 4eb19d9e5bdcc21392dde742073e5449   
773 aff1483c102af14484a9f5c427c7a968 17a3d70bad7c6a0355a9c65e6f694e05   
806 8667828d90ae338afe450911502d82e5 72905c9e40f141402f4f7e71436d9e5e   
828 945e7ce185e28f5cb00ddeb33ae2cd04 1fa06ea1b35d1b360f9a8c1395e07f6a   
831 dfa8222e755272987349d618d9f9d806 8b9cdb41623d86e49ab42ebe03705ee7   
833 d63ecaeea083263581cfc99fbc7e6d39 b7fe7e5f30c78dda9e27fee2446096bb   
  
 Institution Department Email address State \  
6 NaN NaN 6deb878c8a8c61dbd6ab30ff5e257c34 In progress   
27 NaN NaN e0af3fee19091b96f12c4069c4f57330 In progress   
57 NaN NaN 611f6cf5227af7e6d7be14439cf4f163 In progress   
70 NaN NaN de012fde4773590eef41ebd0299a2f85 In progress   
90 NaN NaN de1de4e52f5422814697798822451221 In progress   
92 NaN NaN b80250de7827923c28dc44527740641b In progress   
93 NaN NaN 88145b9ec2ea47edae3cdc29007113a8 In progress   
99 NaN NaN 8f176e336a6ae224dbffb09138b259eb In progress   
110 NaN NaN 47511ef37fe5b31bb28a26bfe7bf8cdf In progress   
115 NaN NaN 7987deae3659b35ffc729bf60e33096f In progress   
.. ... ... ... ...   
747 NaN NaN d6d6cd9a6692281cbfe72c4ec0248776 In progress   
748 NaN NaN 6c8d86e1c7c2a7a4b732ebc369820bf6 In progress   
749 NaN NaN 119b12167752d7041517b30667f8f2a7 In progress   
758 NaN NaN c6c2b66f8a3bda7aac326c45172ab6e7 In progress   
766 NaN NaN 2d9fc8ed04e4c0038c371355db6189a6 In progress   
773 NaN NaN 506f61ba02be6365f9c0866654ffc37a In progress   
806 NaN NaN a9aa2db37f42ad3d927dfc9d8b94f668 In progress   
828 NaN NaN 92b48161555b963b5f204b40448c921a In progress   
831 NaN NaN 0f5584c748970af93e2747556bd16825 In progress   
833 NaN NaN 415227995b096d9e9740b7ff36153ade In progress   
  
 Started on Completed ... Q. 11 /1.00 Q. 12 /1.00 \  
6 2016-01-29 16:03:00 NaT ... 0.00 0.00   
27 2016-02-03 14:09:00 NaT ... 0.00 0.00   
57 2016-02-07 16:31:00 NaT ... 0.00 0.00   
70 2016-02-08 22:36:00 NaT ... 0.00 0.00   
90 2016-02-11 21:20:00 NaT ... 1.00 1.00   
92 2016-02-12 19:11:00 NaT ... 0.00 0.00   
93 2016-02-12 19:21:00 NaT ... 0.00 0.00   
99 2016-02-13 17:17:00 NaT ... 1.00 0.89   
110 2016-02-15 14:30:00 NaT ... 1.00 1.00   
115 2016-02-15 17:35:00 NaT ... 0.00 0.00   
.. ... ... ... ... ...   
747 2016-05-17 15:16:00 NaT ... 0.00 0.00   
748 2016-05-17 19:43:00 NaT ... 0.00 0.00   
749 2016-05-17 20:05:00 NaT ... 1.00 0.89   
758 2016-05-24 22:17:00 NaT ... 0.00 0.00   
766 2016-05-29 01:58:00 NaT ... 1.00 0.00   
773 2016-06-02 23:55:00 NaT ... 0.89 0.89   
806 2016-07-03 22:41:00 NaT ... 0.00 0.00   
828 2016-07-19 05:12:00 NaT ... 1.00 1.00   
831 2016-07-21 22:03:00 NaT ... 0.00 0.00   
833 2016-07-22 14:52:00 NaT ... 0.67 0.00   
  
 Q. 13 /1.00 Q. 14 /1.00 Q. 15 /1.00 Q. 16 /1.00 Q. 17 /1.00 \  
6 0.00 0.00 0.00 0.00 0.00   
27 0.00 0.00 0.00 0.00 0.00   
57 0.00 0.00 0.00 0.00 0.00   
70 0.00 0.00 0.00 0.00 0.00   
90 1.00 0.33 0.00 1.00 0.67   
92 0.00 0.00 0.00 0.00 0.00   
93 0.00 0.00 0.00 0.00 0.00   
99 1.00 1.00 0.00 1.00 0.00   
110 1.00 0.67 0.89 1.00 1.00   
115 0.00 0.00 0.00 0.00 0.00   
.. ... ... ... ... ...   
747 0.00 0.00 0.00 0.00 0.00   
748 0.00 0.00 0.00 0.00 0.00   
749 0.67 1.00 1.00 0.67 1.00   
758 0.00 0.00 0.00 0.00 0.00   
766 1.00 0.67 0.89 1.00 1.00   
773 1.00 0.00 0.00 1.00 1.00   
806 1.00 1.00 0.00 1.00 0.00   
828 1.00 1.00 0.56 1.00 1.00   
831 0.00 0.00 0.00 0.00 0.00   
833 0.00 0.00 0.00 0.00 0.00   
  
 Q. 18 /1.00 Q. 19 /1.00 Q. 20 /1.00   
6 0 0.00 0.00   
27 0 0.00 0.00   
57 0 0.00 0.00   
70 0 0.00 0.00   
90 1 0.00 0.83   
92 0 0.00 0.00   
93 0 0.00 0.00   
99 0 0.00 0.00   
110 0 0.00 0.58   
115 0 0.00 0.00   
.. ... ... ...   
747 0 0.00 0.00   
748 0 0.00 0.00   
749 1 1.00 0.50   
758 0 0.00 0.00   
766 0 0.00 0.00   
773 0 0.44 0.67   
806 0 0.00 0.00   
828 0 0.00 0.17   
831 0 0.00 0.00   
833 0 0.00 0.00   
  
[90 rows x 32 columns]

#we simply select the grade/20 column from unsubmitted\_attempts and plot this, as we've already   
#set the score of non-answered questions to 0 and recalculated the total score   
unsubmitted\_attempts.hist(column='Grade/20.00', bins=range(21))  
#NB plt.title, plt.xlabel and plt.ylabel don't work here. Unable to find solution online.   
#plt.xlabel('Grade/20')  
#plt.ylabel('Total Students')  
#plt.title("Grade/20 for students who didn't submit test")

array([[<matplotlib.axes.\_subplots.AxesSubplot object at 0xafb9922c>]], dtype=object)



png

## Analysis

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

**Write your answer here** *(100 words)* These graphs and data tell us that while most students (74%) attempt and complete the quiz just once, there are a sizable proportion (13.5%) that attempt the quiz but don't complete it, even though by doing so they might have attained a reasonable grade (15 out of 90 students who didn't click submit would have scored 15/20 or higher). Conversely 12% of students complete the quiz multiple times in order to gain a higher score. One student even completed the quiz 11 times, increasing his/her score from 12 to 19. These results probably show differing study patterns in students, with some more motivated than others to score highly on the ICMAs. These results could also indicate students who have completed a quiz, but simply forgot to press submit.

# 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.

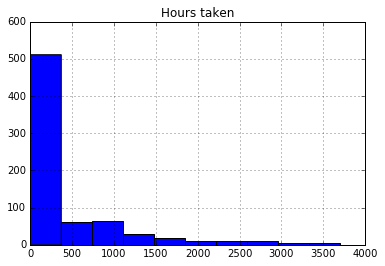
describe the distribution of times. Plot a histogram of number of quiz attempts against time taken.

# create a new 'Hours taken' column based on 'Time taken', and apply function to divide   
# time delta (i.e. a time period) by a 1 hour time delta, to find the number of hours  
icma\_df['Hours taken'] = icma\_df['Time taken'].apply(lambda x: x / np.timedelta64(1, 'h'))  
  
#now describe the distribution of times  
icma\_df['Hours taken'].describe()

count 726.000000  
mean 393.502158  
std 703.000963  
min 0.016667  
25% 0.583333  
50% 1.766667  
75% 522.816667  
max 3696.966667  
Name: Hours taken, dtype: float64

# now plot a histogram of the hours taken  
icma\_df.hist(column='Hours taken')  
#NB plt.title, plt.xlabel and plt.ylabel don't work here. Unable to find solution online.   
#plt.xlabel('Hours taken')  
#plt.ylabel('Total Students')  
#plt.title("Number of hours taken to complete each test")

array([[<matplotlib.axes.\_subplots.AxesSubplot object at 0xafb4220c>]], dtype=object)



png

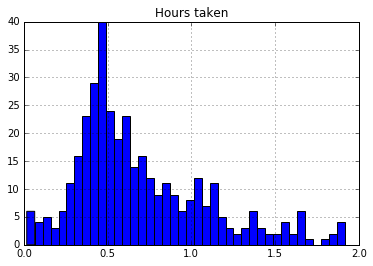
Many quizzes are completed quickly. How many are completed in less than two hours? Plot a histogram of number of quizzes completed by time. Include only the quizzes completed in less than two hours, and use 40 bins to show the detail.

#select all rows where hours taken < 2  
quizzes\_under\_2\_hours = icma\_df[icma\_df['Hours taken']<2]  
quizzes\_under\_2\_hours

Surname First name \  
2 aa8af604652b6b631f11061fe8c69809 ec83707bbca209e51a65e9864ac8baeb   
3 615bbea6dd8739d5aea38087d86175ea 19f2b4e849a31f6f0c43781e40fe14a5   
7 fe0a1584a31c05fc75c52f7fd539fc0c ac6e926ec3ef752a634271124d6328e7   
9 1209bb7c8834eac267c8c79c7cf0adb0 4360e2dcc2592270feb9da346e64a495   
11 c201bd2ac4e934165ef71882b28d4fe3 33957e81e4dc94e74fb958ff1ce34e4e   
12 8933e77b83eee991c3d3aae3c88b8176 d3d2792b4f002c766a99bb7152ccf3ee   
13 cad2ee9ce762fd64e8235c070706f2d0 b0197af0bb2c037fd5c6bbe840b06c6b   
14 fc923dd175c9c06984df402f39ed0ab0 529131b502682340edc778ed6d24c134   
16 f2e29b6dc252b243a626b0d8c44e1608 de425c58493063ed07bcc28a0b7d8e40   
17 1f0a357744ee3b1a21d568e733edccfe 41a8d5bf5bb0a30b1dca315181dc3eb9   
.. ... ...   
819 12aa76b64b26333515b84f3995754eb7 f14c6028d0cc32cc3010073b99450f76   
820 a559c2a31c758cba17c50f0c68879492 7677fe1da1e9a2bd12e783ac1f20a236   
821 a559c2a31c758cba17c50f0c68879492 7677fe1da1e9a2bd12e783ac1f20a236   
822 a559c2a31c758cba17c50f0c68879492 7677fe1da1e9a2bd12e783ac1f20a236   
823 710d376d2209cbfee34f812f5b34d36e 371a59b25a5fabf4fd493bc6300a49f3   
824 964f73c31f31547d0ae3c9effe5cc971 e17b220d8cac5aae18808417eaf21057   
825 9c56e20595cf6a86d9a42d604fea3899 9be228d1bd689abe0667787be2c62fc5   
826 bb3438dc4a8f3943d45a6a8b5b86f1d1 3cc79a93e56bc2c96e3e03596546e0ba   
827 d62b171406df4bfce8fe66f091fc6e6b 4a9f7f2e6c0c90996c883c61796abd87   
832 5a247e123a50b1b183c299d24480b802 9c8783e9cee055cb45c847dd4d6df54b   
  
 OU Computer Username Personal Identifier \  
2 b8fe6559c9ba7bd27819a5460c9359c8 4e74dca9c74ad198eeb162dac5193d9e   
3 18f13b6a9d7fccaf6b4ee35e380f3da2 94a997d5af26916f52f3a8bc4c88cc89   
7 564b23a00ad9052da59f54f1c3c56c8e 28ef47c2253e325c478698287ece1029   
9 4d59ddec2fe415e72c133d6e4ed8ae74 cd0618a9ae302e56f5e5d151378e6e58   
11 fff8aafa0b17b1c0fe1e53c0947d4b81 b8449b69715271304f341ebeb0003fdf   
12 5759a8752ba4984049c0be67c7a6270f 7f30f4954957eb71909d7ec30ccb0e0a   
13 9a0c46024ca1ab2cc18aec172bbfbc9d c93061b5be7add3dde1d71cf78664b69   
14 ddb8b3f6cd12b8667f619444c4644d90 107d2e3e45ee6f04308b9f0c5a20caf4   
16 ebad0a4f525401d570ae6af382244b74 9cbbf8c73e350590adc27b89268eacc6   
17 2016486e0f4f0dc6858b01aba16c3510 ee089b4af0720857d9260689648d42bb   
.. ... ...   
819 4ee56e7d7d7e0cdd5834e72b551ac971 f318d60a7a7fb574a30793abab44ae05   
820 3e9743a3da61bd5dd87c07dd262f750b 8db4ad5565daec67476734944a321ee7   
821 3e9743a3da61bd5dd87c07dd262f750b 8db4ad5565daec67476734944a321ee7   
822 3e9743a3da61bd5dd87c07dd262f750b 8db4ad5565daec67476734944a321ee7   
823 caca043bbc38b91aa76fdd8903092373 4d7f9cfcfb2b91a8411b60dc2a2e1dd1   
824 41e6fdb5d35e23e9cd5fdc688cc2585c bc630412db47c0aafc6151e3a5d96243   
825 4639f76ce301fa94c4a5d97623db16ec b706000f1a4449adbe528013b5bd6e9c   
826 04dabbd3a25552d8ace082f2f5388cb8 51ddad754eee7bb2afddbe6299d805bc   
827 aeb316747bbab0da1f119e65c7a84691 bfade4812577b870c2ab4f338407e7ba   
832 2cd271d80bbed5e5dcb906b8616f59cc 916f55643f042f4ff19d8bfaa816da31   
  
 Institution Department Email address State \  
2 NaN NaN ec01fe85557bab32113d8fbb47b0685d Finished   
3 NaN NaN 860dd783e86cac9751dde00e1319c9b9 Finished   
7 NaN NaN 22704ac9f42eeb956e3b72c8243c255c Finished   
9 NaN NaN 0dfbd8431e881fef15fd6785ad8c2f4d Finished   
11 NaN NaN b236c677b536dd1a592ca1da2e7aedce Finished   
12 NaN NaN ff9114a689c8e5141f46cf1386b5ba9d Finished   
13 NaN NaN 09b9d82717ca0c18f0cf58615c8d3efe Finished   
14 NaN NaN 7e938138a62102a0db2bec13e8167142 Finished   
16 NaN NaN 8ef21cebedf1f002552adf07f7e32e1f Finished   
17 NaN NaN da291a563204563cb0cb8de9f851536f Finished   
.. ... ... ... ...   
819 NaN NaN 85f96eb2237faa9c6d99c90baf779852 Finished   
820 NaN NaN 7be37101bab55f54af95d74112da556b Finished   
821 NaN NaN 7be37101bab55f54af95d74112da556b Finished   
822 NaN NaN 7be37101bab55f54af95d74112da556b Finished   
823 NaN NaN 4a0ae13c5e08a686b447420e5089aaea Finished   
824 NaN NaN 27868f59198067ce268774fcca8bd56f Finished   
825 NaN NaN 8b561b94ec529d1336e3163fbc7d2ce5 Finished   
826 NaN NaN a46e9940fb7bb98604758b40faef42e9 Finished   
827 NaN NaN 6b20344f15d6d055f1dc939ad01739fe Finished   
832 NaN NaN 415d967ab9587be2c743bcb3dc1bcedd Finished   
  
 Started on Completed ... Q. 12 /1.00 \  
2 2016-01-29 11:16:00 2016-01-29 11:57:00 ... 1.00   
3 2016-01-29 12:08:00 2016-01-29 12:51:00 ... 1.00   
7 2016-01-29 22:07:00 2016-01-29 22:51:00 ... 1.00   
9 2016-01-30 13:28:00 2016-01-30 13:44:00 ... 1.00   
11 2016-01-30 16:12:00 2016-01-30 16:40:00 ... 0.89   
12 2016-01-30 22:20:00 2016-01-30 22:36:00 ... 1.00   
13 2016-01-31 08:47:00 2016-01-31 09:27:00 ... 1.00   
14 2016-01-31 10:53:00 2016-01-31 12:01:00 ... 1.00   
16 2016-01-31 21:58:00 2016-01-31 22:25:00 ... 1.00   
17 2016-01-31 22:21:00 2016-01-31 22:46:00 ... 1.00   
.. ... ... ... ...   
819 2016-07-15 14:25:00 2016-07-15 14:52:00 ... 1.00   
820 2016-07-15 16:02:00 2016-07-15 16:21:00 ... 0.00   
821 2016-07-15 16:21:00 2016-07-15 16:41:00 ... 1.00   
822 2016-07-15 16:42:00 2016-07-15 17:07:00 ... 1.00   
823 2016-07-17 12:50:00 2016-07-17 13:15:00 ... 0.89   
824 2016-07-17 18:47:00 2016-07-17 19:21:00 ... 1.00   
825 2016-07-17 18:55:00 2016-07-17 20:00:00 ... 0.89   
826 2016-07-18 00:22:00 2016-07-18 00:49:00 ... 0.67   
827 2016-07-18 15:06:00 2016-07-18 16:01:00 ... 1.00   
832 2016-07-22 11:18:00 2016-07-22 13:09:00 ... 1.00   
  
 Q. 13 /1.00 Q. 14 /1.00 Q. 15 /1.00 Q. 16 /1.00 Q. 17 /1.00 \  
2 1 1.00 1.00 1 1   
3 1 1.00 1.00 1 1   
7 1 1.00 0.89 1 1   
9 1 1.00 1.00 1 1   
11 1 1.00 0.89 1 1   
12 1 1.00 1.00 1 1   
13 1 1.00 1.00 1 1   
14 1 0.00 0.89 1 1   
16 1 1.00 1.00 1 1   
17 1 1.00 0.11 1 1   
.. ... ... ... ... ...   
819 1 1.00 1.00 1 1   
820 0 0.00 0.00 0 0   
821 1 1.00 1.00 1 1   
822 1 1.00 1.00 1 1   
823 1 1.00 0.56 1 1   
824 1 1.00 1.00 1 1   
825 1 1.00 1.00 1 1   
826 1 0.67 0.33 1 1   
827 1 0.33 0.89 1 1   
832 1 1.00 1.00 1 1   
  
 Q. 18 /1.00 Q. 19 /1.00 Q. 20 /1.00 Hours taken   
2 1.00 1.00 1.00 0.683333   
3 1.00 1.00 1.00 0.716667   
7 1.00 1.00 0.50 0.733333   
9 1.00 1.00 1.00 0.266667   
11 0.00 0.44 0.92 0.466667   
12 1.00 1.00 1.00 0.266667   
13 1.00 1.00 0.83 0.666667   
14 0.67 0.89 0.00 1.133333   
16 0.33 1.00 1.00 0.450000   
17 1.00 1.00 0.33 0.416667   
.. ... ... ... ...   
819 1.00 0.44 1.00 0.450000   
820 0.00 0.00 0.00 0.316667   
821 0.00 1.00 1.00 0.333333   
822 1.00 1.00 1.00 0.416667   
823 0.33 1.00 0.17 0.416667   
824 1.00 1.00 0.17 0.566667   
825 0.67 0.67 0.83 1.083333   
826 0.00 0.22 0.00 0.450000   
827 0.00 1.00 0.50 0.916667   
832 1.00 1.00 0.17 1.850000   
  
[370 rows x 33 columns]

#plot histogram of hours taken with bin size of 40  
quizzes\_under\_2\_hours.hist(column='Hours taken', bins=40)  
#NB plt.title, plt.xlabel and plt.ylabel don't work here. Unable to find solution online.   
#plt.xlabel('Hours taken')  
#plt.ylabel('Total Students')  
#plt.title("Number of hours taken to complete each test (for attempts less than 2 hours long)")

array([[<matplotlib.axes.\_subplots.AxesSubplot object at 0xafb6862c>]], dtype=object)

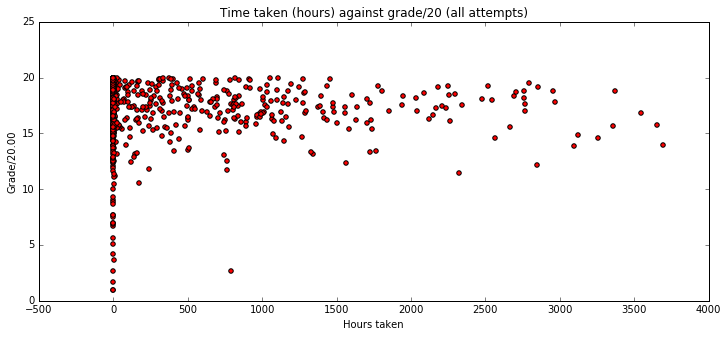


png

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

# scatter plot of time taken against grade for all attempts  
icma\_df.plot.scatter(x='Hours taken', y='Grade/20.00',   
 color='red',   
 title="Time taken (hours) against grade/20 (all attempts)",  
 figsize=(12,5))

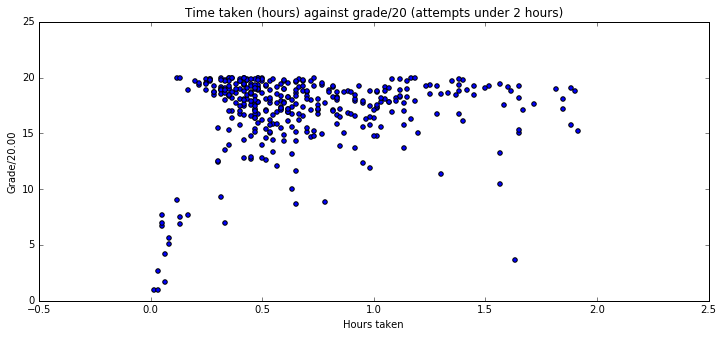
<matplotlib.axes.\_subplots.AxesSubplot at 0xafa454ac>



png

# scatter plot of time taken against grade for attempts less than 2 hours  
quizzes\_under\_2\_hours.plot.scatter(x='Hours taken', y='Grade/20.00',   
 color='blue',   
 title="Time taken (hours) against grade/20 (attempts under 2 hours)",  
 figsize=(12,5))

<matplotlib.axes.\_subplots.AxesSubplot at 0xafa1104c>



png

## 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)* From looking at scatter graphs we can clearly see that the majority of students complete their tests quickly (50% of students within 1.76 hours, 75% of students within 523 hours), and score highly (between 15 and 20). In the 2 hour scatter graph we can also see a clustering of high results around the half hour mark, with the grade falling slightly between half hour and 2 hours. In the other scatter graph we see this pattern more clearly, with the grade visibly dropping as the number of hours increases. Its also interesting to note that the grade also drops for students that completed within 15m. The later results could be explained by students leaving it too long to take the quiz, by which point they have forgotten the material. The former could conversely be explained by students rushing to complete the test and not taking the required to read and think about the questions.

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

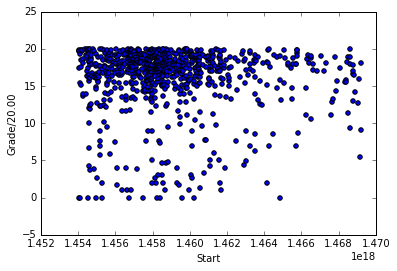
([Contents](#contents))

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')

<matplotlib.axes.\_subplots.AxesSubplot at 0xaf9f744c>

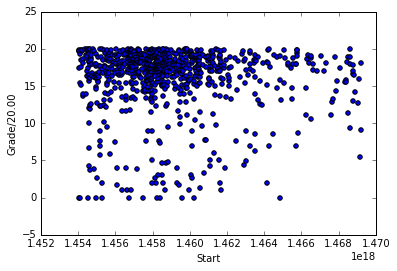


png

Make a scatter plot of completion time against score.

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

<matplotlib.axes.\_subplots.AxesSubplot at 0xaf8e3b2c>



png

# 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

0 \  
Surname b7a1c60935d72dd330a48021b25ec0c4   
First name 34f9e4c9be637c99f9b2cfb3f5d1994c   
OU Computer Username b675bd4a9a46ec179a894866ad80b71d   
Personal Identifier ec0b15df629ce1fc564462e81ba6b267   
Institution NaN   
Department NaN   
Email address b3d15cf86ba4dbbafa458124c09ca816   
State Finished   
Started on 2016-01-29 06:56:00   
Completed 2016-02-08 10:31:00   
... ...   
Q. 15 /1.00 0.89   
Q. 16 /1.00 1   
Q. 17 /1.00 1   
Q. 18 /1.00 1   
Q. 19 /1.00 0.89   
Q. 20 /1.00 0   
Hours taken 243.583   
Start 1454050560000000000   
End 1454927460000000000   
DoW 4   
  
 1 \  
Surname 8933e77b83eee991c3d3aae3c88b8176   
First name d3d2792b4f002c766a99bb7152ccf3ee   
OU Computer Username 5759a8752ba4984049c0be67c7a6270f   
Personal Identifier 7f30f4954957eb71909d7ec30ccb0e0a   
Institution NaN   
Department NaN   
Email address ff9114a689c8e5141f46cf1386b5ba9d   
State Finished   
Started on 2016-01-29 09:14:00   
Completed 2016-01-30 22:10:00   
... ...   
Q. 15 /1.00 0.89   
Q. 16 /1.00 1   
Q. 17 /1.00 1   
Q. 18 /1.00 1   
Q. 19 /1.00 1   
Q. 20 /1.00 1   
Hours taken 36.9333   
Start 1454058840000000000   
End 1454191800000000000   
DoW 4   
  
 2   
Surname aa8af604652b6b631f11061fe8c69809   
First name ec83707bbca209e51a65e9864ac8baeb   
OU Computer Username b8fe6559c9ba7bd27819a5460c9359c8   
Personal Identifier 4e74dca9c74ad198eeb162dac5193d9e   
Institution NaN   
Department NaN   
Email address ec01fe85557bab32113d8fbb47b0685d   
State Finished   
Started on 2016-01-29 11:16:00   
Completed 2016-01-29 11:57:00   
... ...   
Q. 15 /1.00 1   
Q. 16 /1.00 1   
Q. 17 /1.00 1   
Q. 18 /1.00 1   
Q. 19 /1.00 1   
Q. 20 /1.00 1   
Hours taken 0.683333   
Start 1454066160000000000   
End 1454068620000000000   
DoW 4   
  
[36 rows x 3 columns]

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.

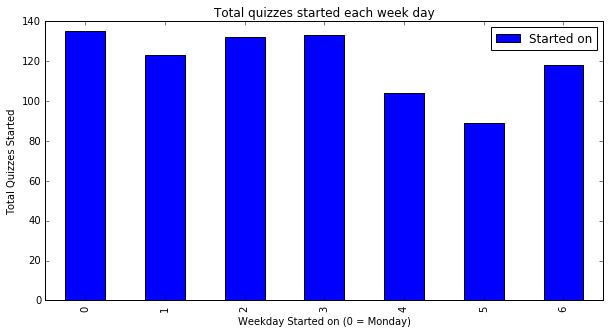
Make a scatter plot of day of week against score.

# Show the number of quizzes started on each day of the week.  
quizzes\_by\_dow = pd.pivot\_table(icma\_df, index=['DoW'], values=['Started on'], aggfunc='count')  
quizzes\_by\_dow

Started on  
DoW   
0 135  
1 123  
2 132  
3 133  
4 104  
5 89  
6 118

# Plot the results above as a bar chart.  
quizzes\_by\_dow.plot.bar(title="Total quizzes started each week day",  
 figsize=(10,5),  
 color='blue')  
plt.xlabel('Weekday Started on (0 = Monday)')  
plt.ylabel('Total Quizzes Started')

<matplotlib.text.Text at 0xaf7589ac>



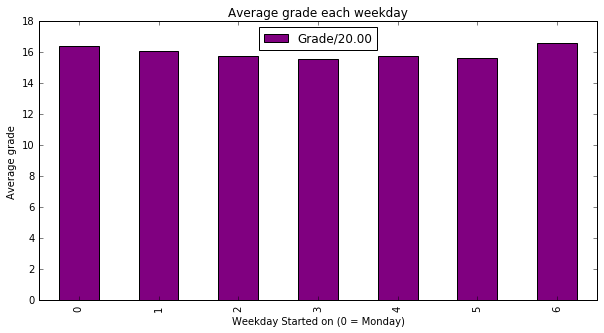
png

# Show the mean score of quizzes started on each day of the week.  
mean\_quiz\_score\_by\_dow = pd.pivot\_table(icma\_df, index=['DoW'],   
 values=['Grade/20.00'],   
 aggfunc='mean')  
mean\_quiz\_score\_by\_dow

Grade/20.00  
DoW   
0 16.388593  
1 16.093984  
2 15.744621  
3 15.585714  
4 15.791635  
5 15.662135  
6 16.618051

# Plot the results above as a bar chart.  
mean\_quiz\_score\_by\_dow.plot.bar(title="Average grade each weekday",   
 figsize=(10,5),  
 color='purple')  
plt.xlabel('Weekday Started on (0 = Monday)')  
plt.ylabel('Average grade')

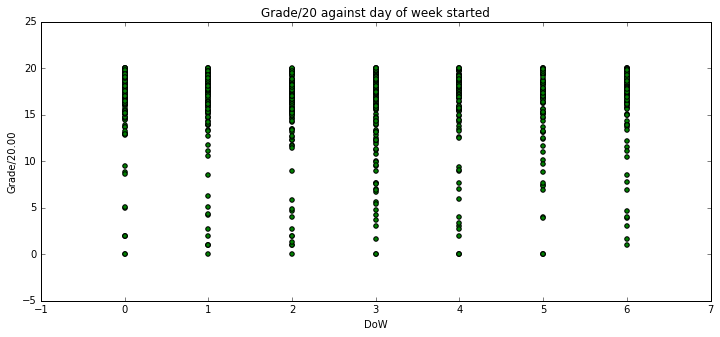
<matplotlib.text.Text at 0xaf758fec>



png

# Plot the results above as a scatter chart, of grade against day of week.  
icma\_df.plot.scatter(x='DoW', y='Grade/20.00',   
 color='green',   
 title="Grade/20 against day of week started",  
 figsize=(12,5))

<matplotlib.axes.\_subplots.AxesSubplot at 0xaf60394c>



png

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.

#add new column, 'Hour started', calculated in the same way as day of week started  
icma\_df['Hour started'] = icma\_df['Started on'].dt.hour

#check new column to make sure it looks sensible  
icma\_df[['Started on', 'Hour started', 'DoW']].head(5)

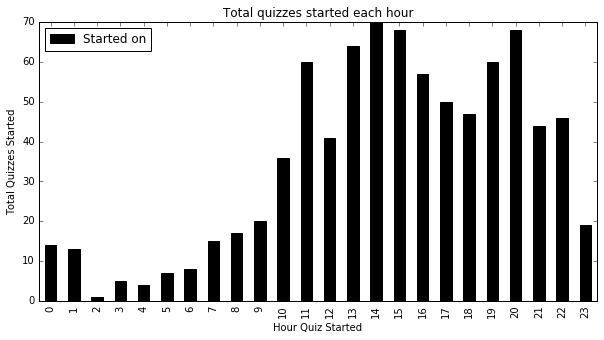
Started on Hour started DoW  
0 2016-01-29 06:56:00 6 4  
1 2016-01-29 09:14:00 9 4  
2 2016-01-29 11:16:00 11 4  
3 2016-01-29 12:08:00 12 4  
4 2016-01-29 12:51:00 12 4

#now lets get the number of quizzes started each hour:  
quizzes\_by\_hour = pd.pivot\_table(icma\_df, index=['Hour started'], values=['Started on'], aggfunc='count')  
quizzes\_by\_hour

Started on  
Hour started   
0 14  
1 13  
2 1  
3 5  
4 4  
5 7  
6 8  
7 15  
8 17  
9 20  
... ...  
14 70  
15 68  
16 57  
17 50  
18 47  
19 60  
20 68  
21 44  
22 46  
23 19  
  
[24 rows x 1 columns]

# Plot the results above as a bar chart.  
quizzes\_by\_hour.plot.bar(title="Total quizzes started each hour",  
 figsize=(10,5),  
 color='black')  
plt.xlabel('Hour Quiz Started')  
plt.ylabel('Total Quizzes Started')

<matplotlib.text.Text at 0xaf51508c>



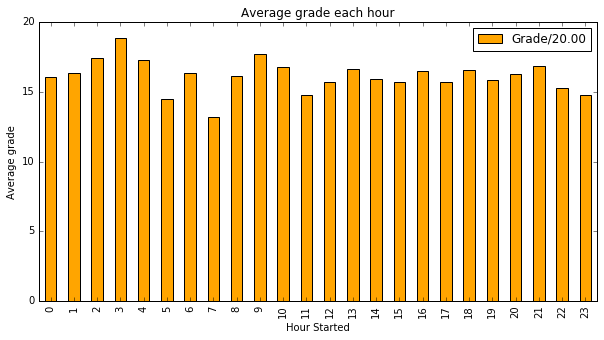
png

#now lets calculate the mean score for every hour the quiz was started  
mean\_quiz\_score\_by\_hour = pd.pivot\_table(icma\_df, index=['Hour started'], values=['Grade/20.00'], aggfunc='mean')  
mean\_quiz\_score\_by\_hour

Grade/20.00  
Hour started   
0 16.025714  
1 16.346154  
2 17.430000  
3 18.872000  
4 17.310000  
5 14.474286  
6 16.341250  
7 13.199333  
8 16.137059  
9 17.723000  
... ...  
14 15.919714  
15 15.693971  
16 16.463509  
17 15.677600  
18 16.554681  
19 15.874000  
20 16.275441  
21 16.864091  
22 15.306522  
23 14.786316  
  
[24 rows x 1 columns]

#and finally, lets plot the above table in a bar chart  
mean\_quiz\_score\_by\_hour.plot.bar(title="Average grade each hour",   
 figsize=(10,5),  
 color='orange')  
plt.xlabel('Hour Started')  
plt.ylabel('Average grade')

<matplotlib.text.Text at 0xaf3fd18c>



png

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

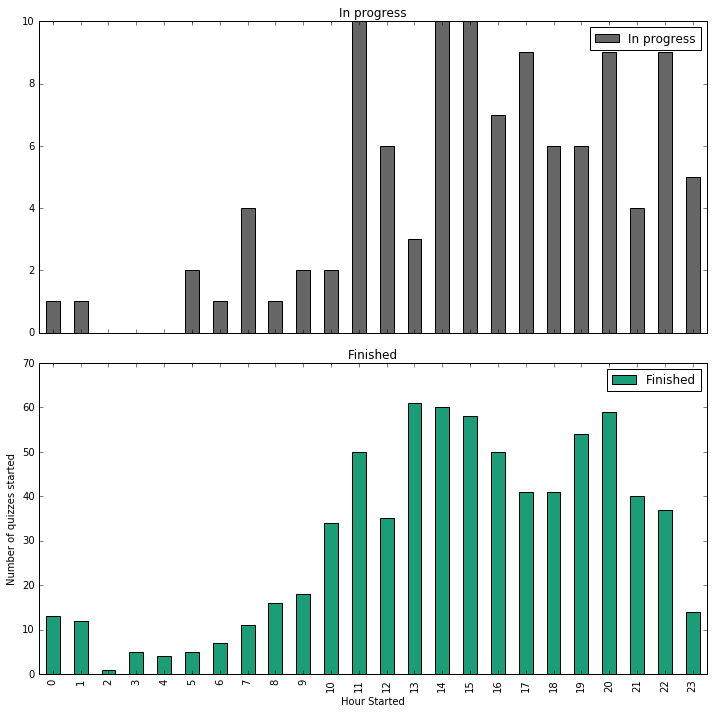
plt.tight\_layout()  
fig.subplots\_adjust(top=0.90)

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')  
  
#lets pivot icma\_df again, this time on Hour started and State  
#NB unstack turns the 2 state rows / hour into 2 columns  
quizzes\_by\_hour\_and\_state = pd.pivot\_table(icma\_df, index=['Hour started', 'State'],   
 values=['Grade/20.00'],   
 aggfunc='count').unstack()   
  
#now we reset the index (so we can plot Hour Started), and fill NaNs with 0s  
quizzes\_by\_hour\_and\_state.reset\_index(inplace=True)  
quizzes\_by\_hour\_and\_state.fillna(0, inplace=True)  
quizzes\_by\_hour\_and\_state.columns = ['Hour started', 'Finished', 'In progress']  
  
#now lets plot the number of quizzes attempted by hour, for finished and in progress students  
quizzes\_by\_hour\_and\_state.plot.bar(  
 x=['Hour started'],  
 y=['In progress', 'Finished'],  
 figsize=(10,10),  
 subplots=True,  
 colormap='Dark2\_r')  
plt.xlabel('Hour Started')  
plt.ylabel('Number of quizzes started')  
  
# Keep these lines at the end  
plt.tight\_layout()  
fig.subplots\_adjust(top=0.90)  
  
# NB would have been much quicker to do a histogram here: E.g.  
#icma\_df.hist(column='Hour started',  
# bins=range(25),  
# by=['State'],  
# figsize=(15,5))  
#which I think gives the same result!

<matplotlib.figure.Figure at 0xaf26aa8c>



png

## 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)* The time of day doesn't seem to be a significant factor in the grade obtained. The (orange) chart which shows the average grade / hour does seem to peak at 3am, and 9am, and dip at around 7am. However, when we look at the distribution of start times we can clearly see the peak time which students choose to take their tests are between 12 and 3pm, and again between 7 and 8pm. This seems to be true for both complete and incomplete quizzes (see above bar chart). The 3am peak in average grade obtained is therefore based on a small number of students (5 or so), and so can be ignored as the only thing it tells us is that a small number of students work best in the middle of the night!

There seems to be another peak of average score at around 9am. This could indicate that students perform better when tested in the morning, but again this is only based on around 20 or so students which is too small a sample (compared to the other students tested) to draw any conclusions. In the end it may only tell us that some students are 'morning people'.

Ultimately, I think this data reflects that there are a mixture of study patterns which suit different students - some work best in the morning, others in the middle of the night. Most students however, will take their tests between 12 and 3pm, or 7 and 8pm, and performance seems reasonably similar regardless.

Interestingly, the day of week charts show that a) less students take quizzes on Fridays and Saturdays, and b) students tend to do better if they take quizzes on Sundays and Mondays.

# e) Difficulty of questions (7 marks)

([Contents](#contents))

Which questions are harder?

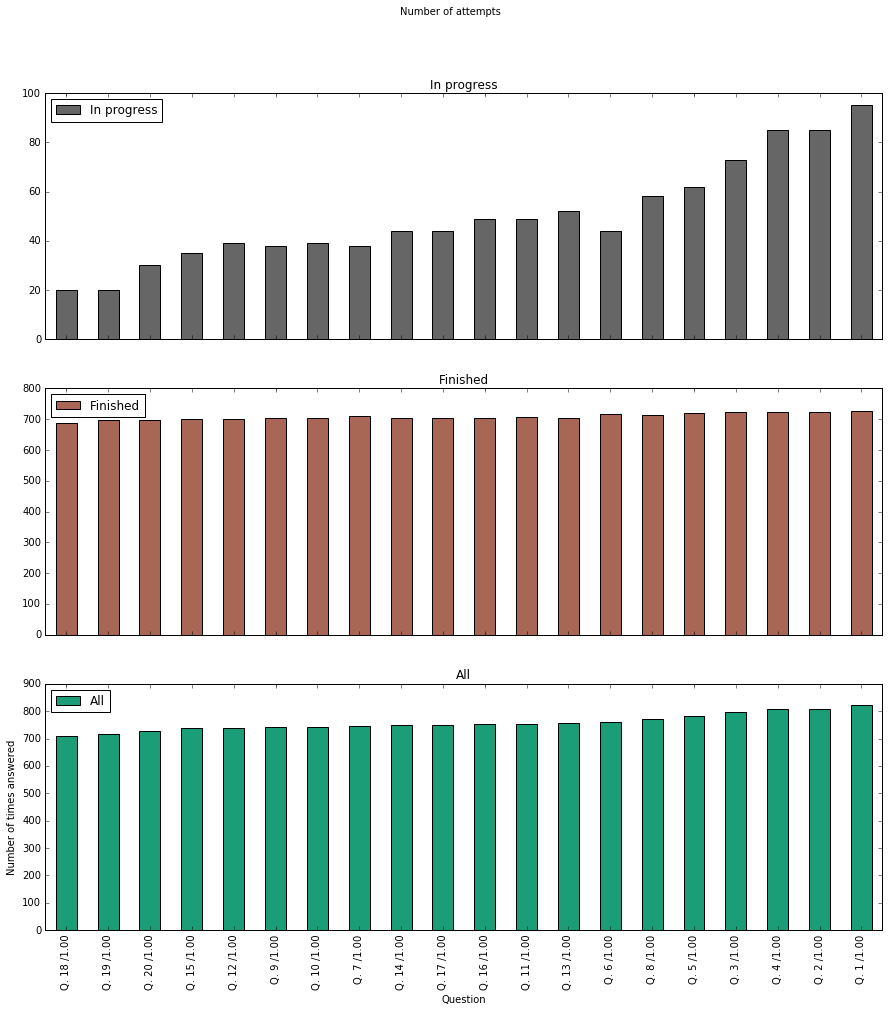
Note that we can judge difficulty in two ways: lower grades for particular questions, or fewer questions answered (students tend to avoid questions they perceive as difficult).

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.

#firstly, we need to read in the csv file again, as we've already filled NA values with 0  
#we need to go back to the raw data, so we can count the number of non-null (i.e. answered)  
#values for each question  
  
icma\_raw\_df = pd.read\_csv('data/icma.csv', parse\_dates=['Started on','Completed'], na\_values=['-'])  
  
#now we can pivot on state, sum all non-NA values, and just report on the question colums  
#margins=true gives us a total column  
questions\_answered = pd.pivot\_table(icma\_raw\_df, index=['State'], values=question\_columns, aggfunc='count', margins=True)  
  
#we need to transpose the table, to get the questions on the bottom.  
questions\_answered = questions\_answered.T  
#we need to reset the index to be able to use the question column  
questions\_answered.reset\_index(inplace=True)  
questions\_answered.rename(columns={'index':'Question'}, inplace = True)  
#lets sort the results by average score to make it easier to interpret the graph  
questions\_answered.sort\_values(by='All', inplace=True)  
#now we can plot the number of scores for each question  
questions\_answered.plot.bar(title="Number of attempts",   
 x=['Question'],  
 y=['In progress', 'Finished', 'All'],  
 figsize=(15,15),  
 subplots=True,  
 colormap='Dark2\_r')  
plt.xlabel('Question')  
plt.ylabel('Number of times answered')

<matplotlib.text.Text at 0xaf117f8c>

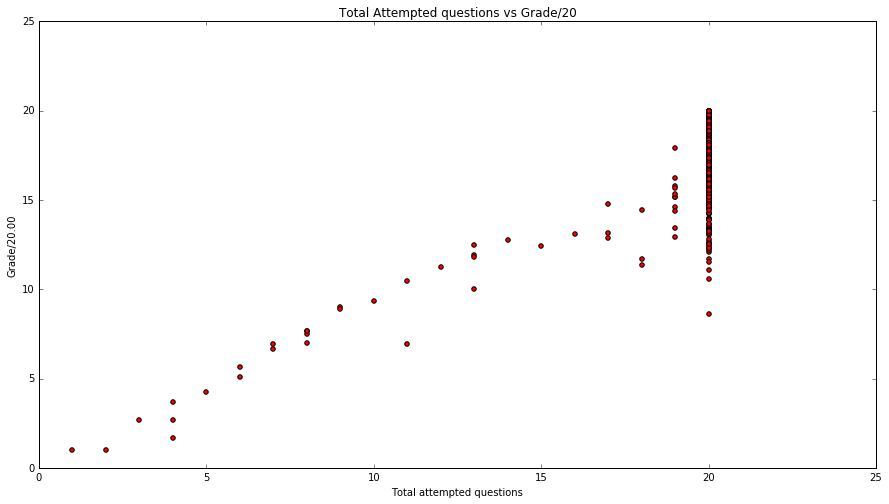


png

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.

# first, lets create a new column 'Total attempted questions'   
icma\_raw\_df['Total attempted questions'] = icma\_raw\_df[question\_columns].count(axis=1)  
  
# Plot the results above as a scatter chart, with no of questions answered on the x-axis,   
#and grade on the y-axis  
icma\_raw\_df.plot.scatter(x='Total attempted questions', y='Grade/20.00',   
 color='red',   
 title="Total Attempted questions vs Grade/20",  
 figsize=(15,8))

<matplotlib.axes.\_subplots.AxesSubplot at 0xaee6e74c>

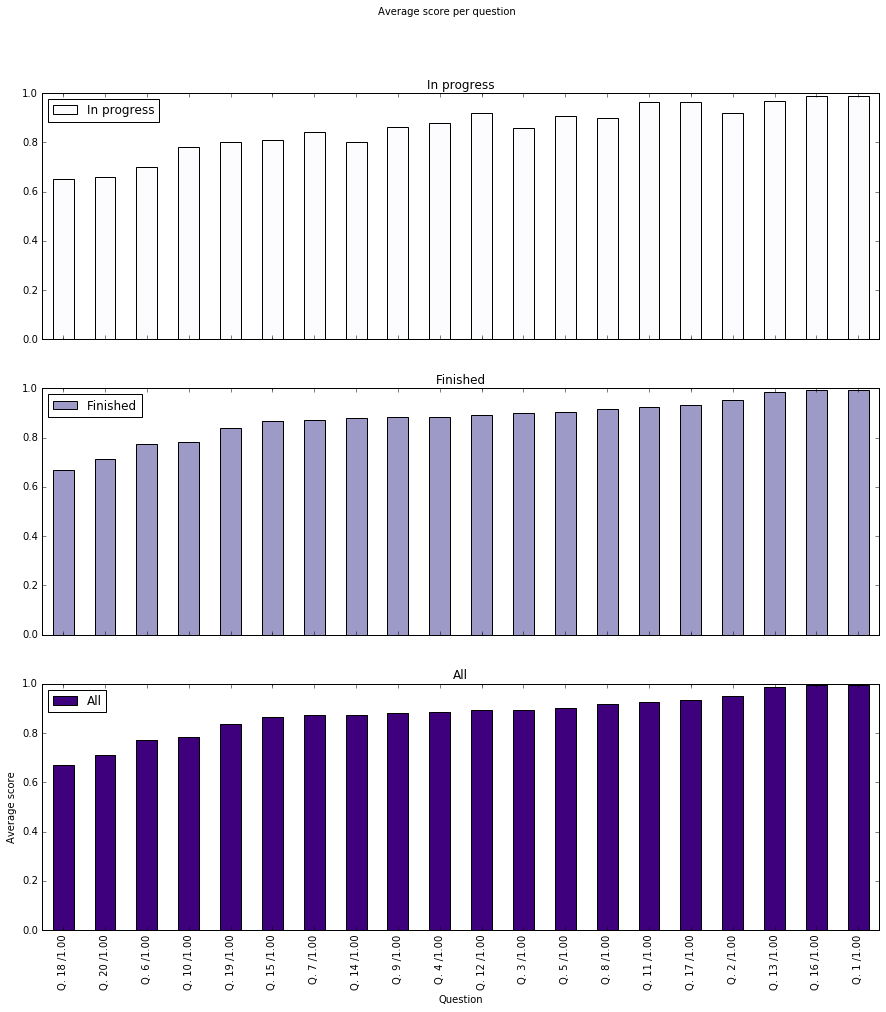


png

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')  
  
#now we can pivot on state, calculate the mean for all non-NA values, and just report on the question colums  
#margins=true gives us a total column  
avg\_question\_score = pd.pivot\_table(icma\_raw\_df, index=['State'], values=question\_columns, aggfunc='mean', margins=True)  
  
#we need to transpose the table, to get the questions on the bottom.  
avg\_question\_score = avg\_question\_score.T  
#we need to reset the index to be able to use the question column  
avg\_question\_score.reset\_index(inplace=True)  
avg\_question\_score.rename(columns={'index':'Question'}, inplace = True)  
#lets sort the results by average score to make it easier to interpret the graph  
avg\_question\_score.sort\_values(by='All', inplace=True)  
  
#now we can plot the average score for each question  
avg\_question\_score.plot.bar(title="Average score per question",   
 x=['Question'],  
 y=['In progress', 'Finished', 'All'],  
 figsize=(15,15),  
 subplots=True,  
 colormap='Purples')  
plt.xlabel('Question')  
plt.ylabel('Average score')

<matplotlib.text.Text at 0xaf0126ac>



png

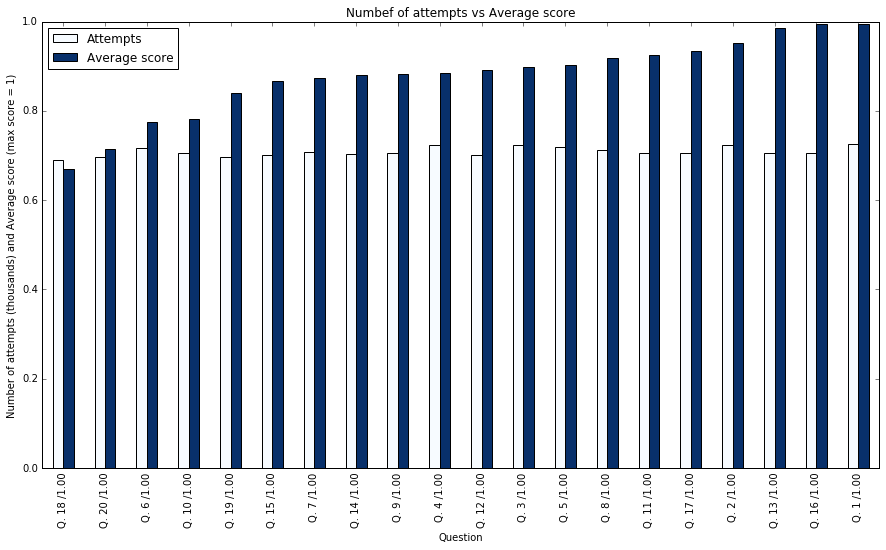
Generate a bar chart that shows two columns for each question. One column should show the number of attempts at that question, the other column should show the mean score. You should rescale the number of question attempts to fit the range 0–1 so that the two types of data a visible on the same graph.

Use only data from completed quiz attempts.

You may need to generate intermediate DataFrames as you go.

from pandas import DataFrame  
#we want a table, with q1 - 20 as rows, and an attempts and average score column  
#first, lets discard all non-completed rows, and only keep the question columns  
question\_attempts = icma\_raw\_df[icma\_raw\_df['State'] == 'Finished']  
question\_attempts = question\_attempts[question\_columns]  
  
#next, lets get the column count and mean from the question\_attempts dataframe  
  
#get total question attempts, divided by 1000  
question\_count = question\_attempts.count() / 1000  
  
#get mean score for each question  
question\_average = question\_attempts.mean()  
  
#now we can build a new dataframe from the count and mean series above  
question\_attempts\_vs\_mean\_score = DataFrame(data={'Attempts':question\_count,   
 'Average score':question\_average})  
#lets sort the dataframe  
question\_attempts\_vs\_mean\_score.sort\_values(by='Average score', inplace=True  
 )  
#now we can plot the above table  
question\_attempts\_vs\_mean\_score.plot.bar(title="Numbef of attempts vs Average score",   
 figsize=(15,8),  
 colormap='Blues')  
plt.xlabel('Question')  
plt.ylabel('Number of attempts (thousands) and Average score (max score = 1)')

<matplotlib.text.Text at 0xaef0d3cc>



png

## Analysis

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

**Write your answer here** *(150 words)* The data plots seem to paint a clear picture as to the difficulty of the questions in this quiz. If we take the first measure of difficulty, the number of times a question is attempted, we can see from the first chart that Questions 18, 19 and 20 have the least attempts, in particular for students who didn't complete the Quiz. Question 6 also proved tricky for in progress students (grey chart), but not so for the completed students (red chart).

If we take the other measure of difficulty (average question score), we can see from the purple bar charts that Question 18, 20 and 6 have the lowest average score, for both in progress and completed students.

These findings are again reflected in the final (blue) chart, which shows number of attempts and average score plotted together. For Question 18, 20 and 6 it seems that the difficulty of the question is reflected in both the low number of attempts and the low average score. Question 19 is interesting however, in that it was the 2nd least attempted question but only the 5th lowest average score, suggesting the question was easier than anticipated by students who attempted it.

# 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)* This analysis has yielded some interesting insights into students' study patterns and their results for this quiz. The data showed performance was very high in this quiz, with 87% of all attempts completed, and only 13% remaining uncompleted. Of those completed attempts 25% scored 19/20 or above, and over half gained 17.8 or above. Of those students who completed the quiz, it was also clear that a good proportion (12.5%) took the quiz multiple times in order to gain a higher score (with 1 student taking the quiz 11 times!).

Of those students who didn't complete a quiz attempt, the data suggests that 17% of those students would have scored 15/20 or above, suggesting that these students would have performed better than they realised, or that in some cases perhaps students forgot to press submit.

Examining the effect of the time and day the test was started, it is clear that most students take their tests between 12 and 3pm, and that students tend not to take tests on a Friday or Saturday. Students who took their tests on a Sunday or Monday tended to get a higher grade on average, though there didn't seem to be any obvious correlation between the time a test was started and the final grade.

The amount of time taken on the quiz was also examined. This showed that 75% of attempts are completed within 22 days, and 50% within 2 hours. There was no obvious correlation between time taken and grade achieved within this time frame, although the data showed a gentle downwards trend in grade for tests which took over 3 weeks to complete (suggesting students had forgotten the material), and a marked deterioration in grade for tests completed in under 15 minutes (suggesting students hadn't taken the time to read or understand the questions properly).

In terms of question difficulty, this was measured in 2 ways: number of times a question was attempted (with harder questions being attempted less), and average score gained for that question. Using both measures this showed that question 6, 18 and 20 were found especially difficult by students. Question 19 was the 2nd least attempted question, but only the 5th lowest average score, suggesting that once students tackled they performed reasonably. This could suggest a poorly worded question.

While these findings are interesting, its important to bear in mind that this analysis is based on a single test, and so its difficult to draw more general conclusions about student study patterns and performance on ICMAs. More broad and general conclusions could be made from analysing more data across several ICMAs for these students (longitudinal data), or by increasing the number of data points across several tests from other modules (cross sectional data). Another interesting area of research would be to investigate the effect of retaking a quiz on the final grade, an effect which was only explored for 1 student in this analysis.

# 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)* Data on students ICMA performance is clearly a requirement for the OU, as this data forms part of the students' academic record. It is also of use in monitoring student performance, study patterns, gauging the difficulty of tests and courses etc. Such data however, is classed as personal data under the Data Protection Act 1998, and therefore places legal obligations on the OU as the 'data controller' in protecting the student's data.

If the OU were planning on releasing the data for analysis without redacting or anonymising any data, they would be breaching the 'fair and lawful' processing principle of the DPA. This principle puts legal responsibly on the OU (the 'data controller') to prevent personal data being disclosed to anyone other than the student it related to (the 'data subject'). 'Personal data' is defined as any data relating to living individuals by which an individual can be identified, either on its own or by cross-referencing with other available data. As the OU's ICMA data contains the names, email addresses and usernames of students, this data is explicitly personal data. It could be used for unsolicited marketing or publication without the student's consent. As the data relates to a course offered by the OU, this data could also be cross-referenced with other publically available information (linked in or facebook for example), to identify individuals.

As long as the data allowed students to be identifiable, there would be a breach of the DPA if this data was release (unless a case could be made for such a release to be considered 'fair and lawful' processing). In order for this data to be released legally the OU would have to break the link between the data, and the identity of the students. This can be achieved through a variety of techniques including redacting (removing) sensitive parts of the data (such as names), or anonymising the data so that it no longer identifies an individual (through aggregation or obfuscation).

## 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())

TM351 f2b9934cd167c59b513af9157ea63b02  
TM352 b379e8f5811f6f59e24a3f0f2307fe86  
tm351 954ed12ba4dc405d3967e086b872f10d

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)* This is an example pseudonymisation, where personal data is anonymised by replacing the personal attributes with another value or code (in this case, by replacing the name, email address, personal identifier and OU username with hashed values).

This approach has the advantage of allowing a record relating to an individual student to be kept intact, retaining all relevant details, and to be linked to other related records, but without revealing the identity of the student. If the hash function is of sufficient strength, and doesn't allow you to work out the original data from the hashed value (one-way), this can be a very effective means of protecting personal data.

There are weaknesses to this approach however. Sometimes, despite pseudonymisation, an individual can still be reidentified. However strong a hashing function is, there may be other data available in the public domain which allows a 'motivated intruder' to combine other data sources together and deduce the identify of an individual (or at least narrow the possibilities down to a small number of individuals). This is especially true in small datasets, where the very presence or absence of a record can allow reidentification when combined with other easily available data such as electoral registers.

Sadly, it is impossible to be certain that reidentification won't be possible now or in the future. The best that can be done is to reduce the risk by taking adequate precautions are taken (such as reviewing existing public sources of information, or performing a 'motivated intruder' test) and weigh up the possible risks and benefits of this form of anonymization over other methods.

## 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)* Other anonymization methods which could be used are suppression (redaction) or aggregation.

Suppression means suppressing any personal data before release. In this example, it could mean removing the student's name, email address etc. This approach carries the same risks as pseudonymisation, in that given a sufficiently motivated individual and other data to cross-reference against, the student could be re-identified.

Aggregation means aggregating all student records together, and releasing summary data which related to the group of students as a whole, rather than individuals (for instance, average mark, average time spent on a quiz etc.). This method has the advantage that it almost impossible to re-identify individuals with a sufficiently large dataset, as data relating to individuals has been discarded and rolled up into summary values. However, as the dataset gets smaller the likelihood of identification increases. For this reason, summary values relating to small populations are often suppressed before release.

### 45 marks in total