Examiners' commentaries 2022

ST2195 Programming for data science: Preliminary examination

Important note

This commentary reflects the examination and assessment arrangements for this course in the academic year 2021–22. The format and structure of the examination may change in future years, and any such changes will be publicised on the virtual learning environment (VLE).

Information about the subject guide and the Essential reading references

Unless otherwise stated, all cross-references will be to the latest version of the course (2021). You should always attempt to use the most recent edition of any Essential reading textbook, even if the commentary and/or online reading list and/or subject guide refer to an earlier edition. If different editions of Essential reading are listed, please check the VLE for reading supplements – if none are available, please use the contents list and index of the new edition to find the relevant section.

General remarks

Learning outcomes

At the end of this course, and having completed the Essential reading and activities, you should be able to:

- convert raw data to relational databases such as SQL
- import data to Python and R, apply data manipulation and visualisation
- program in Python and R
- develop software using version control via Git.

Examination revision strategy

Many candidates are disappointed to find that their examination performance is poorer than they expected. This may be due to a number of reasons. The *Examiners' commentaries* suggest ways of addressing common problems and improving your performance. One particular failing is '**question spotting**', that is, confining your examination preparation to a few questions and/or topics which have come up in past papers for the course. This can have serious consequences.

We recognise that candidates may not cover all topics in the syllabus in the same depth, but you need to be aware that the examiners are free to set questions on **any aspect** of the syllabus. This means that you need to study enough of the syllabus to enable you to answer the required number of examination questions.

The syllabus can be found in the Course information sheet in the section of the VLE dedicated to each course. You should read the syllabus carefully and ensure that you cover sufficient material in preparation for the examination. Examiners will vary the topics and questions from year to year and may well set questions that have not appeared in past papers. Examination papers may legitimately include questions on any topic in the syllabus. So, although past papers can be helpful during your revision, you cannot assume that topics or specific questions that have come up in past examinations will occur again.

If you rely on a question-spotting strategy, it is likely you will find yourself in difficulties when you sit the examination. We strongly advise you not to adopt this strategy.

Examiners' commentaries 2022

ST2195 Programming for data science: Preliminary examination

Important note

This commentary reflects the examination and assessment arrangements for this course in the academic year 2021–22. The format and structure of the examination may change in future years, and any such changes will be publicised on the virtual learning environment (VLE).

Information about the subject guide and the Essential reading references

Unless otherwise stated, all cross-references will be to the latest version of the course (2021). You should always attempt to use the most recent edition of any Essential reading textbook, even if the commentary and/or online reading list and/or subject guide refer to an earlier edition. If different editions of Essential reading are listed, please check the VLE for reading supplements – if none are available, please use the contents list and index of the new edition to find the relevant section.

Comments on specific questions

Section A

Question 1

Define the following objects in Python:

```
A = [2,3]
B = {'python','r','python'}
C = ('george', 6, A)
```

Briefly describe what is happening in each line of the following Python code stating also the type of data structure involved.

```
A[1] = 4
print(B)
C[2]
```

(10 marks)

Approaching the question

The first line changes the 2nd element of the list A, 3, with 4.

The second line prints the set `B` which contains no duplicates so `python` will only appear the first time.

The third line prints the third element of the tuple `C`, which is now the list `[2,4]`. Note that tuples are immutable but they can contain mutable objects.

Question 2

For each of the circumstances below discuss in no more than two sentences whether side-by-side violin plots provide an appropriate choice?

(a) **When we want to study the empirical density of a variable.**

(b) **When we want to compare frequencies of one variable across different categories of another variable.**

(c) **When we want to monitor changes in the distribution of a variable across different categories of another variable.**

(d) **When we want to explore the association between two continuous variables.**

(8 marks)

Approaching the question

Make sure to adhere to the 'at least two sentences' rule. No marks will be awarded otherwise.

(a) Not the most appropriate. Histograms or kernel density plots are sufficient for a single variable.

(b) Not the most appropriate. Violin plots are for continuous variables. Frequencies are better suited to categorical variables.

(c) Appropriate. It provides the empirical density of the continuous variable across the categories of the other variable.

(d) Not the most appropriate. This can be done with a scatterplot.

Question 3

For each of the statements below state whether it is correct. Also, provide a justification for your answer in one sentence.

(a) **In unsupervised learning we aim to minimise the training error.**

(b) **For a categorical input with 3 categories, it suffices to produce 2 dummy variables.**

(c) **In a classification task with a binary target, categories being 'negative' and 'positive', the sensititivity is the probability of negative individuals being classified as negative.**

(d) **Training a machine learning pipeline could involve the task of handling missing values.**

(8 marks)

Approaching the question

Make sure to adhere to the 'at least one sentence' rule. No marks will be awarded otherwise.

(a) Incorrect. Typical tasks aim to identify groups of individuals or variables.

(b) Correct. One category will be set to be the reference and two dummy variables are needed as indicators of each of the two remaining categories.

(c) Incorrect. It is the probability of positive individuals to be classified as positives.

(d) Correct. Choosing the way to handle missing values may be viewed as a parameter of the machine learning pipeline.

Question 4

Which of the following statements are correct? There is at least one correct statement, and negative marks apply for wrong choices.

- **(a) Jupyter notebooks cannot handle R code.**
- **(b) R scripts can only be executed with R Markdown.**
- **(c) Markdown is a programming language.**
- **(d) A `for` loop can be replaced by a `while` loop to perform the same operation.**
- **(e) A scatterplot will illustrate if a continuous variable is responsible for changes in another continuous variable.**

(8 marks)

Approaching the question

In such questions it is not required to provide any justification, it suffices to only give your answer. In this case the answer below would get full marks.

Correct answers: (c), (d) and (e).

Question 5

Note from which language (R or Python) each of the following code chunks is from:

C1. `mat = rbind(c(1, 4),c(7,8))`

C2. `import numpy`

C3. `?lm`

C4. `b[-1]=1`

C5. `for (i in 1:4){k=1}`

C6.

```
def triple(x):
    k=3*x
    return k
```

C7. `f2=f^2`

C8. `if (mark >= 70){print('first')}`

C9. `plot(x,y)`

C10. `apply(df, 2, mean)`

C11. `df = read.csv("data.csv")`

C12. `df.describe()`

C13. `Auto = pd.read_csv("automobileBI.csv")`

C14.

```
for i in range(0,K):
    k[i]=f[i+4]
```

C15. `hist(y)`

C16. `plt.plot(x,y)`

(8 marks)

Approaching the question

Again, as in Question 4, it is not required to provide any justification, it suffices to only give your answer. In this case the answer below would get full marks.

C1: R.

C2: Python.

C3: R.

C4: Python.

C5: R.

C6: Python.

C7: R.

C8: R.

C9: R.

C10: R.

C11: R.

C12: Python.

C13: Python.

C14: Python.

C15: R.

C16: Python.

Section B

Question 6

Using conditional statements write a program using informal code (could be either R or Python or just plain words) that takes the list of numbers `L=[2,4,3,6,7,11,12]`, checks if each of them is even or odd, counts how many of them are even and computes the sum of all even numbers as well as the sum of all odd numbers.

(10 marks)

Approaching the question

By informal code we mean here that the code does not need to be syntactically correct, i.e. to actually work if we run in R or Python. The answer should just be logically correct to illustrate knowledge on what conditional statements do. An example answer is given below:

```
even_sum = 0
odd_sum = 0
even_count = 0
n = length(L)
for i=1:n
  if L[i]/2 is integer
    even_sum = even_sum + L[i]
    even_count = even_count + 1
  else
    odd_sum = odd_sum + L[i]
print('number of even numbers', even_count)
print('sum of even numbers', even_sum)
print('sum of odd numbers', odd_sum)
```

Question 7

Describe what the following chunks of code are doing.

(a)
```
SELECT student_id, mark, department
FROM students
WHERE student_id >= 200 AND average_mark >= 40
ORDER BY average_mark
```

(b)
```
inner_join(student, university, by = "country") %>%
    filter(classification == "first")
```

(10 marks)

Approaching the question

(a) The statement will return all available records for the attributes `student_id`, `average_mark` and `university` such that the student id is greater than or equal to 200, and the average mark is 40 or higher.

(b) Finds all records in tables `student` and `university` that have matching values of `country`, and returns only those records where the classification is a 'first'.

Question 8

Explain in no more than 2 sentences, why the following statements are wrong.

(a) **XML files are usually smaller than JSON files.**

(b) **A matrix in R can contain different types of data.**

(c) **Python lists and dicts (dictionaries) can contain duplicate values.**

(d) **In python the command `print(A[2,1])`, where `A` in numpy 2-dimensional array, will print the first element of the second row.**

(e) **`mlr3` is an R platform used mainly for data visualisation.**

(f) **Histograms and kernel density plots can be used to help us understand the shape of the distribution of categorical variables.**

(g) **Side by side boxplots provide information about the association of two continuous variables.**

(h) **The command `matrix(1:30, nrow = 6)` in R will create a matrix with 6 rows and 4 columns.**

(i) **A pandas data frame can only hold factors and numeric variables.**

(j) **In Python consider a list `L` that contains only real numbers. We can increase all its elements by 1 using the command `L+1`.**

(10 marks)

Approaching the question

Make sure to adhere to the 'no more than two sentences rule'. No marks will be given to answers with more sentences. By the way 1 sentence may suffice, see below for some example answers.

(a) JSON files are generally smaller, since they do not require the end tags used in XML files to define a tree structure

(b) This is only possible for lists and data frames.

(c) Only lists, not dicts.

(d) It will print the 2nd element of the 3rd row.

(e) It is platform used mainly for machine learning pipelines.

(f) They provide information for the distribution of continuous variables.

(g) They are suitable for the association between a continuous and a categorical variable, as they provide a boxplot for each category.

(h) `matrix(1:30, nrow = 5)` creates a matrix with 5 rows and 6 columns.

(i) A pandas data frame can hold multiple variable types.

(j) Cannot perform numerical aperations on lists.

Question 9

Match the commands C1–C4 with the output in O1–O4.

C1. `print(type('exam'))`

C2. `print(type(3))`

C3. `paste("<class", "'str'>")`

C4. `paste("<class", "'int'>")`

C5. `K = 5;K`

O1. `"<class 'str'>"`

O2. `5`

O3. `"<class 'int'>"`

O4. `<class 'str'>`

O5. `<class 'int'>`

(10 marks)

Approaching the question

In this question it suffices to just give the pairings without justification, since none is requested. These are:

- C1 – O4.
- C2 – O5.
- C3 – O1.
- C4 – O3.
- C5 – O2.

Question 10

Consider a dataset consisting of several insurance claims on automobile injuries that contains the following variables:

- `claim`: **the amount claimed by the policyholder**
- `attorney`: **whether an attorney was present when the claim was made (1: yes, 0: no)**
- `gender`: **1: female, 2: male, 3: not disclosed**
- `years_driving`: **age minus age the person obtained their driving licence.**

(a) **Describe what graphs you would produce to demonstrate how the presence of an attorney, gender and years of experience affect the amounts claimed by the policyholders.**

(b) **There exists a debate whether gender information should be included in the procedure of pricing insurance premiums. Suppose you had such data in your posession and were asked to comment on the whether gender information has predictive ability on the claims filed. What plots would you use to extract relevant information from these data?**

(10 marks)

Approaching the question

(a) A scatterplot of `claim` with `year_driving` could be used to get some understanding between these two. Separate versions of this plot with the points labelled according to `attorney` and `gender` will also be useful to explore whether this association changes across the categories. Also, side-by-side boxplots, violin or ridgeline plots of the `claim` across the categories of `gender` and `attorney`.

(b) The side-by-side boxplot (or violin or ridgeline) of `claim` across gender and also the scatterplot of `claim` and `years_driving` with the points labelled by gender, would be potentially useful.

Question 11

Rewrite the following script (in which `f` is a numerical vector) by replacing the `while` loop with a `for` loop in a way so that the code does exactly the same thing.

```
count = 0;
fsum = 0;
half_f_sum = 0.5*sum(f);
while (fsum < half_f_sum){
   count = count + 1;
   fsum = fsum + f[count];
}
```

(10 marks)

Approaching the question

An example answer is given below.

```
count = 0;
fsum = 0;
half_f_sum = 0.5*sum(f)
for (i in 1:length(f)){
   count = count + 1;
   fsum = fsum + f[i];
   if (fsum >= half_f_sum){
     break
   }
}
```