ds205 - entry test practice

Table of Contents

1. Entry Test Review - R and Python

1.1. Creating and subsetting a data frame

1. R: Explain every line in this code block! Write comments in front or next to the commands. Running the code might help.

```
name <- c("Python","R","SQL") # create character vector `name`
type <- c("Practice","Project","Lecture") # create character vector `type`
number <- c(100,430,200) # create numeric vector `number`
courses_df <- data.frame(name,type,number) # create data frame from vectors
subset(courses_df, number < 200 ) # select rows/records whose number < 200
```

2. Python:

    1. Import `pandas` library
    2. Construct data frame from data dictionary (see pandas documentation)
    3. Change row index to start from 1.

```
import pandas as pd  # import pandas library and use `pd` as alias
d = {  # define a dictionary of key:value pairs
    'name': ["Python","R","SQL"],
    'type': ["Practice","Project","Lecture"],
    'number' : [100,430,200]
}
courses_df = pd.DataFrame(data=d) # create data frame from the dictionary
print(courses_df)
courses_df.index=[1,2,3] # move the indices up by one
print(courses_df)
print(courses_df[courses_df['number'] < 200]) # use a flag vector
print(courses_df.query('number < 200')) # use a function
```

1.2. Looking at data

1. R: Show three ways to look at your data, e.g. `courses_df`.

```
courses_df
print(courses_df)
str(courses_df)
summary(courses_df)
```

2. Python:

   - print the `DataFrame` as a whole.
   - `pd.items` iterates over `DataFrame` items (used for loops).
   - `pd.describe()` is the equivalent of R's `summary` function.

```

```

print(courses$_{df}$) print(courses$_{df.items}$) print(courses$_{df.describe}$()) # equivalent of summary() in R

```
      name      type  number
   1  Python  Practice     100
   2       R   Project     430
   3     SQL   Lecture     200
g              number
   count    3.000000
   mean   243.333333
   std    169.213869
   min    100.000000
   25%    150.000000
   50%    200.000000
   75%    315.000000
   max    430.000000
```

1.3. What is vectorisation?

1. What is vectorization? Can you think of an example?

```
foo <- c(1,2,3,4)
100 * foo
```

2. Python:

```
import numpy as np
foo = [1,2,3,4]    # this is a list
print(100 * foo)  # the * operator means replication for lists
foo_np = np.array(foo)  # turn list into numpy array
print(100 * foo_np)  # vectorisation only with numpy arrays
```

1.4. Multiplying vectors of different lengths

1. R: What happens when you multiply two vectors of differing length?

   Multiply a 4- with a 2-element vector:

```
bar <- c(100,200) # length of foo is twice that of bar
foo; bar
foo * bar
```

Multiply a 4- with a 3-element vector:

```
baz <- c(100,200,300)
foo * baz
```

2. Python:

```
foo_np = np.array([1,2,3,4])
bar_np = np.array([100,200])
print(foo_np,bar_np)
print(foo_np * bar_np)
```

1.5. Pattern matching and replacement

1. R: Change the first letter of your first name to lower case using R's pattern matching and replacement function `sub`:

```
my_name <- "Marcus"
sub(pattern = "M",
    replacement = "m",
    x = my_name)
```

2. Python:

```
my_name = 'Marcus'
print(my_name.lower())
```

1.6. Factor levels

1. R: How many levels does `factor(c("a","b","a","a","b"))` have?

```
foo <- factor(c("a","b","a","a","b","c"))
foo
str(foo)
```

2. Python:

```
import pandas as pd
data = ["a","b","a","a","b","c"] # define list
category_data = pd.Categorical(data) # convert list
print(category_data) # categorical data
print(type(category_data))
print(category_data.categories)
print(category_data.codes)
```

1.7. IN PROGRESS Matrix to factor

1. R: What is `factor(matrix("small","medium","large"))`?

```
factor(matrix("small","medium","large")) # error!
```

```
M <- matrix(c("small","medium","large"))
M
factor(M)
```

2. Python:

```
```

1.8. Extracting matrix elements

1. R: Extract "bar" from `M <- matrix(c(a="foo",b="bar"))` in two ways:

```
```

1. Python:

```
m = np.array([["foo"],["bar"]])
print(m)
print(m.shape)
```

```
```

1.9. Concatenating data frames

1. R: Explain every line in this code and in the output:

```
temp <- data.frame(jan=39)
c(temp, sep=57)
```

1. Python:

```
```

1.10. Rownames and column names

1. R: What are the `rownames` of `list(A="a", B=c(1,2,3), C=lst)`?

```
```

1. Python:

```
```

Author: Marcus Birkenkrahe

Created: 2024-03-04 Mon 12:48

Validate