EduData

bigdata-ustc

Nov 26, 2021

INTRODUCTION

Convenient interface for downloading and preprocessing datasets in education.

The datasets include:

- ASSISTments (2009-2010, 2012-2013, 2015, 2017) [Analysis]
- KDD Cup 2010 [Analysis]
- OLI Engineering Statics 2011 [Analysis]
- JunyiAcademy Math Practicing Log *[Analysis]*
- slepemapy.cz
- synthetic
- math2015 [Analysis]
- EdNet *[Analysis]*
- pisa2015math
- workbankr
- critlangacq
- math23k *[Analysis]*
- MOOCCube [Analysis]
- NIPS2020
- OpenLUNA

Your can also visit our datashop BaseData to get those mentioned-above (most of them) datasets.

Except those mentioned-above dataset, we also provide some benchmark dataset for some specified task, which is listed as follows:

- knowledge tracing benchmark dataset
- cognitive diagnosis benchmark dataset

CHAPTER

ONE

INSTALLATION

Git and install by `pip`

```
pip install -e .
```

or install from `pypi`:

```
pip install EduData
```

CHAPTER

TWO

CLI

```
edudata $subcommand $parameters1 $parameters2
```

To see the `help` information:

```
edudata -- --help
edudata $subcommand --help
```

The cli tools is constructed based on fire . Refer to the documentation for detailed usage.

CHAPTER THREE

DOWNLOAD DATASET

Before downloading dataset, first check the available dataset:

```
edudata ls
```

and get:

:: assistment-2009-2010-skill assistment-2012-2013-non-skill assistment-2015 junyi … ktbd ktbd-a0910 ktbd-junyi ktbd-synthetic …

Download the dataset by specifying the name of dataset:

```
edudata download assistment-2009-2010-skill
```

In order to change the storing directory, use the following order:

```
edudata download assistment-2009-2010-skill $dir
```

For detailed information of each dataset, refer to the docs

CHAPTER

FOUR

TASK SPECIFIED TOOLS

4.1 Knowledge Tracing

4.2 Format converter

In Knowledge Tracing task, there is a popular format (we named it `triple line (tl)` format) to represent the interaction sequence records:

```
5
419,419,419,665,665
1,1,1,0,0
```

which can be found in Deep Knowledge Tracing . In this format, three lines are composed of an interaction sequence. The first line indicates the length of the interaction sequence, and the second line represents the exercise id followed by the third line, where each elements stands for correct answer (i.e., 1) or wrong answer (i.e., 0)

In order to deal with the issue that some special symbols are hard to be stored in the mentioned-above format, we offer another one format, named `json sequence` to represent the interaction sequence records:

```
[[419, 1], [419, 1], [419, 1], [665, 0], [665, 0]]
```

Each item in the sequence represent one interaction. The first element of the item is the exercise id (in some works, the exercise id is not one-to-one mapped to one knowledge unit(ku)/concept, but in junyi, one exercise contains one ku) and the second one indicates whether the learner correctly answer the exercise, 0 for wrongly while 1 for correctly One line, one *json* record, which is corresponded to a learner's interaction sequence.

We provide tools for converting two format:

```
# convert tl sequence to json sequence, by default, the exercise tag and answer will be␣
↪converted into int type
edudata tl2json $src $tar
# convert tl sequence to json sequence without converting
edudata tl2json $src $tar False
# convert json sequence to tl sequence
edudata json2tl $src $tar
```

4.3 Dataset Preprocess

The cli tools to quickly convert the "raw" data of the dataset into "mature" data for knowledge tracing task. The "mature" data is in `json sequence` format and can be modeled by XKT and TKT(TBA)

4.4 junyi

```
# download junyi dataset to junyi/
>>> edudata download junyi
# build knolwedge graph
>>> edudata dataset junyi kt extract_relations junyi/ junyi/data/
# prepare dataset for knwoeldge tracing task, which is represented in json sequence
>>> edudata dataset junyi kt build_json_sequence junyi/ junyi/data/ junyi/data/graph_
→vertex.json 1000
# after preprocessing, a json sequence file, named student_log_kt_1000, can be found in␣
→junyi/data/
# further preprocessing like spliting dataset into train and test can be performed
>>> edudata train_valid_test junyi/data/student_log_kt_1000 -- --train_ratio 0.8 --valid_
→ratio 0.1 --test_ratio 0.1
```

4.5 Analysis Dataset

This tool only supports the *json sequence* format. To check the following statical indexes of the dataset:

- knowledge units number
- correct records number
- the number of sequence

```
edudata kt_stat $filename
```

4.6 Evaluation

In order to better verify the effectiveness of model, the dataset is usually divided into `train/valid/test` or using `kfold` method.

```
edudata train_valid_test $filename1 $filename2 --train_ratio 0.8 --valid_ratio 0.1 --
→test_ratio 0.1
edudata kfold $filename1 $filename2 --n_splits 5
```

Refer to longling for more tools and detailed information.

CHAPTER

FIVE

CITATION

If this repository is helpful for you, please cite our work

```
@misc{bigdata2021edudata,
title={EduData},
author={bigdata-ustc},
publisher = {GitHub},
journal = {GitHub repository},
year = {2021},
howpublished = {\url{https://github.com/bigdata-ustc/EduData}},
}
```

CHAPTER

SIX

MORE WORKS

Refer to our website and github for our publications and more projects

6.1 Get Started

6.2

1.
2. `ls download`
3. `dataset kt_stat`

Note: junyiEdNet

4. `graph edge_stat`

6.3

6.3.1

`github clone`

```
$ pip install -e .
```

6.3.2 `pypi`

```
$ pip install EduData
```

6.3.3

```
$ edudata $subcommand $parameters1 $parameters2
```

6.3.4

```
$ edudata -- --help
```

6.3.5 subcommmand

```
$ edudata $subcommand --help
```

6.4

Note: *

6.4.1 download

- dataset* :
- data_dir :
- override : False
- url_dict url

math23k

```
$ edudata download math23k
downloader, INFO http://base.ustc.edu.cn/data/math23k.zip is saved as math23k.zip
Downloading math23k.zip 100.00%: 2.28MB | 2.28MB
downloader, INFO math23k.zip is unzip to math23k
math23k
$ ls
math23k  math23k.zip
```

math32k

```
$ edudata download math23k .. True
downloader, INFO http://base.ustc.edu.cn/data/math23k.zip is saved as ../math23k.zip
Downloading ../math23k.zip 100.00%: 2.28MB | 2.28MB
downloader, INFO ../math23k.zip is unzip to ../math23k
../math23k
```

6.4.2 ls

•

```
$ Code edudata ls
assistment-2009-2010-skill
assistment-2012-2013-non-skill
assistment-2015
assistment-2017
junyi
KDD-CUP-2010
NIPS-2020
...
```

6.4.3 tl2json

.tl

- src* : .tl
- tar* :
- to_int : int True
- left_shift : 1 False

```
$ cat data.tl
15
1,1,1,1,7,7,9,10,10,10,10,11,11,45,54
0,1,1,1,1,1,0,0,1,1,1,1,1,0,0
$ edudata tl2json data.tl data.json
1it [00:00, 2610.02it/s]
$ cat data.json
[[1, 0], [1, 1], [1, 1], [1, 1], [7, 1], [7, 1], [9, 0], [10, 0], [10, 1], [10, 1], [10,␣
↪1], [11, 1], [11, 1], [45, 0], [54, 0]]
```

6.4.4 json2tl

.tl

- src* :
- tar* : .tl

```
$ cat data.json
[[1, 0], [1, 1], [1, 1], [1, 1], [7, 1], [7, 1], [9, 0], [10, 0], [10, 1], [10, 1], [10,␣
↪1], [11, 1], [11, 1], [45, 0], [54, 0]]
$ edudata json2tl data.json data.tl
1it [00:00, 8793.09it/s]
$ data cat data.tl
15
1,1,1,1,7,7,9,10,10,10,10,11,11,45,54
0,1,1,1,1,1,0,0,1,1,1,1,1,0,0
```

6.4.5 kt_stat

- **source***：

```
$ cat data.json
[[1, 0], [1, 1], [1, 1], [1, 1], [7, 1], [7, 1], [9, 0], [10, 0], [10, 1], [10, 1], [10,␣
↪1], [11, 1], [11, 1], [45, 0], [54, 0]]
$ edudata kt_stat data.json
doing statistics: 1it [00:00, 6159.04it/s]
in ['data.json']
knowledge units number: 7
min index: 1; max index: 54
records number: 15
correct records number: 10
the number of sequence: 1
```

```
$ edudata kt_stat data.json data1.json
doing statistics: 2it [00:00, 9218.25it/s]
in ['data.json', 'data1.json']
knowledge units number: 7
min index: 1; max index: 54
records number: 30
correct records number: 20
the number of sequence: 2
```

6.4.6 edge_stat

- **src***： .json
- **threshold**： threshold None

```
$ cat sample_graph.json
[[1, 2, 2], [2, 4, 10], [1, 3, 5], [4, 3, 6], [5, 3, 1]]
$ edudata edge_stat sample_graph.json
in sample_graph.json
5 edges
count     5.000000
mean      4.800000
```

(continues on next page)

(continued from previous page)

```
std       3.563706
min       1.000000
25%       2.000000
50%       5.000000
75%       6.000000
max      10.000000
dtype: float64
```

threshold=3

```
$ edudata edge_stat sample_graph.json 3
in sample_graph.json
3 edges
```

6.4.7 train_valid_test

8:1:1

- files* :

308:1:1

```
$ cat data.json
[0.4086358705691857, 0.5821013717870963, 0.3937663543609674, 0.3596475011511454, 0.
→6269590610755503, 0.5916270350464593, 0.40039551392826145, 0.175949398164154, 0.
→7188498245018131, 0.3353656251326548]
[0.7577482681983009, 0.7823167871569502, 0.7628718209608286, 0.6570446436834679, 0.
→7895185204556635, 0.5802078440735305, 0.27497800873078715, 0.30383370246383956, 0.
→9037409494778825, 0.910175518416613]
[0.408436652871088, 0.3176041020104178, 0.9772468567022291, 0.2958594473962345, 0.
→9400651897265613, 0.7442828330073002, 0.4328292856489826, 0.48221263297826256, 0.
→028567228727882088, 0.06838837638379969]
[0.4367401871654375, 0.9147963293632903, 0.5618913934548003, 0.555425728144243, 0.
→14801367475302585, 0.4753940552854019, 0.35687531862795085, 0.7848409683542806, 0.
→6110589151187046, 0.7982670835419365]
...
$ edudata train_valid_test data.json
dataset, INFO train_valid_test start
dataset, INFO train_valid_test: data.json -> data.train.json,data.valid.json,data.test.
→json
dataset, INFO train_valid_test end
$ wc -l data.json data.train.json data.valid.json data.test.json
30 data.json
```

(continues on next page)

(continued from previous page)

```
24 data.train.json
3 data.valid.json
3 data.test.json
60
```

6.4.8 kfold

5

- files* :

```
$ edudata kfold data.json
dataset, INFO kfold 0 start
dataset, INFO kfold 1 start
dataset, INFO kfold 0: data.json -> data.0.train.json,data.0.test.json
dataset, INFO kfold 2 start
dataset, INFO kfold 1: data.json -> data.1.train.json,data.1.test.json
dataset, INFO kfold 2: data.json -> data.2.train.json,data.2.test.json
...
$ wc -l data.*
 6 data.0.test.json
24 data.0.train.json
 6 data.1.test.json
24 data.1.train.json
 6 data.2.test.json
24 data.2.train.json
 6 data.3.test.json
24 data.3.train.json
 6 data.4.test.json
24 data.4.train.json
30 data.json
```

6.4.9 dataset.junyi.kt.extract_relations

junyi .json

- src_root： ../raw_data/junyi/
- tar_root： ../data/junyi/data/

```
$ edudata dataset junyi kt extract_relations . ./data
837it [00:00, 130560.17it/s]
junyi, INFO vertex num: 835
837it [00:00, 121630.89it/s]
junyi, INFO prerequisite edges: 985
junyi, INFO similarity edges: 1954
junyi, INFO count    1954.000000
mean        4.979990
std         2.436923
min         1.000000
25%         2.603846
50%         5.190909
75%         7.216667
max         9.000000
dtype: float64
junyi, INFO edges: 1954
junyi, INFO count    1954.000000
mean        4.511079
std         1.659825
min         1.000000
25%         3.250000
50%         4.400000
75%         5.666667
max         8.750000
dtype: float64
$ ls ./data
difficulty.json    prerequisite.json
graph_vertex.json  similarity.json
```

6.4.10 dataset.junyi.kt.build_json_sequence

n

- src_root* :
- tar_root* :
- ku_dict_path* :
- n： n 1000

session

session session 1000 59792 session

```
$ edudata dataset junyi kt build_json_sequence . ./data ./data/graph_vertex.json
reading data: 39462201it [03:28, 189554.01it/s]
calculating frequency: 100%|| 247547/247547 [00:00<00:00, 1011762.99it/s]
writing -> data/student_log_kt_1000: 100%|| 1000/1000 [00:03<00:00, 321.59it/s]
$ wc -l student_log_kt_1000
59792 student_log_kt_1000
```

6.4.11 dataset.ednet.kt.build_json_sequence

EdNet

- users_dir* :
- questions_csv* : .csv
- tar* :

```
$ edudata dataset ednet kt build_json_sequence KT1_sample EdNet-Contents/contents/
↪questions.csv sequence.json
building interactions: 100%|| 10/10 [00:00<00:00, 1240.59it/s]
$ wc -l sequence.json
10 sequence.json
```

6.4.12 dataset.ednet.kt.select_n

n

- src* :
- tar* : n
- n*

```
$ edudata dataset ednet kt select_n sequence.json top5.json 5
evaluating length of each row: 10it [00:00, 6238.74it/s]
selecting 5 most active students from sequence.json to top5.json: 10it [00:00, 58254.
↪22it/s]
$ wc -l top5.json
5 top5.json
```

6.4.13 graph.dense

ku_num

- ku_num* :
- tar* :
- undirected : False

```
$ edudata graph dense 5 graph.json
[0, 1]
[0, 2]
[0, 3]
[0, 4]
[1, 0]
...
```

6.4.14 graph.con

- ku_num* :
- src* :
- tar* :

```
$ cat data.json
[[0, 1], [1, 0], [1, 1], [2, 0]]
[[0, 1], [1, 1], [2, 0], [4, 1]]
[[0, 1], [2, 1], [3, 0], [2, 1]]
$ edudata graph con 5 data.json --tar graph.json
/home/huzr/.local/lib/python3.9/site-packages/EduData/Task/KnowledgeTracing/graph.py:529:
↪ UserWarning: do not use this function due to the lack of support from theory
warnings.warn("do not use this function due to the lack of support from theory")
constructing concurrence graph: 3it [00:00, 8701.88it/s]
$ cat graph.json
[
[
    0,
    1,
    0.21049203852953075
],
[
    0,
    2,
    0.07743569350528148
],
...
```

6.4.15 graph.trans

- ku_num* :
- src* :
- tar* :

```
$ cat data.json
[[0, 1], [1, 0], [1, 1], [2, 1]]
[[2, 0], [1, 0], [0, 1], [2, 1]]
$ edudata graph trans 3 data.json -tar result.json
constructing transition graph: 2it [00:00, 6765.01it/s]
[0.0, 0.5, 0.5]
[0.5, 0.0, 0.5]
[0.0, 1.0, 0.0]
```

6.4.16 graph.ctrans

- ku_num* :
- src* :
- tar* :

```
$ cat data.json
[[0, 1], [1, 0], [1, 1], [2, 1]]
[[2, 0], [1, 0], [0, 1], [2, 1]]
$ edudata graph ctrans 3 data.json --tar result.json
constructing coorect transition graph: 2it [00:00, 11351.30it/s]
[0.0, 0.0, 1.0]
[0.0, 0.0, 1.0]
[0.0, 0.0, 0.0]
```

6.4.17 graph.sim

- ku_num*
- src_graph*
- tar* :

```
$ edudata graph sim 5 graph.json result.json
$ cat result.json
[
    [
        0,
        1,
        0.2618280790565648
    ],
    [
        0,
        2,
        0.7264881146529072
    ],
    [
        0,
        3,
        0.4690365472434528
    ],
...
```

6.4.18 graph.ccon

- ku_num* :
- src* :
- tar* :

```
$ cat data.json
[[0, 1], [1, 0], [1, 1], [2, 0]]
[[0, 1], [1, 1], [2, 0], [2, 1]]
[[2, 1], [2, 1], [1, 1], [2, 0]]
[[1, 0], [0, 1], [0, 1], [2, 0]]
[[2, 0], [1, 1], [0, 1], [2, 1]]
$ edudata graph ccon 3 data.json --tar result.json
/home/huzr/.local/lib/python3.9/site-packages/EduData/Task/KnowledgeTracing/graph.py:510:
↪ UserWarning: do not use this function due to the lack of support from theory
warnings.warn("do not use this function due to the lack of support from theory")
constructing coorect transition graph: 5it [00:00, 18927.36it/s]
[[0. 1. 0.]
[1. 0. 0.]
[0. 0. 0.]]
```

6.5

6.5.1

6.5.2

```
[["1", "0/1"], ["2", "0/1"] ...]
[["3", "0/1"], ["4", "0/1"] ...]
...
```

-
-
-

6.6

6.6.1

6.6.2

```
[
        ['1', '2', ''],
        ['3', '4', ''],
        ['5', '6', ''],
        ['7', '8', ''],
        ...
]
```

6.6.3

concurrence_graph		
transition_graph		
correct_transition_graph		
correct_co_influence_graph	co_influence	
similarity_graph		

6.6.4 concurrence_graph(graph.con)

,

:

```
[[1, 0], [2, 0], ...] ---> graph[1][2] += 1, graph[2][1] += 1
```

6.6.5 transition_graph(graph.trans)

:

```
[[1, 0], [2, 1], ...] ---> graph[1][2] += 1
```

6.6.6 correct_transition_graph(graph.trans)

:

```
[[1, 1], [2, 1], [3, 0], ...] ---> graph[1][2] += 1
```

6.6.7 correct_co_influence_graph(graph.ccon)

correct_transition_graph Co-influence

```
Co-influence[i][j] = Co-influence[j][i] = (graph[i][j] + graph[j][i]) /
↪|graph[i][j] - graph[i][j]|
```

Co-influence

6.6.8 similarity_graph(graph.sim)

:

```
[[1, 1, 0],            [[1, 0.94280904, 0.85280287],
 [2, 2, 1],   --->      [0.94280904, 1, 0.90453403],
 [3, 1, 1]]             [0.85280287, 0.90453403, 1]]
```

6.7 DataSet

6.7.1 ASSISTments

2009-2010 ASSISTment Skill Builder Data

Data Description

Column Description

Field	Annotation
order id	Non-chronological id, refer to original problem log
assignment id	Each assignment is specific to single teacher/class.
user id	Id of the student
problem id	Id of the problem
original	Main problem or Scaffolding problem
correct	Correct on the fisrt attempt or Incorrect on the first attempt, or asked for help
attempt count	Number of attempts of the student
ms first reponse	The time in the milliseconds for the student's first response
tutor mode	tutor or test
answer type	choose_1 or algebra or fill_in or open_response
sequence id	Id of the problem set
student class id	Class id
position	Assignment position on the class assignments page
problem set type	Linear or Random or Mastery
base sequence id	If the sequence has been copied, this points to the original copy
skill id	ID of the skill associated with the problem. In this skill builder dataset, records will be duplicated so that each record with one skill.
skill name	Name of the skill
teacher id	ID of the teacher
school id	ID of the school
hint count	Number of student attempts
hint total	Number of possible hints on the problem
overlap time	Time in milliseconds
template id	The template ID of the ASSISTment. ASSISTments with the same template ID have similar questions.
answer id	The answer ID for multi-choice questions.
answer text	The answer text for fill-in questions.
first action	The type of first action: attemp or ask for a hint.
bottom hint	Whether or not the student asks for all hints.
opportunity	The number of opportunities the student has to practice on this skill.
opportunity original	The number of opportunities the student has to practice on this skill counting only original problems.

[1]:
```
import numpy as np
import pandas as pd

import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objs as go
```

[2]:
```
path = "ASSISTments2009-2010.csv"

data = pd.read_csv(path, encoding = "ISO-8859-15",low_memory=False)
```

Record Examples

[3]:
```
pd.set_option('display.max_columns', 500)
data.head()
```

[3]:
```
   order_id  assignment_id  user_id  assistment_id  problem_id  original  \
0  33022537         277618    64525          33139       51424         1   
1  33022709         277618    64525          33150       51435         1   
2  35450204         220674    70363          33159       51444         1   
3  35450295         220674    70363          33110       51395         1   
4  35450311         220674    70363          33196       51481         1   

   correct  attempt_count  ms_first_response tutor_mode answer_type  \
0        1              1              32454      tutor     algebra   
1        1              1               4922      tutor     algebra   
2        0              2              25390      tutor     algebra   
3        1              1               4859      tutor     algebra   
4        0             14              19813      tutor     algebra   

   sequence_id  student_class_id  position            type  base_sequence_id  \
0         5948             13241       126  MasterySection              5948   
1         5948             13241       126  MasterySection              5948   
2         5948             11816        22  MasterySection              5948   
3         5948             11816        22  MasterySection              5948   
4         5948             11816        22  MasterySection              5948   

   skill_id        skill_name  teacher_id  school_id  hint_count  hint_total  \
0       1.0  Box and Whisker       22763         73           0           3   
1       1.0  Box and Whisker       22763         73           0           3   
2       1.0  Box and Whisker       22763         73           0           3   
3       1.0  Box and Whisker       22763         73           0           3   
4       1.0  Box and Whisker       22763         73           3           4   

   overlap_time  template_id  answer_id answer_text  first_action  \
0         32454        30799        NaN          26             0   
1          4922        30799        NaN          55             0   
2         42000        30799        NaN          88             0   
3          4859        30059        NaN          41             0   
4        124564        30060        NaN          65             0   

   bottom_hint  opportunity  opportunity_original  
0          NaN            1                   1.0  
1          NaN            2                   2.0  
2          NaN            1                   1.0  
3          NaN            2                   2.0  
4          0.0            3                   3.0  
```

General features

[4]:
```
data.describe()
```

[4]:
```
           order_id  assignment_id        user_id  assistment_id  \
count  4.017560e+05  401756.000000  401756.000000  401756.000000
mean   3.066256e+07  273701.845882   83414.154542   46443.517526
std    5.264886e+06   11338.460017    7417.814021   11832.443427
min    2.022408e+07  217900.000000      14.000000      86.000000
25%    2.660218e+07  266784.000000   78970.000000   37046.000000
50%    3.110513e+07  271629.000000   80111.000000   44498.000000
75%    3.494364e+07  279158.000000   88142.000000   53142.000000
max    3.831020e+07  291503.000000   96299.000000  106210.000000

          problem_id       original        correct  attempt_count  \
count  401756.000000  401756.000000  401756.000000  401756.000000
mean    81117.030011       0.817140       0.642923       1.596417
std     25426.799662       0.386552       0.479139      12.050437
min        83.000000       0.000000       0.000000       0.000000
25%     58467.000000       1.000000       0.000000       1.000000
50%     80734.000000       1.000000       1.000000       1.000000
75%     93102.000000       1.000000       1.000000       1.000000
max    207348.000000       1.000000       1.000000    3824.000000

       ms_first_response    sequence_id  student_class_id       position  \
count       4.017560e+05  401756.000000     401756.000000  401756.000000
mean        4.748464e+04    7284.411088      12919.115222      57.163649
std         3.614590e+05    1497.941072        783.548291      65.215464
min        -7.759575e+06    5870.000000      11644.000000       1.000000
25%         8.518000e+03    5979.000000      12352.000000       9.000000
50%         1.945300e+04    6910.000000      12574.000000      27.000000
75%         4.457825e+04    8032.000000      13241.000000      92.000000
max         8.407692e+07   13362.000000      14415.000000     295.000000

       base_sequence_id       skill_id     teacher_id      school_id  \
count     401756.000000  338001.000000  401756.000000  401756.000000
mean        6786.020985     127.167032   46875.587322    3031.291025
std         1263.359735     120.427518   15892.975481    1830.451486
min         5870.000000       1.000000   11158.000000       1.000000
25%         5968.000000      39.000000   42999.000000    2770.000000
50%         6094.000000      74.000000   45778.000000    2770.000000
75%         7014.000000     279.000000   59882.000000    5056.000000
max        13362.000000     378.000000   69274.000000    9948.000000

          hint_count     hint_total  overlap_time    template_id  \
count  401756.000000  401756.000000  4.017560e+05  401756.000000
mean        0.487470       2.235817  5.964848e+04   39571.335029
std         1.187255       1.804244  3.822188e+05   12679.439926
min         0.000000       0.000000 -7.759575e+06      86.000000
25%         0.000000       0.000000  1.066900e+04   30244.000000
50%         0.000000       3.000000  2.426450e+04   30987.000000
75%         0.000000       4.000000  5.698925e+04   46399.000000
max        10.000000      10.000000  8.407692e+07  106180.000000
```

(continues on next page)

(continued from previous page)

```
          answer_id   first_action   bottom_hint    opportunity  \
count   45454.000000  401756.000000  67044.000000  401756.000000
mean   145094.431667       0.130012      0.724092      20.553535
std     47127.478285       0.394099      0.446974      62.523994
min         1.000000       0.000000      0.000000       1.000000
25%    104412.000000       0.000000      0.000000       3.000000
50%    136247.000000       0.000000      1.000000       8.000000
75%    184077.000000       0.000000      1.000000      19.000000
max    323181.000000       2.000000      1.000000    3371.000000

       opportunity_original
count         328291.000000
mean              14.403307
std               62.393684
min                1.000000
25%                3.000000
50%                6.000000
75%               13.000000
max             3371.000000
```

[5]:
```
print("The number of records: "+ str(len(data['order_id'].unique())))
```

```
The number of records: 346860
```

[6]:
```
print('Part of missing values for every column')
print(data.isnull().sum() / len(data))
```

```
Part of missing values for every column
order_id                0.000000
assignment_id           0.000000
user_id                 0.000000
assistment_id           0.000000
problem_id              0.000000
original                0.000000
correct                 0.000000
attempt_count           0.000000
ms_first_response       0.000000
tutor_mode              0.000000
answer_type             0.000000
sequence_id             0.000000
student_class_id        0.000000
position                0.000000
type                    0.000000
base_sequence_id        0.000000
skill_id                0.158691
skill_name              0.189466
teacher_id              0.000000
school_id               0.000000
hint_count              0.000000
hint_total              0.000000
overlap_time            0.000000
```

(continues on next page)

(continued from previous page)

```
template_id              0.000000
answer_id                0.886862
answer_text              0.222045
first_action             0.000000
bottom_hint              0.833123
opportunity              0.000000
opportunity_original     0.182860
dtype: float64
```

[7]:
```
len(data.user_id.unique())
```

[7]:
```
4217
```

[8]:
```
ds = data['user_id'].value_counts().reset_index()

ds.columns = [
    'user_id',
    'count'
]

ds['user_id'] = ds['user_id'].astype(str) + '-'
ds = ds.sort_values(['count']).tail(40)

fig = px.bar(
    ds,
    x = 'count',
    y = 'user_id',
    orientation='h',
    title='Top 40 students by number of actions'
)

fig.show("svg")
```

[9]:
```
ds = data['user_id'].value_counts().reset_index()

ds.columns = [
    'user_id',
    'count'
]

ds = ds.sort_values('user_id')

fig = px.histogram(
    ds,
    x = 'user_id',
    y = 'count',
    title = 'User action distribution'
)

fig.show("svg")
```

[10]:
```
ds = data['problem_id'].value_counts().reset_index()

ds.columns = [
    'problem_id',
    'count'
]

ds['problem_id'] = ds['problem_id'].astype(str) + '-'
ds = ds.sort_values(['count']).tail(40)

fig = px.bar(
    ds,
    x = 'count',
    y = 'problem_id',
    orientation = 'h',
    title = 'Top 40 useful problem_ids'
)

fig.show("svg")
```

[11]:
```
ds = data['problem_id'].value_counts().reset_index()

ds.columns = [
    'problem_id',
    'count'
]

ds = ds.sort_values('problem_id')

fig = px.histogram(
    ds,
    x='problem_id',
    y='count',
    title='problem_id action distribution'
)

fig.show("svg")
```

[12]:
```
ds = data['correct'].value_counts().reset_index()

ds.columns = [
    'correct',
    'percent'
]

ds['percent'] /= len(data)
ds = ds.sort_values(['percent'])

fig = px.pie(
    ds,
```

(continues on next page)

(continued from previous page)

```
    names = ['wrong', 'right'],
    values = 'percent',
    title = 'Percent of correct answers'
)

fig.show("svg")
```

Sort by answer types

[13]:
```
ds = data['answer_type'].value_counts().reset_index()

ds.columns = [
    'answer_type',
    'percent'
]

ds['percent'] /= len(data)
ds = ds.sort_values(['percent'])

fig = px.pie(
    ds,
    names = 'answer_type',
    values = 'percent',
    title = 'Problem type',
)

fig.show("svg")
```

[14]:
```
fig = make_subplots(rows=3, cols=2)

traces = [
    go.Bar(
        x = ['wrong', 'right'],
        y = [
            len(data[(data['answer_type'] == item) & (data['correct'] == 0)]),
            len(data[(data['answer_type'] == item) & (data['correct'] == 1)])
        ],
        name = 'Type: ' + str(item),
        text = [
            str(round(100*len(data[(data['answer_type'] == item)&(data['correct'] ==␣
↪0)])/len(data[data['answer_type'] == item]),2)) + '%',
            str(round(100*len(data[(data['answer_type'] == item)&(data['correct'] ==␣
↪1)])/len(data[data['answer_type'] == item]),2)) + '%'
        ],
        textposition = 'auto'
    ) for item in data['answer_type'].unique().tolist()
]
```

(continues on next page)

(continued from previous page)

```
for i in range(len(traces)):
    fig.append_trace(
        traces[i],
        (i //2) + 1,
        (i % 2) + 1
    )

fig.update_layout(
    title_text = 'Percent of correct answers for every problem type',
)

fig.show("svg")
```

Sort by schools

[15]:
```
len(data['school_id'].unique())
```

[15]: 75

[16]:
```
ds = data['school_id'].value_counts().reset_index()

ds.columns = [
    'school_id',
    'percent'
]

ds['percent'] /= len(data)
ds = ds.sort_values(['percent'])

fig = px.pie(
    ds,
    names = 'school_id',
    values = 'percent',
    title = 'Percent of schools',
)

fig.show("svg")
```

[17]:
```
ds = data['school_id'].value_counts().reset_index()

ds.columns = [
    'school_id',
    'count'
]

ds['school_id'] = ds['school_id'].astype(str) + '-'
```

(continues on next page)

(continued from previous page)

```
ds = ds.sort_values(['count']).tail(20)

fig = px.bar(
    ds,
    x = 'count',
    y = 'school_id',
    orientation = 'h',
    title = 'Top 20 useful school_ids'
)

fig.show("svg")
```

Sort by attemp counts

[18]:
```
ds = data['attempt_count'].value_counts().reset_index()

ds.columns = [
    'attempt_count',
    'count'
]

ds['attempt_count'] = ds['attempt_count'].astype(str) + '-'
ds = ds.sort_values(['count']).tail(40)

fig = px.bar(
    ds,
    x = 'count',
    y = 'attempt_count',
    orientation = 'h',
    title = 'Top 20 often attempt count'
)

fig.show("svg")
```

Sort by skills

[19]:
```
ds = data['skill_id'].dropna() # There are less NaNs in 'skill_id' column than 'skill_
↪name' column.
ds = ds.value_counts().reset_index()

ds.columns = [
    'skill_id',
    'count'
]
```

(continues on next page)

(continued from previous page)

```
ds['skill_id'] = ds['skill_id'].astype(str) + '-'
ds = ds.sort_values(['count']).tail(40)

fig = px.bar(
    ds,
    x = 'count',
    y = 'skill_id',
    orientation = 'h',
    title = 'Top 40 useful skill_id'
)

fig.show("svg")
```

[ ]:

```

```

ASSISTments2012-2013 Data Analysis

Data Description

Column Description

Field	Annotation
problem_log_id	Unique ID of the logged actions.
skill	Skill name associated with the problem (different skills are in different rows).
problem_id	The ID of the problem
user_id	The ID of the student doing the problem
assignment_id	Two different assignments can have the same sequence id
assistment_id	The ID of the ASSISTment
start_time	Timestamp when the problem starts
end_time	Timestamp when the problem ends
problem_type	choose_1algebrafill_in or open_response
original	Main problem or Scaffolding problem
correct	Correct on the fisrt attempt or Incorrect on the first attempt, or asked for help
bottom_hint	Whether or not the student asks for all hints
hint_count	Number of hints on this problem
actions	Every action on this problem
attempt_count	Number of student attempts on this problem
ms_first_response	The time in milliseconds for the student's first response
tutor_mode	tutor, test mode, pre-test, or post-test
sequence_id	The content id of the problem set
student_class_id	The class ID
position	Assignment position on the class assignments page
type	This is the type of the head section of the problem set
base_sequence_id	This is to account for if a sequence has been copied
skill_id	ID of the skill associated with the problem (different skills are in different rows)
teacher_id	The ID of the teacher who assigned the problem

continues on next page

Table 1 – continued from previous page

Field	Annotation
school_id	The ID of the school where the problem was assigned
overlap_time	The time in milliseconds for the student's overlap time
template_id	The template ID of the ASSISTments
answer_id	The answer ID for multi-choice questions
answer_text	The answer text for fill-in questions
first_action	The type of first action: attempt or ask for a hint
problemlog_id	Unique ID of the logged actions
Average_confidence(FRUSTRATED)	Predicted Frustration of student for the problem
Average_confidence(CONFUSED)	Predicted Confusion of student for the problem
Average_confidence(CONCENTRATING)	Predicted Engaged Concentration of student for the problem
Average_confidence(BORED)	Predicted Boredom of student for the problem

[26]:
```
import pandas as pd

import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objs as go
```

[28]:
```
path = "2012-2013-data-with-predictions-4-final.csv"

data = pd.read_csv(path, encoding = "ISO-8859-15",low_memory=False)
```

Record Examples

[29]:
```
pd.set_option('display.max_columns', 500)
data.head()
```

[29]:
```
   problem_log_id                             skill  problem_id  user_id  \
0       137792159                               NaN      557460    61394
1       138083797                          Rounding      365981    61394
2       142332619  Multiplication and Division Integers      426415    61394
3       145939397                        Proportion       86686    61394
4       137111284                               NaN      399669    76592

   assignment_id  assistment_id           start_time                 end_time  \
0         565736         341511  2012-09-28 15:11:27  2012-09-28 15:11:36.856
1         573819         204043  2012-10-09 11:01:52  2012-10-09 11:02:13.182
2         734130         247525  2013-03-07 10:53:20  2013-03-07 10:53:28.661
3         821352          48081  2013-08-20 19:54:56  2013-08-20 19:55:21.753
4         557216         227869  2012-09-10 17:20:10  2012-09-10 17:24:56.579

  problem_type  original  correct  bottom_hint  hint_count  \
0     choose_1         1      1.0          0.0           0
1      algebra         1      1.0          0.0           0
2      algebra         1      0.0          0.0           0
3      algebra         1      1.0          0.0           0
4     choose_1         1      1.0          0.0           0

                                             actions  attempt_count  \
```
(continues on next page)

(continued from previous page)

```
0  --- \n- - start\n  - 1348859487561\n  - "95952...           1
1  --- \n- - start\n  - 1349794912269\n  - "62459...           1
2  --- \n- - start\n  - 1362671600405\n  - "74107...           1
3  --- \n- - start\n  - 1377042896503\n  - "73630...           1
4  --- \n- - start\n  - 1347312010563\n  - "69479...           1

   ms_first_response tutor_mode  sequence_id  student_class_id  position  \
0               9852      tutor        55482             23643         4
1              21175      tutor        34221             22967         5
2               8645      tutor        39601             22967        58
3              25728      tutor         6912             26303        21
4             286578      tutor        37143             21696         3

            type  base_sequence_id  skill_id  teacher_id  school_id  \
0  LinearSection             55482       NaN       53472     5048.0
1  LinearSection             34221      54.0       47424     5048.0
2  LinearSection             39601     279.0       47424     5048.0
3 MasterySection              6912      79.0       47424     5048.0
4  LinearSection             37143       NaN      152676     7561.0

   overlap_time  template_id  answer_id        answer_text  first_action  \
0          9852       341511        NaN                she             0
1         21175       204043        NaN              74.29             0
2          8645       247525        NaN                 00             0
3         25728        46362        NaN                3.8             0
4        286578       227869        NaN  C (wr - 1)(wr + 1)             0

   problemlogid  Average_confidence(FRUSTRATED)  Average_confidence(CONFUSED)  \
0     137792159                        0.361323                           0.0
1     138083797                        0.361323                           0.0
2     142332619                        0.361323                           0.0
3     145939397                        0.775000                           0.0
4     137111284                        0.361323                           0.0

   Average_confidence(CONCENTRATING)  Average_confidence(BORED)
0                           0.336529                   0.000000
1                           0.766925                   0.000000
2                           0.766925                   0.442968
3                           0.766925                   0.912281
4                           0.766925                   0.000000
```

General features

[30]:
```
data.describe()
```

[30]:
```
       problem_log_id    problem_id       user_id  assignment_id  \
count    6.123270e+06  6.123270e+06  6.123270e+06   6.123270e+06
mean     1.414932e+08  3.685675e+05  1.770492e+05   6.773074e+05
std      2.693733e+06  2.195421e+05  3.172431e+04   9.425983e+04
min      1.368431e+08  1.000000e+00  2.142100e+04   1.814560e+05
25%      1.391705e+08  1.284030e+05  1.719780e+05   5.863570e+05
50%      1.414916e+08  4.168130e+05  1.791670e+05   6.785645e+05
75%      1.438272e+08  5.644030e+05  1.972510e+05   7.672320e+05
max      1.462357e+08  7.671430e+05  2.282130e+05   8.330540e+05

       assistment_id      original       correct   bottom_hint    hint_count  \
count   6.123270e+06  6.123270e+06  6.123270e+06  6.062922e+06  6.123270e+06
mean    2.202825e+05  9.504296e-01  6.768206e-01  1.200497e-01  3.373479e-01
std     1.393519e+05  2.170557e-01  4.674909e-01  3.250197e-01  9.851956e-01
min     5.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
25%     6.883725e+04  1.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
50%     2.399180e+05  1.000000e+00  1.000000e+00  0.000000e+00  0.000000e+00
75%     3.466830e+05  1.000000e+00  1.000000e+00  0.000000e+00  0.000000e+00
max     4.925890e+05  1.000000e+00  1.000000e+00  1.000000e+00  1.400000e+01

       attempt_count  ms_first_response   sequence_id  student_class_id  \
count   6.123270e+06       6.123270e+06  6.123270e+06      6.123270e+06
mean    1.339212e+00       4.873469e+04  6.689567e+04      2.342511e+04
std     1.056276e+00       2.673557e+05  5.933111e+04      1.612341e+03
min     0.000000e+00      -6.767000e+03  2.000000e+00      1.139300e+04
25%     1.000000e+00       9.436000e+03  1.266200e+04      2.251800e+04
50%     1.000000e+00       2.233100e+04  4.614100e+04      2.314400e+04
75%     1.000000e+00       5.486500e+04  9.901300e+04      2.460300e+04
max     2.900000e+01       3.450552e+08  2.084530e+05      2.738600e+04

           position  base_sequence_id      skill_id    teacher_id  \
count  6.123270e+06      6.123270e+06  2.711813e+06  6.123270e+06
mean   7.402669e+01      6.214174e+04  1.932575e+02  1.210437e+05
std    3.697118e+02      5.687449e+04  1.303155e+02  4.978645e+04
min    0.000000e+00      2.000000e+00  1.000000e+00  1.143600e+04
25%    4.000000e+00      1.189800e+04  6.500000e+01  7.305500e+04
50%    1.200000e+01      4.493100e+04  2.770000e+02  1.285010e+05
75%    3.800000e+01      8.711200e+04  3.100000e+02  1.562940e+05
max    8.533000e+03      2.084530e+05  1.641000e+03  2.205230e+05

          school_id  overlap_time   template_id     answer_id  first_action  \
count  6.123113e+06  6.123270e+06  6.123270e+06  8.275000e+03  6.123270e+06
mean   6.925225e+03  4.907237e+04  2.088952e+05  4.324879e+05  6.151860e-02
std    3.314489e+03  2.884992e+05  1.458227e+05  3.534885e+05  2.635170e-01
min    1.000000e+00 -6.767000e+03  5.000000e+00  1.000000e+00  0.000000e+00
25%    5.260000e+03  9.468000e+03  5.259000e+04  1.060495e+05  0.000000e+00
50%    5.978000e+03  2.241500e+04  2.395460e+05  3.442820e+05  0.000000e+00
75%    9.394000e+03  5.505400e+04  3.434800e+05  7.385615e+05  0.000000e+00
max    1.242800e+04  3.452775e+08  4.925890e+05  1.184706e+06  2.000000e+00
```

(continues on next page)

(continued from previous page)

```
       problemlogid  Average_confidence(FRUSTRATED)  \
count  6.123270e+06                    6.123270e+06
mean   1.414932e+08                    3.894586e-01
std    2.693733e+06                    1.027662e-01
min    1.368431e+08                    3.613230e-01
25%    1.391705e+08                    3.613230e-01
50%    1.414916e+08                    3.613230e-01
75%    1.438272e+08                    3.613230e-01
max    1.462357e+08                    8.671330e-01

       Average_confidence(CONFUSED)  Average_confidence(CONCENTRATING)  \
count                  6.123270e+06                       6.123270e+06
mean                   4.479487e-02                       6.823843e-01
std                    1.924793e-01                       1.713734e-01
min                    0.000000e+00                       1.707320e-01
25%                    0.000000e+00                       7.669250e-01
50%                    0.000000e+00                       7.669250e-01
75%                    0.000000e+00                       7.669250e-01
max                    1.000000e+00                       7.669250e-01

       Average_confidence(BORED)
count               6.123270e+06
mean                2.567723e-01
std                 2.862460e-01
min                 0.000000e+00
25%                 0.000000e+00
50%                 2.214840e-01
75%                 4.429680e-01
max                 1.000000e+00
```

[31]:
```
print("The number of records: "+ str(len(data['problem_log_id'].unique())))
```

```
The number of records: 6123270
```

[32]:
```
print('Part of missing values for every column')
print(data.isnull().sum() / len(data))
```

```
Part of missing values for every column
problem_log_id                        0.000000
skill                                 0.570478
problem_id                            0.000000
user_id                               0.000000
assignment_id                         0.000000
assistment_id                         0.000000
start_time                            0.000000
end_time                              0.000000
problem_type                          0.000000
original                              0.000000
correct                               0.000000
bottom_hint                           0.009856
hint_count                            0.000000
```

(continues on next page)

(continued from previous page)

```
actions                              0.000000
attempt_count                        0.000000
ms_first_response                    0.000000
tutor_mode                           0.000000
sequence_id                          0.000000
student_class_id                     0.000000
position                             0.000000
type                                 0.000000
base_sequence_id                     0.000000
skill_id                             0.557130
teacher_id                           0.000000
school_id                            0.000026
overlap_time                         0.000000
template_id                          0.000000
answer_id                            0.998649
answer_text                          0.056561
first_action                         0.000000
problemlogid                         0.000000
Average_confidence(FRUSTRATED)       0.000000
Average_confidence(CONFUSED)         0.000000
Average_confidence(CONCENTRATING)    0.000000
Average_confidence(BORED)            0.000000
dtype: float64
```

[33]:
```
len(data.user_id.unique())
```

[33]:
```
46674
```

[34]:
```
ds = data['user_id'].value_counts().reset_index()

ds.columns = [
    'user_id',
    'count'
]

ds['user_id'] = ds['user_id'].astype(str) + '-'
ds = ds.sort_values(['count']).tail(40)

fig = px.bar(
    ds,
    x = 'count',
    y = 'user_id',
    orientation='h',
    title='Top 40 students by number of actions'
)

fig.show('svg')
```

[35]:
```
ds = data['user_id'].value_counts().reset_index()

ds.columns = [
```

(continues on next page)

(continued from previous page)

```
    'user_id',
    'count'
]

ds = ds.sort_values('user_id')

fig = px.histogram(
    ds,
    x = 'user_id',
    y = 'count',
    title = 'User action distribution'
)

fig.show('svg')
```

[36]:

```
ds = data['problem_id'].value_counts().reset_index()

ds.columns = [
    'problem_id',
    'count'
]

ds['problem_id'] = ds['problem_id'].astype(str) + '-'
ds = ds.sort_values(['count']).tail(40)

fig = px.bar(
    ds,
    x = 'count',
    y = 'problem_id',
    orientation = 'h',
    title = 'Top 40 useful problem_ids'
)

fig.show('svg')
```

[37]:

```
ds = data['problem_id'].value_counts().reset_index()

ds.columns = [
    'problem_id',
    'count'
]

ds = ds.sort_values('problem_id')

fig = px.histogram(
    ds,
    x='problem_id',
    y='count',
    title='problem_id action distribution'
)
```

(continues on next page)

(continued from previous page)

```
fig.show('svg')
```

[38]:
```
ds = data['correct'].value_counts().reset_index()

ds.columns = [
    'correct',
    'percent'
]

ds['percent'] /= len(data)
ds = ds.sort_values(['correct'])

fig = px.pie(
    ds,
    names = ['0', '0.25','0.375','0.5','0,6','0.625','0.65','0.75','0.85','0.875','0.95',
↪'0.975','1'],
    values = 'percent',
    title = 'Percent of correct answers'
)

fig.show('svg')
"Minor note:we also have Essay questions that teachers can grade. If this value is say 0.
↪25, that means the teacher gave it a 1 out of 4."
```

[38]:
```
'Minor note:we also have Essay questions that teachers can grade. If this value is say 0.
↪25, that means the teacher gave it a 1 out of 4.'
```

Sort by sequence id

[39]:
```
ds = data['sequence_id'].value_counts().reset_index()

ds.columns = [
    'sequence_id',
    'count'
]

ds['sequence_id'] = ds['sequence_id'].astype(str) + '-'
ds = ds.sort_values(['count']).tail(40)

fig = px.bar(
    ds,
    x = 'count',
    y = 'sequence_id',
    orientation = 'h',
    title = 'Top 40 useful sequence_ids'
)
```

(continues on next page)

(continued from previous page)

```
fig.show('svg')
```

sort by problem types

[40]:
```
ds = data['problem_type'].value_counts().reset_index()

ds.columns = [
    'problem_type',
    'percent'
]

ds['percent'] /= len(data)
ds = ds.sort_values(['percent'])

fig = px.pie(
    ds,
    names = 'problem_type',
    values = 'percent',
    title = 'Percent of Problem Types',
)

fig.show('svg')
```

[41]:
```
ds = ds.sort_values(['percent']).tail(6)

fig = make_subplots(rows=3, cols=2)

traces = [
    go.Bar(
        x = ['wrong', 'right'],
        y = [
            len(data[(data['problem_type'] == item) & (data['correct'] == 0)]),
            len(data[(data['problem_type'] == item) & (data['correct'] == 1)])
        ],
        name = 'Type: ' + str(item),
        text = [
            str(round(100*len(data[(data['problem_type'] == item)&(data['correct'] ==␣
↪0)])/len(data[data['problem_type'] == item]),2)) + '%',
            str(round(100*len(data[(data['problem_type'] == item)&(data['correct'] ==␣
↪1)])/len(data[data['problem_type'] == item]),2)) + '%'
        ],
        textposition = 'auto'
    ) for item in ds['problem_type'].unique().tolist()
]

for i in range(len(traces)):
    fig.append_trace(
        traces[i],
```

(continues on next page)

(continued from previous page)

```
        (i //2) + 1,
        (i % 2) + 1
    )

fig.update_layout(
    title_text = 'Percent of correct answers for every problem type',
)

fig.show('svg')
```

Sort by schools

[42]:
```
len(data['school_id'].unique())
```

[42]: 662

[43]:
```
ds = data['school_id'].value_counts().reset_index()

ds.columns = [
    'school_id',
    'percent'
]

ds['percent'] /= len(data)
ds = ds.sort_values(['percent'])

fig = px.pie(
    ds,
    names = 'school_id',
    values = 'percent',
    title = 'Percent of schools',
)

fig.show('svg')
```

[44]:
```
ds = data['school_id'].value_counts().reset_index()

ds.columns = [
    'school_id',
    'count'
]

ds['school_id'] = ds['school_id'].astype(str) + '-'
ds = ds.sort_values(['count']).tail(20)

fig = px.bar(
```

(continues on next page)

(continued from previous page)

```
    ds,
    x = 'count',
    y = 'school_id',
    orientation = 'h',
    title = 'Top 20 useful school_ids'
)

fig.show('svg')
```

Sort by attemp counts

[45]:
```
ds = data['attempt_count'].value_counts().reset_index()

ds.columns = [
    'attempt_count',
    'count'
]

ds['attempt_count'] = ds['attempt_count'].astype(str) + '-'
ds = ds.sort_values(['count']).tail(40)

fig = px.bar(
    ds,
    x = 'count',
    y = 'attempt_count',
    orientation = 'h',
    title = 'Top 20 often attempt count'
)

fig.show('svg')
```

Sort by skills

[46]:
```
ds = data['skill_id'].dropna() # There are less NaNs in 'skill_id' column than 'skill_
↪name' column.
ds = ds.value_counts().reset_index()

ds.columns = [
    'skill_id',
    'count'
]

ds['skill_id'] = ds['skill_id'].astype(str) + '-'
ds = ds.sort_values(['count']).tail(40)
```

(continues on next page)

(continued from previous page)

```
fig = px.bar(
    ds,
    x = 'count',
    y = 'skill_id',
    orientation = 'h',
    title = 'Top 40 useful skill_id'
)

fig.show('svg')
```

[ ]:

```

```

ASSISTments2015 Data Analysis

Data Description

Column Description

Field	Annotation
user id	Id of the student
log id	Unique ID of the logged actions
sequence id	Id of the problem set
correct	Correct on the fisrt attempt or Incorrect on the first attempt, or asked for help

[1]:

```
import numpy as np
import pandas as pd

import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objs as go
```

[2]:

```
path = "2015_100_skill_builders_main_problems.csv"
data = pd.read_csv(path, encoding = "ISO-8859-15",low_memory=False)
```

Record Examples

[3]:

```
pd.set_option('display.max_columns', 500)
data.head()
```

[3]:

	user_id	log_id	sequence_id	correct
0	50121	167478035	7014	0.0
1	50121	167478043	7014	1.0
2	50121	167478053	7014	1.0
3	50121	167478069	7014	1.0
4	50964	167478041	7014	1.0

General features

[4]:
```
data.describe()
```

[4]:
```
             user_id        log_id    sequence_id        correct
count  708631.000000  7.086310e+05  708631.000000  708631.000000
mean   296232.978276  1.695323e+08   22683.474821       0.725502
std     48018.650247  3.608096e+06   41593.028018       0.437467
min     50121.000000  1.509145e+08    5898.000000       0.000000
25%    279113.000000  1.660355e+08    7020.000000       0.000000
50%    299168.000000  1.704579e+08    9424.000000       1.000000
75%    335647.000000  1.723789e+08   14442.000000       1.000000
max    362374.000000  1.754827e+08  236309.000000       1.000000
```

[5]:
```
print("The number of records: "+ str(len(data['log_id'].unique())))
```

```
The number of records: 708631
```

[6]:
```
print('Part of missing values for every column')
print(data.isnull().sum() / len(data))
```

```
Part of missing values for every column
user_id        0.0
log_id         0.0
sequence_id    0.0
correct        0.0
dtype: float64
```

[7]:
```
len(data.user_id.unique())
```

[7]:
```
19917
```

[8]:
```
len(data.sequence_id.unique())
```

[8]:
```
100
```

Sort by user id

[9]:
```
ds = data['user_id'].value_counts().reset_index()

ds.columns = [
    'user_id',
    'count'
]

ds['user_id'] = ds['user_id'].astype(str) + '-'
ds = ds.sort_values(['count']).tail(40)

fig = px.bar(
    ds,
    x = 'count',
    y = 'user_id',
```

(continues on next page)

(continued from previous page)

```
    orientation='h',
    title='Top 40 students by number of actions'
)

fig.show("svg")
```

[10]:
```
ds = data['user_id'].value_counts().reset_index()

ds.columns = [
    'user_id',
    'count'
]

ds = ds.sort_values('user_id')

fig = px.histogram(
    ds,
    x = 'user_id',
    y = 'count',
    title = 'User action distribution'
)

fig.show("svg")
```

Correct answers

[11]:
```
ds = data['correct'].value_counts().reset_index()

ds.columns = [
    'correct',
    'percent'
]

ds['percent'] /= len(data)
ds = ds.sort_values(['correct'])

fig = px.pie(
    ds,
    names = ['0', '1/10','1/5', '1/4','1/3', '1/2','2/3', '3/4','4/5','9/10', '1'],
    values = 'percent',
    title = 'Percent of correct answers'
)

fig.show("svg")
```

Minor note: we also have Essay questions that teachers can grade. If this value is say .25 that means the teacher gave it a 1 our of 4.

Sort by sequence id

[12]:
```
ds = data['sequence_id'].value_counts().reset_index()

ds.columns = [
    'sequence_id',
    'count'
]

ds['sequence_id'] = ds['sequence_id'].astype(str) + '-'
ds = ds.sort_values(['count']).tail(40)

fig = px.bar(
    ds,
    x = 'count',
    y = 'sequence_id',
    orientation = 'h',
    title = 'Top 40 useful sequence_ids'
)

fig.show("svg")
```

[13]:
```
ds = data.groupby('sequence_id')['correct'].mean()
ds = ds.reset_index()

ds['sequence_id'] = ds['sequence_id'].astype(str) + '-'
ds1 = ds.sort_values(['correct']).tail(20)

fig1 = px.bar(
    ds1,
    x = 'correct',
    y = 'sequence_id',
    orientation = 'h',
    title = 'Average number correct answers of problem sets (top 20)'
)

fig1.show("svg")

ds2 = ds.sort_values(['correct']).head(20)

fig2 = px.bar(
    ds2,
    x = 'correct',
    y = 'sequence_id',
    orientation = 'h',
    title = 'Average number correct answers of problem sets (bottom 20)'
)
fig2.show("svg")
```

This figure presents the average number correct answers of problem sets. These low-average problem sets deserve more attention from teachers and students.

ASSISTments2017 Data Analysis

Data Description

Column Description

Field	Annotation
student id	a deidentified ID/tag used for identifying an individual student
SY ASSISTments Usage	the academic years the student used ASSISTments
AveKnow	average student knowledge level (according to Bayesian Knowledge Tracing algorithm – cf.
AveCarelessness	average student carelessness (according to San Pedro, Baker, & Rodrigo, 2011 model)
AveCorrect	average student correctness
NumActions	total number of student actions in system
AveResBored	average student affect: boredom (see Pardos, Baker, San Pedro, Gowda, & Gowda, 2014)
AveResEngcon	average student affect:engaged concentration (see Pardos, Baker, San Pedro, Gowda, & Gov
AveResConf	average student affect:confusion (see Pardos, Baker, San Pedro, Gowda, & Gowda, 2014)
AveResFrust	average student affect:frustration (see Pardos, Baker, San Pedro, Gowda, & Gowda, 2014)
AveResOfftask	average student affect: off task (see Pardos, Baker, San Pedro, Gowda, & Gowda, 2014 and
AveResGaming	average student affect:gaming the system (see Pardos, Baker, San Pedro, Gowda, & Gowda,
actionId	the unique id of this specific action
skill	a tag used for identifying the cognitive skill related to the problem (see Razzaq, Heffernan,
problemId	a unique ID used for identifying a single problem
assignmentId	a unique ID used for identifying an assignment
assistmentId	a unique ID used for identifying an assistment (a instance of a multi-part problem)
startTime	when did the student start the problem (UNIX time, seconds)
endTime	when did the student end the problem (UNIX time, seconds)
timeTaken	Time spent on the current step
correct	Answer is correct
original	Problem is original not a scaffolding problem
hint	Action is a hint response
hintCount	Total number of hints requested so far
hintTotal	total number of hints requested for the problem
scaffold	Problem is a scaffolding problem
bottomHint	Bottom-out hint is used
attemptCount	Total problems attempted in the tutor so far.
problemType	the type of the problem
frIsHelpRequest	First response is a help request
frPast5HelpRequest	Number of last 5 First responses that included a help request
frPast8HelpRequest	Number of last 8 First responses that included a help request
stlHintUsed	Second to last hint is used an indicates a hint that gives considerable detail but is not quite b
past8BottomOut	Number of last 8 problems that used the bottom-out hint.
totalFrPercentPastWrong	Percent of all past problems that were wrong on this KC.
totalFrPastWrongCount	Total first responses wrong attempts in the tutor so far.
frPast5WrongCount	Number of last 5 First responses that were wrong
frPast8WrongCount	Number of last 8 First responses that were wrong
totalFrTimeOnSkill	Total first response time spent on this KC across all problems
timeSinceSkill	Time since the current KC was last seen.
frWorkingInSchool	First response Working during school hours (between 7:00 am and 3:00 pm)
totalFrAttempted	Total first responses attempted in the tutor so far.
totalFrSkillOpportunities	Total first response practice opportunities on this KC so far.

Table 2 – continued from previous p

Field	Annotation
responseIsFillIn	Response is filled in (No list of answers available)
responseIsChosen	Response is chosen from a list of answers (Multiple choice, etc).
endsWithScaffolding	Problem ends with scaffolding
endsWithAutoScaffolding	Problem ends with automatic scaffolding
frTimeTakenOnScaffolding	First response time taken on scaffolding problems
frTotalSkillOpportunitiesScaffolding	Total first response practice opportunities on this skill so far
totalFrSkillOpportunitiesByScaffolding	Total first response scaffolding opportunities for this KC so far
frIsHelpRequestScaffolding	First response is a help request Scaffolding
timeGreater5Secprev2wrong	Long pauses after 2 Consecutive wrong answers
sumRight	NaN
helpAccessUnder2Sec	Time spent on help was under 2 seconds
timeGreater10SecAndNextActionRight	Long pause after correct answer
consecutiveErrorsInRow	Total number of 2 wrong answers in a row across all the problems
sumTime3SDWhen3RowRight	NaN
sumTimePerSkill	NaN
totalTimeByPercentCorrectForskill	Total time spent on this KC across all problems divided by percent correct for the same KC
prev5count	NaN
timeOver80	NaN
manywrong	NaN
confidence(BORED)	the confidence of the student affect prediction: bored
confidence(CONCENTRATING)	the confidence of the student affect prediction: concecntrating
confidence(CONFUSED)	the confidence of the student affect prediction: confused
confidence(FRUSTRATED)	the confidence of the student affect prediction: frustrated
confidence(OFF TASK)	the confidence of the student affect prediction: off task
confidence(GAMING)	the confidence of the student affect prediction: gaming
RES_BORED	rescaled of the confidence of the student affect prediction: boredom
RES_CONCENTRATING	rescaled of the confidence of the student affect prediction: concentration
RES_CONFUSED	rescaled of the confidence of the student affect prediction: confusion
RES_FRUSTRATED	rescaled of the confidence of the student affect prediction: frustration
RES_OFFTASK	rescaled of the confidence of the student affect prediction: off task
RES_GAMING	rescaled of the confidence of the student affect prediction: gaming
Ln-1	baysian knowledge tracing's knowledge estimate at the previous time step
Ln	baysian knowledge tracing's knowledge estimate at the time step
schoolID	the id (anonymized) of the school the student was in during the year the data was collected
MCAS	Massachusetts Comprehensive Assessment System test score. In short, this number is the st

[1]:
```
import numpy as np
import pandas as pd

import plotly.express as asz
from plotly.subplots import make_subplots
import plotly.graph_objs as go
```

[70]:
```
path = "anonymized_full_release_competition_dataset.csv"
data = pd.read_csv(path, encoding = "ISO-8859-15",low_memory=False)
```

[36]:
```
pd.set_option('display.max_columns', 500)
data.head()
```

[36]:
```
   studentId  MiddleSchoolId InferredGender SY ASSISTments Usage   AveKnow  \
0          8               2           Male           2004-2005  0.352416
1          8               2           Male           2004-2005  0.352416
2          8               2           Male           2004-2005  0.352416
3          8               2           Male           2004-2005  0.352416
4          8               2           Male           2004-2005  0.352416

   AveCarelessness  AveCorrect  NumActions  AveResBored  AveResEngcon  \
0         0.183276    0.483902        1056     0.208389      0.679126
1         0.183276    0.483902        1056     0.208389      0.679126
2         0.183276    0.483902        1056     0.208389      0.679126
3         0.183276    0.483902        1056     0.208389      0.679126
4         0.183276    0.483902        1056     0.208389      0.679126

   AveResConf  AveResFrust  AveResOfftask  AveResGaming  action_num  \
0    0.115905     0.112408       0.156503      0.196561        9950
1    0.115905     0.112408       0.156503      0.196561        9951
2    0.115905     0.112408       0.156503      0.196561        9952
3    0.115905     0.112408       0.156503      0.196561        9953
4    0.115905     0.112408       0.156503      0.196561        9954

                                     skill  problemId        problemType  \
0         properties-of-geometric-figures       1118  textfieldquestion
1         properties-of-geometric-figures       1119         noprobtype
2  sum-of-interior-angles-more-than-3-sides       1120         noprobtype
3  sum-of-interior-angles-more-than-3-sides       1120         noprobtype
4  sum-of-interior-angles-more-than-3-sides       1121         noprobtype

   assignmentId  assistmentId   startTime     endTime  timeTaken  correct  \
0      20405010     104051118  1096470301  1096470350       49.0        0
1      20405010     104051119  1096470350  1096470354        4.0        1
2      20405010     104051120  1096470354  1096470360        6.0        0
3      20405010     104051120  1096470360  1096470378       18.0        0
4      20405010     104051121  1096470378  1096470380        2.0        1

   original  hint  hintCount  hintTotal  scaffold  bottomHint  attemptCount  \
0         1     1          1          1         0           0             1
1         0     0          0          0         1           0             1
2         0     0          0          0         0           0             1
3         0     0          0          0         0           0             2
4         0     0          0          1         0           0             1

   frIsHelpRequest  frPast5HelpRequest  frPast8HelpRequest  stlHintUsed  \
0                1                   0                   0            0
1                1                   1                   1            0
2                0                   0                   0            0
3                0                   0                   0            0
4                0                   0                   0            0

   past8BottomOut  totalFrPercentPastWrong  totalFrPastWrongCount  \
0               0                      0.0                      0
1               0                      0.0                      0
2               0                      0.0                      0
```

(continues on next page)

(continued from previous page)

```
3              0                     0.0                      1
4              0                     1.0                      1

   frPast5WrongCount  frPast8WrongCount  totalFrTimeOnSkill  timeSinceSkill  \
0                  0                  0                 0.0             0.0
1                  0                  0                49.0             0.0
2                  0                  0                 0.0             0.0
3                  0                  0                 0.0             0.0
4                  1                  1                 6.0             0.0

   frWorkingInSchool  totalFrAttempted  totalFrSkillOpportunities  \
0                  1                 0                          0
1                  1                 1                          1
2                  1                 2                          0
3                  1                 3                          1
4                  1                 3                          1

   responseIsFillIn  responseIsChosen  endsWithScaffolding  \
0                 0                 0                    0
1                 0                 0                    1
2                 0                 0                    0
3                 0                 0                    0
4                 0                 0                    0

   endsWithAutoScaffolding  frTimeTakenOnScaffolding  \
0                        0                       0.0
1                        0                       4.0
2                        0                       6.0
3                        0                       6.0
4                        0                       2.0

   frTotalSkillOpportunitiesScaffolding  \
0                                     0
1                                     0
2                                     0
3                                     1
4                                     1

   totalFrSkillOpportunitiesByScaffolding  frIsHelpRequestScaffolding  \
0                                     0.0                           0
1                                     0.0                           1
2                                     0.0                           0
3                                     0.0                           0
4                                     1.0                           0

   timeGreater5Secprev2wrong  sumRight  helpAccessUnder2Sec  \
0                          0         0                    0
1                          0         1                    0
2                          0         1                    0
3                          0         1                    0
4                          0         2                    0
```

(continues on next page)

(continued from previous page)

```
   timeGreater10SecAndNextActionRight  consecutiveErrorsInRow  \
0                                   0                       0
1                                   1                       0
2                                   0                       0
3                                   0                       1
4                                   1                       0

   sumTime3SDWhen3RowRight  sumTimePerSkill  \
0                      0.0             49.0
1                      0.0             53.0
2                      0.0              6.0
3                      0.0             24.0
4                      0.0             26.0

   totalTimeByPercentCorrectForskill  Prev5count  timeOver80  manywrong  \
0                           0.000000           0           0          0
1                         106.000000           1           0          0
2                           0.000000           2           0          0
3                           0.000000           3           0          0
4                          77.999999           4           0          1

   confidence(BORED)  confidence(CONCENTRATING)  confidence(CONFUSED)  \
0           0.597865                   0.234294                0.0000
1           0.355694                   0.992585                0.9375
2           0.355694                   0.992585                0.9375
3           0.355694                   0.617065                0.0000
4           0.355694                   0.617065                0.0000

   confidence(FRUSTRATED)  confidence(OFF TASK)  confidence(GAMING)  \
0                     0.0              0.838710            0.008522
1                     0.0              0.600000            0.047821
2                     0.0              0.600000            0.047821
3                     0.0              0.204082            0.343996
4                     0.0              0.204082            0.343996

   RES_BORED  RES_CONCENTRATING  RES_CONFUSED  RES_FRUSTRATED  RES_OFFTASK  \
0   0.376427           0.320317      0.000000             0.0     0.785585
1   0.156027           0.995053      0.887452             0.0     0.468252
2   0.156027           0.995053      0.887452             0.0     0.468252
3   0.156027           0.744520      0.000000             0.0     0.108417
4   0.156027           0.744520      0.000000             0.0     0.108417

   RES_GAMING         Ln-1           Ln  MCAS  Enrolled  Selective  isSTEM
0    0.000264         0.13  0.061190409    45         0          0     NaN
1    0.001483  0.061190409  0.213509945    45         0          0     NaN
2    0.001483        0.116  0.033305768    45         0          0     NaN
3    0.010665        0.116  0.033305768    45         0          0     NaN
4    0.010665  0.033305768  0.118385889    45         0          0     NaN
```

General features

[23]:
```
data.describe()
```

[23]:
```
          studentId  MiddleSchoolId         AveKnow  AveCarelessness  \
count 942816.000000   942816.000000   942816.000000    942816.000000
mean    3844.844105        2.515472        0.195155         0.109436
std     2250.484065        1.039785        0.116451         0.059952
min        8.000000        1.000000        0.028057         0.007801
25%     1952.000000        2.000000        0.110542         0.068760
50%     3766.000000        2.000000        0.159285         0.094513
75%     5781.000000        4.000000        0.247704         0.137316
max     7783.000000        4.000000        0.752498         0.430576

          AveCorrect      NumActions     AveResBored    AveResEngcon  \
count 942816.000000   942816.000000   942816.000000   942816.000000
mean       0.372681      869.850594        0.232949        0.658442
std        0.107367      530.210725        0.030637        0.027440
min        0.000000        2.000000        0.170871        0.403309
25%        0.294989      478.000000        0.209035        0.642060
50%        0.345575      754.000000        0.230394        0.660669
75%        0.428822     1151.000000        0.252082        0.676588
max        0.932990     3057.000000        0.440870        0.723990

          AveResConf    AveResFrust   AveResOfftask    AveResGaming  \
count 942816.000000   942816.000000   942816.000000   942816.000000
mean       0.098940        0.131406        0.172212        0.192703
std        0.034788        0.038875        0.057992        0.153455
min        0.005075        0.000000        0.083167        0.001974
25%        0.076385        0.107278        0.131467        0.060724
50%        0.096357        0.127504        0.159598        0.156245
75%        0.119282        0.150582        0.198533        0.298914
max        0.402483        0.543463        0.837402        0.709200

         action_num      problemId  assignmentId  assistmentId     startTime  \
count  9.428160e+05  942816.000000  9.428160e+05  9.428160e+05  9.428160e+05
mean   1.849329e+06    1899.719319  1.198773e+07  6.061572e+07  1.120793e+09
std    1.726001e+06    2579.212724  1.434706e+07  5.128829e+07  1.940359e+07
min    9.950000e+03       1.000000  2.000000e+00  5.000000e+00  1.095421e+09
25%    7.221708e+05     721.000000  7.230000e+02  2.213000e+03  1.103136e+09
50%    9.578745e+05    1116.000000  2.040501e+07  1.040504e+08  1.112980e+09
75%    2.722813e+06    1419.000000  2.040510e+07  1.040511e+08  1.138371e+09
max    6.355811e+06   22761.000000  1.000000e+09  1.040515e+08  1.180218e+09

            endTime      timeTaken        correct       original  \
count  9.428160e+05  942816.000000  942816.000000  942816.000000
mean   1.120793e+09      29.747869       0.372681       0.264214
std    1.940354e+07      72.019768       0.483519       0.440914
min    1.095421e+09       0.000000       0.000000       0.000000
25%    1.103136e+09       5.000000       0.000000       0.000000
50%    1.112980e+09      11.000000       0.000000       0.000000
75%    1.138371e+09      30.000000       1.000000       1.000000
max    1.180218e+09    9999.000000       1.000000       1.000000
```

(continues on next page)

(continued from previous page)

```
             hint      hintCount      hintTotal       scaffold  \
count  942816.000000  942816.000000  942816.000000  942816.000000
mean        0.331025       1.218490       1.953967       0.385732
std         0.470582       1.980665       2.929242       0.486768
min         0.000000       0.000000       0.000000       0.000000
25%         0.000000       0.000000       0.000000       0.000000
50%         0.000000       0.000000       1.000000       0.000000
75%         1.000000       2.000000       3.000000       1.000000
max         1.000000      56.000000      56.000000       1.000000

          bottomHint   attemptCount  frIsHelpRequest  frPast5HelpRequest  \
count  942816.000000  942816.000000    942816.000000       942816.000000
mean        0.062794       2.673605         0.268104            1.947322
std         0.242592       2.929801         0.442972            1.712580
min         0.000000       1.000000         0.000000            0.000000
25%         0.000000       1.000000         0.000000            0.000000
50%         0.000000       2.000000         0.000000            2.000000
75%         0.000000       3.000000         1.000000            3.000000
max         1.000000      91.000000         1.000000            5.000000

       frPast8HelpRequest    stlHintUsed  past8BottomOut  \
count       942816.000000  942816.000000   942816.000000
mean             2.575323       0.004012        0.241678
std              2.457799       0.063217        0.674613
min              0.000000       0.000000        0.000000
25%              0.000000       0.000000        0.000000
50%              2.000000       0.000000        0.000000
75%              4.000000       0.000000        0.000000
max              8.000000       1.000000        8.000000

       totalFrPercentPastWrong  totalFrPastWrongCount  frPast5WrongCount  \
count            942816.000000          942816.000000      942816.000000
mean                  0.227882               1.988008           0.719380
std                   0.271404               3.390149           0.832699
min                   0.000000               0.000000           0.000000
25%                   0.000000               0.000000           0.000000
50%                   0.166667               1.000000           1.000000
75%                   0.333333               2.000000           1.000000
max                   1.000000              73.000000           5.000000

       frPast8WrongCount  totalFrTimeOnSkill  timeSinceSkill  \
count      942816.000000       942816.000000    9.428160e+05
mean            0.944750          376.213405    4.850802e+05
std             1.076276          689.302924    2.075599e+06
min             0.000000            0.000000   -9.928014e+06
25%             0.000000           37.000000    0.000000e+00
50%             1.000000          140.000000    0.000000e+00
75%             1.000000          396.000000    8.000000e+00
max             8.000000        11663.000000    4.840000e+07

       frWorkingInSchool  totalFrAttempted  totalFrSkillOpportunities  \
```

(continues on next page)

(continued from previous page)

```
count     942816.000000     942816.000000               942816.000000
mean           0.974588        193.316005                    8.381019
std            0.157373        164.898869                   11.998292
min            0.000000          0.000000                    0.000000
25%            1.000000         67.000000                    2.000000
50%            1.000000        151.000000                    4.000000
75%            1.000000        276.000000                   10.000000
max            1.000000       1378.000000                  221.000000

       responseIsFillIn  responseIsChosen  endsWithScaffolding  \
count     942816.000000          942816.0        942816.000000
mean           0.023131               0.0             0.636085
std            0.150319               0.0             0.481125
min            0.000000               0.0             0.000000
25%            0.000000               0.0             0.000000
50%            0.000000               0.0             1.000000
75%            0.000000               0.0             1.000000
max            1.000000               0.0             1.000000

       endsWithAutoScaffolding  frTimeTakenOnScaffolding  \
count            942816.000000             942816.000000
mean                  0.005724                 24.060853
std                   0.075443                 71.665655
min                   0.000000                  0.000000
25%                   0.000000                  0.000000
50%                   0.000000                  8.000000
75%                   0.000000                 23.000000
max                   1.000000               9999.000000

       frTotalSkillOpportunitiesScaffolding  \
count                         942816.000000
mean                               3.989371
std                                6.581897
min                                0.000000
25%                                0.000000
50%                                2.000000
75%                                5.000000
max                              105.000000

       totalFrSkillOpportunitiesByScaffolding  frIsHelpRequestScaffolding  \
count                           942816.000000               942816.000000
mean                                 1.036189                    0.670047
std                                  1.184489                    0.470196
min                                  0.000000                    0.000000
25%                                  0.000000                    0.000000
50%                                  1.176471                    1.000000
75%                                  1.656250                    1.000000
max                                 37.000000                    1.000000

       timeGreater5Secprev2wrong       sumRight  helpAccessUnder2Sec  \
count              942816.000000  942816.000000        942816.000000
mean                    0.045388     145.982059             0.055674
```

(continues on next page)

(continued from previous page)

```
std                    0.208155    124.342503            0.229291
min                    0.000000      0.000000            0.000000
25%                    0.000000     50.000000            0.000000
50%                    0.000000    113.000000            0.000000
75%                    0.000000    210.000000            0.000000
max                    1.000000    962.000000            1.000000

       timeGreater10SecAndNextActionRight  consecutiveErrorsInRow  \
count                       942816.000000           942816.000000
mean                             0.207414                0.155307
std                              0.405455                0.885692
min                              0.000000                0.000000
25%                              0.000000                0.000000
50%                              0.000000                0.000000
75%                              0.000000                0.000000
max                              1.000000               56.000000

       sumTime3SDWhen3RowRight  sumTimePerSkill  \
count            942731.000000    942816.000000
mean                  0.087042       601.665586
std                   1.619202       953.900687
min                 -11.332080         0.000000
25%                   0.000000       107.000000
50%                   0.000000       275.000000
75%                   0.000000       669.000001
max                  92.709045     12459.000000

       totalTimeByPercentCorrectForskill     Prev5count     timeOver80  \
count                      942816.000000  942816.000000  942816.000000
mean                         2166.143744       4.972689       0.081323
std                          4601.435964       0.315281       0.273331
min                             0.000000       0.000000       0.000000
25%                           189.736571       5.000000       0.000000
50%                           840.000006       5.000000       0.000000
75%                          2404.499998       5.000000       0.000000
max                        310590.000100       5.000000       1.000000

           manywrong  confidence(BORED)  confidence(CONCENTRATING)  \
count  942816.000000      942816.000000               9.428160e+05
mean        0.715638           0.436958               5.400894e-01
std         0.451110           0.120751               1.830483e-01
min         0.000000           0.355694               6.500000e-07
25%         0.000000           0.355694               3.743169e-01
50%         1.000000           0.355694               5.676439e-01
75%         1.000000           0.597865               6.591692e-01
max         1.000000           0.680982               1.000000e+00

       confidence(CONFUSED)  confidence(FRUSTRATED)  confidence(OFF TASK)  \
count         942816.000000           942816.000000         942816.000000
mean               0.134450                0.164114              0.256006
std                0.292877                0.326057              0.213177
min                0.000000                0.000000              0.000000
```

(continues on next page)

(continued from previous page)

```
25%              0.000000             0.000000            0.090909
50%              0.000000             0.000000            0.230769
75%              0.000000             0.091463            0.230769
max              1.000000             1.000000            1.000000

       confidence(GAMING)      RES_BORED  RES_CONCENTRATING   RES_CONFUSED  \
count        942816.000000  942816.000000       9.428160e+05  942816.000000
mean              0.337888       0.232949       6.584415e-01       0.098940
std               0.335292       0.116371       1.734275e-01       0.249505
min               0.000039       0.156027       8.890000e-07       0.000000
25%               0.047821       0.156027       5.117519e-01       0.000000
50%               0.186970       0.156027       7.115475e-01       0.000000
75%               0.614582       0.376427       7.726099e-01       0.000000
max               0.999676       0.505313       1.000000e+00       1.000000

       RES_FRUSTRATED    RES_OFFTASK     RES_GAMING           MCAS  \
count   942816.000000  942816.000000  942816.000000  942816.000000
mean         0.131406       0.172212       0.192703     -95.982302
std          0.300351       0.216997       0.340232     332.827628
min          0.000000       0.000000       0.000001    -999.000000
25%          0.000000       0.048295       0.001483      14.000000
50%          0.000000       0.122595       0.005797      23.000000
75%          0.009561       0.122595       0.259648      34.000000
max          1.000000       1.000000       0.999377      54.000000

            Enrolled      Selective         isSTEM
count  942816.000000  942816.000000  316974.000000
mean        0.641147       0.300434       0.204178
std         0.479664       0.458447       0.403100
min         0.000000       0.000000       0.000000
25%         0.000000       0.000000       0.000000
50%         1.000000       0.000000       0.000000
75%         1.000000       1.000000       0.000000
max         1.000000       1.000000       1.000000
```

[40]:

```
print("The number of records: " + str(len(data['action_num'].unique())))
#or use print(data['action_num'].count())
```

```
The number of records: 942816
```

[37]:

```
print('Part of missing values for every column')
print(data.isnull().sum() / len(data))
```

```
Part of missing values for every column
studentId                0.000000
MiddleSchoolId           0.000000
InferredGender           0.184189
SY ASSISTments Usage     0.000000
AveKnow                  0.000000
                           ...
Ln                       0.000000
MCAS                     0.000000
```

(continues on next page)

(continued from previous page)

```
Enrolled                0.000000
Selective               0.000000
isSTEM                  0.663801
Length: 82, dtype: float64
studentId               942816
MiddleSchoolId          942816
InferredGender          769160
SY ASSISTments Usage    942816
AveKnow                 942816
                         ...
Ln                      942816
MCAS                    942816
Enrolled                942816
Selective               942816
isSTEM                  316974
Length: 82, dtype: int64
```

[28]:
```
len(data.studentId.unique())
```

[28]:
```
1709
```

[29]:
```
len(data.MiddleSchoolId.unique())
```

[29]:
```
4
```

Sort by student id

[86]:
```
ds = data['studentId'].value_counts().reset_index() #value_countsstudentidindexreset

ds.columns = [
    'studentId',
    'count'
]

ds['studentId'] = ds['studentId'].astype(str) + '-' #str
ds = ds.sort_values(['count']).tail(40)

fig = px.bar(
    ds,
    x = 'count',
    y = 'studentId',
    orientation='h',
    title='Top 40 students by number of actions'
)

fig.show("svg")
```

[58]:
```
ds = data['studentId'].value_counts().reset_index()

ds.columns = [
    'studentId',
    'count'
]
## Correct answers
ds = ds.sort_values('studentId')

fig = px.histogram(
    ds,
    x = 'studentId',
    y = 'count',
    title = 'User action distribution'
)

fig.show("svg")
```

Sort by MiddleSchoolId

[92]:
```
ds = data['MiddleSchoolId'].value_counts().reset_index()

ds.columns = [
    'MiddleSchoolId',
    'percent'
]

ds['percent'] /= len(data)
ds = ds.sort_values(['percent'])

fig = px.pie(
    ds,
    names = 'MiddleSchoolId',
    values = 'percent',
    title = 'Percent of schools',
)

fig.show("svg")
```

Sort by correct answers

[93]:
```
ds = data['correct'].value_counts().reset_index()

ds.columns = [
    'correct',
    'percent'
]

ds['percent'] /= len(data)
ds = ds.sort_values(['percent'])

fig = px.pie(
    ds,
    names = ['0', '1'],
    values = 'percent',
    title = 'Percent of correct answers'
)

fig.show("svg")
```

Sort by problem id

[94]:
```
ds = data['problemId'].value_counts().reset_index()

ds.columns = [
    'problemId',
    'count'
]

ds['problemId'] = ds['problemId'].astype(str) + '-'
ds = ds.sort_values(['count']).tail(40)

fig = px.bar(
    ds,
    x = 'count',
    y = 'problemId',
    orientation = 'h',
    title = 'Top 40 useful problem ids'
)

fig.show("svg")
```

[90]:
```
ds = data['problemId'].value_counts().reset_index()

ds.columns = [
    'problemId',
```

(continues on next page)

(continued from previous page)

```
    'count'
]

ds = ds.sort_values('problemId')

fig = px.histogram(
    ds,
    x='problemId',
    y='count',
    title='problemid action distribution'
)

fig.show("svg")
```

[83]:
```
ds = data['problemType'].value_counts().reset_index()

ds.columns = [
    'problemType',
    'percent'
]

ds['percent'] /= len(data)
ds = ds.sort_values(['percent'])

fig = px.pie(
    ds,
    names = 'problemType',
    values = 'percent',
    title = 'Percent of problem types',
)

fig.show("svg")
```

[85]:
```
ds = ds.sort_values(['percent']).tail(6)

fig = make_subplots(rows=3, cols=2)

traces = [
    go.Bar(
        x = ['wrong', 'right'],
        y = [
            len(data[(data['problemType'] == item) & (data['correct'] == 0)]),
            len(data[(data['problemType'] == item) & (data['correct'] == 1)])
        ],
        name = 'Type: ' + str(item),
        text = [
            str(round(100*len(data[(data['problemType'] == item)&(data['correct'] ==␣
↪0)])/len(data[data['problemType'] == item]),2)) + '%',
            str(round(100*len(data[(data['problemType'] == item)&(data['correct'] ==␣
↪1)])/len(data[data['problemType'] == item]),2)) + '%'
```

(continues on next page)

(continued from previous page)

```
        ],
        textposition = 'auto'
    ) for item in ds['problemType'].unique().tolist()
]

for i in range(len(traces)):
    fig.append_trace(
        traces[i],
        (i //2) + 1,
        (i % 2) + 1
    )

fig.update_layout(
    title_text = 'Percent of correct answers for top 6 problem type',
)

fig.show("svg")
```

Sort by skills

[67]:
```
ds = data['skill'].dropna() # There are less NaNs in 'skill_id' column than 'skill_name'␣
↪column.
ds = ds.value_counts().reset_index()

ds.columns = [
    'skill',
    'count'
]

ds['skill'] = ds['skill'].astype(str) + '-'
ds = ds.sort_values(['count']).tail(40)

fig = px.bar(
    ds,
    x = 'count',
    y = 'skill',
    orientation = 'h',
    title = 'Top 40 useful skills'
)

fig.show("svg")
```

[ ]:
```
```

6.7.2 EdNet

EdNet-KT1 Data Analysis

Columns Description

Field	Annotation
user_id	student's id
timestamp	the moment the question was given, represented as Unix timestamp in milliseconds
solving_id	represents each learning session of students corresponds to each bunle. It is a form of single integer, starting from 1
question_id	the ID of the question that given to student, which is a form of q{integer}
user_answer	the answer that the student submitted, recorded as a character between a and d inclusively
elapsed_time	the time that the students spends on each question in milliseconds

Statement for Our Data Set

There are 784309 tables in our data set. Each table describes a student's question-solving log. There is no difference in the information dimension between the tables. Each table contains the `timestamp`,`solving_id`,`question_id`,`user_answer` and `elapsed_time` as described in the above `Columns Description` section.

[1]:
```
import numpy as np
import pandas as pd

import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objs as go
```

Record Example

We randomly selected 5000 tables from all the students for analysis,which accounted for about 0.64% of the total data set, and added a column named `user_id` to the original table

[2]:
```
import os
path=r'D:\EdNet-KT1\KT1'
d=[]
table_list=[]
s=pd.Series(os.listdir(path))
file_selected=s.sample(5000).to_numpy()
for file_name in file_selected:
    data_raw=pd.read_csv(path+'\\'+file_name,encoding = "ISO-8859-15")
    data_raw['user_id']=pd.Series([file_name[:-4]]*len(data_raw))
    d.append([file_name[:-4],len(data_raw)])
    data=pd.DataFrame(data_raw,columns=['user_id']+data_raw.columns.to_list()[:-1])
    table_list.append(data)
df=pd.concat(table_list)
pd.set_option('display.max_rows',10)
```

(continues on next page)

(continued from previous page)

```
df=df.reset_index(drop=True)
df
```

[2]:
```
       user_id      timestamp  solving_id question_id user_answer  \
0      u717875  1565332027449           1       q4862           d
1      u717875  1565332057492           2       q6747           d
2      u717875  1565332085743           3        q326           c
3      u717875  1565332116475           4       q6168           a
4      u717875  1565332137148           5        q847           a
...        ...            ...         ...         ...         ...
574251 u177603  1530371808931          15       q6984           b
574252 u177603  1530372197614          16       q7335           c
574253 u177603  1530372198181          16       q7336           a
574254 u177603  1530372198879          16       q7337           c
574255 u177603  1530372199425          16       q7338           b

        elapsed_time
0              45000
1              24000
2              25000
3              27000
4              17000
...              ...
574251         44250
574252         95750
574253         95750
574254         95750
574255         95750

[574256 rows x 6 columns]
```

General Feature

[3]:
```
df.describe()
```

[3]:
```
          timestamp     solving_id  elapsed_time
count  5.742560e+05  574256.000000  5.742560e+05
mean   1.546425e+12     875.902859  2.599017e+04
std    2.019656e+10    1941.978009  3.376126e+04
min    1.494451e+12       1.000000  0.000000e+00
25%    1.531720e+12      77.000000  1.600000e+04
50%    1.548410e+12     311.000000  2.100000e+04
75%    1.564817e+12     900.000000  3.000000e+04
max    1.575306e+12   18039.000000  7.650000e+06
```

[4]:
```
len(df.question_id.unique())
```

[4]:
```
11838
```

This shows there are totally 11838 questions.

Missing Value

[5]:
```
print('Part of missing values for every column')
print(df.isnull().sum() / len(df))
```

```
Part of missing values for every column
user_id          0.000000
timestamp        0.000000
solving_id       0.000000
question_id      0.000000
user_answer      0.000556
elapsed_time     0.000000
dtype: float64
```

This indicates that there are no missing values in all columns except `user_answer`. A missing value in `user_answer` indicates that some students did not choose an option.

[6]:
```
df.fillna('not choose',inplace=True)
```

Fill in `not choose` in the position of the missing value

Sort user_id

[7]:
```
user_count_table=pd.DataFrame(d,columns=['user_id','count'])
ds=user_count_table.sort_values(by=['count'],axis=0).tail(40)
fig = px.bar(
    ds,
    x = 'count',
    y = 'user_id',
    orientation='h',
    title='Top 40 active students'
)

fig.show("svg")
```

We use the number of questions that students have done as an indicator of whether a student is active. This figure shows the 40 most active students.

[8]:
```
ds=df.loc[:,['user_id','elapsed_time']].groupby('user_id').mean()
ds=ds.reset_index(drop=False)
ds.columns=['user_id','avg_elapsed_time']
ds_tail=ds.sort_values(by=['avg_elapsed_time'],axis=0).tail(40)

fig_tail = px.bar(
    ds_tail,
    x = 'avg_elapsed_time',
    y = 'user_id',
    orientation='h',
    title='Bottom 40 fast-solving students '
)
fig_tail.show("svg")
ds_head=ds.sort_values(by=['avg_elapsed_time'],axis=0).head(40)
```

(continues on next page)

(continued from previous page)

```
fig_head = px.bar(
    ds_head,
    x = 'avg_elapsed_time',
    y = 'user_id',
    orientation='h',
    title='Top 40 fast-solving students'
)
fig_head.show("svg")
```

We take the average time it takes students to do a question as an indicator of how fast students do it. These two figures respectively show the fastest and slowest students among the 5000 students, and the average time they spent doing the problems.

Note that some students spend very little time doing the questions, and the time is almost zero. We can almost judge that these students did not do the questions at all, and they chose blindly. We remove these students and rearrange them

[9]:
```
bound=5000 # If the average time of doing the topic is less than 5000, it means that the␣
↪student is most likely to be bad
ds=df.loc[:,['user_id','elapsed_time']].groupby('user_id').mean()
ds=ds.reset_index(drop=False)
ds.columns=['user_id','avg_elapsed_time']
bad_user_ids=ds[ds['avg_elapsed_time']<bound]['user_id'].to_list()
df_drop=df.drop(df[df['user_id'].isin(bad_user_ids)].index)
print('bad students number is ',len(bad_user_ids))
print('length of table after dropping is ',len(df_drop))
```

```
bad students number is  61
length of table after dropping is  567778
```

After dropping

[10]:
```
ds=df_drop['user_id'].value_counts().reset_index(drop=False)
ds.columns=['user_id','count']
ds_tail=ds.sort_values(by=['count'],axis=0).tail(40)
fig_tail = px.bar(
    ds_tail,
    x = 'count',
    y = 'user_id',
    orientation='h',
    title='Top 40 active students after dropping some students'
)
fig_tail.show("svg")
```

This figure shows the 40 most active students after dropping some bad students.

[11]:
```
ds=df_drop.loc[:,['user_id','elapsed_time']].groupby('user_id').mean()
ds=ds.reset_index(drop=False)
ds.columns=['user_id','avg_elapsed_time']

ds_head=ds.sort_values(by=['avg_elapsed_time'],axis=0).head(40)
```

(continues on next page)

(continued from previous page)

```
fig_head = px.bar(
    ds_head,
    x = 'avg_elapsed_time',
    y = 'user_id',
    orientation='h',
    title='Top 40 fast-solving students after dropping some students'
)
fig_head.show("svg")
```

This figure respectively show the more reasonable fastest students among the 5000 students than before, and the average time they spent doing the problems.

Sort question_id

[12]:
```
ds=df.loc[:,['question_id','elapsed_time']].groupby('question_id').mean()
ds=ds.reset_index(drop=False)
ds_tail=ds.sort_values(by=['elapsed_time'],axis=0).tail(40)
fig_tail = px.bar(
    ds_tail,
    x = 'elapsed_time',
    y = 'question_id',
    orientation='h',
    title='Top 40 question_id by the average of elapsed_time'
)
fig_tail.show("svg")
ds_head=ds.sort_values(by=['elapsed_time'],axis=0).head(40)
fig_head = px.bar(
    ds_head,
    x = 'elapsed_time',
    y = 'question_id',
    orientation='h',
    title='Bottom 40 question_id by the average of elapsed_time'
)
fig_head.show("svg")
```

We can judge the difficulty of this question from the average time spent on a question.

These two figures reflect the difficulty of the questions and shows the ids of the 40 most difficult and 40 easiest questions.s

Appearence of Questions

[13]:
```
ds=df['question_id'].value_counts().reset_index(drop=False)
ds.columns=['question_id','count']
ds_tail=ds.sort_values(by=['count'],axis=0).tail(40)
fig_tail = px.bar(
    ds_tail,
    x = 'count',
    y = 'question_id',
    orientation='h',
    title='Top 40 question_id by the number of appearance'
)
fig_tail.show("svg")
ds_head=ds.sort_values(by=['count'],axis=0).head(40)
fig_head = px.bar(
    ds_head,
    x = 'count',
    y = 'question_id',
    orientation='h',
    title='Bottom 40 question_id by the number of appearance'
)
fig_head.show("svg")
```

These two images reflect the 40 questions that were drawn the most frequently and the 40 questions that were drawn the least frequently

[14]:
```
ds2=df['question_id'].value_counts().reset_index(drop=False)
ds2.columns=['question_id','count']
def convert_id2int(x):
    return pd.Series(map(lambda t:int(t[1:]),x))
ds2['question_id']=convert_id2int(ds2['question_id'])
ds2.sort_values(by=['question_id'])
fig = px.histogram(
    ds2,
    x = 'question_id',
    y = 'count',
    title='question distribution'
)
fig.show("svg")
```

Question's Option Selected Most Frequently

[15]:
```
ds=df.loc[:,['question_id','user_answer','user_id']].groupby(['question_id','user_answer
↪']).count()

most_count_dict={}
for id in df.question_id.unique():
    most_count=ds.loc[id].apply(lambda x:x.max())[0]
    most_count_dict[id]=most_count
```

(continues on next page)

(continued from previous page)

```
ds2=ds.apply(lambda x:x-most_count_dict[x.name[0]],axis=1)
ds2=ds2[ds2.user_id==0]
ds2=ds2.reset_index(drop=False).loc[:,['question_id','user_answer']]
ds2.columns=['question_id','most_answer']
ds2.index=ds2['question_id']
ds2['most_answer']
```

[15]:
```
question_id
q1          b
q10         d
q100        c
q1000       c
q10000      b
           ..
q9995       d
q9996       a
q9997       d
q9998       a
q9999       b
Name: most_answer, Length: 12215, dtype: object
```

This shows the most selected options (including `not choose`) for each question.
Note that if there are multiple options for a question to be selected most frequently, the table will also contain them.

Choices Distribution

[16]:
```
ds = df['user_answer'].value_counts().reset_index(drop=False)
ds.columns = ['user_answer', 'percent']

ds['percent']=ds['percent']/len(df)
ds = ds.sort_values(by=['percent'])

fig = px.pie(
    ds,
    names = ds['user_answer'],
    values = 'percent',
    title = 'Percent of Choice'
)

fig.show("svg")
```

We use a pie chart to show the distribution of the proportions of `a`, `b`, `c`, `d` and `not choose` among the options selected by the 5000 students.

Sort By Time Stamp

[17]:
```
import time
import datetime
```

[18]:
```
df_time=df.copy()
columns=df.columns.to_list()
columns[1]='time'
df_time.columns=columns
df_time['time'] /= 1000
df_time['time']=pd.Series(map(datetime.datetime.fromtimestamp,df_time['time']))
df_time
```

[18]:
```
        user_id                    time  solving_id question_id user_answer  \
0       u717875 2019-08-09 14:27:07.449           1       q4862           d
1       u717875 2019-08-09 14:27:37.492           2       q6747           d
2       u717875 2019-08-09 14:28:05.743           3        q326           c
3       u717875 2019-08-09 14:28:36.475           4       q6168           a
4       u717875 2019-08-09 14:28:57.148           5        q847           a
...         ...                     ...         ...         ...         ...
574251  u177603 2018-06-30 23:16:48.931          15       q6984           b
574252  u177603 2018-06-30 23:23:17.614          16       q7335           c
574253  u177603 2018-06-30 23:23:18.181          16       q7336           a
574254  u177603 2018-06-30 23:23:18.879          16       q7337           c
574255  u177603 2018-06-30 23:23:19.425          16       q7338           b

        elapsed_time
0              45000
1              24000
2              25000
3              27000
4              17000
...              ...
574251         44250
574252         95750
574253         95750
574254         95750
574255         95750

[574256 rows x 6 columns]
```

This table shows the result of converting unix timestamp to datetime format

question distribution by time

[19]:
```
ds_time_question=df_time.loc[:,['time','question_id']]
ds_time_question=ds_time_question.sort_values(by=['time'])
ds_time_question
```

[19]:
```
                          time question_id
503014 2017-05-11 05:17:10.922        q129
503015 2017-05-11 05:17:34.561       q8058
```

(continues on next page)

(continued from previous page)

```
503016 2017-05-11 05:17:56.806       q8120
503017 2017-05-11 05:18:22.591        q157
503018 2017-05-11 05:18:43.085         q52
...                        ...         ...
108215 2019-12-03 00:48:27.437        q776
108216 2019-12-03 00:59:38.437      q10847
108217 2019-12-03 00:59:38.437      q10844
108218 2019-12-03 00:59:38.437      q10845
108219 2019-12-03 00:59:38.437      q10846

[574256 rows x 2 columns]
```

This table shows the given questions in chronological order.And we can see that the earliest question q127 is on May 11, 2017, and the latest question q10846 is on December 3, 2019.

[20]:
```
ds_time_question['year']=pd.Series(map(lambda x :x.year,ds_time_question['time']))
ds_time_question['month']=pd.Series(map(lambda x :x.month,ds_time_question['time']))
ds=ds_time_question.loc[:,['year','month']].value_counts()

years=ds_time_question['year'].unique()
years.sort()
fig=make_subplots(
    rows=2,
    cols=2,
    start_cell='top-left',
    subplot_titles=tuple(map(str,years))
)
traces=[
    go.Bar(
        x=ds[year].reset_index().sort_values(by=['month'],axis=0)['month'].to_list(),
        y=ds[year].reset_index().sort_values(by=['month'],axis=0)[0].to_list(),
        name='Year: '+str(year),
        text=[ds[year][month] for month in ds[year].reset_index().sort_values(by=['month
↪'],axis=0)['month'].to_list()],
        textposition='auto'
    ) for year in years
]
for i in range(len(traces)):
    fig.append_trace(traces[i],(i//2)+1,(i%2)+1)

fig.update_layout(title_text='Bar of the distribution of the number of question solved␣
↪in {} years'.format(len(traces)))
fig.show('svg')
```

1. These three figures show the distribution of the number of problems solved in each month of 2017, 2018, and 2019.
2. And the number of questions solved is gradually increasing.
3. And the number of questions solved in March, 4, May, and June is generally small.

user distribution by time

[21]:
```
ds_time_user=df_time.loc[:,['user_id','time']]
ds_time_user=ds_time_user.sort_values(by=['time'])
ds_time_user
```

[21]:
```
       user_id                    time
503014  u21056 2017-05-11 05:17:10.922
503015  u21056 2017-05-11 05:17:34.561
503016  u21056 2017-05-11 05:17:56.806
503017  u21056 2017-05-11 05:18:22.591
503018  u21056 2017-05-11 05:18:43.085
...        ...                     ...
108215   u9476 2019-12-03 00:48:27.437
108216   u9476 2019-12-03 00:59:38.437
108217   u9476 2019-12-03 00:59:38.437
108218   u9476 2019-12-03 00:59:38.437
108219   u9476 2019-12-03 00:59:38.437

[574256 rows x 2 columns]
```

This table shows the students who did the questions in order of time.And we can see that the first student who does the problem is u21056, and the last student who does the problem is u9476.

[22]:
```
ds_time_user=df_time.loc[:,['user_id','time']]
ds_time_user=ds_time_user.sort_values(by=['time'])
ds_time_user['year']=pd.Series(map(lambda x :x.year,ds_time_user['time']))
ds_time_user['month']=pd.Series(map(lambda x :x.month,ds_time_user['time']))
ds_time_user.drop(['time'],axis=1,inplace=True)
ds=ds_time_user.groupby(['year','month']).nunique()

years=ds_time_user['year'].unique()
years.sort()
fig=make_subplots(
    rows=2,
    cols=2,
    start_cell='top-left',
    subplot_titles=tuple(map(str,years))
)
traces=[
    go.Bar(
        x=ds.loc[year].reset_index()['month'].to_list(),
        y=ds.loc[year].reset_index()['user_id'].to_list(),
        name='Year: '+str(year),
        text=[ds.loc[year].loc[month,'user_id'] for month in ds.loc[year].reset_index()[
↪'month'].to_list()],
        textposition='auto'
    ) for year in years
]
for i in range(len(traces)):
    fig.append_trace(traces[i],(i//2)+1,(i%2)+1)

fig.update_layout(title_text='Bar of the distribution of the number of active students␣
↪in {} years'.format(len(traces)))
```

(continues on next page)

(continued from previous page)

```
fig.show('svg')
```

1. These three graphs respectively show the number of students active on the system in each month of 2017, 2018, and 2019.
2. And we can see that the number of active students in 2019 is generally more than that in 2018, and there are more in 2018 than in 2017, indicating that the number of users of the system is gradually increasing.
3. Note that the number of students is not repeated here

6.7.3 junyi

Junyi

data source

Authorization

```
Any form of commercial usage is not allowed!
Please cite the following paper if you publish your work:

Haw-Shiuan Chang, Hwai-Jung Hsu and Kuan-Ta Chen,
"Modeling Exercise Relationships in E-Learning: A Unified Approach,"
International Conference on Educational Data Mining (EDM), 2015.
```

Introduction

The dataset contains the problem log and exercise-related information on the Junyi Academy (http://www.junyiacademy.org/), an E-learning platform established in 2012 on the basis of the open-source code released by Khan Academy. In addition, the annotations of exercise relationship we collected for building models are also available.

Data Description

Column Description

Field	Annotation
name	Exercise name (The name is also an id of exercise, so each name is unique in the dataset). If you want to access the exercise on the website, please append this name after url, http://www.junyiacademy.org/exercise/ (e.g., http://www.junyiacademy.org/exercise/similar_triangles_1). Please note that Junyi Academy are constantly changing their contents as Khan Academy did, so some url of exercises might be unavaible when you access them.
live	Whether the exercise is still accessible on the website on Jan. 2015
pre-requisite	Indicate its prerequisite exericse (parent shown in its knowledge map)
h_position	The coordiate on the x axis of the knowledge map
v_position	The coordiate on the y axis of the knowledge map
creation_date	The date this exercise is created
seconds_per_fast_problem	The website judge a student finish the exercise fast if he/she takes less then this time to answer the question. This number is manually assigned by the experts in Junyi Academy.
pretty_display_name	The chinese name of exercise shown in the knowledge map (Please use UTF-8 to decode the chinese characters)
short_display_name	Another chinese name of exercise (Please use UTF-8 to decode the chinese characters)
topic	The topic of each exercise, and the topic would be shown as a larger node in the knowledge map.
area:	The area of each exercise (Each area contains several topics)

[1]:
```
import numpy as np
import dask.dataframe as dd
import pandas as pd

import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objs as go
```

[2]:
```
path = "./junyi/junyi_Exercise_table.csv"

data = pd.read_csv(path, encoding = "utf-8",low_memory=False)
data.head()
```

[2]:
```
                               name  live              prerequisites  \
0              parabola_intuition_1  True  recognizing_conic_sections
1                  circles_and_arcs  True                         NaN
2                inscribed_angles_3  True          inscribed_angles_2
3  solving_quadratics_by_factoring  True      factoring_polynomials_1
4              graphing_parabolas_1  True       graphing_parabolas_0.5

   h_position  v_position                  creation_date  \
0          47           2   2012-10-11 17:55:24.8056 UTC
1          40         -20  2012-10-11 17:55:33.41014 UTC
2          44         -22  2012-10-11 17:55:44.11836 UTC
```

(continues on next page)

(continued from previous page)

```
3          50           -2  2012-10-11 17:54:59.28029 UTC
4          52            0  2012-10-11 17:55:00.48268 UTC

   seconds_per_fast_problem pretty_display_name short_display_name  \
0                      13.0               ? 1               ?1
1                      27.0
2                       5.0           3         3
3                       7.0
4                      24.0                1                1

                       topic      area
0             conic-sections   algebra
1  area-perimeter-and-volume  geometry
2          circle-properties  geometry
3                quadtratics   algebra
4                quadtratics   algebra
```

[3]:
```
data.describe()
```

[3]:
```
       h_position  v_position  seconds_per_fast_problem
count  837.000000  837.000000                837.000000
mean    25.402628   -5.704898                 10.782557
std     15.876667   12.721159                  8.935352
min    -15.000000  -34.000000                  0.000000
25%     15.000000  -17.000000                  5.000000
50%     26.000000   -5.000000                  8.000000
75%     36.000000    5.000000                 13.000000
max     60.000000   19.000000                 60.000000
```

[4]:
```
data["area"] = [item if item != "null" and item !='nan' else "unknown"
                            for item in data["area"].apply(str)]

fig = px.scatter(
    data,
    x = 'h_position',
    y = 'v_position',
    color='area',
    title='Exercises distribution on area in knowledge map'
)

fig.show('svg')
```

[5]:
```
data["topic"] = [item if item != "null" and item !='nan' else "unknown"
                            for item in data["topic"].apply(str)]

fig = px.scatter(
    data,
    x = 'h_position',
    y = 'v_position',
    color='topic',
    title='Exercises distribution on topics in knowledge map'
```

(continues on next page)

(continued from previous page)

```
)

fig.show('svg')
```

[6]:
```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# Creating two subplots
def makeplot(title='Average time And Exercises count spent on area',groupByItem='area'):
    ds=data.groupby(groupByItem,as_index=False).agg(exercise_count=('topic','count'))

    ds = ds.sort_values('exercise_count')

    fig = px.bar(
    ds,
    x = 'exercise_count',
    y = groupByItem,
    orientation='h',
    title=title
)



    fig.show('svg')
makeplot(title='Exercise count on area',groupByItem='area')
makeplot(title='Exercise count on topics',groupByItem='topic')
```

Field	description
Exercise_A, Exercise_B	The exercise names being compared
Similarity_avg, Difficulty_avg, Prequesite_avg	The mean opinion scores of different relationships. This is also the ground truth we used to train/test our model.
Similarity_raw, Difficulty_raw, Prequesite_raw	The raw scores given by workers (delimiter is "_")

[8]:
```
path = "./junyi/relationship_annotation_training.csv"

data = dd.read_csv(path, encoding = "utf-8",low_memory=False)
data.head()
```

[8]:
```
                          Exercise_A  \
0  radius_diameter_and_circumference
1  radius_diameter_and_circumference
2  radius_diameter_and_circumference
3               vertex_of_a_parabola
4               vertex_of_a_parabola

                              Exercise_B  Similarity_avg  \
0                 arithmetic_word_problems_1        1.857143
1                            parts_of_circles        6.785714
```

(continues on next page)

(continued from previous page)

```
2          perimeter_of_squares_and_rectangles        3.571429
3  solving_quadratics_by_taking_the_square_root        5.923077
4                        completing_the_square_1        5.692308

               Similarity_raw  Difficulty_avg              Difficulty_raw  \
0  1_4_1_1_1_1_2_1_1_1_3_1_3_5        2.857143  4_5_1_1_1_1_7_1_1_4_2_5_2_5
1  6_9_6_6_7_8_7_8_8_8_4_6_5_7        2.428571  3_5_1_3_2_1_5_1_1_1_1_2_5_3
2  2_6_4_1_1_2_4_4_7_2_3_4_4_6        2.285714  2_5_1_1_1_1_3_2_1_1_5_2_3_4
3    6_7_6_7_8_4_5_4_3_6_6_8_7        3.307692    3_3_3_1_2_2_4_4_4_3_5_5_4
4    7_5_7_8_3_4_5_5_3_6_7_7_7        3.307692    2_3_3_4_2_2_4_4_5_3_4_4_3

   Prerequisite_avg             Prerequisite_raw
0          3.000000  1_6_1_1_1_3_2_1_9_2_3_2_8_2
1          7.285714  6_7_7_6_8_8_9_5_9_9_7_7_5_9
2          5.000000  2_6_5_4_2_8_3_5_9_5_5_3_7_6
3          5.846154    5_8_7_7_6_2_6_5_6_7_3_7_7
4          5.461538    6_4_6_8_2_2_5_6_5_6_7_7_7
```

[9]:
```
data.describe().compute()
```

[9]:
```
       Similarity_avg  Difficulty_avg  Prerequisite_avg
count     1131.000000     1131.000000       1131.000000
mean         5.088256        4.402577          4.801077
std          2.248680        1.586114          1.934648
min          1.000000        1.000000          1.166667
25%          3.100000        3.153846          3.160256
50%          5.333333        4.333333          4.777778
75%          7.000000        5.538462          6.333333
max          9.000000        8.454545          8.800000
```

user_id	An number represents an user
exercise	Exercise name
problem_type	Some exercises would record what template of problem this student encounters at this time
problem_number	How many times this student practices this exercise (e.g., the number would be 1 if the student tries to answer this exercise at the first time)
topic_mode	Whether the student is assigned this exercise by clicking the topic icon (This function has been closed now)
suggested	Whether the exercise is suggested by the system according to prerequisite relationships on the knowledge map
review_mode	Whether the exercise is done by the student after he/she earn proficiency
time_done	Unix timestamp in microsecends
time_taken	Second the student spend on this exercise
time_taken_attempts	Seconds the student spend on each answering attempt
correct	Whether the student's first attempt is correct, and the field would be false if any hint is requested
count_attempts	How many times student attempt to answer the problem
hint_used	Whether student request hints
count_hints	How many times student request hints
hint_time_taken_list	Seconds the student spend on each requested hints
earned_proficiency	Whether the student reaches proficiency. Please refer to http://david-hu.com/2011/11/02/how-khan-academy-is-using-machine-learning-to-assess-student-mastery.html for the algorithm of determining proficiency
points_earned	How many points students earn for this practice

[11]:
```
path = "./junyi/junyi_ProblemLog_original.csv"

data = dd.read_csv(path, encoding = "utf-8",low_memory=False, dtype={'hint_time_taken_
↪list': 'object'})
data.head()
```

[11]:
```
   user_id             exercise   problem_type  problem_number  topic_mode  \
0    12884     time_terminology    analog_word               1       False
1   239464     multiplication_1              0               6       False
2   147359  adding_decimals_0.5              0               6       False
3   158155     multiplication_1              0               3       False
4   147151        subtraction_2  subtraction-2              10        True

   suggested  review_mode         time_done  time_taken time_taken_attempts  \
0      False        False  1420714810324490           4                 3&1
1      False        False  1403098400836660           2                   2
2      False        False  1418890695540340          16                  16
3      False        False  1400469444264040           2                   2
4       True        False  1382650905730160           4                   4

   correct  count_attempts  hint_used  count_hints hint_time_taken_list  \
0    False               2      False            0                  NaN
1     True               1      False            0                  NaN
2     True               1      False            0                  NaN
3     True               1      False            0                  NaN
4     True               1      False            0                  NaN
```

(continues on next page)

(continued from previous page)

```
   earned_proficiency  points_earned
0               False              0
1               False             14
2               False             75
3               False             75
4               False            225
```

[12]:
```
data.describe().compute()
```

[12]:
```
            user_id  problem_number     time_done    time_taken  \
count  2.592599e+07    2.592599e+07  2.592599e+07  2.592599e+07
mean   1.236557e+05    2.859253e+01  3.263023e+11  9.955710e+01
std    7.121600e+04    9.871659e+01  1.248303e+13  2.157362e+05
min    0.000000e+00    1.000000e+00  1.350004e+15 -5.049212e+08
25%    6.199900e+04    4.000000e+00  1.395736e+15  4.000000e+00
50%    1.242630e+05    9.000000e+00  1.405395e+15  8.000000e+00
75%    1.856380e+05    2.200000e+01  1.415168e+15  1.800000e+01
max    2.476050e+05    5.174000e+03  1.421000e+15  4.067572e+08

       count_attempts   count_hints  points_earned
count    2.592599e+07  2.592599e+07   2.592599e+07
mean     1.363888e+00  2.850791e-01   8.219998e+01
std      2.391150e+00  1.276758e+00   9.056150e+01
min      0.000000e+00  0.000000e+00   0.000000e+00
25%      1.000000e+00  0.000000e+00   5.000000e+00
50%      1.000000e+00  0.000000e+00   5.000000e+01
75%      1.000000e+00  0.000000e+00   1.950000e+02
max      1.000000e+03  2.000000e+01   2.250000e+02
```

[13]:
```
data['user_id'].nunique().compute()
```

[13]:
```
247606
```

[14]:
```
total_count = len(data)
total_count
```

[14]:
```
25925992
```

[15]:
```
ds = data['earned_proficiency'].value_counts().reset_index().compute()

ds.columns = [
    'earned_proficiency',
    'percent'
]

ds['percent'] /= total_count
ds = ds.sort_values(['percent'])
```

[16]:
```
ds
```

[16]:
```
   earned_proficiency   percent
1                True  0.046066
0               False  0.953934
```

[17]:
```
fig = px.pie(
    ds,
    names = ['mastered','not mastered'],
    values = 'percent',
    title = 'Percent of mastered exercises',
)

fig.show('svg')
```

[18]:
```
ds = data['correct'].value_counts().reset_index().compute()
ds.columns = [
    'correct',
    'percent'
]
ds['percent'] /= total_count
ds = ds.sort_values(['percent'])
```

[19]:
```
ds
```

[19]:
```
   correct   percent
1    False  0.172126
0     True  0.827874
```

[20]:
```
fig = px.pie(
    ds,
    names = ['wrong','correct'],
    values = 'percent',
    title = 'Percent of answer correctly at first attempt',
)
fig.show('svg')
```

The tab delimited format used in PSLC datashop, please refer to their document (https://pslcdatashop.web.cmu.edu/help?page=importFormatTd) The size of the text file is too large (9.1 GB) to analyze using tools of websites, so we compress the text file and put it as an extra file of the dataset. We also upload a small subset of data into the website for the illustration purpose. Note that there are some assumptions when converting the data into this format, please read the description of our dataset for more details.

[22]:
```
path = "./junyi/junyi_ProblemLog_for_PSLC.txt"
data = dd.read_csv(path, sep='\t',encoding = "utf-8")
pd.set_option('display.max_columns', 2000)
data.head()
```

[22]:
```
   Anon Student Id  Session Id           Time Student Response Type  \
0            12884      148691  1420714809324               ATTEMPT
1            12884      148691  1420714810324               ATTEMPT
2           239464       93497  1403098400837               ATTEMPT
3           147359      145156  1418890695540               ATTEMPT
4           158155      105559  1400469444264               ATTEMPT
```

(continues on next page)

(continued from previous page)

```
  Tutor Response Type              Level (Unit)      Level (Section)  \
0              RESULT              telling-time     time_terminology
1              RESULT              telling-time     time_terminology
2              RESULT  multiplication-division     multiplication_1
3              RESULT                  decimals  adding_decimals_0.5
4              RESULT  multiplication-division     multiplication_1

                     Problem Name  Problem Start Time  \
0  time_terminology--analog_word       1420714806324
1  time_terminology--analog_word       1420714809324
2             multiplication_1--0       1403098398837
3          adding_decimals_0.5--0       1418890679540
4             multiplication_1--0       1400469442264

                        Step Name    Outcome   Condition Name  Condition Type  \
0  time_terminology--analog_word  INCORRECT  Choose_Exercise             NaN
1  time_terminology--analog_word  INCORRECT  Choose_Exercise             NaN
2             multiplication_1--0    CORRECT  Choose_Exercise             NaN
3          adding_decimals_0.5--0    CORRECT  Choose_Exercise             NaN
4             multiplication_1--0    CORRECT  Choose_Exercise             NaN

  Selection  Action  Input         KC (Exercise)               KC (Topic)  \
0       NaN     NaN    NaN     time_terminology             telling-time
1       NaN     NaN    NaN     time_terminology             telling-time
2       NaN     NaN    NaN     multiplication_1  multiplication-division
3       NaN     NaN    NaN  adding_decimals_0.5                 decimals
4       NaN     NaN    NaN     multiplication_1  multiplication-division

    KC (Area)  CF (points_earned)  CF (earned_proficiency)
0  arithmetic                   0                        0
1  arithmetic                   0                        0
2  arithmetic                  14                        0
3  arithmetic                  75                        0
4  arithmetic                  75                        0
```

Questions and Collaboration:

```
1. If you have any question to this dataset, please e-mail to hschang@cs.umass.edu.
2. If you have intention to acquire more data which fit your research purpose, please contact Junyi Academy directly for discussing the further cooperation opportunites by emailing to support@junyiacademy.org
```

Note:

```
1. The dataset we used in our paper (Modeling Exercise Relationships in E-Learning: A␣
↪Unified Approach) is extracted from Junyi Academy on July 2014, and this dataset is␣
↪extracted on Jan 2015. After applying our method on the new dataset, we got similar␣
↪observation with that in our paper, even though this dataset contains more users and␣
↪exercises.
2. After uncompress the original problem log and problem log using PLSC format, the text␣
↪files will take around 2.6 GB and 9.1 GB respectively. Please prepare enough space in␣
↪your disk.
```

Annotaion:

1. PSLCoriginaltime_taken_attemptsPSLCoriginal

Analysis

[23]:
```
len(data)
```

[23]:
```
39462201
```

[24]:
```
ds=data.groupby('Anon Student Id').agg({'Session Id':'count'}).describe().compute()
```

[25]:
```
ds
```

[25]:
```
          Session Id
count  247547.000000
mean      159.412964
std       598.876158
min         1.000000
25%         7.000000
50%        19.000000
75%        82.000000
max     55984.000000
```

[26]:
```
data1=data.sample(frac=0.01).compute()
```

[27]:
```
# session(1%)
nunique = dd.Aggregation(
    name="nunique",
    chunk=lambda s: s.apply(lambda x: list(set(x))),
    agg=lambda s0: s0.obj.groupby(level=list(range(s0.obj.index.nlevels))).sum(),
    finalize=lambda s1: s1.apply(lambda final: len(set(final))),
)
ds = data1.groupby('Session Id').agg({'KC (Exercise)':'nunique', 'KC (Topic)':'nunique',
↪'Time':lambda x: x.max()-x.min()})
ds.describe()
```

[27]:
```
      KC (Exercise)     KC (Topic)          Time
count  164015.000000  164015.000000  1.640150e+05
```

(continues on next page)

(continued from previous page)

```
mean        1.994543       1.564345  5.521162e+08
std         2.457822       1.288020  2.618666e+09
min         1.000000       1.000000  0.000000e+00
25%         1.000000       1.000000  0.000000e+00
50%         1.000000       1.000000  0.000000e+00
75%         2.000000       2.000000  1.515309e+06
max       121.000000      35.000000  6.488305e+10
```

[ ]:

6.7.4 math2015

math2015-Math1 Data Analysis

[1]:
```
import numpy as np
import pandas as pd

import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objs as go
```

[2]:
```
path = "C:/Users/Administrator/Desktop/Math1/rawdata.txt"
data = pd.read_table(path,header=None)
```

RECORDS

The learning records are saved in a score matrix. Each row corresponds to the records of a student on different test items.

[3]:
```
pd.set_option('display.max_rows',10)
data
```

[3]:
```
      0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17  \
0     4   4   4   4   4   4   4   4   4   4   4   4   4   4   0   5   8   0
1     4   0   4   0   4   4   4   4   0   4   4   4   0   0   0   5   0   8
2     4   4   0   0   0   0   4   0   0   4   0   4   0   0   0   0   0   0
3     4   4   4   0   0   0   4   4   4   0   4   4   4   4   0   5   0   0
4     0   4   4   0   4   0   4   0   0   0   4   4   0   4   0   3   0   0
...  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
4204  4   0   4   4   4   0   4   0   0   4   0   0   0   0   0   4   2   0
4205  0   4   4   0   4   4   4   0   0   0   0   0   0   0   0   3   0   0
4206  4   4   0   0   0   4   4   4   0   0   0   0   0   0   0   3   2   0
4207  4   0   4   4   0   4   0   0   4   0   0   4   0   4   0   4   1   2
4208  4   0   4   0   4   0   0   4   0   0   0   4   0   0   0   3   0   0

      18  19
0     5   4
1     2   0
2     0   0
```

(continues on next page)

(continued from previous page)

```
3       0   0
4       0   0
...    ..  ..
4204    1   0
4205    1   0
4206    4   0
4207    0   0
4208    1   0

[4209 rows x 20 columns]
```

For example, the first row presents the learning records of student 0 where he gets 4 point on item 0 and 5 point on item 18.

General features

[4]:
```
data.describe()
```

[4]:
```
                 0            1            2            3            4   \
count  4209.000000  4209.000000  4209.000000  4209.000000  4209.000000
mean      2.983131     2.421478     2.532668     2.827275     2.205750
std       1.741888     1.955317     1.927991     1.821100     1.989625
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       0.000000     0.000000     0.000000     0.000000     0.000000
50%       4.000000     4.000000     4.000000     4.000000     4.000000
75%       4.000000     4.000000     4.000000     4.000000     4.000000
max       4.000000     4.000000     4.000000     4.000000     4.000000

                 5            6            7            8            9   \
count  4209.000000  4209.000000  4209.000000  4209.000000  4209.000000
mean      2.109765     3.573295     2.387265     1.588976     1.663103
std       1.997223     1.234951     1.962381     1.957542     1.971655
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       0.000000     4.000000     0.000000     0.000000     0.000000
50%       4.000000     4.000000     4.000000     0.000000     0.000000
75%       4.000000     4.000000     4.000000     4.000000     4.000000
max       4.000000     4.000000     4.000000     4.000000     4.000000

                10           11           12           13           14  \
count  4209.000000  4209.000000  4209.000000  4209.000000  4209.000000
mean      2.438584     2.986933     1.044429     2.305536     0.022808
std       1.951550     1.739736     1.757162     1.976759     0.301222
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       0.000000     0.000000     0.000000     0.000000     0.000000
50%       4.000000     4.000000     0.000000     4.000000     0.000000
75%       4.000000     4.000000     4.000000     4.000000     0.000000
max       4.000000     4.000000     4.000000     4.000000     4.000000

                15           16           17           18           19
count  4209.000000  4209.000000  4209.000000  4209.000000  4209.000000
mean      4.247327     2.692564     1.226657     2.333333     0.953196
```

(continues on next page)

(continued from previous page)

```
std        1.476007      2.789518      2.238923      2.110050      1.378800
min        0.000000      0.000000      0.000000      0.000000      0.000000
25%        3.000000      0.000000      0.000000      1.000000      0.000000
50%        5.000000      2.000000      0.000000      2.000000      0.000000
75%        5.000000      4.000000      1.000000      4.000000      2.000000
max        6.000000      8.000000      8.000000      9.000000      9.000000
```

[5]:
```
print('The number of records:' + str(len(data)))
```

```
The number of records:4209
```

[6]:
```
data['count']=data.apply(lambda x: x.sum(),axis=1)
data['index1']=data.index
ds=data.loc[:, ['count', 'index1']]
ds['index1'] = ds['index1'].astype(str) + '-'
ds = ds.sort_values(['count']).tail(50)
fig = px.bar(
    ds,
    x = 'count',
    y = 'index1',
    orientation='h',
    title='Top 50 students by score'
)

fig.show("svg")
```

This figure shows the total score of Top 50 students.

[7]:
```
ds=data.loc[:, ['count', 'index1']]
ds = ds.sort_values(['index1'])
fig = px.histogram(
    ds,
    x='index1',
    y='count',
    title='Students score distribution'
)
fig.show("svg")
```

Sort by correct rate

[8]:
```
path = "C:/Users/Administrator/Desktop/Math1/data.txt"
data = pd.read_table(path,header=None)
```

[9]:
```
ds = data.mean()
ds1=pd.DataFrame(columns=['problem_id','count'])
for i in range(len(ds)):
    new=pd.DataFrame({
        'problem_id':int(i),
        'count':ds[i]
```

(continues on next page)

(continued from previous page)

```
    },index=[0]
    )
    ds1=ds1.append(new,ignore_index=True)

ds1=ds1.sort_values(['count'])
ds1['problem_id'] = (ds1['problem_id']).astype(str) + '-'
fig = px.bar(
    ds1,
    x = 'count',
    y = 'problem_id',
    orientation = 'h',
    title = 'Average correct rate of questions'
)

fig.show("svg")
```

This figure presents the average correct rate of questions.It's obvious that students do the best on item 6 but need to improve on item 14.

Sort by problem type

[10]:
```
ds = data.mean()
ds1=pd.DataFrame(columns=['problem_id','count'])
for i in range(len(ds)):
    new=pd.DataFrame({
        'problem_id':int(i),
        'count':ds[i]
    },index=[0]
    )
    ds1=ds1.append(new,ignore_index=True)

data2= [('Obj',ds1[ds1['problem_id']<15]['count'].mean()),
    ('Sub',ds1[ds1['problem_id']>=14]['count'].mean())]
ds2 = pd.DataFrame(
    data=data2,
    columns=['Type','Percent']
)

fig = make_subplots(rows=1,cols=2)
traces = [
    go.Bar(
        x=['wrong','right'],
        y=[
        1-float(ds2[ds2['Type']==item]['Percent']),
        float(ds2[ds2['Type']==item]['Percent'])
        ],
        name='Type:' + str(item),
        text=[
        str(round(100*(1-float(ds2[ds2['Type']==item]['Percent'])))) + '%',
        str(round(100*float(ds2[ds2['Type']==item]['Percent']))) + '%'
```

(continues on next page)

(continued from previous page)

```
        ],
        textposition='auto'
    ) for item in ds2['Type'].tolist()
]
for i in range(len(traces)):
    fig.append_trace(
        traces[i],
        (i //2) + 1,
        (i % 2) + 1
    )
fig.update_layout(
    title_text = 'Average correct rate of questions for every problem type',
)

fig.show("svg")
```

This figure shows that students do better in objective questions but there is a huge gap in the average correct rate of subjective and objective questions. The subjective questions are a challenge to students and their ability to answer them needs to be strengthened.

[11]:
```
path = "C:/Users/Administrator/Desktop/Math1/problemdesc.txt"
data1 = pd.read_table(path,header=0)
```

[12]:
```
count = data1['Full Score'].sum()
data2= [('Obj',data1[data1['Type']=='Obj']['Full Score'].sum()),
    ('Sub',data1[data1['Type']=='Sub']['Full Score'].sum())]
ds = pd.DataFrame(
    data=data2,
    columns=['Type','Percent']
)

ds['Percent']/=count
ds=ds.sort_values('Percent')

fig=px.pie(
    ds,
    names='Type',
    values='Percent',
    title='Problem type',
)
fig.show("svg")
```

Sort by skills

[13]:
```
path1 = "C:/Users/Administrator/Desktop/Math1/q.txt"
data1 = pd.read_table(path1,header=None)
```

[14]:
```
ds1 = data1.sum()
```

[15]:
```
path2 = "C:/Users/Administrator/Desktop/Math1/qnames.txt"
data2 = pd.read_table(path2,header=0)
```

[16]:
```
ds=pd.DataFrame(columns=['skill','count'])
for i in range(len(ds1)):
    new=pd.DataFrame({
        'skill':data2['Skill Names'][i],
        'count':ds1[i]
    },index=[0]
    )
    ds=ds.append(new,ignore_index=True)
```

[17]:
```
ds=ds.sort_values(['count'])
fig = px.bar(
    ds,
    x='count',
    y='skill',
    orientation='h',
    title='Skill count'
)
fig.show("svg")
```

This figure shows that calculation is the most important skill in the test as almost every item is related to it. Besides, reasoning and demonstration is also the key to do better in the test.

math2015-Math2 Data Analysis

[1]:
```
import numpy as np
import pandas as pd

import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objs as go
```

[2]:
```
path = "C:/Users/Administrator/Desktop/Math2/rawdata.txt"
data = pd.read_table(path,header=None)
```

RECORDS

The learning records are saved in a score matrix. Each row corresponds to the records of a student on different test items.

[3]:
```
pd.set_option('display.max_rows',10)
data
```

[3]:
```
      0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17  \
0     3   0   0   0   3   0   3   3   3   3   0   0   0   0   0   0   0   0
1     3   3   3   3   3   3   3   0   3   0   0   3   4   0   0   4  12  12
2     3   3   3   0   0   0   3   3   0   0   0   3   4   0   0   0   8   0
3     0   3   3   3   0   3   3   0   3   3   0   3   0   0   0   0  12  12
4     3   3   0   0   3   3   0   0   3   0   0   3   0   0   0   4   4  12
...  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
3906  3   3   0   3   3   0   0   0   3   0   0   3   0   0   0   0   6   0
3907  3   0   3   3   3   3   3   3   0   0   3   3   4   4   0   0   6   2
3908  3   3   0   3   3   3   3   0   3   3   0   0   0   0   0   0   2  12
3909  3   3   0   3   3   3   0   0   3   0   0   3   0   0   4   0   1  12
3910  3   3   3   3   0   3   0   3   3   0   0   0   0   0   0   0   6   8

      18  19
0      6   0
1      6   1
2      6   0
3      6   0
4      6   0
...   ..  ..
3906   6   1
3907   6   0
3908   6   4
3909   3   0
3910   6   2

[3911 rows x 20 columns]
```

For example, the first row presents the learning records of student 0 where he gets 3 point on item 0 and 6 point on item 18.

General features

[4]:
```
data.describe()
```

[4]:
```
                 0            1            2            3            4  \
count  3911.000000  3911.000000  3911.000000  3911.000000  3911.000000
mean      2.633342     2.120174     1.608540     2.162363     1.300946
std       0.982743     1.365965     1.496259     1.346009     1.486924
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       3.000000     0.000000     0.000000     0.000000     0.000000
50%       3.000000     3.000000     3.000000     3.000000     0.000000
75%       3.000000     3.000000     3.000000     3.000000     3.000000
max       3.000000     3.000000     3.000000     3.000000     3.000000
```

(continues on next page)

(continued from previous page)

```
                 5            6            7            8            9  \
count  3911.000000  3911.000000  3911.000000  3911.000000  3911.00000
mean      1.893122     1.765789     0.441064     2.145487     1.72360
std       1.447754     1.476453     1.062517     1.354184     1.48343
min       0.000000     0.000000     0.000000     0.000000     0.00000
25%       0.000000     0.000000     0.000000     0.000000     0.00000
50%       3.000000     3.000000     0.000000     3.000000     3.00000
75%       3.000000     3.000000     0.000000     3.000000     3.00000
max       3.000000     3.000000     3.000000     3.000000     3.00000

                10           11           12           13           14  \
count  3911.000000  3911.000000  3911.000000  3911.000000  3911.000000
mean      1.418307     1.548709     1.557658     0.413194     0.805932
std       1.497965     1.499401     1.950719     1.217549     1.604637
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       0.000000     0.000000     0.000000     0.000000     0.000000
50%       0.000000     3.000000     0.000000     0.000000     0.000000
75%       3.000000     3.000000     4.000000     0.000000     0.000000
max       3.000000     3.000000     4.000000     4.000000     4.000000

                15           16           17           18           19
count  3911.000000  3911.000000  3911.000000  3911.000000  3911.000000
mean      0.773204     4.384301     5.333419     6.609051     1.724367
std       1.579750     3.812531     4.820757     3.085368     2.542170
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       0.000000     0.000000     0.000000     6.000000     0.000000
50%       0.000000     4.000000     5.000000     6.000000     0.000000
75%       0.000000     6.000000    11.000000     7.000000     3.000000
max       4.000000    12.000000    12.000000    12.000000    10.000000
```

[5]:
```
print('The number of records:' + str(len(data)))
```

```
The number of records:3911
```

[6]:
```
data['count']=data.apply(lambda x: x.sum(),axis=1)
data['index1']=data.index
ds=data.loc[:, ['count', 'index1']]
ds['index1'] = ds['index1'].astype(str) + '-'
ds = ds.sort_values(['count']).tail(50)
fig = px.bar(
    ds,
    x = 'count',
    y = 'index1',
    orientation='h',
    title='Top 50 students by score'
)

fig.show("svg")
```

This figure shows the total score of Top 50 students.

[7]:
```
ds=data.loc[:, ['count', 'index1']]
ds = ds.sort_values(['index1'])
fig = px.histogram(
    ds,
    x='index1',
    y='count',
    title='Students score distribution'
)
fig.show("svg")
```

Sort by correct rate

[8]:
```
path = "C:/Users/Administrator/Desktop/Math2/data.txt"
data = pd.read_table(path,header=None)
```

[9]:
```
ds = data.mean()
ds1=pd.DataFrame(columns=['problem_id','count'])
for i in range(len(ds)):
    new=pd.DataFrame({
        'problem_id':int(i),
        'count':ds[i]
    },index=[0]
    )
    ds1=ds1.append(new,ignore_index=True)

ds1=ds1.sort_values(['count'])
ds1['problem_id'] = (ds1['problem_id']).astype(str) + '-'
fig = px.bar(
    ds1,
    x = 'count',
    y = 'problem_id',
    orientation = 'h',
    title = 'Average correct rate of questions'
)

fig.show("svg")
```

This figure presents the average correct rate of questions.It's obvious that students do the best on item 0 but need to improve on item 13.

Sort by problem type

[10]:
```
ds = data.mean()
ds1=pd.DataFrame(columns=['problem_id','count'])
for i in range(len(ds)):
    new=pd.DataFrame({
        'problem_id':int(i),
        'count':ds[i]
    },index=[0]
    )
    ds1=ds1.append(new,ignore_index=True)

data2= [('Obj',ds1[ds1['problem_id']<15]['count'].mean()),
    ('Sub',ds1[ds1['problem_id']>=14]['count'].mean())]
ds2 = pd.DataFrame(
    data=data2,
    columns=['Type','Percent']
)

fig = make_subplots(rows=1,cols=2)
traces = [
    go.Bar(
        x=['wrong','right'],
        y=[
        1-float(ds2[ds2['Type']==item]['Percent']),
        float(ds2[ds2['Type']==item]['Percent'])
        ],
        name='Type:' + str(item),
        text=[
        str(round(100*(1-float(ds2[ds2['Type']==item]['Percent'])))) + '%',
        str(round(100*float(ds2[ds2['Type']==item]['Percent']))) + '%'
        ],
        textposition='auto'
    ) for item in ds2['Type'].tolist()
]
for i in range(len(traces)):
    fig.append_trace(
        traces[i],
        (i //2) + 1,
        (i % 2) + 1
    )
fig.update_layout(
    title_text = 'Average correct rate of questions for every problem type',
)

fig.show("svg")
```

This figure shows that students do better in objective questions and their ability to answer subjective questions needs to be strengthened.

[11]:
```
path = "C:/Users/Administrator/Desktop/Math2/problemdesc.txt"
data1 = pd.read_table(path,header=0)
```

[12]:
```
count = data1['Full Score'].sum()
data2= [('Obj',data1[data1['Type']=='Obj']['Full Score'].sum()),
    ('Sub',data1[data1['Type']=='Sub']['Full Score'].sum())]
ds = pd.DataFrame(
    data=data2,
    columns=['Type','Percent']
)

ds['Percent']/=count
ds=ds.sort_values('Percent')

fig=px.pie(
    ds,
    names='Type',
    values='Percent',
    title='Problem type',
)
fig.show("svg")
```

Sort by skills

[13]:
```
path1 = "C:/Users/Administrator/Desktop/Math2/q.txt"
data1 = pd.read_table(path1,header=None)
```

[14]:
```
ds1 = data1.sum()
```

[15]:
```
path2 = "C:/Users/Administrator/Desktop/Math2/qnames.txt"
data2 = pd.read_table(path2,header=0)
```

[16]:
```
ds=pd.DataFrame(columns=['skill','count'])
for i in range(len(ds1)):
    new=pd.DataFrame({
        'skill':data2['Skill Names'][i],
        'count':ds1[i]
    },index=[0]
    )
    ds=ds.append(new,ignore_index=True)
```

[17]:
```
ds=ds.sort_values(['count'])
fig = px.bar(
    ds,
    x='count',
    y='skill',
    orientation='h',
    title='Skill count'
)
fig.show("svg")
```

This figure shows that calculation is the most important skills in the test as almost every item is related to it. This proves that calculation is a necessary skill for math tests.

6.7.5 OLI_Fall2011

OLI data in fall, 2011problem

[1]:
```
%matplotlib inline
import pandas as pd
import numpy as np
# global configuration: show every rows and cols
pd.set_option('display.max_rows', None)
pd.set_option('max_colwidth',None)
pd.set_option('display.max_columns', None)
```

1. Data Description

1.1 Column Description

[2]:
```
# help_table3: the description for data by problems
df3 = pd.read_csv('OLI_data/help_table3.csv',sep=',',encoding="gbk")
df3 = df3.loc[:, ['Field', 'Annotation']]
df3
```

[2]:
```
                          Field  \
0                           Row
1                        Sample
2               Anon Student ID
3             Problem Hierarchy
4                  Problem Name
5                  Problem View
6            Problem Start Time
7              Problem End Time
8                 Latency (sec)
9    Steps Missing Start Times
10                        Hints
11                   Incorrects
12                     Corrects
13                 Avg Corrects
14                        Steps
15         Avg Assistance Score
16        Correct First Attempts
17                    Condition
18                          KCs
19            Steps without KCs
20                      KC List

                                                                                                   ␣
↪                                                                                                  ␣
↪                                                                                                  ␣
↪                                                                                                  ␣
↪                                                                                                  ␣
↪                                                                                               ␣
↪Annotation
```

(continues on next page)

(continued from previous page)

```
0                                                              A row
counter.
1                                                   The sample that
includes this problem. If you select more than one sample to export,
 problems that occur in more than one sample will be duplicated in the export.
2                                         The student that worked on the
problem.
3                 The location in the curriculum hierarchy where this problem
occurs.
4                                                      The name of the
problem.
5                                The number of times the student encountered the
problem so far. This counter                  increases with each instance of
the same problem. See "Problem View" in the "By                  Student-Step"
table above.
6   If the problem start time is not given in the original log data, then it is set
             to the time of the last transaction of the prior problem. If there is no
prior problem                    for the session, the time of the earliest transaction
is used. Earliest transaction time                    is equivalent to the minimum
transaction time for the earliest step of the problem.                For more
detail on how problem start time is determined, see Determining
Problem Start Time.
7                   Derived from the maximum transaction time of the latest step of the
problem.
```

(continues on next page)

(continued from previous page)

```
8 The amount of time the student spent on this problem. Specifically, the difference between the problem start time and the last transaction on this problem.
9 The number of steps (from the student-step table) with "Step Start Time" values of "null".
10 Total number of hints the student requested for this problem.
11 Total number of incorrect attempts the student made on this problem.
12 Total number of correct attempts the student made for this problem.
13 The total number of correct attempts / total number of steps in the problem.
14 Total number of steps the student took while working on the problem.
15 Calculated as (total hints requested + total incorrect attempts) / total steps.
```

(continues on next page)

(continued from previous page)

```
16                                                                                                    ␣
↪                                                                                                     ␣
↪                                                                                                     ␣
↪                                                                                                     ␣
↪                                                                                                     ␣
↪                    Total number of correct first attempts made by the student for this␣
↪problem.
17                                                                                                    ␣
↪                                                                                      The name and␣
↪type of the condition the student is assigned to. In the case of a student                          ␣
↪       assigned to multiple conditions (factors in a factorial design), condition␣
↪names are                     separated by a comma and space. This differs from the␣
↪transaction format, which optionally                    has "Condition Name" and
↪"Condition Type" columns.
18                                                                                                    ␣
↪                                                                                                     ␣
↪                                                                                                     ␣
↪                                                                                                     ␣
↪                                                                                                     ␣
↪                                  Total number of KCs practiced by the student for this␣
↪problem.
19                                                                                                    ␣
↪                                                                                                     ␣
↪                                                                                                     ␣
↪                                                                                                     ␣
↪                                                                                                     ␣
↪      Total number of steps in this problem (performed by the student) without an␣
↪assigned KC.
20                                                                                                    ␣
↪                                                                                                     ␣
↪                                                                                                     ␣
↪                                                                                                     ␣
↪                                                                                                     ␣
↪                          Comma-delimited list of KCs practiced by the student for this␣
↪problem.
```

1.2 Summarization of Data

This table organizes the data as student-problem

[3]:
```
df_problem =  pd.read_csv('OLI_data/AllData_problem_2011F.csv',low_memory=False) # sep="\
↪t"
df_problem.head(2)
```

[3]:
```
   Row    Sample                       Anon Student Id  \
0    1  All Data  Stu_00b2b35fd027e7891e8a1a527125dd65
1    2  All Data  Stu_00b2b35fd027e7891e8a1a527125dd65

                                                                                                      ␣
↪Problem Hierarchy  \
0  sequence Statics, unit Concentrated Forces and Their Effects, module Introduction to␣
↪Free Body Diagrams
```

(continues on next page)

(continued from previous page)

```
1                     sequence Statics, unit Concentrated Forces and Their Effects,␣
↪module Effects of Force

  Problem Name  Problem View Problem Start Time Problem End Time  \
0   _m2_assess             1     2011/9/21 17:35  2011/9/21 17:35
1  tutor_03_01             1     2011/9/21 17:49  2011/9/21 17:49

   Latency (sec)  Steps Missing Start Times  Hints  Incorrects  Corrects  \
0              0                          0      0           9        12
1              9                          0      0           0         3

   Avg Corrects  Steps  Avg Assistance Score  Correct First Attempts  \
0         0.571     21                 0.429                      12
1         1.000      3                 0.000                       3

   Condition  KCs (F2011)  Steps without KCs (F2011)  \
0        NaN            5                          0
1        NaN            1                          0

                                                                                     ␣
↪            KC List (F2011)  \
0  gravitational_forces, identify_interaction, represent_interaction_cord, represent_
↪interaction_spring, simple_step
1                                                                               ␣
↪distinguish_rotation_translation

   KCs (Single-KC)  Steps without KCs (Single-KC) KC List (Single-KC)  \
0                1                              0           Single-KC
1                1                              0           Single-KC

   KCs (Unique-step)  Steps without KCs (Unique-step) KC List (Unique-step)
0                  0                               21                     .
1                  3                                0    KC523, KC680, KC768
```

2. Data Analysis

[4]:
```
df_problem.describe()
```

[4]:
```
                Row  Problem View  Latency (sec)  Steps Missing Start Times  \
count  45002.000000  45002.000000   45002.000000               45002.000000
mean   22501.500000      1.221146      85.639883                   0.007000
std    12991.102744      1.140622     301.895374                   0.106748
min        1.000000      1.000000       0.000000                   0.000000
25%    11251.250000      1.000000       0.000000                   0.000000
50%    22501.500000      1.000000      20.000000                   0.000000
75%    33751.750000      1.000000      73.000000                   0.000000
max    45002.000000     32.000000   20426.000000                   8.000000

              Hints    Incorrects      Corrects  Avg Corrects         Steps  \
count  45002.000000  45002.000000  45002.000000  45002.000000  45002.000000
mean       0.620217      1.644460      4.176325      0.959571      4.331963
```

(continues on next page)

(continued from previous page)

```
std       1.956302     3.378211     5.125742     0.358850     5.079484
min       0.000000     0.000000     0.000000     0.000000     1.000000
25%       0.000000     0.000000     1.000000     1.000000     1.000000
50%       0.000000     1.000000     3.000000     1.000000     3.000000
75%       0.000000     2.000000     5.000000     1.000000     5.000000
max      50.000000   413.000000   232.000000    19.333000    32.000000

       Avg Assistance Score  Correct First Attempts  Condition   KCs (F2011)  \
count          45002.000000            45002.000000        0.0  45002.000000
mean               0.928014                3.219479        NaN      1.223923
std                2.221907                4.603916        NaN      1.733856
min                0.000000                0.000000        NaN      0.000000
25%                0.000000                1.000000        NaN      0.000000
50%                0.250000                2.000000        NaN      1.000000
75%                1.000000                4.000000        NaN      2.000000
max              210.500000               32.000000        NaN      9.000000

       Steps without KCs (F2011)  KCs (Single-KC)  \
count               45002.000000          45002.0
mean                    1.798920              1.0
std                     3.830471              0.0
min                     0.000000              1.0
25%                     0.000000              1.0
50%                     0.000000              1.0
75%                     2.000000              1.0
max                    32.000000              1.0

       Steps without KCs (Single-KC)  KCs (Unique-step)  \
count                        45002.0       45002.000000
mean                             0.0           4.289654
std                              0.0           5.084490
min                              0.0           0.000000
25%                              0.0           1.000000
50%                              0.0           3.000000
75%                              0.0           5.000000
max                              0.0          32.000000

       Steps without KCs (Unique-step)
count                     45002.000000
mean                          0.042309
std                           0.557118
min                           0.000000
25%                           0.000000
50%                           0.000000
75%                           0.000000
max                          29.000000
```

1Analysis for Null and Unique value of column attributes

[5]:
```
def work_col_analysis(df_work):
    num_nonull_toal = df_work.notnull().sum()  # Not Null
    dict_col_1 = {'col_name':num_nonull_toal.index,'num_nonull':num_nonull_toal.values}
    df_work_col_1 = pd.DataFrame(dict_col_1)

    num_null_toal = df_work.isnull().sum()  # Null
    dict_col_2 = {'col_name':num_null_toal.index,'num_null':num_null_toal.values}
    df_work_col_2 = pd.DataFrame(dict_col_2)

    num_unique_toal = df_work.apply(lambda col: len(col.unique()))   # axis=0
    print(type(num_unique_toal))
    dict_col_3 = {'col_name':num_unique_toal.index,'num_unique':num_unique_toal.values}
    df_work_col_3 = pd.DataFrame(dict_col_3)

    df_work_col = pd.merge(df_work_col_1, df_work_col_2, on=['col_name'])
    df_work_col = pd.merge(df_work_col, df_work_col_3, on=['col_name'])
    return df_work_col
print("-------------------num_unique_toal and num_nonull_toal----------------------")
df_result = work_col_analysis(df_problem)
df_result
```

```
-------------------num_unique_toal and num_nonull_toal----------------------
<class 'pandas.core.series.Series'>
```

[5]:

	col_name	num_nonull	num_null	num_unique
0	Row	45002	0	45002
1	Sample	45002	0	1
2	Anon Student Id	45002	0	333
3	Problem Hierarchy	45002	0	27
4	Problem Name	45002	0	300
5	Problem View	45002	0	32
6	Problem Start Time	45002	0	25983
7	Problem End Time	45002	0	25884
8	Latency (sec)	45002	0	1290
9	Steps Missing Start Times	45002	0	8
10	Hints	45002	0	35
11	Incorrects	45002	0	37
12	Corrects	45002	0	51
13	Avg Corrects	45002	0	195
14	Steps	45002	0	31
15	Avg Assistance Score	45002	0	335
16	Correct First Attempts	45002	0	33
17	Condition	0	45002	1
18	KCs (F2011)	45002	0	10
19	Steps without KCs (F2011)	45002	0	31
20	KC List (F2011)	45002	0	170
21	KCs (Single-KC)	45002	0	1
22	Steps without KCs (Single-KC)	45002	0	1
23	KC List (Single-KC)	45002	0	1
24	KCs (Unique-step)	45002	0	32
25	Steps without KCs (Unique-step)	45002	0	16
26	KC List (Unique-step)	45002	0	1470

2Analysis for Discrete value of column attributes

Columns with a small number of discrete values may represent very informative, so identify these columns first and analyze them one by one

[6]:
```
discrete_cols = []
series = []
cols = list(df_problem.columns.values)

for col in cols:
    if len(df_problem[col].unique().tolist()) <= 20 and len(df_problem[col].unique().
↪tolist()) >= 2:
        discrete_cols.append(col)
        series.append(df_problem[col].unique().tolist())

for a,b in zip(discrete_cols,series):
    print(a," : ",b)
    print("-"*80)
```

```
Steps Missing Start Times  :  [0, 1, 2, 5, 7, 6, 3, 8]
--------------------------------------------------------------------------------
KCs (F2011)  :  [5, 1, 4, 2, 3, 9, 0, 8, 6, 7]
--------------------------------------------------------------------------------
Steps without KCs (Unique-step)  :  [21, 0, 17, 15, 9, 2, 5, 1, 4, 3, 12, 10, 8, 11, 14,␣
↪29]
--------------------------------------------------------------------------------
```

3Data Cleaning

Data Cleaning Suggestions - Redundant columns: Columns that are all NULL or Single value. - Others

[7]:
```
df_problem_clear = df_problem.copy(deep=True) # deep copy
```

[8]:
```
# Clear all redundant columns directly.
cols = list(df_problem.columns.values)
drop_cols = []
for col in cols:
    if len(df_problem_clear[col].unique().tolist()) == 1:
        df_problem_clear.drop(col,axis =1,inplace=True)
        drop_cols.append(col)

print("the cols num before clear: ",len(df_problem.columns.to_list()))
print("the cols num after clear:",len(df_problem_clear.columns.to_list()))
for col in drop_cols:
    print("drop:---",col)
```

```
the cols num before clear:  27
the cols num after clear: 22
drop:--- Sample
drop:--- Condition
drop:--- KCs (Single-KC)
drop:--- Steps without KCs (Single-KC)
drop:--- KC List (Single-KC)
```

[9]:
```
df_problem_clear.head(2)
```

[9]:
```
   Row                           Anon Student Id  \
0    1  Stu_00b2b35fd027e7891e8a1a527125dd65
1    2  Stu_00b2b35fd027e7891e8a1a527125dd65

                                                                                 ␣
↪Problem Hierarchy  \
0  sequence Statics, unit Concentrated Forces and Their Effects, module Introduction to␣
↪Free Body Diagrams
1                    sequence Statics, unit Concentrated Forces and Their Effects,␣
↪module Effects of Force

  Problem Name  Problem View Problem Start Time Problem End Time  \
0   _m2_assess             1     2011/9/21 17:35  2011/9/21 17:35
1  tutor_03_01             1     2011/9/21 17:49  2011/9/21 17:49

   Latency (sec)  Steps Missing Start Times  Hints  Incorrects  Corrects  \
0             0                          0      0           9        12
1             9                          0      0           0         3

   Avg Corrects  Steps  Avg Assistance Score  Correct First Attempts  \
0         0.571     21                 0.429                      12
1         1.000      3                 0.000                       3

   KCs (F2011)  Steps without KCs (F2011)  \
0            5                          0
1            1                          0

                                                                                 ␣
↪            KC List (F2011)  \
0  gravitational_forces, identify_interaction, represent_interaction_cord, represent_
↪interaction_spring, simple_step
1                                                                              ␣
↪distinguish_rotation_translation

   KCs (Unique-step)  Steps without KCs (Unique-step) KC List (Unique-step)
0                  0                               21                     .
1                  3                                0    KC523, KC680, KC768
```

[10]:
```
# the remaining columns
print("-------------------num_unique_toal and num_nonull_toal----------------------")
df_result = work_col_analysis(df_problem_clear)
df_result
```

```
-------------------num_unique_toal and num_nonull_toal----------------------
<class 'pandas.core.series.Series'>
```

[10]:
```
                          col_name  num_nonull  num_null  num_unique
0                              Row       45002         0       45002
1                  Anon Student Id       45002         0         333
2                Problem Hierarchy       45002         0          27
3                     Problem Name       45002         0         300
4                     Problem View       45002         0          32
```

(continues on next page)

(continued from previous page)

```
5               Problem Start Time       45002        0      25983
6                 Problem End Time       45002        0      25884
7                   Latency (sec)        45002        0       1290
8          Steps Missing Start Times     45002        0          8
9                            Hints       45002        0         35
10                      Incorrects       45002        0         37
11                        Corrects       45002        0         51
12                    Avg Corrects       45002        0        195
13                           Steps       45002        0         31
14            Avg Assistance Score       45002        0        335
15           Correct First Attempts      45002        0         33
16                    KCs (F2011)        45002        0         10
17         Steps without KCs (F2011)     45002        0         31
18                 KC List (F2011)       45002        0        170
19                KCs (Unique-step)      45002        0         32
20  Steps without KCs (Unique-step)      45002        0         16
21            KC List (Unique-step)      45002        0       1470
```

3. Data Visualization

[11]:
```
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objs as go
import matplotlib.pyplot as plt
```

[12]:
```
# The distribution of continuous values
def show_value_counts_histogram(colname, sort = True):
    # create the bins
    start = int(df_problem_clear[colname].min()/10)*10
    end = int(df_problem_clear[colname].quantile(q=0.95)/10+1)*10
    problem = int((end - start)/20)
    print(start, end, problem)
    counts, bins = np.histogram(df_problem_clear[colname],bins=range(start, end,␣
↪problem))
    bins = 0.5 * (bins[:-1] + bins[1:])

    fig = px.bar(x=bins, y=counts, labels={'x': colname, 'y':'count'})
    fig.show("svg")

# Box distribution of continuous values
def show_value_counts_box(colname, sort = True):
    # way1: plotly (too costy for box-plot)
    # fig = px.box(df_problem_clear, y=colname)
    # fig.show("svg")
    # way2: matplotlib
    plt.figure(figsize=(10,5))
    plt.title('Box-plot for '+ colname,fontsize=20)#
    plt.boxplot([df_problem_clear[colname].tolist()])
    plt.show("svg")
```

(continues on next page)

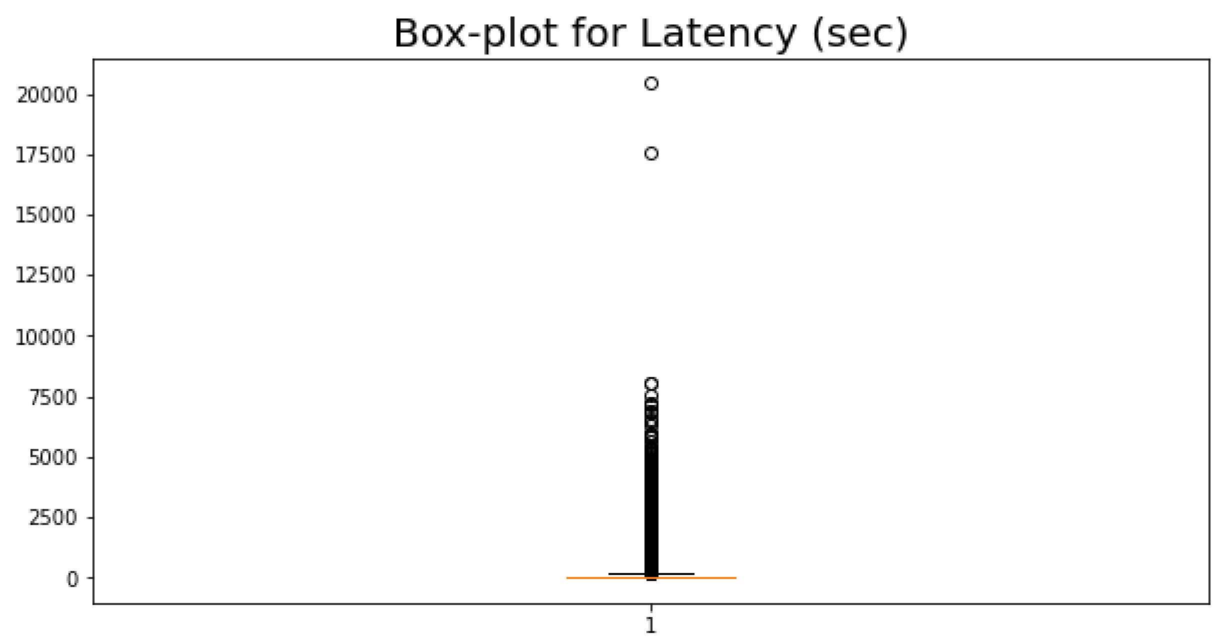
(continued from previous page)

```
# Histogram of discrete values
def show_value_counts_bar(colname, sort = True):
    ds = df_problem_clear[colname].value_counts().reset_index()
    ds.columns = [
        colname,
        'Count'
    ]
    if sort:
        ds = ds.sort_values(by='Count', ascending=False)
    # histogram
    fig = px.bar(
        ds,
        x = colname,
        y = 'Count',
        title = colname + ' distribution'
    )
    fig.show("svg")


# Pie of discrete values
def show_value_counts_pie(colname, sort = True):
    ds = df_problem_clear[colname].value_counts().reset_index()
    ds.columns = [
        colname,
        'percent'
    ]
    ds['percent'] /= len(df_problem_clear)
    if sort:
        ds = ds.sort_values(by='percent', ascending=False)
    fig = px.pie(
        ds,
        names = colname,
        values = 'percent',
        title = colname+ 'Percentage',
    )
    fig.show("svg")
```

1sort by single attributes

[13]:
```
# Bar
show_value_counts_bar('KCs (F2011)')
show_value_counts_bar('Problem Hierarchy')
```

[14]:
```
# analysis for "duration"
# It is obvious that there are unreasonable outliers

show_value_counts_box('Latency (sec)')
show_value_counts_histogram('Latency (sec)')
```




```
0 340 17
```

2group by Problem Name, sorted by meam(avg corrects)

[15]:
```
# Classification Statistic

# Problem Name,Avg Corrects, Avg Assistance Score
df_problem_group1 = df_problem_clear.groupby(['Problem Name'])['Avg Corrects'].mean().
↪reset_index()
df_problem_group1.columns = ["Problem Name","Avg Corrects"]
df_problem_group1 = df_problem_group1.sort_values(by='Avg Corrects', ascending=False)
fig = px.bar(df_problem_group1, x="Problem Name", y="Avg Corrects", title="Questions␣
↪sorted by the average accuracy")
fig.show("svg")
```

OLI data in fall, 2011step

[1]:
```
%matplotlib inline
import pandas as pd
import numpy as np
# global configuration: show every rows and cols
pd.set_option('display.max_rows', None)
pd.set_option('max_colwidth',None)
pd.set_option('display.max_columns', None)
```

1. Data Description

1.1 Column Description

[2]:
```
# help_table2: the description for data by steps
df2 = pd.read_csv('OLI_data/help_table2.csv',sep=',',encoding="gbk")
df2 = df2.loc[:, ['Field', 'Annotation']]
df2
```

[2]:
```
                               Field  \
0                                Row
1                             Sample
2                     Anon Student ID
3                  Problem Hierarchy
4                       Problem Name
5                       Problem View
6                          Step Name
7                    Step Start Time
8             First Transaction Time
9           Correct Transaction Time
10                     Step End Time
11               Step Duration (sec)
12       Correct Step Duration (sec)
13         Error Step Duration (sec)
14                     First Attempt
15                        Incorrects
16                             Hints
17                          Corrects
18                         Condition
19                  KC (model_name)
20         Opportunity (model_name)
21  Predicted Error Rate (model_name)

                                                                          Annotation
0
```

(continues on next page)

(continued from previous page)

```
1 ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪ The sample that
↪includes this step. If you select more than one sample to export,
↪steps that occur in more than one sample will be duplicated in the export.
2 ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪ The student that performed the step.
3 ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪ The location in the curriculum hierarchy where this step occurs.
4 ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪ ↪
```

(continues on next page)

(continued from previous page)

```
  The number of times the student encountered the problem so far. This counter increases with each instance of the same problem.   Note that problem view increases regardless of whether or not the step was encountered in previous problem views. For example, a step can have a "Problem View" of "3", indicating the problem was viewed three times by this student, but that same step need not have been encountered by that student in all instances of the problem. If this number does not increase as you expect it to, it might be that DataShop has identified similar problems as distinct: two problems with the same "Problem Name" are considered different "problems" by DataShop if the following logged values are not identical: problem name, context, tutor_flag (whether or not the problem or activity is tutored) and "other" field. For more on the logging of these fields, see the description of the "problem" element in the Guide to the Tutor Message Format. For more detail on how problem view is determined, see Determining Problem View.
 Formed by concatenating the "selection" and "action". Also see the glossary entry for "step".
 The step start time is determined one of three ways: If it's the first step of the problem, the step start time is the same as the problem start time If it's a subsequent step, then the step start time is the time of the preceding transaction, if that transaction is within 10 minutes. If it's a subsequent step and the elapsed time between the previous transaction and the first transaction of this step is more than 10 minutes, then the step start time is set to null as it's considered an unreliable value. For a visual example, see the Examples page.

```

(continues on next page)

(continued from previous page)

```
The time of the correct attempt toward the step, if there was one.

The time of the last transaction toward the step.

The elapsed time of the step in seconds, calculated by adding all of the durations for transactions that were attributed to the step. See the glossary entry for more detail. This column                    was previously labeled "Assistance Time". It differs from "Assistance Time" in that its                    values are derived by summing transaction durations, not finding the difference between only two points in time (step start time and the last correct attempt).

The step duration if the first attempt for the step was correct. This might                    also be described as "reaction time" since it's the duration of time from the previous                    transaction or problem start event to the correct attempt. See the glossary entry for more detail.
```

(continues on next page)

(continued from previous page)

```
13                                                                                            ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                    The step duration if the first attempt for the␣
↪step was an error (incorrect                    attempt or hint request).
14                                                                                            ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                  The tutor's response to the student's first attempt on the step.␣
↪Example values                           are "hint", "correct", and "incorrect".
15                                                                                            ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪        Total number of incorrect attempts by the student on the step.
16                                                                                            ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
↪                                                                                             ␣
```

(continues on next page)

(continued from previous page)

```
                                Total correct attempts by the student for the step. (Only increases if the step                    is encountered more than once.)
                                                            The name and type of the condition the student is assigned to. In the case of a                     student assigned to multiple conditions (factors in a factorial design), condition names are separated by a comma and space. This differs from the transaction format, which                  optionally has "Condition Name" and "Condition Type" columns.
                                                                 (Only shown when the "Knowledge Components" option is selected.) Knowledge                    component(s) associated with the correct performance of this step. In the case of                     multiple KCs assigned to a single step, KC names are separated by two tildes ("~~").
                    (Only shown when the "Knowledge Components" option is selected.) An opportunity                    is the first chance on a step for a student to demonstrate whether he or she has learned                    the associated knowledge component. Opportunity number is therefore a count that                    increases by one each time the student encounters a step with the listed knowledge      component. In the case of multiple KCs assigned to a single step, opportunity             number values are separated by two tildes ("~~") and are given in the same order as                    the KC names. Check here to see how opportunity
```

(continues on next page)

(continued from previous page)

```
21
↪
↪
↪
↪
↪                                   A hypothetical error rate based on the Additive
↪Factor Model (AFM)                     algorithm. A value of "1" is a prediction that a
↪student's first attempt will be an                    error (incorrect attempt or hint
↪request); a value of "0" is a prediction that the                    student's first
↪attempt will be correct. For specifics, see below "Predicted Error Rate" and how it's
↪calculated.                          In the case of multiple KCs assigned to a single step,
↪Datashop implements a compensatory sum across all of the KCs,                    thus
↪a single value of predicted error rate is provided (i.e., the same predicted error
↪rate for each KC assigned to a step).                       For more detail on Datashop
↪'s implementation for multi-skilled step, see                        Model Values page.
```

1.2 Summarization of Data

This table organizes the data as student-problem-step

[3]:
```
df_step =  pd.read_csv('OLI_data/AllData_student_step_2011F.csv',low_memory=False) # sep=
↪"\t"
df_step.head(2)
```

[3]:
```
   Row    Sample                         Anon Student Id  \
0    1  All Data  Stu_00b2b35fd027e7891e8a1a527125dd65
1    2  All Data  Stu_00b2b35fd027e7891e8a1a527125dd65


↪Problem Hierarchy  \
0  sequence Statics, unit Concentrated Forces and Their Effects, module Introduction to
↪Free Body Diagrams
1  sequence Statics, unit Concentrated Forces and Their Effects, module Introduction to
↪Free Body Diagrams

  Problem Name  Problem View                    Step Name  Step Start Time  \
0   _m2_assess             1  q1_point1i1 UpdateComboBox  2011/9/21 17:35
1   _m2_assess             1  q1_point3i3 UpdateComboBox  2011/9/21 17:35

  First Transaction Time Correct Transaction Time    Step End Time  \
0        2011/9/21 17:35          2011/9/21 17:35  2011/9/21 17:35
1        2011/9/21 17:35          2011/9/21 17:35  2011/9/21 17:35

  Step Duration (sec) Correct Step Duration (sec) Error Step Duration (sec)  \
0              23.13                        23.13                         .
1              23.13                        23.13                         .

  First Attempt  Incorrects  Hints  Corrects Condition            KC (F2011)  \
0       correct           0      0         1         .  identify_interaction
1       correct           0      0         1         .  gravitational_forces
```

(continues on next page)

(continued from previous page)

```
  Opportunity (F2011)  Predicted Error Rate (F2011) KC (Single-KC)  \
0                   1                        0.3991      Single-KC
1                   1                        0.1665      Single-KC

   Opportunity (Single-KC)  Predicted Error Rate (Single-KC) KC (Unique-step)  \
0                        1                            0.4373              NaN
1                        2                            0.4373              NaN

   Opportunity (Unique-step)  Predicted Error Rate (Unique-step)
0                        NaN                                 NaN
1                        NaN                                 NaN
```

2. Data Analysis

[4]:
```
df_step.describe()
```

[4]:
```
                  Row   Problem View     Incorrects          Hints  \
count   194947.000000  194947.000000  194947.000000  194947.000000
mean     97474.000000       1.133154       0.379611       0.143172
std      56276.495801       0.760515       1.373797       0.852520
min          1.000000       1.000000       0.000000       0.000000
25%      48737.500000       1.000000       0.000000       0.000000
50%      97474.000000       1.000000       0.000000       0.000000
75%     146210.500000       1.000000       0.000000       0.000000
max     194947.000000      32.000000     413.000000      43.000000

             Corrects  Predicted Error Rate (F2011)  Opportunity (Single-KC)  \
count   194947.000000                113992.000000            194947.000000
mean         0.964072                     0.237508               419.751066
std          0.480346                     0.158128               288.365862
min          0.000000                     0.002900                 1.000000
25%          1.000000                     0.117900               171.000000
50%          1.000000                     0.201400               382.000000
75%          1.000000                     0.319500               635.000000
max         86.000000                     0.969300              1410.000000

       Predicted Error Rate (Single-KC)  Opportunity (Unique-step)  \
count                     194947.000000              193043.000000
mean                           0.252233                   1.035971
std                            0.086406                   0.384182
min                            0.038600                   1.000000
25%                            0.188100                   1.000000
50%                            0.240500                   1.000000
75%                            0.294700                   1.000000
max                            0.773600                  24.000000

       Predicted Error Rate (Unique-step)
count                                 0.0
mean                                  NaN
std                                   NaN
min                                   NaN
```

(continues on next page)

(continued from previous page)

```
25%                                      NaN
50%                                      NaN
75%                                      NaN
max                                      NaN
```

[5]:
```
num_total = len(df_step)
num_students = len(df_step['Anon Student Id'].unique())
num_problems = len(df_step['Problem Name'].unique())
num_kcs = len(df_step['KC (F2011)'].unique())
num_null_condition = df_step['Condition'].isnull().sum()  #
print("num_total:",num_total)
print("num_students:",num_students)
print("num_problems:",num_problems)
print("num_kcs:",num_kcs)
print("num_null_condition:",num_null_condition)

n_incorrects = df_step['Incorrects'].sum()
n_hints = df_step['Hints'].sum()
n_corrects = df_step['Corrects'].sum()
print("\n","*"*30,"\n")
print(n_incorrects,n_hints,n_corrects)
print(n_corrects / (n_incorrects + n_hints + n_corrects))
```

```
num_total: 194947
num_students: 333
num_problems: 300
num_kcs: 98
num_null_condition: 0

 ******************************

74004 27911 187943
0.6483968011923079
```

1Analysis for Null and Unique value of column attributes

[6]:
```
def work_col_analysis(df_work):
    num_nonull_toal = df_work.notnull().sum()  # Not Null
    dict_col_1 = {'col_name':num_nonull_toal.index,'num_nonull':num_nonull_toal.values}
    df_work_col_1 = pd.DataFrame(dict_col_1)

    num_null_toal = df_work.isnull().sum()  # Null
    dict_col_2 = {'col_name':num_null_toal.index,'num_null':num_null_toal.values}
    df_work_col_2 = pd.DataFrame(dict_col_2)

    num_unique_toal = df_work.apply(lambda col: len(col.unique()))   # axis=0
    print(type(num_unique_toal))
    dict_col_3 = {'col_name':num_unique_toal.index,'num_unique':num_unique_toal.values}
    df_work_col_3 = pd.DataFrame(dict_col_3)

```

(continues on next page)

(continued from previous page)

```
    # df_work_col = pd.concat([df_work_col_1, df_work_col_2], axis=1)
    df_work_col = pd.merge(df_work_col_1, df_work_col_2, on=['col_name'])
    df_work_col = pd.merge(df_work_col, df_work_col_3, on=['col_name'])
    return df_work_col
print("-------------------num_unique_toal and num_nonull_toal----------------------")
df_result = work_col_analysis(df_step)
df_result
```

```
-------------------num_unique_toal and num_nonull_toal----------------------
<class 'pandas.core.series.Series'>
```

[6]:

	col_name	num_nonull	num_null	num_unique
0	Row	194947	0	194947
1	Sample	194947	0	1
2	Anon Student Id	194947	0	333
3	Problem Hierarchy	194947	0	27
4	Problem Name	194947	0	300
5	Problem View	194947	0	32
6	Step Name	194947	0	382
7	Step Start Time	194632	315	33098
8	First Transaction Time	194947	0	34578
9	Correct Transaction Time	182132	12815	33501
10	Step End Time	194947	0	34351
11	Step Duration (sec)	194947	0	2521
12	Correct Step Duration (sec)	194947	0	2187
13	Error Step Duration (sec)	194947	0	2105
14	First Attempt	194947	0	3
15	Incorrects	194947	0	32
16	Hints	194947	0	30
17	Corrects	194947	0	17
18	Condition	194947	0	1
19	KC (F2011)	113992	80955	98
20	Opportunity (F2011)	113992	80955	1206
21	Predicted Error Rate (F2011)	113992	80955	7623
22	KC (Single-KC)	194947	0	1
23	Opportunity (Single-KC)	194947	0	1410
24	Predicted Error Rate (Single-KC)	194947	0	317
25	KC (Unique-step)	193043	1904	1179
26	Opportunity (Unique-step)	193043	1904	25
27	Predicted Error Rate (Unique-step)	0	194947	1

3Data Cleaning

Data Cleaning Suggestions

- Redundant columns: Columns that are all NULL or Single value.
- rows that KC (F2011) == nullDo not know the knowledge source
- rows that Step Start Time == nullThis step is too short or more than 10mins, so the data is not reliable
- Others

[7]:
```
df_step_clear = df_step.copy(deep=True) # deep copy
```

[8]:
```
# ""
cols = list(df_step.columns.values)
drop_cols = []
for col in cols:
    if len(df_step_clear[col].unique().tolist()) == 1:
        df_step_clear.drop(col,axis =1,inplace=True)
        drop_cols.append(col)

print("the cols num before clear: ",len(df_step.columns.to_list()))
print("the cols num after clear:",len(df_step_clear.columns.to_list()))
for col in drop_cols:
    print("drop:---",col)
```

```
the cols num before clear:  28
the cols num after clear: 24
drop:--- Sample
drop:--- Condition
drop:--- KC (Single-KC)
drop:--- Predicted Error Rate (Unique-step)
```

[9]:
```
# Others'KC (F2011)','Step Start Time' with null value
df_step_clear.dropna(axis=0, how='any', subset=['KC (F2011)','Step Start Time'],inplace
↪= True)
```

[10]:
```
# the remaining columns
print("-------------------num_unique_toal and num_nonull_toal----------------------")
df_result = work_col_analysis(df_step_clear)
df_result
```

```
-------------------num_unique_toal and num_nonull_toal----------------------
<class 'pandas.core.series.Series'>
```

[10]:

	col_name	num_nonull	num_null	num_unique
0	Row	113817	0	113817
1	Anon Student Id	113817	0	331
2	Problem Hierarchy	113817	0	26
3	Problem Name	113817	0	154
4	Problem View	113817	0	32
5	Step Name	113817	0	240
6	Step Start Time	113817	0	18856
7	First Transaction Time	113817	0	19745
8	Correct Transaction Time	103454	10363	19146
9	Step End Time	113817	0	19623
10	Step Duration (sec)	113817	0	2382
11	Correct Step Duration (sec)	113817	0	2093
12	Error Step Duration (sec)	113817	0	1949
13	First Attempt	113817	0	3
14	Incorrects	113817	0	25
15	Hints	113817	0	25
16	Corrects	113817	0	15
17	KC (F2011)	113817	0	97

(continues on next page)

(continued from previous page)

```
18              Opportunity (F2011)      113817         0        1205
19      Predicted Error Rate (F2011)     113817         0        7622
20             Opportunity (Single-KC)   113817         0        1164
21  Predicted Error Rate (Single-KC)     113817         0         315
22                  KC (Unique-step)     112869       948         625
23         Opportunity (Unique-step)     112869       948          25
```

Outlier Analysis

- It is found that there is a non-numeric type in duration that is '.' , which should represent 0
- In addition, box diagrams can be used to analyze whether some outliers need to be removed

[11]:
```
print(df_step_clear.columns.tolist())
print("-"*100)
print(df_step_clear.describe().columns.tolist()) #object
print("-"*100)
print(df_step_clear.dtypes)
```

```
['Row', 'Anon Student Id', 'Problem Hierarchy', 'Problem Name', 'Problem View', 'Step␣
↪Name', 'Step Start Time', 'First Transaction Time', 'Correct Transaction Time', 'Step␣
↪End Time', 'Step Duration (sec)', 'Correct Step Duration (sec)', 'Error Step Duration␣
↪(sec)', 'First Attempt', 'Incorrects', 'Hints', 'Corrects', 'KC (F2011)', 'Opportunity␣
↪(F2011)', 'Predicted Error Rate (F2011)', 'Opportunity (Single-KC)', 'Predicted Error␣
↪Rate (Single-KC)', 'KC (Unique-step)', 'Opportunity (Unique-step)']
-----------------------------------------------------------------------------------------
↪-----------
['Row', 'Problem View', 'Incorrects', 'Hints', 'Corrects', 'Predicted Error Rate (F2011)
↪', 'Opportunity (Single-KC)', 'Predicted Error Rate (Single-KC)', 'Opportunity (Unique-
↪step)']
-----------------------------------------------------------------------------------------
↪-----------
Row                               int64
Anon Student Id                  object
Problem Hierarchy                object
Problem Name                     object
Problem View                      int64
Step Name                        object
Step Start Time                  object
First Transaction Time           object
Correct Transaction Time         object
Step End Time                    object
Step Duration (sec)              object
Correct Step Duration (sec)      object
Error Step Duration (sec)        object
First Attempt                    object
Incorrects                        int64
Hints                             int64
Corrects                          int64
KC (F2011)                       object
Opportunity (F2011)              object
Predicted Error Rate (F2011)    float64
```

(continues on next page)

(continued from previous page)

```
Opportunity (Single-KC)              int64
Predicted Error Rate (Single-KC)    float64
KC (Unique-step)                    object
Opportunity (Unique-step)          float64
dtype: object
```

[12]:
```
# Change . to 0 in "xxx-duration"
rectify_cols = ['Step Duration (sec)', 'Correct Step Duration (sec)', 'Error Step␣
↪Duration (sec)']
for col in rectify_cols:
    df_step_clear[col] = df_step_clear[col].apply(lambda x: 0 if x=='.' else x)
    df_step_clear[col] = df_step_clear[col].astype(float)
print(df_step_clear[rectify_cols].dtypes)
```

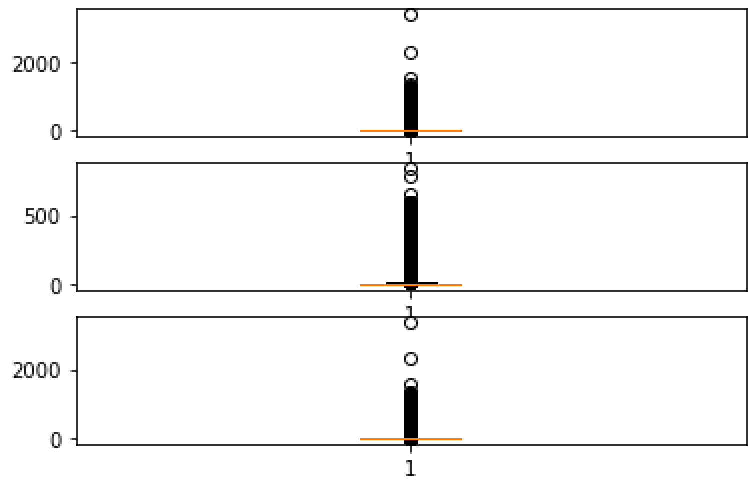
```
Step Duration (sec)            float64
Correct Step Duration (sec)    float64
Error Step Duration (sec)      float64
dtype: object
```

3. Data Visualization

[13]:
```
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objs as go
import matplotlib.pyplot as plt
%matplotlib inline
```

[14]:
```
# Outlier analysis for each column

fig=plt.figure()
box_cols = ['Step Duration (sec)', 'Correct Step Duration (sec)','Error Step Duration␣
↪(sec)']
for i, col in enumerate(box_cols):
    ax=fig.add_subplot(3, 1, i+1)
    ax.boxplot(df_step_clear[df_step_clear[col].notnull()][col].tolist())
fig.show("svg")
```

```
D:\MySoftwares\Anaconda\envs\data\lib\site-packages\ipykernel_launcher.py:8: UserWarning:

Matplotlib is currently using module://ipykernel.pylab.backend_inline, which is a non-
↪GUI backend, so cannot show the figure.
```



[15]:
```
# The distribution of continuous values
def show_value_counts_histogram(colname, sort = True):
    # create the bins
    start = int(df_step_clear[colname].min()/10)*10
    end = int(df_step_clear[colname].quantile(q=0.95)/10+1)*10
    step = int((end - start)/20)
    print(start, end, step)
    counts, bins = np.histogram(df_step_clear[colname],bins=range(start, end, step))
    bins = 0.5 * (bins[:-1] + bins[1:])

    fig = px.bar(x=bins, y=counts, labels={'x': colname, 'y':'count'})
    fig.show("svg")


# Box distribution of continuous values
def show_value_counts_box(colname, sort = True):
#     fig = px.box(df_step_clear, y=colname)
#     fig.show("svg")
    plt.figure(figsize=(10,5))
    plt.title('Box-plot for '+ colname,fontsize=20)#
    plt.boxplot([df_step_clear[colname].tolist()])
    plt.show("svg")


# Histogram of discrete values
def show_value_counts_bar(colname, sort = True):
    ds = df_step_clear[colname].value_counts().reset_index()
    ds.columns = [
        colname,
        'Count'
    ]
    if sort:
        ds = ds.sort_values(by='Count', ascending=False)
    # histogram
    fig = px.bar(
        ds,
```

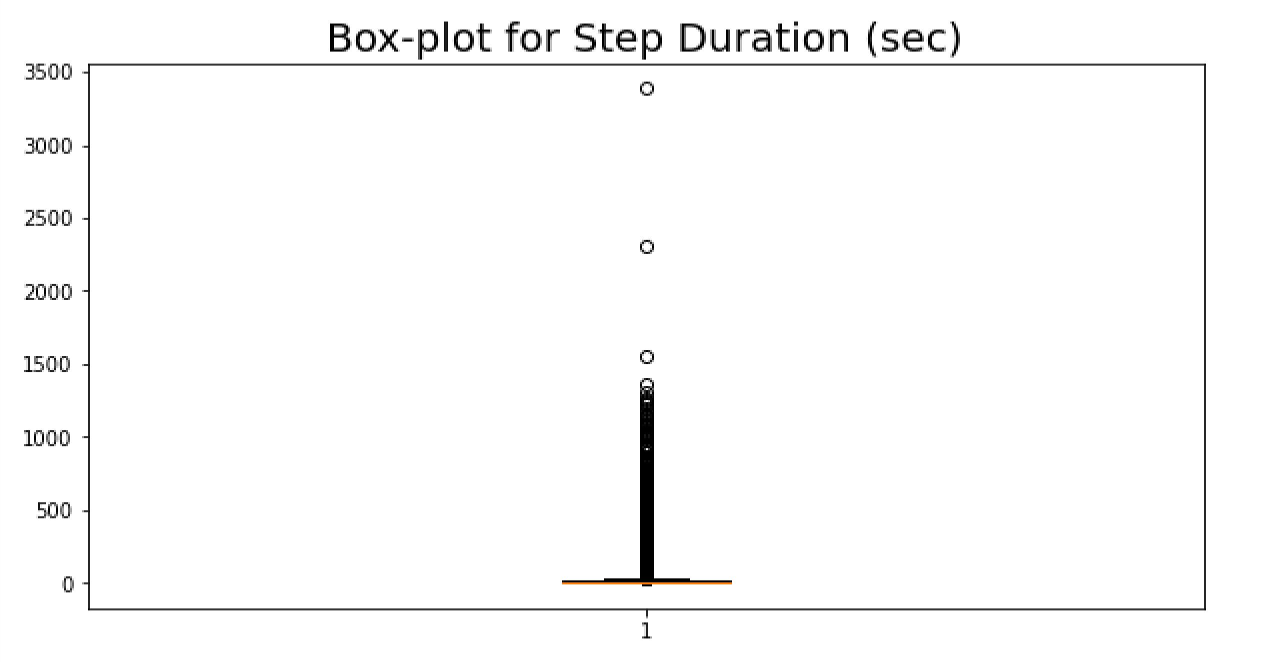
(continues on next page)

(continued from previous page)

```
        x = colname,
        y = 'Count',
        title = colname + ' distribution'
    )
    fig.show("svg")


# Pie of discrete values
def show_value_counts_pie(colname, sort = True):
    ds = df_step_clear[colname].value_counts().reset_index()
    ds.columns = [
        colname,
        'percent'
    ]
    ds['percent'] /= len(df_step_clear)
    if sort:
        ds = ds.sort_values(by='percent', ascending=False)
    fig = px.pie(
        ds,
        names = colname,
        values = 'percent',
        title = colname+ ' Percentage',
    )
    fig.update_traces(textposition='inside', textinfo='percent+label',showlegend=False)
    fig.show("svg")
```

[16]:
```
# Bar
show_value_counts_bar('First Attempt')
show_value_counts_histogram('Step Duration (sec)')
show_value_counts_box('Step Duration (sec)')
```

```
0 70 3
```



[17]:
```
# Pie
# show_value_counts_pie('KC (F2011)')
show_value_counts_pie('Problem Hierarchy')
show_value_counts_pie('Problem Name')
# show_value_counts_pie('Step Name')
```

[19]:
```
# four column labels are individually distributed as follows

topnum_max = 50 # show top 50 for each type
fig = make_subplots(rows=2, cols=2,   # 2*2
      start_cell="top-left",
      subplot_titles=('KC (F2011)','Problem Hierarchy','Problem Name','Step Name'),
      column_widths=[0.5, 0.5])
traces = [
    go.Bar(
        x = df_step[colname].value_counts().reset_index().index.tolist()[:topnum_max],
        y = df_step[colname].value_counts().reset_index()[colname].tolist()[:topnum_max],
        name = 'Type: ' + str(colname),
        text = df_step[colname].value_counts().reset_index()['index'].tolist()[:topnum_
↪max],
        textposition = 'auto',
    ) for colname in ['KC (F2011)','Problem Hierarchy','Problem Name','Step Name']
]
for i in range(len(traces)):
    fig.append_trace(
        traces[i],
        (i //2) + 1, # pos_row
        (i % 2) + 1  # pos_col
    )
```

(continues on next page)

(continued from previous page)

```
fig.update_layout(
    title_text = 'Bar of top 50 distributions for each type ',
)

fig.show("svg")
```

OLI data in fall, 2011transaction

[1]:
```
%matplotlib inline
import pandas as pd
import numpy as np
# global configuration: show every rows and cols
pd.set_option('display.max_rows', None)
pd.set_option('max_colwidth',None)
pd.set_option('display.max_columns', None)
```

1. Data Description

1.1 Column Description

[2]:
```
# help_table1: the description for data by transactions
df1 = pd.read_csv('OLI_data/help_table1.csv',sep=',',encoding="gbk")
df1 = df1.loc[:, ['Field', 'Annotation']]
df1
```

[2]:
```
                          Field  \
0                           Row
1                   Sample Name
2                Transaction Id
3               Anon Student Id
4                    Session Id
5                          Time
6                     Time Zone
7                Duration (sec)
8         Student Response Type
9      Student Response Subtype
10          Tutor Response Type
11       Tutor Response Subtype
12           Level (level_type)
13                 Problem Name
14                 Problem View
15           Problem Start Time
16                    Step Name
17              Attempt at Step
18                      Outcome
19                    Selection
20                       Action
21                        Input
```

(continues on next page)

(continued from previous page)

```
22              Feedback Text
23    Feedback Classification
24                 Help Level
25            Total Num Hints
26             Condition Name
27             Condition Type
28            KC (model_name)
29   KC Category (model_name)
30                     School
31                      Class
32     CF (custom_field_name)
33                 Event Type

                                                                                   ␣
↪                                                                                  ␣
↪                                                                                  ␣
↪                                                                                  ␣
↪                                                                                  ␣
↪                                                                                  ␣
↪                                                                                  ␣
↪                                                                                  ␣
↪                        Annotation
0                                                                                  ␣
↪                                                                                  ␣
↪                                                                                  ␣
↪                                                                                  ␣
↪                                                                                  ␣
↪                                                                                  ␣
↪                                                                                  ␣
↪                                                                                  ␣
↪                     A row counter
1                                                                                  ␣
↪                                                                                  ␣
↪                                                                                  ␣
↪                                                                                  ␣
↪                                                                                  ␣
↪                                                                                  ␣
↪                                  The sample that contains the transaction. If a␣
↪transaction appears in multiple samples,                  the transaction will be␣
↪repeated, but with a different sample name.
2                                                                                  ␣
↪                                                                                  ␣
↪                                                                                  ␣
↪                                                                                  ␣
↪                                                                                  ␣
↪                                                                                  ␣
↪                                              A unique ID that identifies the␣
↪transaction. Currently used for annotating transactions with                    custom␣
↪fields via                    web services.
3                                                                                  ␣
↪                                                                                  ␣
↪                                                                                  ␣
↪                                                                                  ␣
↪                                                                                  ␣
↪                                                                                  ␣
↪              DataShop-generated anonymous student ID. To obtain original student␣
↪identifiers or to                     learn more about data anonymization, see About␣
↪data anonymization                     below.
```

(continues on next page)

(continued from previous page)

```
4                                       A dataset-unique string that identifies the user's session with the tutor.
5                                       Time the transaction occurred. For instance, if a student types "25" and presses return, the                     transaction time is at the point in which they press return.
6                                       The local time zone (e.g., EST, PST, US/Eastern).
7   Duration of the transaction in seconds. This is the time of the current transaction minus                     that of the preceding transaction or problem start event-whichever is closer in time to the                     current transaction. If this difference is greater than 10 minutes, or if the prior transaction occurred                     during a different user session, DataShop reports the duration as null                     (a dot). If the current transaction is preceded by neither another transaction or a problem start event,                     duration is shown as null. The duration is formatted without decimal places if the two times used                     in the calculation were without millisecond precision.
8                                       The type of attempt made by the student (e.g., "ATTEMPT" or "HINT_REQUEST").                     This is logged in the                     semantic_event element.
9                                       A more detailed classification of the student attempt. For example,                     the CTAT software describes actions taken by the tutor on behalf of the student                     as having subtype "tutor-performed".
```

(continues on next page)

(continued from previous page)

```
                                                                                          The type of response made by the tutor (e.g., "RESULT" or "HINT_MSG").
                                                                                          A more detailed classification of the tutor response.
                                                                                          The problem hierarchy name (e.g., "Understanding Fractions") of the type                    specified in the column header (e.g., "Unit"). There may be multiple "Level" columns if                    the problem hierarchy is more than one level deep. Level is logged in                    the                     level element.
                                                                                          The name of the problem. Two problems with the same "Problem Name" are                    considered different "problems" by DataShop if the following logged values are not                    identical: problem name, context, tutor_flag (whether or not the problem                    or activity is tutored) and "other" field. These fields are logged in the                    problem element.
        The number of times the student encountered the problem so far. This counter increases                    with each instance of the same problem. See "Problem View" in the "By Student-Step" table below.
                                                    If the problem start time is not given in the original log data, then it is set to the time                    of the last transaction of the prior problem. If there is no prior problem for the session,                    the time of the earliest transaction is used. Earliest transaction time is equivalent                    to the minimum transaction time for the earliest step of the problem. For more detail                    on how problem start time is determined, see Determining                    Problem Start Time.
```

(continues on next page)

(continued from previous page)

```
                     Formed by concatenating the "selection" and "action". Also see the
                    glossary entry for "step".
                                                                              As of this transaction, the current number of
attempts toward the identified step.
                                                  The tutor's evaluation of the student's attempt. For example,
"CORRECT", "INCORRECT", or "HINT".                                        This is logged in the
                    action_evaluation element.
                                                                                                  A description of the
interface element(s) that the student selected or                   interacted with
(for example, "LowestCommonDenominatorCell"). This is logged in the
event_descriptor                                    element.
                                                                                                          A description of the
manipulation applied to the selection.
                                        The input the student submitted (e.g., the text entered, the text
of a menu item or a combobox entry).
```

(continues on next page)

(continued from previous page)

```
The body of a hint, success, or incorrect action message shown to the student.                    It is generally a text value, logged in the tutor_advice element.
The type of error (e.g., "sign error") or type of hint.
In the case of hierarchical hints, this is the depth of the hint. "1", for example, is an initial hint, while "3" is the third hint.
The total number of hints available. This is logged in the                    action_evaluation                    element.
The name of the condition (e.g., "Unworked").
A condition classification (e.g., "Experimental", "Control"); optional at the time of logging.
```

(continues on next page)

(continued from previous page)

```
The knowledge component for this transaction. It is a member of the knowledge component model named in the column header. One "KC (model_name)" column should appear in the export for each KC model in the dataset.
The knowledge component "category" logged by some tutors. It is a member of the knowledge component model named in the column header. One "KC Category (model_name)" column should appear in the export for each KC model in the dataset.
The name of the school where the student used the tutor to create this transaction.
The name of the class the student was in when he or she used the tutor to create this transaction.
The value of a custom field. This is usually information that did not fit into any of the other logging fields (i.e., any of the other columns), and so was logged in this special container.
Allowed values are "assess", "instruct" and "assess_instruct". Blank is also allowed. Only "instruct" and "assess_instruct" values are treated as learning opportunities.
```

(continues on next page)

(continued from previous page)

1.2 Summarization of Data

This table organizes the data as student-problem-step-transaction

[3]:
```
df_transaction =  pd.read_csv('OLI_data/AllData_transaction_2011F.csv',low_memory=False)
↪# sep="\t"
df_transaction.head(5)
```

[3]:
```
   Row Sample Name                    Transaction Id  \
0    1    All Data  2adbe4abefd649d48862d3f62b1abf5e
1    2    All Data  4393251e32a6f00502f3f1ef894af8fe
2    3    All Data  e2fb2cb788d10ebaa6f288e0757d1b09
3    4    All Data  e7e150d423862e346dc7e36a95e394e4
4    5    All Data  684b1f770a225f21745c6c4c977ddc32

                      Anon Student Id                        Session Id  \
0  Stu_00b2b35fd027e7891e8a1a527125dd65  8dd109e680020ca6016f8e64290b5610
1  Stu_00b2b35fd027e7891e8a1a527125dd65  8dd109e680020ca6016f8e64290b5610
2  Stu_00b2b35fd027e7891e8a1a527125dd65  8dd109e680020ca6016f8e64290b5610
3  Stu_00b2b35fd027e7891e8a1a527125dd65  8dd109e680020ca6016f8e64290b5610
4  Stu_00b2b35fd027e7891e8a1a527125dd65  8dd109e680020ca6016f8e64290b5610

                  Time   Time Zone Duration (sec) Student Response Type  \
0  2011-09-21 17:26:36  US/Eastern              1             VIEW_PAGE
1  2011-09-21 17:35:28  US/Eastern          23.13               ATTEMPT
2  2011-09-21 17:35:28  US/Eastern          23.13               ATTEMPT
3  2011-09-21 17:35:28  US/Eastern          23.13               ATTEMPT
4  2011-09-21 17:35:28  US/Eastern          23.13               ATTEMPT

  Student Response Subtype Tutor Response Type  Tutor Response Subtype  \
0                 UI Event                 NaN                     NaN
1                      NaN              RESULT                     NaN
2                      NaN              RESULT                     NaN
3                      NaN              RESULT                     NaN
4                      NaN              RESULT                     NaN

  Level (Sequence)                           Level (Unit)  \
0          Statics  Concentrated Forces and Their Effects
1          Statics  Concentrated Forces and Their Effects
2          Statics  Concentrated Forces and Their Effects
3          Statics  Concentrated Forces and Their Effects
4          Statics  Concentrated Forces and Their Effects

                       Level (Module) Level (Section1) Problem Name  \
0  Introduction to Free Body Diagrams              NaN   _m2_assess
1  Introduction to Free Body Diagrams              NaN   _m2_assess
2  Introduction to Free Body Diagrams              NaN   _m2_assess
3  Introduction to Free Body Diagrams              NaN   _m2_assess
4  Introduction to Free Body Diagrams              NaN   _m2_assess
```

(continues on next page)

(continued from previous page)

```
   Problem View   Problem Start Time                   Step Name  \
0             1  2011-09-21 17:26:35                         NaN
1             1  2011-09-21 17:26:35  q1_point1i1 UpdateComboBox
2             1  2011-09-21 17:26:35  q1_point3i3 UpdateComboBox
3             1  2011-09-21 17:26:35  q1_point6i2 UpdateComboBox
4             1  2011-09-21 17:26:35  q1_point1i2 UpdateComboBox

   Attempt At Step  Is Last Attempt    Outcome    Selection           Action  \
0              NaN              NaN        NaN   Navigation  SelectPageNumber
1              1.0              1.0    CORRECT  q1_point1i1    UpdateComboBox
2              1.0              1.0    CORRECT  q1_point3i3    UpdateComboBox
3              1.0              1.0  INCORRECT  q1_point6i2    UpdateComboBox
4              1.0              1.0    CORRECT  q1_point1i2    UpdateComboBox

                              Input Input.1 Feedback Text  \
0                                 1     NaN           NaN
1         <material>cord c</material>     NaN           NaN
2         <material>120 lb</material>     NaN           NaN
3  <material>no interaction</material>     NaN           NaN
4             <material>up</material>     NaN           NaN

   Feedback Classification  Help Level  Total Num Hints KC (Single-KC)  \
0                      NaN         NaN             NaN            NaN
1                      NaN         NaN             NaN      Single-KC
2                      NaN         NaN             NaN      Single-KC
3                      NaN         NaN             NaN      Single-KC
4                      NaN         NaN             NaN      Single-KC

   KC Category (Single-KC) KC (Unique-step)  KC Category (Unique-step)  \
0                      NaN              NaN                        NaN
1                      NaN              NaN                        NaN
2                      NaN              NaN                        NaN
3                      NaN              NaN                        NaN
4                      NaN              NaN                        NaN

                  KC (F2011)  KC Category (F2011) KC (F2011).1  \
0                        NaN                  NaN          NaN
1         identify_interaction                  NaN          NaN
2         gravitational_forces                  NaN          NaN
3  represent_interaction_spring                  NaN          NaN
4    represent_interaction_cord                  NaN          NaN

   KC Category (F2011).1 KC (F2011).2  KC Category (F2011).2  \
0                    NaN          NaN                    NaN
1                    NaN          NaN                    NaN
2                    NaN          NaN                    NaN
3                    NaN          NaN                    NaN
4                    NaN          NaN                    NaN

                  School        Class CF (oli:activityGuid)  \
0  Marion Technical College  MET2010B-01                  NaN
1  Marion Technical College  MET2010B-01                  NaN
```

(continues on next page)

(continued from previous page)

```
2  Marion Technical College  MET2010B-01                   NaN
3  Marion Technical College  MET2010B-01                   NaN
4  Marion Technical College  MET2010B-01                   NaN

  CF (oli:highStakes) CF (oli:purpose) CF (oli:resourceType)
0                 NaN              NaN                   NaN
1                 NaN              NaN                   NaN
2                 NaN              NaN                   NaN
3                 NaN              NaN                   NaN
4                 NaN              NaN                   NaN
```

2. Data Analysis

[4]:
```
df_transaction.describe()
```

[4]:
```
                 Row  Tutor Response Subtype   Problem View  Attempt At Step  \
count  361092.000000                     0.0  361092.000000    289858.000000
mean   180546.500000                     NaN       1.180192         2.382867
std    104238.426039                     NaN       0.907172         9.948941
min         1.000000                     NaN       1.000000         1.000000
25%     90273.750000                     NaN       1.000000         1.000000
50%    180546.500000                     NaN       1.000000         1.000000
75%    270819.250000                     NaN       1.000000         2.000000
max    361092.000000                     NaN      32.000000       427.000000

       Is Last Attempt  Feedback Classification  Help Level  Total Num Hints  \
count    289858.000000                      0.0         0.0              0.0
mean          0.658678                      NaN         NaN              NaN
std           0.474154                      NaN         NaN              NaN
min           0.000000                      NaN         NaN              NaN
25%           0.000000                      NaN         NaN              NaN
50%           1.000000                      NaN         NaN              NaN
75%           1.000000                      NaN         NaN              NaN
max           1.000000                      NaN         NaN              NaN

       KC Category (Single-KC)  KC Category (Unique-step)  \
count                      0.0                        0.0
mean                       NaN                        NaN
std                        NaN                        NaN
min                        NaN                        NaN
25%                        NaN                        NaN
50%                        NaN                        NaN
75%                        NaN                        NaN
max                        NaN                        NaN

       KC Category (F2011)  KC Category (F2011).1  KC Category (F2011).2
count                  0.0                    0.0                    0.0
mean                   NaN                    NaN                    NaN
std                    NaN                    NaN                    NaN
min                    NaN                    NaN                    NaN
25%                    NaN                    NaN                    NaN
```

(continues on next page)

(continued from previous page)

```
50%                 NaN                 NaN                 NaN
75%                 NaN                 NaN                 NaN
max                 NaN                 NaN                 NaN
```

1 Analysis for Null and Unique value of column attributes

[5]:
```python
def work_col_analysis(df_work):
    num_nonull_toal = df_work.notnull().sum()  # Not Null
    dict_col_1 = {'col_name':num_nonull_toal.index,'num_nonull':num_nonull_toal.values}
    df_work_col_1 = pd.DataFrame(dict_col_1)

    num_null_toal = df_work.isnull().sum()  # Null
    dict_col_2 = {'col_name':num_null_toal.index,'num_null':num_null_toal.values}
    df_work_col_2 = pd.DataFrame(dict_col_2)

    num_unique_toal = df_work.apply(lambda col: len(col.unique()))   # axis=0
    print(type(num_unique_toal))
    dict_col_3 = {'col_name':num_unique_toal.index,'num_unique':num_unique_toal.values}
    df_work_col_3 = pd.DataFrame(dict_col_3)

    df_work_col = pd.merge(df_work_col_1, df_work_col_2, on=['col_name'])
    df_work_col = pd.merge(df_work_col, df_work_col_3, on=['col_name'])
    return df_work_col
print("-------------------num_unique_toal and num_nonull_toal----------------------")
df_result = work_col_analysis(df_transaction)
df_result
```

```
-------------------num_unique_toal and num_nonull_toal----------------------
<class 'pandas.core.series.Series'>
```

[5]:

	col_name	num_nonull	num_null	num_unique
0	Row	361092	0	361092
1	Sample Name	361092	0	1
2	Transaction Id	361092	0	361092
3	Anon Student Id	361092	0	335
4	Session Id	361092	0	6656
5	Time	361092	0	263172
6	Time Zone	361092	0	1
7	Duration (sec)	361092	0	2565
8	Student Response Type	361092	0	5
9	Student Response Subtype	71234	289858	2
10	Tutor Response Type	289858	71234	3
11	Tutor Response Subtype	0	361092	1
12	Level (Sequence)	361092	0	1
13	Level (Unit)	361092	0	7
14	Level (Module)	361092	0	19
15	Level (Section1)	59480	301612	10
16	Problem Name	361092	0	300
17	Problem View	361092	0	32
18	Problem Start Time	361092	0	46473
19	Step Name	289858	71234	383

(continues on next page)

(continued from previous page)

```
20             Attempt At Step      289858     71234         428
21             Is Last Attempt      289858     71234           3
22                     Outcome      289858     71234           4
23                   Selection      361082        10         287
24                      Action      361082        10          10
25                       Input      302086     59006        6827
26                     Input.1           1    361091           2
27               Feedback Text      231063    130029        1579
28     Feedback Classification           0    361092           1
29                  Help Level           0    361092           1
30              Total Num Hints           0    361092           1
31              KC (Single-KC)      289858     71234           2
32     KC Category (Single-KC)           0    361092           1
33             KC (Unique-step)     283336     77756        1179
34   KC Category (Unique-step)           0    361092           1
35                  KC (F2011)      152592    208500          81
36         KC Category (F2011)           0    361092           1
37                KC (F2011).1       16904    344188          19
38       KC Category (F2011).1           0    361092           1
39                KC (F2011).2        6690    354402           9
40       KC Category (F2011).2           0    361092           1
41                      School      361092         0           7
42                       Class      361092         0           9
43       CF (oli:activityGuid)       45002    316090        1244
44         CF (oli:highStakes)       45002    316090           3
45            CF (oli:purpose)       44516    316576           4
46       CF (oli:resourceType)       45002    316090           3
```

2Analysis for Discrete value of column attributes

Columns with a small number of discrete values may represent very informative, so identify these columns first and analyze them one by one

[6]:
```
discrete_cols = []
series = []
cols = list(df_transaction.columns.values)

for col in cols:
    if len(df_transaction[col].unique().tolist()) <= 20 and len(df_transaction[col].
↪unique().tolist()) >= 2:
        discrete_cols.append(col)
        series.append(df_transaction[col].unique().tolist())

for a,b in zip(discrete_cols,series):
    print(a," : ",b)
    print("-"*80)
```

```
Student Response Type  :  ['VIEW_PAGE', 'ATTEMPT', 'SAVE_ATTEMPT', 'SUBMIT_ATTEMPT',
↪'HINT_REQUEST']
--------------------------------------------------------------------------------
Student Response Subtype  :  ['UI Event', nan]
```

(continues on next page)

(continued from previous page)

```
--------------------------------------------------------------------------------
Tutor Response Type  :  [nan, 'RESULT', 'HINT_MSG']
--------------------------------------------------------------------------------
Level (Unit)  :  ['Concentrated Forces and Their Effects', 'Engineering Systems - Single␣
↪Body Equilibrium', 'Complex Interactions Between Bodies', 'Multiple Body Equilibrium -␣
↪Frames', 'Multiple Body Equilibrium - Trusses', 'Friction', 'Moments of Inertia']
--------------------------------------------------------------------------------
Level (Module)  :  ['Introduction to Free Body Diagrams', 'Effects of Force',
↪'Representing Interactions Between Bodies', 'Effects of Multiple Forces', 'Equilibrium␣
↪Under 2D Concentrated Forces', 'Equilibrium of a Single Subsystem', 'Couples',
↪'Statically Equivalent Loads', 'Applications of Static Equivalency to Distributed␣
↪Forces', 'Representing Engineering Connections', 'Drawing FBDs of a Single Subsystem',
↪'Choosing a Solvable Subsystem', 'Drawing FBDs of Multiple Subsystems', 'Solving␣
↪Multiple Subsystems', 'Method of Joints', 'Method of Sections', 'Friction', 'Second␣
↪Moment of Area', 'Mass Moment of Inertia']
--------------------------------------------------------------------------------
Level (Section1)  :  [nan, 'Combining Concurrent Forces', 'Combining Moments', 'Applying␣
↪Force Equilibrium', 'Applying Force and Moment Equilibrium', 'Simplifying 3D loadings␣
↪to 2D or 1D loading', 'Fixed Connections', 'Pin Connections', 'Other Connections',
↪'Center of Gravity and Centroid']
--------------------------------------------------------------------------------
Is Last Attempt  :  [nan, 1.0, 0.0]
--------------------------------------------------------------------------------
Outcome  :  [nan, 'CORRECT', 'INCORRECT', 'HINT']
--------------------------------------------------------------------------------
Action  :  ['SelectPageNumber', 'UpdateComboBox', 'Click', 'UpdateRadioButton',
↪'UpdateCheckbox', 'UpdateNumberField', 'UpdateShortAnswer', 'UpdateHotspotSingle',
↪'UpdateHotspotMultiple', nan]
--------------------------------------------------------------------------------
Input.1  :  [nan, 'No, the forces of B on A and A on B shown on the diagram on the right␣
↪are not correct because body B and body A are interacting on one another when ???F???␣
↪is applied to the body ???B??? but A opposite senses on each other. In this case B␣
↪will push A in a']
--------------------------------------------------------------------------------
KC (Single-KC)  :  [nan, 'Single-KC']
--------------------------------------------------------------------------------
KC (F2011).1  :  [nan, 'rotation_sense_of_force', 'identify_interaction', 'motion_
↪dependence_on_force', 'couple_represents_net_zero_force', 'recognize_equivalence_from_
↪motion', 'relate_direction_normal_force_and_contact', 'moment_sign_sense_relation',
↪'possible_interaction_for_nonuniform_contact', 'represent_interaction_contacting_body',
↪ 'represent_forces_two-force_member', 'represent_interaction_cord', 'identify_enabling_
↪unknown', 'identify_equation_isolates_specific_unknown', 'sense_if_assuming_tension',
↪'determine_joint_is_solvable', 'judge_force_sense_based_on_sign', 'identify_internal_
↪load_points_on_section', 'identify_external_load_points_on_section']
--------------------------------------------------------------------------------
KC (F2011).2  :  [nan, 'rotation_sense_of_force', 'statics_problem_force_and_moment',
↪'represent_interaction_cord', 'represent_interaction_pin_connection', 'recognize_
↪variable_solvable_from_subsystem', 'tension_vs_compression_given_force_senses', 'sense_
↪if_assuming_tension', 'identify_internal_load_points_on_section']
--------------------------------------------------------------------------------
School  :  ['Marion Technical College', 'Sinclair Community College', 'Carnegie Mellon␣
↪University', 'Kettering University', 'Miami University', 'University of Maryland␣
↪Eastern Shore', 'University of Mississippi']
```

(continues on next page)

(continued from previous page)

```
--------------------------------------------------------------------------------
Class  :  ['MET2010B-01', 'F11-E213-01', '24-261Fall11', 'FEA-Fall11', 'DesignFall11',
↪'F11-E213-50', 'F11-MME-211', 'ENGE260-F11', 'ENGR309H-F11']
--------------------------------------------------------------------------------
CF (oli:highStakes)  :  [nan, False, True]
--------------------------------------------------------------------------------
CF (oli:purpose)  :  [nan, 'quiz', 'didigetthis', 'learnbydoing']
--------------------------------------------------------------------------------
CF (oli:resourceType)  :  [nan, 'x-oli-assessment2', 'x-oli-inline-assessment']
--------------------------------------------------------------------------------
```

3Data Cleaning

Data Cleaning Suggestions - Redundant columns: Columns that are all NULL or Single value. - Others

[7]:
```
df_transaction_clear = df_transaction.copy(deep=True) # deep copy
```

[8]:
```
# ""
cols = list(df_transaction.columns.values)
drop_cols = []
for col in cols:
    if len(df_transaction_clear[col].unique().tolist()) == 1:
        df_transaction_clear.drop(col,axis =1,inplace=True)
        drop_cols.append(col)

print("the cols num before clear: ",len(df_transaction.columns.to_list()))
print("the cols num after clear:",len(df_transaction_clear.columns.to_list()))
for col in drop_cols:
    print("drop:---",col)
```

```
the cols num before clear:  47
the cols num after clear: 35
drop:--- Sample Name
drop:--- Time Zone
drop:--- Tutor Response Subtype
drop:--- Level (Sequence)
drop:--- Feedback Classification
drop:--- Help Level
drop:--- Total Num Hints
drop:--- KC Category (Single-KC)
drop:--- KC Category (Unique-step)
drop:--- KC Category (F2011)
drop:--- KC Category (F2011).1
drop:--- KC Category (F2011).2
```

[9]:
```
df_transaction_clear.head()
```

[9]:
```
   Row                    Transaction Id  \
0    1  2adbe4abefd649d48862d3f62b1abf5e
1    2  4393251e32a6f00502f3f1ef894af8fe
2    3  e2fb2cb788d10ebaa6f288e0757d1b09
```

(continues on next page)

(continued from previous page)

```
3    4  e7e150d423862e346dc7e36a95e394e4
4    5  684b1f770a225f21745c6c4c977ddc32

                       Anon Student Id                        Session Id  \
0  Stu_00b2b35fd027e7891e8a1a527125dd65  8dd109e680020ca6016f8e64290b5610
1  Stu_00b2b35fd027e7891e8a1a527125dd65  8dd109e680020ca6016f8e64290b5610
2  Stu_00b2b35fd027e7891e8a1a527125dd65  8dd109e680020ca6016f8e64290b5610
3  Stu_00b2b35fd027e7891e8a1a527125dd65  8dd109e680020ca6016f8e64290b5610
4  Stu_00b2b35fd027e7891e8a1a527125dd65  8dd109e680020ca6016f8e64290b5610

                 Time Duration (sec) Student Response Type  \
0  2011-09-21 17:26:36            1            VIEW_PAGE
1  2011-09-21 17:35:28        23.13              ATTEMPT
2  2011-09-21 17:35:28        23.13              ATTEMPT
3  2011-09-21 17:35:28        23.13              ATTEMPT
4  2011-09-21 17:35:28        23.13              ATTEMPT

  Student Response Subtype Tutor Response Type  \
0                 UI Event                 NaN
1                      NaN              RESULT
2                      NaN              RESULT
3                      NaN              RESULT
4                      NaN              RESULT

                         Level (Unit)                  Level (Module)  \
0  Concentrated Forces and Their Effects  Introduction to Free Body Diagrams
1  Concentrated Forces and Their Effects  Introduction to Free Body Diagrams
2  Concentrated Forces and Their Effects  Introduction to Free Body Diagrams
3  Concentrated Forces and Their Effects  Introduction to Free Body Diagrams
4  Concentrated Forces and Their Effects  Introduction to Free Body Diagrams

  Level (Section1) Problem Name  Problem View   Problem Start Time  \
0              NaN   _m2_assess            1  2011-09-21 17:26:35
1              NaN   _m2_assess            1  2011-09-21 17:26:35
2              NaN   _m2_assess            1  2011-09-21 17:26:35
3              NaN   _m2_assess            1  2011-09-21 17:26:35
4              NaN   _m2_assess            1  2011-09-21 17:26:35

                     Step Name  Attempt At Step  Is Last Attempt    Outcome  \
0                          NaN              NaN              NaN        NaN
1  q1_point1i1 UpdateComboBox              1.0              1.0    CORRECT
2  q1_point3i3 UpdateComboBox              1.0              1.0    CORRECT
3  q1_point6i2 UpdateComboBox              1.0              1.0  INCORRECT
4  q1_point1i2 UpdateComboBox              1.0              1.0    CORRECT

     Selection           Action                               Input Input.1  \
0   Navigation  SelectPageNumber                                  1     NaN
1  q1_point1i1    UpdateComboBox          <material>cord c</material>     NaN
2  q1_point3i3    UpdateComboBox          <material>120 lb</material>     NaN
3  q1_point6i2    UpdateComboBox  <material>no interaction</material>     NaN
4  q1_point1i2    UpdateComboBox              <material>up</material>     NaN
```

(continues on next page)

(continued from previous page)

```
  Feedback Text KC (Single-KC) KC (Unique-step)                     KC (F2011)  \
0           NaN             NaN              NaN                            NaN
1           NaN       Single-KC              NaN           identify_interaction
2           NaN       Single-KC              NaN            gravitational_forces
3           NaN       Single-KC              NaN  represent_interaction_spring
4           NaN       Single-KC              NaN    represent_interaction_cord

  KC (F2011).1 KC (F2011).2                    School        Class  \
0          NaN          NaN  Marion Technical College  MET2010B-01
1          NaN          NaN  Marion Technical College  MET2010B-01
2          NaN          NaN  Marion Technical College  MET2010B-01
3          NaN          NaN  Marion Technical College  MET2010B-01
4          NaN          NaN  Marion Technical College  MET2010B-01

  CF (oli:activityGuid) CF (oli:highStakes) CF (oli:purpose)  \
0                   NaN                 NaN              NaN
1                   NaN                 NaN              NaN
2                   NaN                 NaN              NaN
3                   NaN                 NaN              NaN
4                   NaN                 NaN              NaN

  CF (oli:resourceType)
0                   NaN
1                   NaN
2                   NaN
3                   NaN
4                   NaN
```

[10]:

```
# the remaining columns
print("------------------num_unique_toal and num_nonull_toal---------------------")
df_result = work_col_analysis(df_transaction_clear)
df_result
```

```
------------------num_unique_toal and num_nonull_toal----------------------
<class 'pandas.core.series.Series'>
```

[10]:

	col_name	num_nonull	num_null	num_unique
0	Row	361092	0	361092
1	Transaction Id	361092	0	361092
2	Anon Student Id	361092	0	335
3	Session Id	361092	0	6656
4	Time	361092	0	263172
5	Duration (sec)	361092	0	2565
6	Student Response Type	361092	0	5
7	Student Response Subtype	71234	289858	2
8	Tutor Response Type	289858	71234	3
9	Level (Unit)	361092	0	7
10	Level (Module)	361092	0	19
11	Level (Section1)	59480	301612	10
12	Problem Name	361092	0	300
13	Problem View	361092	0	32
14	Problem Start Time	361092	0	46473
15	Step Name	289858	71234	383

(continues on next page)

(continued from previous page)

```
16            Attempt At Step      289858     71234          428
17            Is Last Attempt      289858     71234            3
18                    Outcome      289858     71234            4
19                  Selection      361082        10          287
20                     Action      361082        10           10
21                      Input      302086     59006         6827
22                    Input.1           1    361091            2
23              Feedback Text      231063    130029         1579
24             KC (Single-KC)      289858     71234            2
25            KC (Unique-step)     283336     77756         1179
26                  KC (F2011)     152592    208500           81
27                KC (F2011).1      16904    344188           19
28                KC (F2011).2       6690    354402            9
29                      School     361092         0            7
30                       Class     361092         0            9
31     CF (oli:activityGuid)        45002    316090         1244
32       CF (oli:highStakes)        45002    316090            3
33          CF (oli:purpose)        44516    316576            4
34     CF (oli:resourceType)        45002    316090            3
```

Outlier Analysis

- It is found that there is a non-numeric type in duration that is '.' , which should represent 0

[11]:

```
# Change . to 0 in "duration"
rectify_cols = ['Duration (sec)']
for col in rectify_cols:
    df_transaction_clear[col] = df_transaction_clear[col].apply(lambda x: 0 if x=='.'␣
↪else x)
    df_transaction_clear[col] = df_transaction_clear[col].astype(float)
print(df_transaction_clear[rectify_cols].dtypes)
```

```
Duration (sec)    float64
dtype: object
```

3. Data Visualization

[12]:

```
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objs as go
```

[13]:

```
# Histogram of discrete values
def show_value_counts_bar(colname, sort = True):
    ds = df_transaction[colname].value_counts().reset_index()
    ds.columns = [
        colname,
        'Count'
    ]
    if sort:
```

(continues on next page)

(continued from previous page)

```
        ds = ds.sort_values(by='Count', ascending=False)
    # histogram
    fig = px.bar(
        ds,
        x = colname,
        y = 'Count',
        title = colname + ' distribution'
    )
    fig.show("svg")


# Pie of discrete values
def show_value_counts_pie(colname, sort = True):
    ds = df_transaction[colname].value_counts().reset_index()
    ds.columns = [
        colname,
        'percent'
    ]
    ds['percent'] /= len(df_transaction)
    if sort:
        ds = ds.sort_values(by='percent', ascending=False)
    fig = px.pie(
        ds,
        names = colname,
        values = 'percent',
        title = colname+ ' Percentage',
    )
    fig.show("svg")
```

[14]:
```
col_pies = ['Student Response Type','Tutor Response Type','Outcome']
for col in col_pies:
    show_value_counts_pie(col)
```

Analysis by label description: > - If the Student Response Type == ATTEMPT, then the Tutor Response Type == Result, then the Student Response Type => correct or incorrect

- If the Student Response Type == HINT_REQUEST, then the Tutor Response Type == HINT_MSG, then the outCome => hint
- If Student Response Type == other,then the Tutor Response Type == NaN, then the outCome => NaN

[15]:
```
%matplotlib inline
def show_value_counts_pie2(col1,type1,col2, sort = True):
    df_tmp = df_transaction[df_transaction[col1] == type1]
    ds = df_tmp[col2].value_counts().reset_index()
    ds.columns = [
        col2,
        'percent'
    ]
    ds['percent'] /= len(df_tmp)
    if sort:
```

(continues on next page)

(continued from previous page)

```
        ds = ds.sort_values(by='percent', ascending=False)
    fig = px.pie(
        ds,
        names = col2,
        values = 'percent',
        title = col2+ ' Percentage when ' + col1 + ' =='+ type1,
    )
    fig.show("svg")

# Take Student Response Type as an example
col1 = 'Student Response Type'
col2 = 'Outcome'
# col1 = 'Tutor Response Type'
# col2 = 'Outcome'

show_value_counts_pie2(col1,"ATTEMPT",col2)
show_value_counts_pie2(col1,"HINT_REQUEST",col2)
```

[16]:
```
col_bars = ['Level (Unit)','Level (Module)','Level (Section1)','KC (F2011)']

fig = make_subplots(rows=3, cols=2,   # 2*2
      start_cell="top-left",
      subplot_titles=col_bars,
      column_widths=[0.5, 0.5])
traces = [
    go.Bar(
        x = df_transaction[colname].value_counts().reset_index().index.tolist(),
        y = df_transaction[colname].value_counts().reset_index()[colname].tolist(),
        name = 'Type: ' + str(colname),
        text = df_transaction[colname].value_counts().reset_index()['index'].tolist(),
        textposition = 'auto',
    ) for colname in col_bars
]
for i in range(len(traces)):
    fig.append_trace(
        traces[i],
        (i //2) + 1, # pos_row
        (i % 2) + 1  # pos_col
    )

fig.update_layout(
    title_text = 'Bar of distributions for every type',
)

fig.show("svg")
```

According to the chart below, there are 3 schools with a smaller sample of students.

[17]:
```
#
```

(continues on next page)

(continued from previous page)

```
schools = [item for item in df_transaction_clear['School'].unique().tolist()]
students = [len(df_transaction_clear[df_transaction_clear['School'] == sch]['Anon␣
↪Student Id'].unique()) for sch in schools]
fig = go.Figure(data=[go.Bar(
        x = schools,
        y = students,
        name = 'The number of students is counted by school',
        text = schools,
        textposition = 'auto',
    )])
fig.show("svg")
fig = go.Figure(data=[go.Pie(
        labels = schools,
        values = students,
        name = 'The number of students is counted by school',
        text = schools,
        textposition = 'auto',
    )])
fig.show("svg")
```

6.7.6 KDD Cup 2010

KDD Cup 2010 —— Data Analysis on algebra_2006_2007_train

Data Description

Column Description

Attribute	Annotaion
Row	The row number
Anon Student Id	Unique, anonymous identifier for a student
Problem Hierarchy	The hierarchy of curriculum levels containing the problem
Problem Name	Unique identifier for a problem
Problem View	The total number of times the student encountered the problem so far
Step Name	Unique identifier for one of the steps in a problem
Step Start Time	The starting time of the step (Can be null)
First Transaction Time	The time of the first transaction toward the step
Correct Transaction Time	The time of the correct attempt toward the step, if there was one
Step End Time	The time of the last transaction toward the step
Step Duration (sec)	The elapsed time of the step in seconds, calculated by adding all of the durations for transactions that were attributed to the step (Can be null if step start time is null)
Correct Step Duration (sec)	The step duration if the first attempt for the step was correct
Error Step Duration (sec)	The step duration if the first attempt for the step was an error (incorrect attempt or hint request)
Correct First Attempt	The tutor's evaluation of the student's first attempt on the step—1 if correct, 0 if an error
Incorrects	Total number of incorrect attempts by the student on the step
Hints	Total number of hints requested by the student for the step
Corrects	Total correct attempts by the student for the step (only increases if the step is encountered more than once)
KC(KC Model Name)	The identified skills that are used in a problem, where available
Opportunity(KC Model Name)	A count that increases by one each time the student encounters a step with the listed knowledge component
	Additional KC models, which exist for the challenge data sets, will appear as additional pairs of columns (KC and Opportunity columns for each model)

For the test portion of the challenge data sets, values will not be provided for the following columns:

Step Start Time

First Transaction Time

Correct Transaction Time

Step End Time

Step Duration (sec)

Correct Step Duration (sec)

Error Step Duration (sec)

Correct First Attempt

Incorrects

Hints

Corrects

[1]:
```
import pandas as pd
import plotly.express as px
```

[2]:
```
path = "algebra_2006_2007_train.txt"
data = pd.read_table(path, encoding="ISO-8859-15", low_memory=False)
```

Record Examples

[3]:
```
pd.set_option('display.max_column', 500)
data.head()
```

[3]:
```
   Row Anon Student Id                Problem Hierarchy Problem Name  \
0    1           JG4Tz  Unit CTA1_01, Section CTA1_01-1   LDEMO_WKST
1    2           JG4Tz  Unit CTA1_01, Section CTA1_01-1   LDEMO_WKST
2    3           JG4Tz  Unit CTA1_01, Section CTA1_01-1   LDEMO_WKST
3    4           JG4Tz  Unit CTA1_01, Section CTA1_01-1   LDEMO_WKST
4    5           JG4Tz  Unit CTA1_01, Section CTA1_01-1   LDEMO_WKST

   Problem View Step Name        Step Start Time First Transaction Time  \
0             1      R1C1  2006-10-26 09:51:58.0  2006-10-26 09:52:30.0
1             1      R1C2  2006-10-26 09:53:30.0  2006-10-26 09:53:41.0
2             1      R2C1  2006-10-26 09:53:41.0  2006-10-26 09:53:46.0
3             1      R2C2  2006-10-26 09:53:46.0  2006-10-26 09:53:50.0
4             1      R4C1  2006-10-26 09:53:50.0  2006-10-26 09:54:05.0

  Correct Transaction Time          Step End Time  Step Duration (sec)  \
0    2006-10-26 09:53:30.0  2006-10-26 09:53:30.0                 92.0
1    2006-10-26 09:53:41.0  2006-10-26 09:53:41.0                 11.0
2    2006-10-26 09:53:46.0  2006-10-26 09:53:46.0                  5.0
3    2006-10-26 09:53:50.0  2006-10-26 09:53:50.0                  4.0
4    2006-10-26 09:54:05.0  2006-10-26 09:54:05.0                 15.0

   Correct Step Duration (sec)  Error Step Duration (sec)  \
0                          NaN                       92.0
1                         11.0                        NaN
2                          5.0                        NaN
3                          4.0                        NaN
4                         15.0                        NaN

   Correct First Attempt  Incorrects  Hints  Corrects        KC(Default)  \
0                      0           2      0         1                NaN
1                      1           0      0         1                NaN
2                      1           0      0         1  Identifying units
3                      1           0      0         1  Identifying units
4                      1           0      0         1   Entering a given

  Opportunity(Default)
0                  NaN
1                  NaN
2                    1
```

(continues on next page)

(continued from previous page)

```
3                    2
4                    1
```

[4]:
```
data.describe()
```

[4]:
```
                 Row  Problem View  Step Duration (sec)  \
count  2.270384e+06  2.270384e+06         2.267551e+06
mean   1.513120e+06  1.092910e+00         1.958364e+01
std    8.736198e+05  3.448857e-01         4.768345e+01
min    1.000000e+00  1.000000e+00         0.000000e+00
25%    7.577408e+05  1.000000e+00         3.000000e+00
50%    1.511844e+06  1.000000e+00         7.000000e+00
75%    2.269432e+06  1.000000e+00         1.700000e+01
max    3.025933e+06  1.000000e+01         3.208000e+03

       Correct Step Duration (sec)  Error Step Duration (sec)  \
count                 1.751638e+06              515913.000000
mean                  1.171716e+01                  46.292087
std                   2.645318e+01                  81.817794
min                   0.000000e+00                   0.000000
25%                   3.000000e+00                  11.000000
50%                   5.000000e+00                  22.000000
75%                   1.100000e+01                  47.000000
max                   1.204000e+03                3208.000000

       Correct First Attempt    Incorrects         Hints      Corrects
count           2.270384e+06  2.270384e+06  2.270384e+06  2.270384e+06
mean            7.722359e-01  4.455044e-01  1.184311e-01  1.062878e+00
std             4.193897e-01  2.000914e+00  6.199071e-01  6.894285e-01
min             0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
25%             1.000000e+00  0.000000e+00  0.000000e+00  1.000000e+00
50%             1.000000e+00  0.000000e+00  0.000000e+00  1.000000e+00
75%             1.000000e+00  0.000000e+00  0.000000e+00  1.000000e+00
max             1.000000e+00  3.600000e+02  1.020000e+02  9.200000e+01
```

[5]:
```
print("Part of missing values for every column")
print(data.isnull().sum() / len(data))
```

```
Part of missing values for every column
Row                               0.000000
Anon Student Id                   0.000000
Problem Hierarchy                 0.000000
Problem Name                      0.000000
Problem View                      0.000000
Step Name                         0.000000
Step Start Time                   0.001103
First Transaction Time            0.000000
Correct Transaction Time          0.034757
Step End Time                     0.000000
Step Duration (sec)               0.001248
Correct Step Duration (sec)       0.228484
Error Step Duration (sec)         0.772764
```

(continues on next page)

(continued from previous page)

```
Correct First Attempt          0.000000
Incorrects                     0.000000
Hints                          0.000000
Corrects                       0.000000
KC(Default)                    0.203407
Opportunity(Default)           0.203407
dtype: float64
```

[6]:

```
print("the number of records:")
print(len(data))
```

```
the number of records:
2270384
```

[7]:

```
print("how many students are there in the table:")
print(len(data['Anon Student Id'].unique()))
```

```
how many students are there in the table:
1338
```

[8]:

```
print("how many problems are there in the table:")
print(len(data['Problem Name'].unique()))
```

```
how many problems are there in the table:
91913
```

Sort by Anon Student Id

[9]:

```
ds = data['Anon Student Id'].value_counts().reset_index()
ds.columns = [
    'Anon Student Id',
    'count'
]
ds['Anon Student Id'] = ds['Anon Student Id'].astype(str) + '-'
ds = ds.sort_values('count').tail(40)

fig = px.bar(
    ds,
    x='count',
    y='Anon Student Id',
    orientation='h',
    title='Top 40 students by number of steps they have done'
)
fig.show("svg")
```

Percent of corrects, hints and incorrects

[10]:
```
count_corrects = data['Corrects'].sum()
count_hints = data['Hints'].sum()
count_incorrects = data['Incorrects'].sum()

total = count_corrects + count_hints + count_incorrects

percent_corrects = count_corrects / total
percent_hints = count_hints / total
percent_incorrects = count_incorrects / total

dfl = [['corrects', percent_corrects], ['hints', percent_hints], ['incorrects', percent_
↪incorrects]]

df = pd.DataFrame(dfl, columns=['transaction type', 'percent'])

fig = px.pie(
    df,
    names=['corrects', 'hints', 'incorrects'],
    values='percent',
    title='Percent of corrects, hints and incorrects'
)
fig.show("svg")
```

Sort by Problem Name

[11]:
```
storeProblemCount = [1]
storeProblemName = [data['Problem Name'][0]]
currentProblemName = data['Problem Name'][0]
currentStepName = [data['Step Name'][0]]
lastIndex = 0

for i in range(1, len(data), 1):
    pbNameI = data['Problem Name'][i]
    stNameI = data['Step Name'][i]
    if pbNameI != data['Problem Name'][lastIndex]:
        currentStepName = [stNameI]
        currentProblemName = pbNameI
        if pbNameI not in storeProblemName:
            storeProblemName.append(pbNameI)
            storeProblemCount.append(1)
        else:
            storeProblemCount[storeProblemName.index(pbNameI)] += 1
        lastIndex = i
    elif stNameI not in currentStepName:
        currentStepName.append(stNameI)
        lastIndex = i
```

(continues on next page)

(continued from previous page)

```
    else:
        currentStepName = [stNameI]
        storeProblemCount[storeProblemName.index(pbNameI)] += 1
        lastIndex = i

dfData = {
    'Problem Name': storeProblemName,
    'count': storeProblemCount
}
df = pd.DataFrame(dfData).sort_values('count').tail(40)
df["Problem Name"] += '-'

fig = px.bar(
    df,
    x='count',
    y='Problem Name',
    orientation='h',
    title='Top 40 useful problem'
)
fig.show("svg")
```

[12]:

```
data['total transactions'] = data['Incorrects'] + data['Hints'] + data['Corrects']
df1 = data.groupby('Problem Name')['total transactions'].sum().reset_index()
df2 = data.groupby('Problem Name')['Corrects'].sum().reset_index()
df1['Corrects'] = df2['Corrects']
df1['Correct rate'] = df1['Corrects'] / df1['total transactions']

df1 = df1.sort_values('total transactions')
count = 0
standard = 500
for i in df1['total transactions']:
    if i > standard:
        count += 1
df1 = df1.tail(count)

df1 = df1.sort_values('Correct rate')

df1['Problem Name'] = df1['Problem Name'].astype(str) + "-"

df_px = df1.tail(20)

fig = px.bar(
    df_px,
    x='Correct rate',
    y='Problem Name',
    orientation='h',
    title='Correct rate of each problem (top 20)  (total transactions of \
each problem are required to be more than 500)',
    text='Problem Name'
)
fig.update_layout(title_font_size=10)
```

(continues on next page)

(continued from previous page)

```
fig.show("svg")

df_px = df1.head(20)

fig = px.bar(
    df_px,
    x='Correct rate',
    y='Problem Name',
    orientation='h',
    title='Correct rate of each problem (bottom 20)  (total transactions of \
each problem are required to be more than 500)',
    text='Problem Name'
)
fig.update_layout(title_font_size=10)
fig.show("svg")
```

These two figures present the correct rate of problems. Problems with low correct rate deserve more attention from teachers and students.

Sort by KC

[13]:
```
data.dropna(subset=['KC(Default)'], inplace=True)

data['total transactions'] = data['Corrects'] + data['Hints'] + data['Incorrects']
df1 = data.groupby('KC(Default)')['total transactions'].sum().reset_index()
df2 = data.groupby('KC(Default)')['Corrects'].sum().reset_index()
df1['Corrects'] = df2['Corrects']
df1['correct rate'] = df1['Corrects'] / df1['total transactions']

count = 0
standard = 300
for i in df1['total transactions']:
    if i > standard:
        count += 1
df1 = df1.sort_values('total transactions').tail(count)

df1 = df1.sort_values('correct rate')

df1['KC(Default)'] = df1['KC(Default)'].astype(str) + '-'

df_px = df1.tail(20)

fig = px.bar(
    df_px,
    x='correct rate',
    y='KC(Default)',
    orientation='h',
    title='Correct rate of each KC(Default) (top 20)  (total transactions of \
each KC are required to be more than 300)',
    text='KC(Default)'
```

(continues on next page)

(continued from previous page)

```
)
fig.update_yaxes(visible=False)
fig.update_layout(title_font_size=10)
fig.show("svg")

df_px = df1.head(20)

fig = px.bar(
    df_px,
    x='correct rate',
    y='KC(Default)',
    orientation='h',
    title='Correct rate of each KC(Default) (bottom 20)  (total transactions of \
each KC are required to be more than 300)',
    text='KC(Default)'
)
fig.update_yaxes(visible=False)
fig.update_layout(title_font_size=10)
fig.show("svg")
```

These two figures present the correct rate of KCs. KCs with low correct rate deserve more attention from teachers and students.

Postscript

Given that the whole data package is composed of 5 data sets and data files in these 5 data sets that can be used to conduct data analysis share the same data format, the following analysis based on "algebra_2006_2007_train" is just an example of data analysis on KDD Cup, and the code can be used to analyse other data files with some small changes on the file path and column names.

6.7.7 Math23k

Math23k Analysis Report

Data Description

Field	Annotation
id	Id of the problem
original_text	Original text of the problem
equation	Solution to the problem
segmented_text	Chinese word segmentation of the problem

[1]:
```
import numpy as np
import pandas as pd
import jieba

import plotly.express as px
```

(continues on next page)

(continued from previous page)

```
from plotly.subplots import make_subplots
import plotly.graph_objs as go
```

[2]:
```
path1 = "./math23k/raw/train23k.json"
path2 = "./math23k/raw/test23k.json"
path3 = "./math23k/raw/valid23k.json"

data  = pd.read_json(path1, orient='records')
data2 = pd.read_json(path2, orient='records')
data3 = pd.read_json(path3, orient='records')
data = pd.concat([data, data2, data3])
```

Record Examples

[3]:
```
data.head()
```

[3]:
```
      id                                          original_text  \
0    946               1.5204=
1  21227  AB5...
2  16892             30(1/5)5
3   8502  2303354...
4  23021                           30020%

                     equation  \
0            x=20/(4-1.5)*1.5
1  x=196/(80%+((3)/(3+2))-1)
2            x=30*(1-(1/5))+5
3             x=(230-35)/3-48
4               x=300/(1+20%)

                                        segmented_text
0          1.5       20      ...
1       A   B       5     ...
2      30      (1/5)       5...
3         230         3   ...
4            300       20%
```

The number of problems

[4]:
```
len(data['id'].unique())
```

[4]:
```
23162
```

Part of missing values for every column

[5]:
```
data.isnull().sum() / len(data)
```

[5]:
```
id                 0.0
original_text      0.0
equation           0.0
segmented_text     0.0
dtype: float64
```

Cut words and find verbs in problems

Verbs may be quite useful for solving math word problems, because sometimes a verb means an operator in equation.

[6]:
```
import jieba.posseg as pseg
def cut_word(text):
    return jieba.lcut(text)

def find_verbs(text):
    words = pseg.cut(text)
    return [word for word,flag in words if flag == 'v']

data['content']=data['original_text'].apply(cut_word)
data['verbs']=data['original_text'].apply(find_verbs)
data.head()
```

```
Building prefix dict from the default dictionary ...
Loading model from cache /tmp/jieba.cache
Loading model cost 0.563 seconds.
Prefix dict has been built successfully.
```

[6]:
```
      id                                      original_text  \
0    946              1.5204=
1  21227  AB5...
2  16892            30(1/5)5
3   8502  2303354...
4  23021                            30020%

                  equation  \
0          x=20/(4-1.5)*1.5
1  x=196/(80%+((3)/(3+2))-1)
2          x=30*(1-(1/5))+5
3           x=(230-35)/3-48
4             x=300/(1+20%)
```

(continues on next page)

(continued from previous page)

```
                                     segmented_text  \
0          1.5      20      ...
1       A   B       5     ...
2      30       (1/5)        5...
3         230         3    ...
4            300       20%

                                            content  \
0  [, , , , , 1.5, , , , , 20, ,...
1  [, , , , , A, , B, , , , , ,...
2  [, , , 30, , , , , , (, 1, /, 5, ...
3  [, , , , , , , 230, , , , , ...
4  [, , , , 300, , , , , , 20%, , ...

                                   verbs
0                                [, ]
1  [, , , , , , , , , , ]
2                             [, , ]
3                       [, , , ]
4                                  [, ]
```

Count of words of problems

```
[7]: def getsize(ser):
         return len(ser)

     data['word_count']=data['content'].apply(getsize)
     data.head()
```

```
[7]:      id                                         original_text  \
0    946              1.5204=
1  21227  AB5...
2  16892          30(1/5)5
3   8502  2303354...
4  23021                        30020%

                      equation  \
0           x=20/(4-1.5)*1.5
1  x=196/(80%+((3)/(3+2))-1)
2           x=30*(1-(1/5))+5
3            x=(230-35)/3-48
4              x=300/(1+20%)

                                     segmented_text  \
0          1.5      20      ...
1       A   B       5     ...
2      30       (1/5)        5...
3         230         3    ...
4            300       20%

                                            content  \
```

(continues on next page)

(continued from previous page)

```
0  [, , , , , 1.5, , , , , 20, ,...
1  [, , , , , A, , B, , , , , ,...
2  [, , , 30, , , , , , (, 1, /, 5, ...
3  [, , , , , , , 230, , , , , ...
4  [, , , , 300, , , , , , 20%, , ...

                                    verbs  word_count
0                                  [, ]          24
1  [, , , , , , , , , ]          63
2                               [, , ]          28
3                         [, , , ]          38
4                                   [, ]          17
```

The length of problems

This picture shows that the length of most problems are about 20 to 40 chinese words.It may be helpful for design of model's input

[8]:
```
cnt = data['word_count'].value_counts().reset_index()
cnt.columns = [ 'word_count' , 'problem_count']

fig = px.bar(
    cnt , x = 'word_count', y = 'problem_count' ,
    title = 'The length of problems'
)
fig.show('svg')
```

Delete stopword

[ ]:
```
```

[9]:
```
def get_stopword():
    s = set()
    with open("../raw_data/stopword/stopword.txt","r",encoding="UTF-8") as f:
        for line in f:
            s.add(line.strip())
    return s

def delete_stopword(words):
    return [w for w in words if (w not in stopword)]

stopword=get_stopword()
data['content']=data['content'].apply(delete_stopword)
data.head()
```

[9]:
```
      id                                      original_text  \
0    946              1.5204=
1  21227  AB5...
2  16892          30(1/5)5
```

(continues on next page)

(continued from previous page)

```
3   8502  2303354...
4  23021                        30020%

                   equation  \
0          x=20/(4-1.5)*1.5
1  x=196/(80%+((3)/(3+2))-1)
2          x=30*(1-(1/5))+5
3           x=(230-35)/3-48
4             x=300/(1+20%)

                                     segmented_text  \
0          1.5       20       ...
1       A   B       5      ...
2      30       (1/5)        5...
3         230        3    ...
4            300       20%

                                            content  \
0    [, , , , 1.5, , , 20, , , , ]
1  [, , A, B, , , , , , , , ,...
2       [, , , 30, , , , , , , ]
3  [, , , , 230, , , , , , , ...
4              [, , , 300, , , 20%, , ]

                                   verbs  word_count
0                                 [, ]          24
1  [, , , , , , , , , , ]           63
2                               [, , ]          28
3                         [, , , ]          38
4                                  [, ]          17
```

The keywords

Keywords may show us the topic of problem sometimes.They are useful for our analysis. This report use textrank algorithm in 'jieba'. because length of problem are usually short ,TF/IDF may be not suitable for this dataset.

[10]:
```
import jieba.analyse
def get_keyword(text):
    topk = min(3,len(text))
    keyword = [word for word in jieba.analyse.textrank(text, topK = topk)]
    return keyword

data['keywords'] = data['original_text'].apply(get_keyword)
data.head()
```

[10]:
```
      id                                original_text  \
0    946              1.5204=
1  21227  AB5...
2  16892          30(1/5)5
3   8502  2303354...
4  23021                        30020%
```

(continues on next page)

(continued from previous page)

```
                    equation  \
0           x=20/(4-1.5)*1.5
1  x=196/(80%+((3)/(3+2))-1)
2           x=30*(1-(1/5))+5
3            x=(230-35)/3-48
4              x=300/(1+20%)

                                    segmented_text  \
0          1.5      20      ...
1       A   B      5     ...
2      30       (1/5)        5...
3         230        3   ...
4            300      20%

                                           content  \
0    [, , , , 1.5, , , 20, , , , ]
1  [, , A, B, , , , , , , , ,...
2       [, , , 30, , , , , , , , ]
3  [, , , , 230, , , , , , , ...
4               [, , , 300, , , 20%, , ]

                                  verbs  word_count      keywords
0                                [, ]          24  [, , ]
1  [, , , , , , , , , , ]          63  [, , ]
2                              [, , ]          28     [, ]
3                        [, , , ]          38  [, , ]
4                                 [, ]          17     [, ]
```

Topic Prediction

Classify problems by their topics may be helpful for models and analyse the result of models in different fields. Because there're no labels in original data, unsupervised algorithm LDA may be suitable.

[11]:
```
from gensim import corpora, models

all_words = []
for text in data['content']:
    all_words.append(text)
#print(all_words)
dictionary = corpora.Dictionary(all_words)
corpus = [dictionary.doc2bow(text) for text in all_words]

lda = models.ldamodel.LdaModel(corpus = corpus, id2word = dictionary, num_topics = 5)

print('keywords of topics')
for topic in lda.print_topics(num_words = 5):
    print(topic)
```

```
keywords of topics
(0, '0.067*"" + 0.018*"" + 0.016*"" + 0.013*"" + 0.013*""')
(1, '0.060*"" + 0.057*"" + 0.026*"" + 0.024*"" + 0.020*""')
(2, '0.026*"" + 0.025*"" + 0.025*"" + 0.024*"" + 0.014*""')
(3, '0.041*"" + 0.031*"" + 0.022*"" + 0.017*"" + 0.015*""')
(4, '0.117*"" + 0.030*"" + 0.019*"" + 0.014*"" + 0.011*""')
```

[12]:

```
topic = []
for i,values in enumerate(lda.inference(corpus)[0]):
    topic_val = 0
    topic_id = 0
    for tid, val in enumerate(values):
        if val > topic_val:
            topic_val = val
            topic_id = tid
    topic.append(topic_id)
data['topic'] = topic
data.head(10)
```

[12]:

```
      id                                       original_text  \
0    946              1.5204=
1  21227  AB5...
2  16892          30(1/5)5
3   8502  2303354...
4  23021                          30020%
5   5901  20%(2/7...
6  12815            36408
7  19584  720154...
8  10773                               94
9  22037            (3/5)(1/4)

                      equation  \
0             x=20/(4-1.5)*1.5
1     x=196/(80%+((3)/(3+2))-1)
2             x=30*(1-(1/5))+5
3              x=(230-35)/3-48
4                x=300/(1+20%)
5   x=(5-2)/(20%+(2/7)+(3/5)-1)
6                    x=36*40/8
7                   x=720/15/4
8                        x=9+4
9          x=(3/5)+(1/4)+(3/5)

                                       segmented_text  \
0          1.5       20       ...
1       A   B       5      ...
2      30       (1/5)          5...
3         230         3    ...
4            300       20%
5                 20% ...
6     36        40           ...
7            720      ...
```

(continues on next page)

(continued from previous page)

```
8            9         4
9        (3/5)          (1/4)  ...

                                                  content  \
0    [, , , , 1.5, , , 20, , , , ]
1  [, , A, B, , , , , , , , ,...
2      [, , , 30, , , , , , , , ]
3  [, , , , 230, , , , , , , ...
4              [, , , 300, , , 20%, , ]
5  [, , , , , , , , , 20%, ...
6  [, 36, , , , , 40, , , , , , ...
7  [, , , , , , , , 720, , ,...
8                              [, , , ]
9                  [, , , , , , , , ]

                              verbs  word_count        keywords  topic
0                            [, ]          24    [, , ]      3
1  [, , , , , , , , , ]          63    [, , ]      1
2                          [, , ]          28       [, ]      3
3                      [, , , ]          38    [, , ]      1
4                            [, ]          17       [, ]      1
5                [, , , , , , ]          55    [, , ]      0
6                        [, , , ]          27    [, , ]      3
7                  [, , , , , ]          36  [, , ]      2
8                            [, ]          16              []      2
9                        [, , , ]          32    [, , ]      0
```

[13]:
```
output = data['topic'].value_counts().reset_index()
output.columns=['topic_id','number of problems']

fig = px.pie(
    output,
    names = 'topic_id',
    values = 'number of problems',
    title = 'Topic of problems'
)

fig.show("svg")
```

Number of operators

If you know how many operators are there in equations, it may be much easier for you to solve math word problems.Especially when your algorithm are based on equation templates.

[14]:
```
def num_of_operators(equation):
    cnt = 0
    for op in equation:
        if op =='(' or op == '+' or op == '-' or op == '*' or op == '/' or op == '^':
            cnt += 1
    return cnt
```

(continues on next page)

(continued from previous page)

```
tmp = data.loc[:,['equation']]
tmp['operators_cnt'] = tmp['equation'].apply(num_of_operators)
cnt = tmp['operators_cnt'].value_counts().reset_index()
output = cnt.head(10)
other_sum = cnt['operators_cnt'].sum() - output['operators_cnt'].sum()

output = output.sort_values(['operators_cnt'])
output.loc[10] = ['other', other_sum]

output.columns=['number of operators','number of problems']

fig = px.pie(
    output,
    names = 'number of operators',
    values = 'number of problems',
    title = 'Number of operators'
)

fig.show("svg")
```

Evaluate difficulty

Different problems have different difficulty.People may choose different way to solve problems when difficulty of problems are different,and so is AI.To evaluate difficulty of problems, the kinds of operators in equations may be useful.Value of them are as follows.

[9]:
```
def calc_difficulty(equation):
    difficulty = 0

    def eval(x):
        if x == '+' : return 2
        elif x == '-' : return 3
        elif x == '*' : return 5
        elif x == '/' : return 7
        elif x == '(' : return 8
        elif x == '%' : return 5
        elif x == '^' : return 6
        else : return 0

    for op in equation:
        difficulty += eval(op)
    return difficulty


data['difficulty'] = data['equation'].apply(calc_difficulty)

cnt = data['difficulty'].value_counts().reset_index()
cnt.columns = [ 'difficulty' , 'problem_count']
```

(continues on next page)

(continued from previous page)

```
fig = px.bar(
    cnt , x = 'difficulty', y = 'problem_count' ,
    title = 'The difficulty of problems'
)
fig.show('svg')
```

The most difficult problems

[10]:
```
tmp = data[['id','original_text','difficulty']]
tmp = tmp.sort_values(['difficulty']).tail(10)
tmp ['id'] = tmp['id'] . astype(str)
fig = px.bar(
    tmp , x = 'difficulty', y = 'id' ,
    orientation = 'h',
    title = 'The difficulty of problems'
)
fig.show('svg')
```

Simplify expressions

Algorithm based on templates will find templates in equations at first.To find templates, we should simplify expressions first.'+' means operator '+' or '-', '*' means operator '*' or '/', 'n' means a number.

[17]:
```
from pythonds.basic.stack import Stack


def simplify(expr):
    n = len(expr)
    output = ''
    flag = True
    for i in range(2,n):
        if flag and (expr[i].isdigit() or expr[i] == '.' or expr[i] == '%'):
            output = output + 'n'
            flag = False
        if not (expr[i].isdigit() or expr[i] == '.' or expr[i] == '%'):
            if expr[i] == '[' or expr[i] == '{':
                output = output + '('
            elif expr[i] == ']' or expr[i] == '}':
                output = output + ')'
            elif expr[i] == '-':
                output = output + '+'
            elif expr[i] == '/':
                output = output + '*'
            else: output = output + expr[i]
            flag = True
    return output

data['post_expression'] = data['equation'].apply(simplify)
```

Count of numbers in equations

[18]:
```
def CountNum(expr):
    cnt = 0
    for x in expr:
        if x == 'n':
            cnt = cnt + 1
    return cnt


tmp = data.loc[:,['post_expression','original_text']]
tmp['number_cnt'] = tmp['post_expression'].apply(CountNum)


cnt = tmp['number_cnt'].value_counts().reset_index()
output = cnt.head(10)
other_sum = cnt['number_cnt'].sum() - output['number_cnt'].sum()

output = output.sort_values(['number_cnt'])
output.loc[10] = ['other', other_sum]

output.columns=['count of numbers','number of problems']

fig = px.pie(
    output,
    names = 'count of numbers',
    values = 'number of problems',
    title = 'Count of numbers in equations'
)

fig.show("svg")
```

Are numbers in equations as many as in problems?

This result shows that about half of problems have useless parameters or potential parameters in problems

[19]:
```
def NuminProb(text):
    prob = str(text)
    cnt = 0
    flag = True

    for w in prob:
        if w.isdigit() or w == '.' or w == '%':
            if flag:
                cnt += 1
                flag = False
        else:
            flag =True
```

(continues on next page)

(continued from previous page)

```
    return cnt

def isSame(a, b):
    if a == b:
        return True
    else:
        return False

tmp['num_in_prob'] = tmp['original_text'].apply(NuminProb)
tmp['same count'] = tmp.apply(lambda row: isSame(row['number_cnt'], row['num_in_prob']),␣
↪axis=1)
same = tmp['same count'].value_counts().reset_index()

fig = px.pie(
    same,
    names = 'index',
    values = 'same count',
    title = 'Are numbers in equation as many as in problems?'
)
fig.show("svg")
```

Postfix expressions

Some algorithm need postfix expressions instead of infix expressions.The reasons for that may be postfix expressions can help us build expression trees,and there are no brackets in postfix expressions,so postfix expressions can merge some template.

[20]:
```
def InfixToPostfix(infixexpr):
    prec = {}
    prec['^'] = 4
    prec["*"] = 3
    prec["/"] = 3
    prec["+"] = 2
    prec["-"] = 2
    prec["("] = 1

    opstack = Stack()
    postfixList = []

    for token in infixexpr:
        if token == 'n':
            postfixList.append(token)
        elif token == "(":
            opstack.push(token)
        elif token == ")":
            topstack = opstack.pop()
            while topstack != "(":
                postfixList.append(topstack)
                if opstack.isEmpty():
                    print(infixexpr)
```

(continues on next page)

(continued from previous page)

```
                else :
                    topstack = opstack.pop()
            else:
                while (not opstack.isEmpty()) and (prec[opstack.peek()] >= prec[token]):
                    postfixList.append(opstack.pop())
                opstack.push(token)
    while not opstack.isEmpty():
        postfixList.append(opstack.pop())
    return ''.join(postfixList)


data['post_expression'] = data['post_expression'].apply(InfixToPostfix)

data.head()
```

[20]:

```
      id                                     original_text  \
0    946              1.5204=
1  21227  AB5...
2  16892          30(1/5)5
3   8502  2303354...
4  23021                            30020%

                  equation  \
0          x=20/(4-1.5)*1.5
1  x=196/(80%+((3)/(3+2))-1)
2          x=30*(1-(1/5))+5
3           x=(230-35)/3-48
4             x=300/(1+20%)

                                    segmented_text  \
0          1.5       20      ...
1       A   B       5      ...
2      30      (1/5)        5...
3         230         3    ...
4            300       20%

                                           content  \
0    [, , , , 1.5, , , 20, , , , ]
1   [, , A, B, , , , , , , , ,...
2       [, , , 30, , , , , , , , ]
3   [, , , , 230, , , , , , , ...
4                [, , , 300, , , 20%, , ]

                                verbs  word_count      keywords  topic  \
0                                [, ]          24   [, , ]      3
1  [, , , , , , , , , , ]          63  [, , ]       1
2                              [, , ]          28      [, ]      3
3                        [, , , ]          38  [, , ]       1
4                                 [, ]          17      [, ]      1

   difficulty post_expression
0          23         nnn+*n*
```

(continues on next page)

(continued from previous page)

```
1          58      nnnnn+*+n+*
2          33        nnnn*+*n+
3          21          nn+n*n+
4          22            nnn+*
```

Templates of postfix expressions

Template may be useful to solve math word problems. In fact,many algorithms are based on them.The result shows that 15 kinds of postfix templates can help us solve about 70% of problems.

[21]:
```
ds = data['post_expression'].value_counts().reset_index()
ds = ds.sort_values(['post_expression'])

output = ds.tail(15)
other_sum = ds['post_expression'].sum() - output['post_expression'].sum()

output.columns = [
    'post_expression',
    'percent'
]

output = output.sort_values(['percent'])
output.loc[15] = ['others', other_sum]

fig = px.pie(
    output,
    names = 'post_expression',
    values = 'percent',
    title = 'Templates of postfix expressions',
)

fig.show("svg")
```

Reference

@inproceedings{Liu2019TreestructuredDF, title={Tree-structured Decoding for Solving Math Word Problems}, author={Qianying Liu and Wenyv Guan and Sujian Li and Daisuke Kawahara}, booktitle={EMNLP/IJCNLP}, year={2019} }

@inproceedings{Xie2019AGT, title={A Goal-Driven Tree-Structured Neural Model for Math Word Problems}, author={Zhipeng Xie and Shichao Sun}, booktitle={IJCAI}, year={2019} }

@inproceedings{Wang2019TemplateBasedMW, title={Template-Based Math Word Problem Solvers with Recursive Neural Networks}, author={Lei Wang and D. Zhang and Jipeng Zhang and Xing Xu and L. Gao and B. Dai and H. Shen}, booktitle={AAAI}, year={2019} }

@article{Lee2020SolvingAW, title={Solving Arithmetic Word Problems with a Templatebased Multi-Task Deep Neural Network (T-MTDNN)}, author={D. Lee and G. Gweon}, journal={2020 IEEE International Conference on Big Data and Smart Computing (BigComp)}, year={2020}, pages={271-274} }

6.7.8 pisa2015math

[1]:
```
import pandas as pd
import numpy as np
import plotly.express as px
```

[2]:
```
file_path = "/home/huzr/pisa2015_science/pisa2015_science/cog_science.csv"
df = pd.read_csv(file_path, low_memory=False)
```

Data Description

Field	annotation
CNTRYID	The country which the student comes from
CNTSTUID	Student ID
Region	The region which the student comes from
STRATUM	Types of students classified by school and place of birth, etc.
SUBNATIO	Further division of country
ADMINMODE	The means by which the student answer questions(paper or computer)
LANGTEST_COG	The language which the student uses
BOOKID	Form ID
CBASCI	Science Cluster Combination Random Number(S)
CS601Q01S, CS601Q02S … DS626Q04C	Question ID

Record Examples

[3]:
```
df.head()
```

[3]:
```
   CNTRYID  CNTSTUID   Region                                   STRATUM  \
0  Albania    803627  Albania  ALB - stratum 05: Urban \ South \ Public
1  Albania    800454  Albania  ALB - stratum 05: Urban \ South \ Public
2  Albania    800893  Albania  ALB - stratum 05: Urban \ South \ Public
3  Albania    804180  Albania  ALB - stratum 05: Urban \ South \ Public
4  Albania    800491  Albania  ALB - stratum 05: Urban \ South \ Public

  SUBNATIO ADMINMODE LANGTEST_COG         BOOKID CBASCI DS269Q01C  ...  \
0  Albania     Paper     Albanian  Form 27 (PBA)    NaN       NaN  ...
1  Albania     Paper     Albanian   Form 9 (PBA)    NaN       NaN  ...
2  Albania     Paper     Albanian  Form 18 (PBA)    NaN       NaN  ...
3  Albania     Paper     Albanian  Form 10 (PBA)    NaN       NaN  ...
4  Albania     Paper     Albanian  Form 19 (PBA)    NaN       NaN  ...

  CS601Q01S CS601Q02S CS601Q04S DS610Q01C CS610Q02S CS610Q04S CS626Q01S  \
0       NaN       NaN       NaN       NaN       NaN       NaN       NaN
1       NaN       NaN       NaN       NaN       NaN       NaN       NaN
2       NaN       NaN       NaN       NaN       NaN       NaN       NaN
3       NaN       NaN       NaN       NaN       NaN       NaN       NaN
4       NaN       NaN       NaN       NaN       NaN       NaN       NaN
```

(continues on next page)

(continued from previous page)

```
  CS626Q02S CS626Q03S DS626Q04C
0       NaN       NaN       NaN
1       NaN       NaN       NaN
2       NaN       NaN       NaN
3       NaN       NaN       NaN
4       NaN       NaN       NaN

[5 rows x 192 columns]
```

[4]:
```
print("{} Students from {} regions around the world participate in this assessment.".
↪format(len(df['CNTSTUID']), len(df['Region'].unique())))
```

```
519334 Students from 118 regions around the world participate in this assessment.
```

The columns begin with “DS” or “CS” are the questions.

[5]:
```
question_ids = [col for col in df.columns if 'DS' in col or 'CS' in col]
print("There are {} questions involved.".format(len(question_ids)))
```

```
There are 183 questions involved.
```

Data cleaning

Drop useless columns

[6]:
```
useless_columns = ["CBASCI", "BOOKID", "ADMINMODE", "SUBNATIO", "LANGTEST_COG"]
df.drop(columns=useless_columns, inplace=True)
df.head()
```

[6]:
```
   CNTRYID  CNTSTUID   Region                                   STRATUM  \
0  Albania    803627  Albania  ALB - stratum 05: Urban \ South \ Public
1  Albania    800454  Albania  ALB - stratum 05: Urban \ South \ Public
2  Albania    800893  Albania  ALB - stratum 05: Urban \ South \ Public
3  Albania    804180  Albania  ALB - stratum 05: Urban \ South \ Public
4  Albania    800491  Albania  ALB - stratum 05: Urban \ South \ Public

  DS269Q01C DS269Q03C CS269Q04S CS408Q01S DS408Q03C CS408Q04S  ... CS601Q01S  \
0       NaN       NaN       NaN       NaN       NaN       NaN  ...       NaN
1       NaN       NaN       NaN       NaN       NaN       NaN  ...       NaN
2       NaN       NaN       NaN       NaN       NaN       NaN  ...       NaN
3       NaN       NaN       NaN       NaN       NaN       NaN  ...       NaN
4       NaN       NaN       NaN       NaN       NaN       NaN  ...       NaN

  CS601Q02S CS601Q04S DS610Q01C CS610Q02S CS610Q04S CS626Q01S CS626Q02S  \
0       NaN       NaN       NaN       NaN       NaN       NaN       NaN
1       NaN       NaN       NaN       NaN       NaN       NaN       NaN
2       NaN       NaN       NaN       NaN       NaN       NaN       NaN
3       NaN       NaN       NaN       NaN       NaN       NaN       NaN
4       NaN       NaN       NaN       NaN       NaN       NaN       NaN

  CS626Q03S DS626Q04C
```

(continues on next page)

(continued from previous page)

```
0       NaN       NaN
1       NaN       NaN
2       NaN       NaN
3       NaN       NaN
4       NaN       NaN

[5 rows x 187 columns]
```

Transform the result to score

[7]:
```
def transform_to_number(ans):
    if isinstance(ans, float):
        return np.NaN
    if 'No credit' in ans:
        return 0
    if 'Partial credit' in ans:
        return 1
    if 'Full credit' in ans:
        return 2
    if 'No Response' in ans:
        return np.NaN
    if 'Not Reached' in ans:
        return np.NaN
    if 'Not Applicable' in ans:
        return np.NaN

    return np.NaN

for q in question_ids:

    df[q] = df[q].map(transform_to_number)

df_question = df[question_ids]
```

[8]:
```
df_difficulty = df_question.transpose().mean(1) / 2
# define difficulty equals average score divided by full score(2)
df_question = df_question / df_difficulty
# The more difficult questions will bring more score
```

[9]:
```
print("5 least answered questions:")
df_question.count().nsmallest()
```

```
5 least answered questions:
```

[9]:
```
DS327Q02C        0
DS438Q03C    38225
DS425Q04C    39228
DS524Q07C    39777
DS268Q02C    39958
dtype: int64
```

The question "DS327Q02C" is bad, because no one has given the correct answer.

[10]:
```
df_question.drop(columns=['DS327Q02C'], inplace=True)
print("The number of students who answer each question:")
df_question.count()
```

```
The number of students who answer each question:
```

[10]:
```
DS269Q01C    44197
DS269Q03C    43324
CS269Q04S    48415
CS408Q01S    48440
DS408Q03C    42613
             ...
CS610Q04S    91081
CS626Q01S    92980
CS626Q02S    91846
CS626Q03S    91033
DS626Q04C    89762
Length: 182, dtype: int64
```

Calculate average score for each student

[11]:
```
df_student = df.copy()
df_student['Avg score'] = df_question.mean(1)
df_student['Question count'] = df_question.count(1)
df_student.drop(columns=question_ids, inplace=True)
df_student.drop(df_student[df_student['Question count'] == 0].index, inplace=True)
```

[12]:
```
print("{} students give at least one valid answer.".format(len(df_student)))
```

```
446051 students give at least one valid answer.
```

[13]:
```
df_student.head()
```

[13]:
```
         CNTRYID  CNTSTUID     Region                         STRATUM  \
10734  Australia   3610676  Australia  AUS - stratum 21: QLD-Gov, Y10
10735  Australia   3611874  Australia  AUS - stratum 21: QLD-Gov, Y10
10736  Australia   3601769  Australia  AUS - stratum 21: QLD-Gov, Y10
10737  Australia   3605996  Australia  AUS - stratum 21: QLD-Gov, Y10
10738  Australia   3608147  Australia  AUS - stratum 21: QLD-Gov, Y10

       Avg score  Question count
10734   2.945319              19
10735   2.693977              27
10736   2.982012              23
10737   2.330750              32
10738   1.752320              32
```

Sort questions by average score

[27]:
```
q_top20 = df_difficulty.nlargest(20)
fig = px.bar(
    q_top20.iloc[::-1],
    labels={"value": "Difficulty", "index": "Question Name"},
    orientation='h',
    title="Top 20 easy questions"
            )
fig.update_layout(showlegend=False)
fig.show('svg')
```

[14]:
```
q_bottom20 = df_difficulty.nsmallest(20)
fig = px.bar(
    q_bottom20.iloc[::-1],
    labels={"value": "Difficulty", "index": "Question Name"},
    orientation='h',
    title="Top 20 difficult questions"
)
fig.update_layout(showlegend=False)
fig.show('svg')
```

Sort regions by average score

[15]:
```
df_region = \
    df_student[["Region", "Avg score", "Question count"]] \
        .groupby("Region").mean().rename(columns={"Avg score": "Region avg score"})
```

[16]:
```
df_region.describe()
```

[16]:

	Region avg score	Question count
count	103.000000	103.000000
mean	1.988160	29.004939
std	0.397446	1.280883
min	0.822475	22.413025
25%	1.782955	28.684978
50%	2.119420	29.295016
75%	2.259783	29.780795
max	2.635470	30.473813

[17]:
```
region_top20 = df_region["Region avg score"].nlargest(20)
fig = px.bar(
    region_top20.iloc[::-1],
    labels={"value": "Region average Score", "index": "Region"},
    orientation='h',
    title="Top 20 regions"
)
fig.update_layout(showlegend=False)
fig.show("svg")
```

[18]:
```
region_bottom20 = df_region["Region avg score"].nsmallest(20)
fig = px.bar(
    region_bottom20.iloc[::-1],
    labels={"index": "Region", "value": "Region"},
    orientation="h",
    title="Bottom 20 regions"
)
fig.update_layout(showlegend=False)
fig.show("svg")
```

The distrbution of scores

[19]:
```
df_student.describe()
```

[19]:
```
           CNTSTUID      Avg score  Question count
count  4.460510e+05  446051.000000   446051.000000
mean   4.689134e+07       1.955394       28.581337
std    3.016672e+07       1.059917        5.101046
min    3.600001e+06       0.000000        1.000000
25%    1.880574e+07       1.101785       26.000000
50%    4.400022e+07       1.816121       29.000000
75%    7.520527e+07       2.694062       33.000000
max    9.731129e+07       8.116385       55.000000
```

[21]:
```
def level(score):
    return int(score * 10)

df_student_level = df_student["Avg score"].apply(level)
df_student_level_dist = df_student_level.groupby(df_student_level).size()
fig = px.bar(
    df_student_level_dist,
    labels={"value": "Students counts", "index": "Score range"},
    title="Score distribution",
    hover_data={
        "variable":False,
    }
)
fig.update_layout(showlegend=False)
fig.show('svg')

df_student_level_dist
```

[21]:
```
Avg score
0      2055
1      2159
2      4048
3      6144
4      8811
```

(continues on next page)

(continued from previous page)

```
      ...
67        1
70        1
72        1
79        1
81        3
Name: Avg score, Length: 72, dtype: int64
```

Sort students by average score

[22]:
```
df_student["CNTSTUID"] = df_student["CNTSTUID"].apply(str)
s_top500 = \
    df_student[df_student["Question count"] >= 10] \
        .nlargest(500, ["Avg score"]).set_index("CNTSTUID")
# We only select students who have answered more than 10 questions
s_top500.index = s_top500.index.map(str)
fig = px.bar(
    s_top500[:20].iloc[::-1],
    x = "Avg score",
    orientation="h",
    title="The score of top 20 students"
)
fig.update_layout(yaxis_type='category')
fig.update_layout(showlegend=False)
fig.show('svg')
```

[24]:
```
s_top500_dist = s_top500.groupby("CNTRYID").size()
s_top500_dist = s_top500_dist[s_top500_dist > 10]
fig = px.pie(
    values=s_top500_dist,
    names=s_top500_dist.index,
    title="Top 500 students come from"
)
fig.show('svg')
```

[25]:
```
s_bottom500 = df_student[(df_student["Question count"] > 10) & (df_student["Avg score"] >
↪ 0.1)]\
    .nsmallest(500, "Avg score")
# A too low score indicates that the student may answer questions blindly
s_bottom500.set_index("CNTSTUID")
s_bottom500.index = s_bottom500.index.map(str)
fig = px.bar(
    s_bottom500.iloc[:20].iloc[::-1],
    x = "Avg score",
    orientation="h",
    title="The score of bottom 20 students"
)
```

(continues on next page)

(continued from previous page)

```
fig.update_layout(yaxis_type='category')
fig.show('svg')
```

[26]:
```
s_bottom500_dist = s_bottom500.groupby("CNTRYID").size()
s_bottom500_dist = s_bottom500_dist[s_bottom500_dist > 10]
fig = px.pie(
    names=s_bottom500_dist.index,
    values=s_bottom500_dist,
    title="Bottom 500 students come from",
)
fig.show('svg')
```

[ ]:
```

```

6.8 Task

6.9 EduData.DataSet

6.9.1 EduData.DataSet.download_data

`EduData.DataSet.`**`get_data`**(*dataset*, *data_dir='.'*, *override=False*, *url_dict: Optional[dict] = None*)

Parameters

- **dataset** (*str*) –
- **data_dir** (*str*) –
- **override** (*bool*) –
- **url_dict** –

This script is used to convert the original junyi dataset into json sequence, which can be applied in kt task.

`EduData.DataSet.junyi.KnowledgeTracing.`**`select_n_most_frequent_students`**(*source: str*, *target_prefix: str*, *ku_dict_path: str*, *n: (<class 'int'>, <class 'list'>)*)

None in n means select all students

`EduData.DataSet.EdNet.utils.`**`get_question_id`**(*question_str*)

Examples

```
>>> get_question_id("q123")
123
```

6.10 EduData.Task

EduData.Task.KnowledgeTracing.format.**tl2json**(*src: str*, *tar: str*, *to_int=True*, *left_shift=False*)

convert the dataset in *tl* sequence into *json* sequence

.tl format The first line is the number of exercises a student attempted. The second line is the exercise tag sequence. The third line is the response sequence.

```
15
1,1,1,1,7,7,9,10,10,10,10,11,11,45,54
0,1,1,1,1,1,0,0,1,1,1,1,1,0,0
```

.json format Each sample contains several response elements, and each element is a two-element list. The first is the exercise tag and the second is the response.

```
[[1,0],[1,1],[1,1],[1,1],[7,1],[7,1],[9,0],[10,0],[10,1],[10,1],[10,1],[11,1],[11,
↪1],[45,0],[54,0]]
```

EduData.Task.KnowledgeTracing.graph.**correct_co_influence_graph**(*ku_num*, **src*, *tar=None*, *input_is_file=True*)

Co-influence graph

A co-influence pair is defined as two vertexes that the sum of transition count is large and the difference is small.

Diagonal_value is always 0

Parameters

- **ku_num** –
- **src** –
- **tar** –
- **input_is_file** –

Examples

```
>>> _seq = [
...     [[0, 1], [1, 0], [1, 1], [2, 0]],
...     [[0, 1], [1, 1], [2, 0], [2, 1]],
...     [[2, 1], [2, 1], [1, 1], [2, 0]],
...     [[1, 0], [0, 1], [0, 1], [2, 0]],
...     [[2, 0], [1, 1], [0, 1], [2, 1]],
... ]
>>> correct_co_influence_graph(3, _seq, input_is_file=False)
array([[0., 1., 0.],
       [1., 0., 0.],
       [0., 0., 0.]])
```

EduData.Task.KnowledgeTracing.graph.**correct_transition_count_graph**(*ku_num*, **src*, *tar=None*, *input_is_file=True*)

Parameters

- **ku_num** –
- **src** –
- **tar** –
- **input_is_file** –

Examples

```
>>> _seq = [[[0, 1], [1, 0], [1, 1], [2, 1]], [[2, 0], [1, 0], [0, 1], [2, 1]]]
>>> correct_transition_count_graph(3, _seq, input_is_file=False)
[[0, 0, 1], [0, 0, 1], [0, 0, 0]]
>>> _seq = [[[0, 1], [1, 1], [1, 1], [2, 1]]]
>>> correct_transition_count_graph(3, _seq, input_is_file=False)
[[0, 1, 0], [0, 0, 1], [0, 0, 0]]
```

EduData.Task.KnowledgeTracing.graph.**correct_transition_graph**(*ku_num*, **src*, *tar=None*, *input_is_file=True*, *diagonal_value=0.0*)

When a concept is mastered, how much probability is it to be transferred to another concept.

For example,

` [[0, 1], [1, 0], [1, 1], [2, 1]] [[2, 0], [1, 0], [0, 1], [2, 1]] ` When concept #0 is mastered (i.e., 1st in seq #1, 3rd in seq #2), only concept # 2 can be mastered (4th in seq #2). Thus, the transition probabilty for concept #0 is [0, 0, 1], which mastering concept #0 can influence mastering concept #2 more thant concept #1.

Parameters

- **ku_num** –
- **src** –
- **tar** –
- **input_is_file** –
- **diagonal_value** –

Examples

```
>>> _seq = [[[0, 1], [1, 0], [1, 1], [2, 1]], [[2, 0], [1, 0], [0, 1], [2, 1]]]
>>> correct_transition_graph(3, _seq, input_is_file=False)
[[0.0, 0.0, 1.0], [0.0, 0.0, 1.0], [0.0, 0.0, 0.0]]
>>> _seq = [[[0, 1], [1, 1], [1, 1], [2, 1]]]
>>> correct_transition_graph(3, _seq, input_is_file=False)
[[0.0, 1.0, 0.0], [0.0, 0.0, 1.0], [0.0, 0.0, 0.0]]
```

EduData.Task.KnowledgeTracing.graph.**dense_graph**(*ku_num: int*, *tar=None*, *undirected: bool = False*)

Dense graph where any two vertex have a link

No self loop is reserved.

Parameters

- **ku_num** (*int*) –
- **tar** –
- **undirected** –

Examples

Target file is a json file, json.load can be used to read it.

Demo of target file with undirected tag is False: [

[0, 1], [0, 2], [1, 0], … [2, 0], [2, 1]

]

Demo of target file with undirected tag is True: [

[0, 1], [1, 2], [0, 2]

]

```
>>> dense_graph(3)
[[0, 1], [0, 2], [1, 0], [1, 2], [2, 0], [2, 1]]
>>> dense_graph(3, undirected=True)
[[0, 1], [0, 2], [1, 2]]
```

EduData.Task.KnowledgeTracing.graph.**posterior_correct_probability_graph**(*ku_num*, **src*, *tar=None*, *input_is_file=True*, *fill_na_with=0.0*)

When a concept is mastered, how much probability is another concept correctly answered.

For example,

` [[0, 1], [1, 1], [1, 1], [2, 1]] [[2, 0], [1, 0], [0, 1], [2, 1]] ` When concept #0 is mastered (i.e., 1st in seq #1, 3rd in seq #2), concept #1 and # 2 can both be mastered (1th in seq # 1, 4th in seq #2). Thus, the posterior correct probability for concept #0 is [0, 1, 1].

Parameters

- **ku_num** –
- **src** –
- **tar** –
- **input_is_file** –
- **fill_na_with** –

Returns

- >>> *_seq = [[[0, 1], [1, 0], [1, 1], [2, 1]], [[2, 0], [1, 0], [0, 1], [2, 1]]]*
- >>> *posterior_correct_probability_graph(3, _seq, input_is_file=False)*
- *[[0.0, 1.0, 1.0], [0.0, 0.0, 1.0], [0.0, 0.0, 0.0]]*

`EduData.Task.KnowledgeTracing.graph.posterior_correct_transition_graph`(*ku_num*, **src*, *tar=None*, *input_is_file=True*, *diagonal_value=None*)

Correct transition graph based on posterior correct graph

For example,

` [[0, 1], [1, 1], [1, 1], [2, 1]] [[2, 0], [1, 0], [0, 1], [2, 1]] ` When concept #0 is mastered (i.e., 1st in seq #1, 3rd in seq #2), concept #1 and # 2 can both be mastered (1th in seq # 1, 4th in seq #2). Thus, the posterior correct probability for concept #0 is [0, 1, 1].

Parameters

- **ku_num** –
- **src** –
- **tar** –
- **input_is_file** –
- **diagonal_value** –

Returns

- *>>> _seq = [[[0, 1], [1, 0], [1, 1], [2, 1]], [[2, 0], [1, 0], [0, 1], [2, 1]]]*
- *>>> posterior_correct_transition_graph(3, _seq, input_is_file=False)*
- *[[0.0, 0.5, 0.5], [0.0, 0.0, 1.0], [0.0, 0.0, 0.0]]*
- *>>> _seq = [[[0, 1], [1, 0], [1, 1], [2, 1]], [[2, 0], [1, 0], [0, 1], [2, 1]]]*
- *>>> posterior_correct_transition_graph(3, _seq, input_is_file=False)*
- *[[0.0, 0.5, 0.5], [0.0, 0.0, 1.0], [0.0, 0.0, 0.0]]*

`EduData.Task.KnowledgeTracing.graph.similarity_graph`(*ku_num*, *src_graph*, *tar*)

construct similarity graph based on transition graph

`EduData.Task.KnowledgeTracing.graph.transition_graph`(*ku_num*, **src*, *tar=None*, *input_is_file=True*, *diagonal_value=0.0*)

When a concept is learned, how much probability does another concept appear.

For example,

` [[0, 1], [1, 0], [1, 1], [2, 1]] [[2, 0], [1, 0], [0, 1], [2, 1]] ` When concept #0 is learned (i.e., 1st in seq #1, 3rd in seq #2), concept #2 and #1 could appear (2nd in seq #1, 4th in seq #2) Thus, the transition probabilty for concept #0 is [0, 0.5, 0.5].

Parameters

- **ku_num** –
- **src** –
- **tar** –
- **input_is_file** –
- **diagonal_value** –

Examples

```
>>> _seq = [[[0, 1], [1, 0], [1, 1], [2, 1]], [[2, 0], [1, 0], [0, 1], [2, 1]]]
>>> transition_graph(3, _seq, input_is_file=False)
[[0.0, 0.5, 0.5], [0.5, 0.0, 0.5], [0.0, 1.0, 0.0]]
>>> _seq = [[[0, 1], [1, 1], [1, 1], [2, 1]]]
>>> transition_graph(3, _seq, input_is_file=False)
[[0.0, 1.0, 0.0], [0.0, 0.0, 1.0], [0.0, 0.0, 0.0]]
```

PYTHON MODULE INDEX

e

INDEX