Part I - PISA Data

by Boris Livshits

PISA Dataset contains results of PISA 2012 assesments or students of 65 economies in math, reading and science.

Brief description

There are a lot of materials related to this dataset and results of assesment on OECD website - https://www.oecd-ilibrary.org/education/pisa-2012-assessment-and-analytical-framework_9789264190511-en

Dataset contains full list of answers, estimates and results of assesment. A lot of questions are in form of categorital data (i.e. "Agree"/"Disagree" and so on).

However, the suggested data dictionary doesn't provide sufficient description of dataset variables. More detailed description of questions and possible answers is here - https://read.oecd-ilibrary.org/education/pisa-2012-assessment-and-analytical-framework/background-questionnaires_9789264190511-9-en and here https://www.oecd.org/pisa/pisaproducts/PISA-2012-technical-report-final.pdf

Preliminary Wrangling

What is the structure of your dataset?

As we mentioned earlier - dataset conctains a lot of variables obtained from PISA assesment

What is/are the main feature(s) of interest in your dataset?

We want to look deeper to results in math, reading and science and going to look are deeper - are there any factors that can determine/predict results? There are 50 vartiables with results of test:

- 5 plausible values (PV) in math, reading and science (totally - 15 variables)
- 35 additional PV in subsections of math. I must admire that theory behind PV is rather complicated (it is described here - https://www.oecd-ilibrary.org/docserver/9789264056275-7-en.pdf?expires=1673076434&id=id&accname=guest&checksum=61AEC646C2FA7383532899657DDFAD3D). That's why we are going to focus our attention on PV1 in math, reading and science, respectively.

In [1]:
```
# import all packages and set plots to be embedded inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

%matplotlib inline
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('max_colwidth', 120)
pd.set_option('display.width', 1500)
```

> Load and brief description

In [2]:
```
pisa_raw_data=pd.read_csv('pisa2012.csv',encoding = "cp1252", low_memory=False)
```

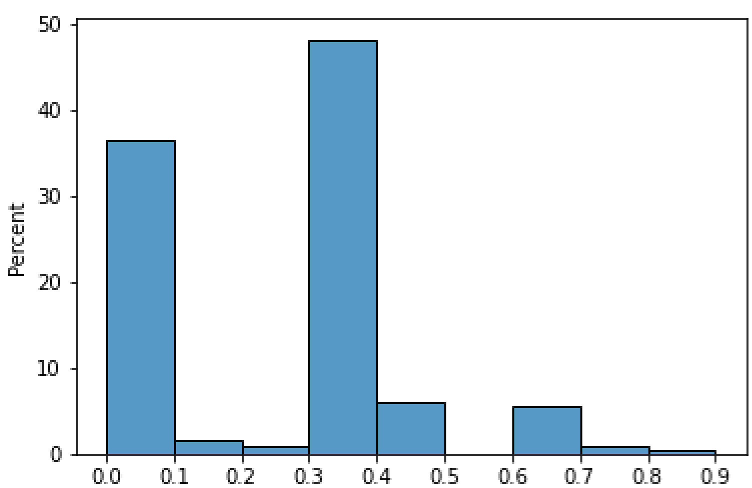
In [3]:
```
#how many rows (observations) and how many columns (properties) do we have?
pisa_raw_data.shape
```

Out[3]:
```
(485490, 636)
```

- #### We have about half million records (485,490 exactly) and 636 columns, i.e. different characteristics of the data. Really huge database.
- Our dataset is a set of answer of students and their parents all over the world - so it is highly likely that we have missing values. Let's check it:)

In [4]:
```
checking_miss_data=pisa_raw_data.isnull().sum()/pisa_raw_data.shape[0]
```

In [5]:
```
bins=np.arange(0,1,0.1)
plt.xticks(ticks=np.arange(0,1,0.1));
yticks=np.arange(0,51,10);
#ylabels = for y in yticks: '()%.'format(y)
plt.yticks(ticks=yticks);
sb.histplot(checking_miss_data, bins=bins, stat='percent');
```



What does it mean?

around half of our data have missing values for 30-40% of variables. In other word, it means that for aproximatly 50% of records answers weren't given for 30-40% of questions. It rather big figure, so we have to take into account while we will provide any insights and conclusions based on data. Most probably - answer were not given for some countries due to possible difference in assesment procedure.

In [6]:
```
#let's look how our data looks
pisa_raw_data.head()
```

Out[6]:

	Unnamed: 0	CNT	SUBNATIO	STRATUM	OECD	NC	SCHOOLID	STIDSTD	ST01Q01	ST02Q01	ST03Q01
0	1	Albania	80000	ALB0006	Non-OECD	Albania	1	1	10	1.0	2

1	2	Albania	80000	ALB0006	Non-OECD	Albania	1	2	10	1.0	2
2	3	Albania	80000	ALB0006	Non-OECD	Albania	1	3	9	1.0	9
3	4	Albania	80000	ALB0006	Non-OECD	Albania	1	4	9	1.0	8
4	5	Albania	80000	ALB0006	Non-OECD	Albania	1	5	9	1.0	10

In [92]:
```
pisa_raw_data.tail()
```

Out[92]:

	Unnamed: 0	CNT	SUBNATIO	STRATUM	OECD	NC	SCHOOLID	STIDSTD	ST01Q01	ST02Q01	ST0
485485	485486	Vietnam	7040000	VNM0317	Non-OECD	Viet Nam	162	4955	10	3.0	
485486	485487	Vietnam	7040000	VNM0317	Non-OECD	Viet Nam	162	4956	10	3.0	
485487	485488	Vietnam	7040000	VNM0317	Non-OECD	Viet Nam	162	4957	10	3.0	
485488	485489	Vietnam	7040000	VNM0317	Non-OECD	Viet Nam	162	4958	10	3.0	

485489	485490	Vietnam	7040000	VNM0317	Non-OECD	Viet Nam	162	4959	10	3.0

In [7]:
```
#database is so huge - it doesn't allow us to look deeper into details.
pisa_raw_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 485490 entries, 0 to 485489
Columns: 636 entries, Unnamed: 0 to VER_STU
dtypes: float64(250), int64(18), object(368)
memory usage: 2.3+ GB
```

What features in the dataset do you think will help support your investigation into your feature(s) of interest?

Our dataset contains a lot of very interesting variables, so it not easy "to cut" our interest. We are going to look how distribited values of different variables which could be interesting for us.

First of all - some parameters which can be correlated with results in math:

> 'ST62Q01' - Familiarity with Math Concepts - Exponential Function, and 'ST62Q19' - Familiarity with Math Concepts - Probability. possible answers - from 'Never heard of it' to 'Know it well, understand the concept'

In [8]:
```
pisa_raw_data['ST62Q01'].value_counts(normalize=True, dropna=False)
```

Out[8]:
```
NaN                                     0.368712
Never heard of it                       0.242479
Heard of it once or twice               0.113829
Heard of it a few times                 0.107650
Heard of it often                       0.087623
Know it well,  understand the concept   0.079707
Name: ST62Q01, dtype: float64
```

In [9]:
```
pisa_raw_data['ST62Q19'].value_counts(normalize=True, dropna=False)
```

Out[9]:
```
NaN                                     0.366148
Know it well,  understand the concept   0.283732
Heard of it often                       0.138013
Heard of it a few times                 0.084360
Never heard of it                       0.068125
Heard of it once or twice               0.059622
Name: ST62Q19, dtype: float64
```

for both variables share of NaN is about 37%, let's check if these NaN occure simultaneously.

In [10]:
```
pisa_raw_data[pisa_raw_data['ST62Q19'].isna()].ST62Q01.value_counts(normalize=True, drop
```

Out[10]:
```
NaN                                     0.986667
Never heard of it                       0.005879
Heard of it once or twice               0.002593
Heard of it a few times                 0.002070
Heard of it often                       0.001513
```

```
Know it well,  understand the concept    0.001277
Name: ST62Q01, dtype: float64
```

So, it looks like if Q01 is NaN then Q19 is NaN too. We should do not forget change datatype for these variables to categorical.

But now let's look for other variables that can be useful in our analysis.

> we want to see at 'ST69Q02' - duration of math lesson, it is open type question, and ST70Q02 - how many math lessons are in week.

In [97]:
```python
pisa_raw_data['ST69Q02'].value_counts(normalize=True, dropna=False)
```

Out[97]:
```
NaN      0.384949
45.0     0.208280
50.0     0.128460
60.0     0.084451
40.0     0.062541
55.0     0.038196
90.0     0.018007
120.0    0.011034
75.0     0.010651
30.0     0.009205
35.0     0.008435
80.0     0.006515
70.0     0.005308
100.0    0.003166
65.0     0.002130
180.0    0.001530
20.0     0.001075
52.0     0.000917
48.0     0.000886
110.0    0.000760
53.0     0.000694
59.0     0.000659
160.0    0.000641
85.0     0.000641
58.0     0.000632
56.0     0.000616
54.0     0.000597
57.0     0.000587
72.0     0.000529
25.0     0.000470
42.0     0.000410
47.0     0.000398
43.0     0.000381
15.0     0.000381
105.0    0.000377
95.0     0.000301
150.0    0.000297
46.0     0.000290
68.0     0.000272
51.0     0.000247
49.0     0.000243
115.0    0.000210
62.0     0.000206
84.0     0.000204
88.0     0.000204
78.0     0.000194
44.0     0.000185
77.0     0.000167
37.0     0.000167
64.0     0.000165
38.0     0.000136
```

130.0	0.000117
82.0	0.000109
135.0	0.000103
63.0	0.000099
83.0	0.000097
66.0	0.000095
140.0	0.000095
73.0	0.000095
76.0	0.000093
86.0	0.000080
41.0	0.000072
36.0	0.000060
79.0	0.000058
74.0	0.000051
39.0	0.000049
67.0	0.000045
96.0	0.000045
69.0	0.000041
125.0	0.000041
61.0	0.000041
87.0	0.000041
71.0	0.000039
118.0	0.000039
81.0	0.000037
33.0	0.000035
32.0	0.000025
89.0	0.000023
175.0	0.000019
24.0	0.000019
145.0	0.000016
104.0	0.000014
34.0	0.000014
170.0	0.000012
28.0	0.000012
27.0	0.000012
108.0	0.000012
93.0	0.000012
16.0	0.000010
23.0	0.000008
92.0	0.000008
165.0	0.000008
17.0	0.000008
106.0	0.000006
18.0	0.000006
91.0	0.000006
111.0	0.000006
103.0	0.000004
102.0	0.000004
94.0	0.000004
29.0	0.000004
116.0	0.000004
31.0	0.000004
156.0	0.000004
21.0	0.000004
22.0	0.000004
19.0	0.000004
128.0	0.000002
114.0	0.000002
155.0	0.000002
138.0	0.000002
127.0	0.000002
172.0	0.000002
152.0	0.000002
139.0	0.000002
112.0	0.000002
117.0	0.000002

```
113.0    0.000002
177.0    0.000002
Name: ST69Q02, dtype: float64
```

In [77]:
```
pisa_raw_data['ST70Q02'].value_counts(normalize=True, dropna=False)
```

Out[77]:
```
NaN     0.385489
4.0     0.191689
5.0     0.157604
3.0     0.111765
6.0     0.054623
2.0     0.042773
7.0     0.019819
8.0     0.013096
1.0     0.008268
10.0    0.004538
9.0     0.004101
0.0     0.003638
12.0    0.000968
11.0    0.000630
14.0    0.000311
16.0    0.000146
13.0    0.000134
15.0    0.000130
20.0    0.000078
30.0    0.000049
18.0    0.000039
40.0    0.000027
17.0    0.000021
25.0    0.000010
34.0    0.000010
35.0    0.000010
22.0    0.000008
23.0    0.000008
19.0    0.000004
24.0    0.000004
31.0    0.000002
21.0    0.000002
32.0    0.000002
33.0    0.000002
Name: ST70Q02, dtype: float64
```

In [78]:
```
pisa_raw_data['ST69Q02'].describe()
```

Out[78]:
```
count    298601.000000
mean         52.911273
std          17.007616
min          15.000000
25%          45.000000
50%          50.000000
75%          55.000000
max         180.000000
Name: ST69Q02, dtype: float64
```

In [79]:
```
pisa_raw_data['ST70Q02'].describe()
```

Out[79]:
```
count    298339.000000
mean          4.350246
std           1.652565
min           0.000000
25%           3.000000
50%           4.000000
75%           5.000000
max          40.000000
Name: ST70Q02, dtype: float64
```

once again - about 38% of answers for both variables are missed, but possible it corresponds to missing data about others math related questions. However, it seems that here we have some "incorrect" data. For example, zero lessons of math per week sounds unrealistic. For purpore of our analysis, it'll be useful to create new variable - math learning time per week.

In [12]:
```
#create new variable 'math_time'
pisa_raw_data['math_time']=pisa_raw_data['ST69Q02']*pisa_raw_data['ST70Q02']
pisa_raw_data['math_time'].describe()
```

Out[12]:
```
count    283303.000000
mean        226.007056
std          97.448421
min           0.000000
25%         180.000000
50%         220.000000
75%         250.000000
max        3000.000000
Name: math_time, dtype: float64
```

Also it may be useful to check:

> 'ST26Q06' - Possessions - Internet with possible answers Yes or No

In [99]:
```
pisa_raw_data['ST26Q06'].value_counts(normalize=True, dropna=False)
```

Out[99]:
```
Yes    0.828112
No     0.146536
NaN    0.025352
Name: ST26Q06, dtype: float64
```

here we have relatively small number of NaN. However, most of students have internet connection at home, so it seems that just as simple dummy it doesn't determine something.

> Let's look on 'IC04Q01' - How old you are when you first accessed the Internet? with possible answer - '6 years old or younger', '7-9 years old', '10-12 years old', '13 years or older', 'I have never accessed the Internet'

In [24]:
```
pisa_raw_data['IC04Q01'].value_counts(normalize=True, dropna=False)
```

Out[24]:
```
NaN                         0.389678
7-9 years old               0.233964
10-12 years old             0.217976
13 years old  or older      0.078539
6 years old or younger      0.075728
Never                       0.004115
Name: IC04Q01, dtype: float64
```

> one more variable of possible interest is 'ST28Q01' - 'How many books at home' with possible answers: 0-10 books, 11-25, 26-100, 101-200, 201-500, more than 500.

In [30]:
```
pisa_raw_data['ST28Q01'].value_counts(normalize=True, dropna=False)
```

Out[30]:
```
26-100 books           0.278449
11-25 books            0.200488
0-10 books             0.195765
101-200 books          0.140786
201-500 books          0.101479
More than 500 books    0.058883
```

```
NaN                        0.024151
Name: ST28Q01, dtype: float64
```

so, we are going to look how following veriables:

- ST62Q01 (knowing the concept of exponential function) - *categorical*
- math_time (minutes of math lessons in school per week) - *continuous*
- IC04Q01 (age at first time access the Internet) - *categorical*
- ST28Q01 (how many books at home) - *categorical*

can tell us something about possible test results -

- PV1MATH
- PV1READ
- PV1SCIE

In [13]:
```
#let's "cut" our dataset only to variables of out interest
pisa_data=pisa_raw_data[['PV1MATH','PV1READ','PV1SCIE','ST62Q01','math_time','IC04Q01','
```

We are going to convert categorical data into categories and order it - from worst to best. So, for example, we assune that earlier access to the Internet is better.

In [14]:
```
#before converting data to categorical datatupe, we have to do some preliminary work (I
pisa_data.loc[:,'ST28Q01']=pisa_data['ST28Q01'].str.rstrip();
```

```
C:\Users\Livshits\AppData\Local\Temp\ipykernel_4692\2255024264.py:2: SettingWithCopyWarn
ing:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  pisa_data.loc[:,'ST28Q01']=pisa_data['ST28Q01'].str.rstrip();
```

In [15]:
```
# convert categorical  into ordered categorical types
ordinal_var_dict = {
    "ST62Q01": ["Never heard of it", "Heard of it once or twice", "Heard of it a few tim
    "IC04Q01": ["Never","13 years old  or older", "10-12 years old", "7-9 years old", "6
    "ST28Q01": ["0-10 books", "11-25 books", "26-100 books", "101-200 books", "201-500 b
}

# set and order categories

for col in ordinal_var_dict:
    pisa_data[col]=pisa_data[col].astype("category")
    pisa_data[col]=pisa_data[col].cat.set_categories(ordinal_var_dict[col], ordered=True
```

```
C:\Users\Livshits\AppData\Local\Temp\ipykernel_4692\3632817947.py:11: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  pisa_data[col]=pisa_data[col].astype("category")
C:\Users\Livshits\AppData\Local\Temp\ipykernel_4692\3632817947.py:12: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  pisa_data[col]=pisa_data[col].cat.set_categories(ordinal_var_dict[col], ordered=True)
C:\Users\Livshits\AppData\Local\Temp\ipykernel_4692\3632817947.py:11: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  pisa_data[col]=pisa_data[col].astype("category")
C:\Users\Livshits\AppData\Local\Temp\ipykernel_4692\3632817947.py:12: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  pisa_data[col]=pisa_data[col].cat.set_categories(ordinal_var_dict[col], ordered=True)
C:\Users\Livshits\AppData\Local\Temp\ipykernel_4692\3632817947.py:11: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  pisa_data[col]=pisa_data[col].astype("category")
C:\Users\Livshits\AppData\Local\Temp\ipykernel_4692\3632817947.py:12: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  pisa_data[col]=pisa_data[col].cat.set_categories(ordinal_var_dict[col], ordered=True)
```

In [16]:

```
for col in ordinal_var_dict:
    display(pisa_data[col].cat.categories)
```

```
Index(['Never heard of it', 'Heard of it once or twice', 'Heard of it a few times', 'Hea
rd of it often', 'Know it well,  understand the concept'], dtype='object')
Index(['Never', '13 years old  or older', '10-12 years old', '7-9 years old', '6 years o
ld or younger'], dtype='object')
Index(['0-10 books', '11-25 books', '26-100 books', '101-200 books', '201-500 books', 'M
ore than 500 books'], dtype='object')
```

In [17]:

```
mean_score = np.round(pisa_data[['PV1MATH','PV1READ','PV1SCIE']].mean(), 2)
mean_score
```

Out[17]:

```
PV1MATH    469.62
PV1READ    472.00
PV1SCIE    475.77
dtype: float64
```

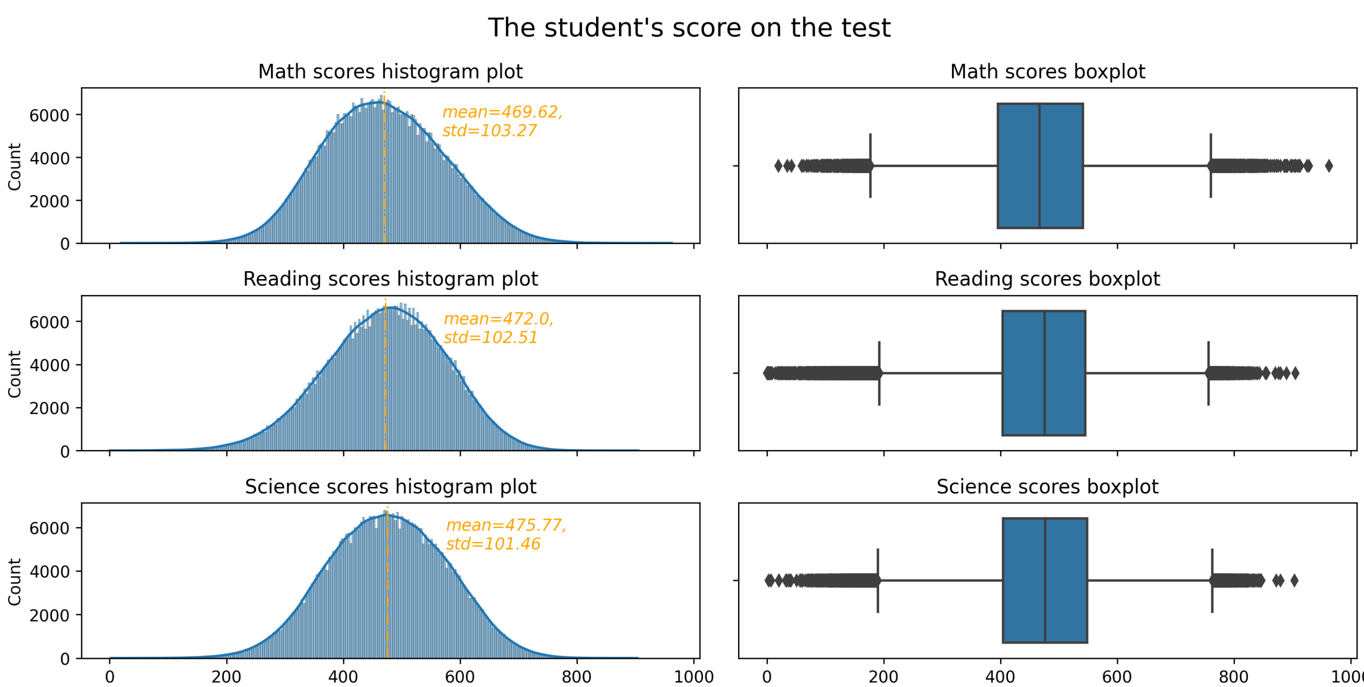
Univariate Exploration

> In this section, investigate distributions of individual variables. If you see unusual points or outliers, take a deeper look to clean things up and prepare yourself to look at relationships between variables.

In [18]:

```
# creating 6 subplots
fig, axes = plt.subplots(3, 2, figsize=[12, 6], dpi=250, sharex=True)
#fig, axes = plt.subplots(3, 2, figsize=[12, 6], dpi=250)
```

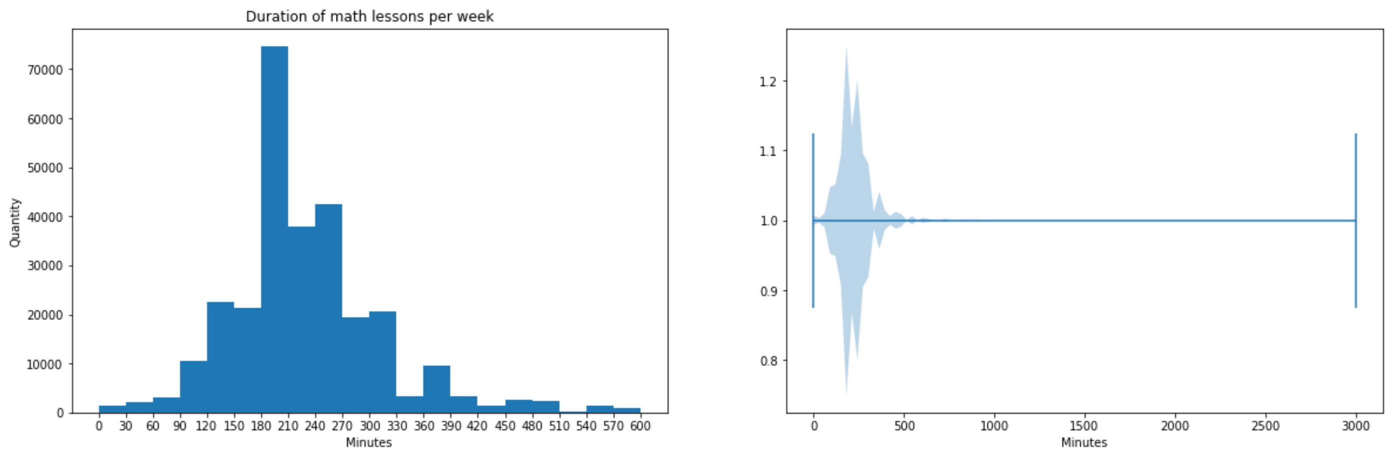
```python
# main title:
fig.suptitle("The student's score on the test", fontsize=16)

# the plots:
sb.histplot(data=pisa_data, x="PV1MATH", ax=axes[0][0], kde=True)
sb.boxplot(data=pisa_data, x="PV1MATH", ax=axes[0][1])
sb.histplot(data=pisa_data, x="PV1READ", ax=axes[1][0], kde=True)
sb.boxplot(data=pisa_data, x="PV1READ", ax=axes[1][1])
sb.histplot(data=pisa_data, x="PV1SCIE", ax=axes[2][0], kde=True)
sb.boxplot(data=pisa_data, x="PV1SCIE", ax=axes[2][1])


mean_score = np.round(pisa_data[['PV1MATH','PV1READ','PV1SCIE']].mean(), 2)
std_score = np.round(pisa_data[['PV1MATH','PV1READ','PV1SCIE']].std(), 2)
# histogram customization:
axes[0][0].axvline(mean_score['PV1MATH'], color="orange", linestyle="-.", linewidth=1)
axes[0][0].text(x=mean_score['PV1MATH'] + 100, y=5000, s=f"mean={mean_score['PV1MATH']},
                , style="italic", color="orange")
axes[1][0].axvline(mean_score['PV1READ'], color="orange", linestyle="-.", linewidth=1)
axes[1][0].text(x=mean_score['PV1READ'] + 100, y=5000, s=f"mean={mean_score['PV1READ']},
                , style="italic", color="orange")
axes[2][0].axvline(mean_score['PV1SCIE'], color="orange", linestyle="-.", linewidth=1)
axes[2][0].text(x=mean_score['PV1SCIE'] + 100, y=5000, s=f"mean={mean_score['PV1SCIE']},
                , style="italic", color="orange")

# titles and labels
axes[0][0].set_title("Math scores histogram plot")
axes[0][1].set_xlabel("")
axes[0][1].set_title("Math scores boxplot")
axes[1][0].set_title("Reading scores histogram plot")
axes[1][1].set_xlabel("")
axes[1][1].set_title("Reading scores boxplot")
axes[2][0].set_title("Science scores histogram plot")
axes[2][0].set_xlabel("")
axes[2][1].set_xlabel("")
axes[2][1].set_title("Science scores boxplot")
plt.tight_layout()
plt.savefig("score.png")
```



What we can say about test scores?

- the first and, probably, most important - all three scores is normally distributed
- reading scores looks slightly left-skewed, while math scores - slightly right skewed.

In [19]:
```
fig, axes = plt.subplots(1,2, figsize=[20, 6])
bins=np.arange(0,601,30)
axes[0].set_xticks(ticks=np.arange(0,601,30));
axes[0].set_title('Duration of math lessons per week');
axes[0].set_xlabel('Minutes');
axes[0].set_ylabel('Quantity');
axes[0].hist(data=pisa_data, x='math_time',bins=bins);
#violinplot and boxplot in pyplot don't work correcty with data with NaN, so we have to
axes[1].violinplot(pisa_data[pisa_data['math_time'].isna()==False].math_time, vert=False
axes[1].set_xlabel('Minutes');
```



It's clear that some data should be considered as "low-quality" data, typical working week is about 40 hours, that's whay highly unlikely to have about 50 hours of math per week! Seem's reasonable to remove all data with math_time more than 10 hours (600 minutes) per week. On the other hand, minimal duration of lesson is 15 minutes, so it's reasonable to suppose that overall duration couldn't be less than 30 minutes per week.

In [20]:
```
drop_indexes=pisa_data[(pisa_data['math_time']>600)|(pisa_data['math_time']<30)].index
((pisa_data['math_time']>600)|(pisa_data['math_time']<30)).sum()
((pisa_data['math_time']>600)|(pisa_data['math_time']<30)).sum()/pisa_data.shape[0]
```

Out[20]:
```
0.006924962409112443
```

In [21]:
```
pisa_data.drop(drop_indexes,inplace=True)
```

```
C:\Users\Livshits\AppData\Local\Temp\ipykernel_4692\1480749985.py:1: SettingWithCopyWarn
ing:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  pisa_data.drop(drop_indexes,inplace=True)
```

As we mentioned earlier - for several categorical variables we have about 35%-40% of missing data, let's have a look - if we drop such record will it change the distribution of our "variables of interest"

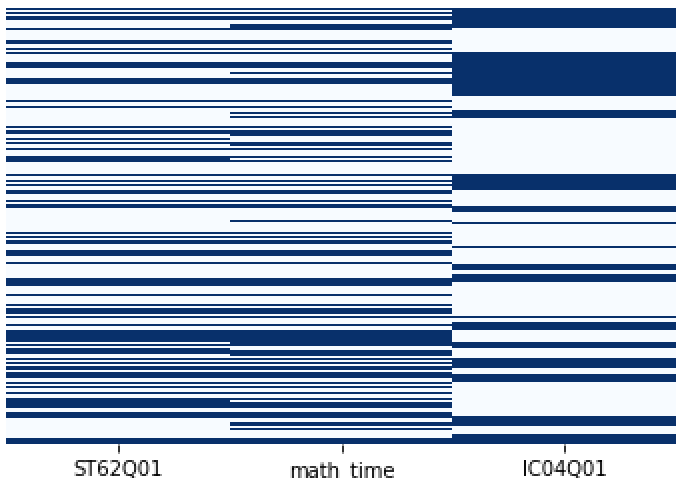
In [22]:
```
pisa_data.isnull().sum()/pisa_data.shape[0]
```

Out[22]:
```
PV1MATH      0.000000
PV1READ      0.000000
PV1SCIE      0.000000
ST62Q01      0.371070
math_time    0.419364
IC04Q01      0.388855
```

```
ST28Q01      0.024201
dtype: float64
```

In [23]:
```
sb.heatmap(pisa_data[['ST62Q01','math_time','IC04Q01']].isnull(),cmap="Blues",yticklabel
missed_data=pisa_data['ST62Q01'].isnull()|pisa_data['math_time'].isnull()|pisa_data['IC0
missed_data.describe()
```

Out[23]:
```
count      482128
unique          2
top          True
freq       306677
dtype: object
```



It's a pity, but NaN for IC04Q01 (age at first time access the Internet) often has NaN when math-related questions aren't missed. So, we are not going to drop all records with NaN from our dataset.

Discuss the distribution(s) of your variable(s) of interest. Were there any unusual points? Did you need to perform any transformations?

- Frankly speaking, test results doesn't have any unusual points - all of them is normally distributed. So we don't need to do any transformation.

Of the features you investigated, were there any unusual distributions? Did you perform any operations on the data to tidy, adjust, or change the form of the data? If so, why did you do this?

Yes, we changed data type to categorical - for categorical variables:

- ST62Q01 (knowing the concept of exponential function)
- IC04Q01 (age at first time access the Internet)
- ST28Q01 (how many books at home)

Secondly, we construct a new variable - overall time for math per week. It is a product of two variables - ST69Q02 and ST70Q02 (duration of math lesson and how many math lessons per week students have). Moreover, we drop all records with total duration more than 10 hours and less than 30 minutes - total 3362 records (about 0.7%).

Bivariate Exploration

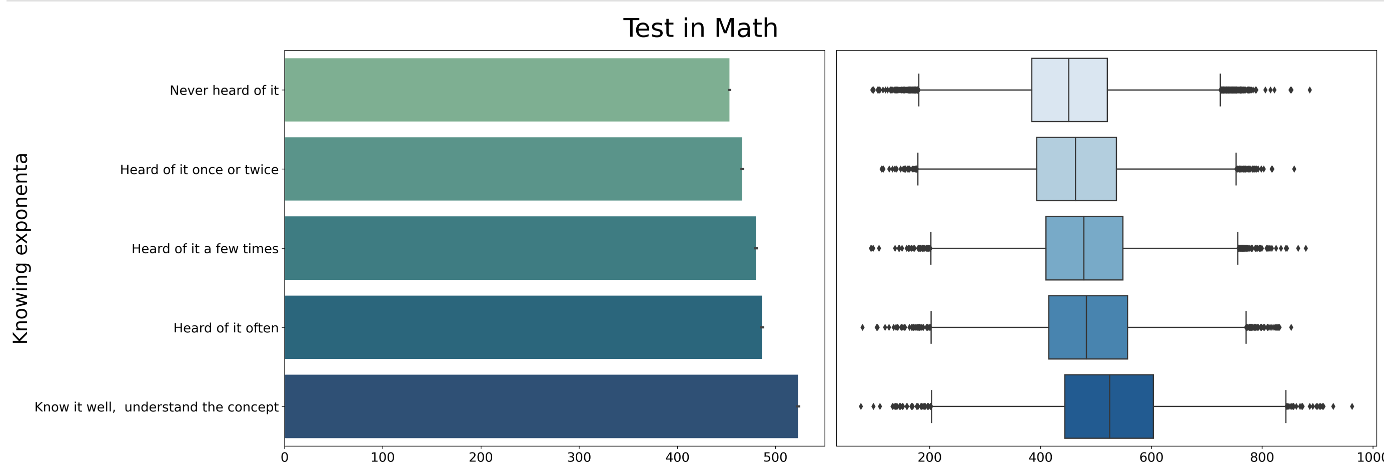
> In this section, investigate relationships between pairs of variables in your data. Make sure the variables that you cover here have been introduced in some fashion in the previous section

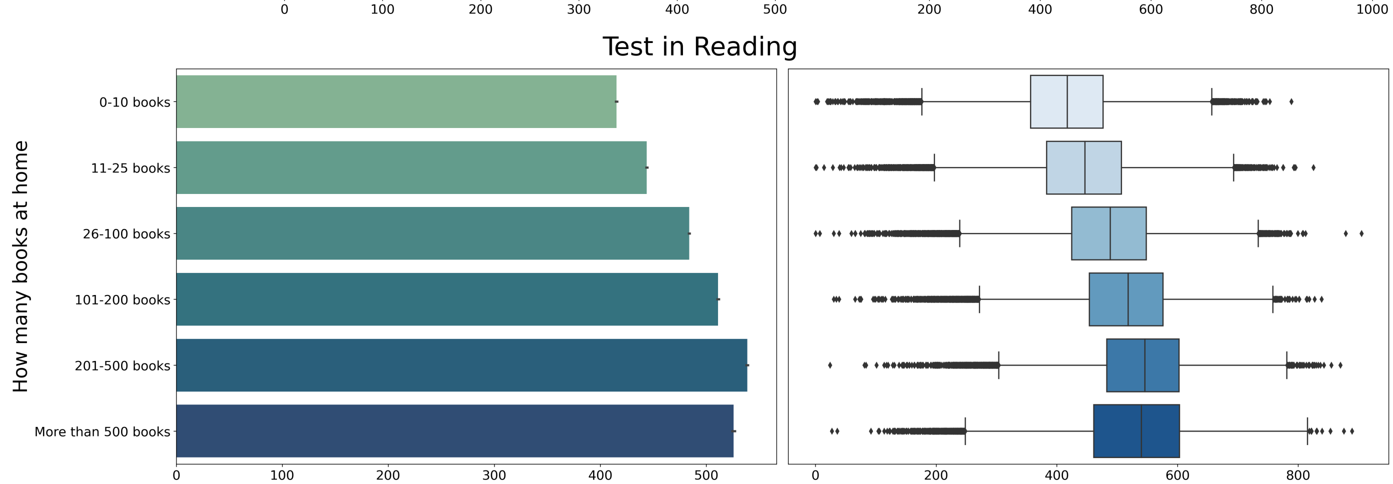
(univariate exploration).

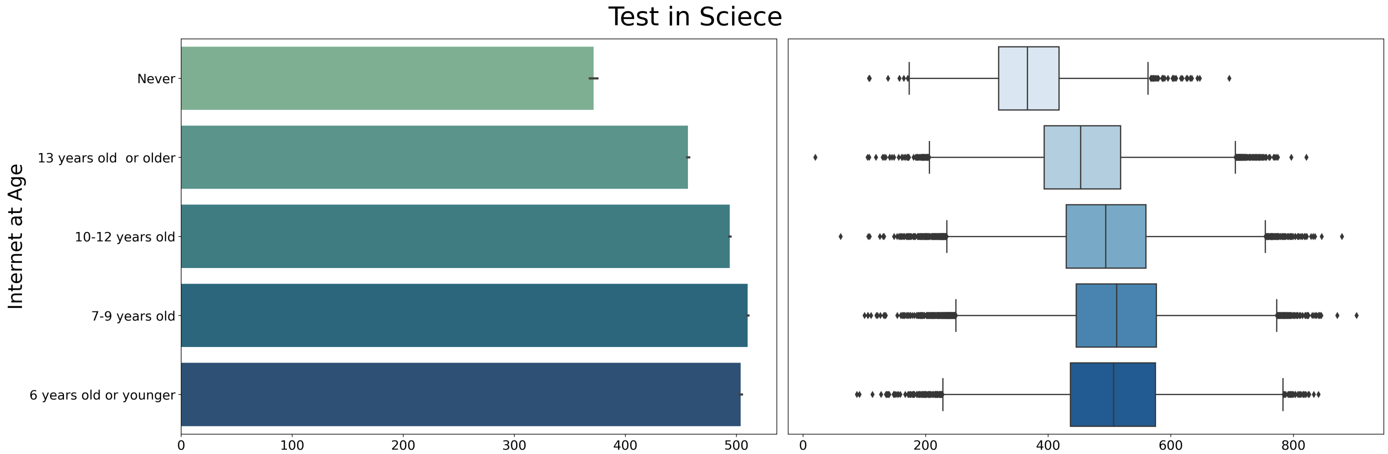
Categorical variables

In [24]:
```python
categ_dict={
    'ST62Q01':'PV1MATH',
    'ST28Q01':'PV1READ',
    'IC04Q01':'PV1SCIE'
}
labels={
    'ST62Q01':['Test in Math','Knowing exponenta'],
    'ST28Q01':['Test in Reading','How many books at home'],
    'IC04Q01':['Test in Science','Internet at Age']
}
```

In [25]:
```python
for key, value in categ_dict.items():
    fig, axes = plt.subplots(1, 2, figsize=[24, 8], dpi=250)
    fig.suptitle(labels[key][0], fontsize=32)
    sb.barplot(data=pisa_data, y=key, x=value, ax=axes[0], palette="crest")
    sb.boxplot(data=pisa_data, x=value, y=key, ax=axes[1], palette="Blues")
    axes[0].tick_params(axis='both', which='both', labelsize=16)
    axes[1].tick_params(axis='both', which='both', labelsize=16)
    axes[0].set_ylabel(labels[key][1], fontsize=24)
    axes[0].set_xlabel("")
    axes[1].set_xlabel("")
    axes[1].set_ylabel("")
    axes[1].set_yticks([])
    plt.tight_layout()


plt.savefig("bivar2.png")
```







A couple of comments:

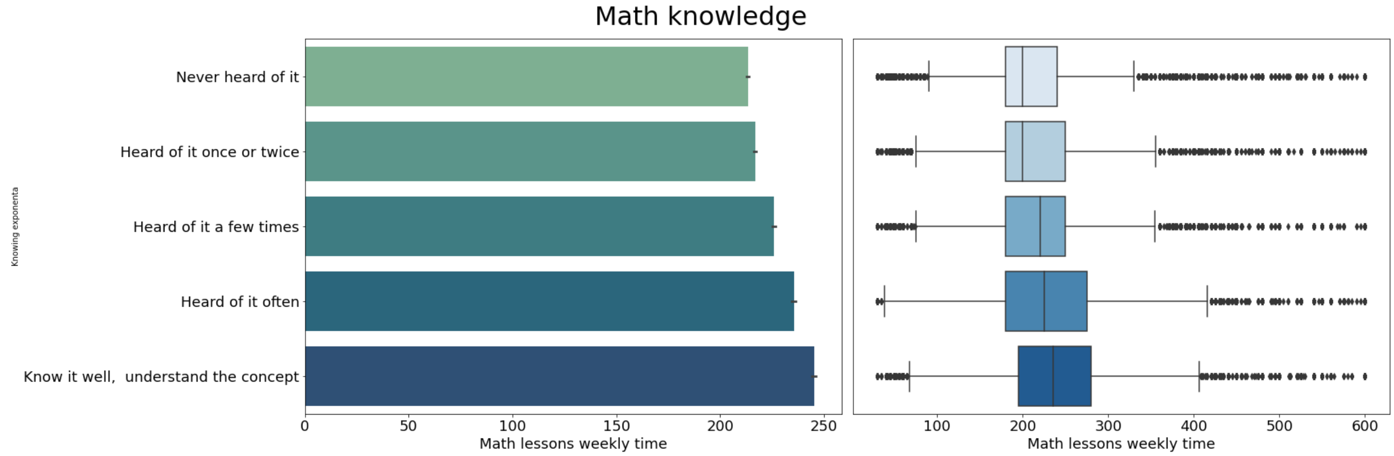
- results of math test and knowlegde of exponentional function don't look surprisingly - better knowledge corresponds with better results.
- in reading test we have something unexpected - group with 200-500 books at home show on average higher results than the group with 500+ books at home. There can be several expanations:

  > - first of all - it maybe difficult to estimate how many books you have, so, maybe some students, didn't provide accurate answers.
  > - secondly, quantity doesn't mean quality - if you have books at home it doesn't necessary mean that you are reading them.
  > - probable some kind of "threshold" maybe 100 books at home.

- in science we see that student without access to Internet show suffiently lower results in test. However, the difference between early age access and access in early tineage seems to be relatively small.

Now, let's look to math time per week and test results in math

In [26]:
```
fig, axes = plt.subplots(1, 2, figsize=[24, 8]);

fig.suptitle("Math knowledge", fontsize=32);
sb.barplot(data=pisa_data, y='ST62Q01', x='math_time', ax=axes[0], palette="crest");
sb.boxplot(data=pisa_data, x='math_time', y='ST62Q01', ax=axes[1], palette="Blues");
axes[0].tick_params(axis='both', which='both', labelsize=18);
axes[1].tick_params(axis='both', which='both', labelsize=18);
axes[0].set_ylabel(labels['ST62Q01'][1]);
axes[0].set_xlabel('Math lessons weekly time', fontsize=18);
axes[1].set_xlabel('Math lessons weekly time', fontsize=18);
axes[1].set_ylabel("");
axes[1].set_yticks([]);
plt.tight_layout();
plt.savefig("bivar3.png");
```



Most interesting is boxplot - it's clear that top "math timers" are in group who never heard about exponenta. It looks strange - you spend a lot of time for math lesson but doesn't know one of the most common and basic function.

In [496…

```
#Let's have a look on some stats of math time for our groups
def quantile99(ins):
    return ins.quantile(q=0.99)

pisa_data.groupby(by='ST62Q01').math_time.agg([pd.Series.max,quantile99])
```

Out[496]:

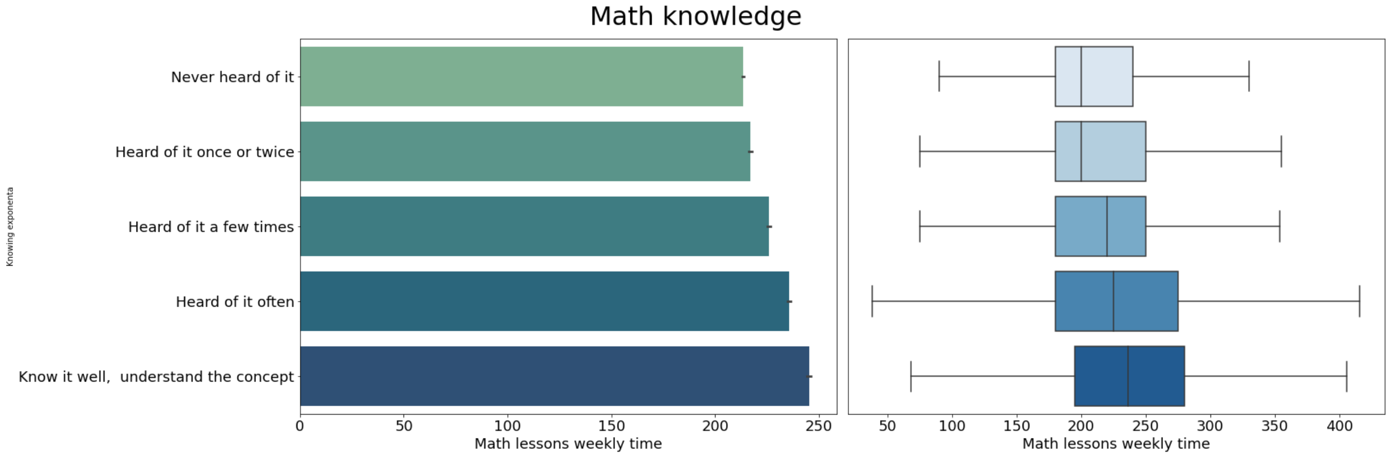
ST62Q01	max	quantile99
Never heard of it	2400.0	540.0
Heard of it once or twice	2220.0	571.1
Heard of it a few times	1600.0	600.0
Heard of it often	1800.0	600.0
Know it well, understand the concept	1800.0	600.0

It is clear that groups "Never heard of it" and "Heard of it once or twice" have more great outliers than the others.

So, let's look on boxplot without outliers

In [27]:

```
fig, axes = plt.subplots(1, 2, figsize=[24, 8]);

fig.suptitle("Math knowledge", fontsize=32);
sb.barplot(data=pisa_data, y='ST62Q01', x='math_time', ax=axes[0], palette="crest");
sb.boxplot(data=pisa_data, x='math_time', y='ST62Q01', ax=axes[1], palette="Blues",showf
axes[0].tick_params(axis='both', which='both', labelsize=18);
axes[1].tick_params(axis='both', which='both', labelsize=18);
axes[0].set_ylabel(labels['ST62Q01'][1]);
axes[0].set_xlabel('Math lessons weekly time', fontsize=18);
axes[1].set_xlabel('Math lessons weekly time', fontsize=18);
axes[1].set_ylabel("");
axes[1].set_yticks([]);
plt.tight_layout();
plt.savefig("bivar4.png");
```



Without outliers we have more "expected" picture - the more you learn, than more you know. One more interesting point - it seems that math lessons weekly time distribution seems to be right-skewed (IQC is not symmetrical aroud the mean). Let's get some metrics.

In [28]:
```
pisa_data.groupby(by='ST62Q01').math_time.agg([pd.Series.mean,pd.Series.mode])
```

Out[28]:

ST62Q01	mean	mode
Never heard of it	213.392967	180.0
Heard of it once or twice	216.898075	180.0
Heard of it a few times	225.814187	180.0
Heard of it often	235.549630	180.0
Know it well, understand the concept	245.243292	180.0

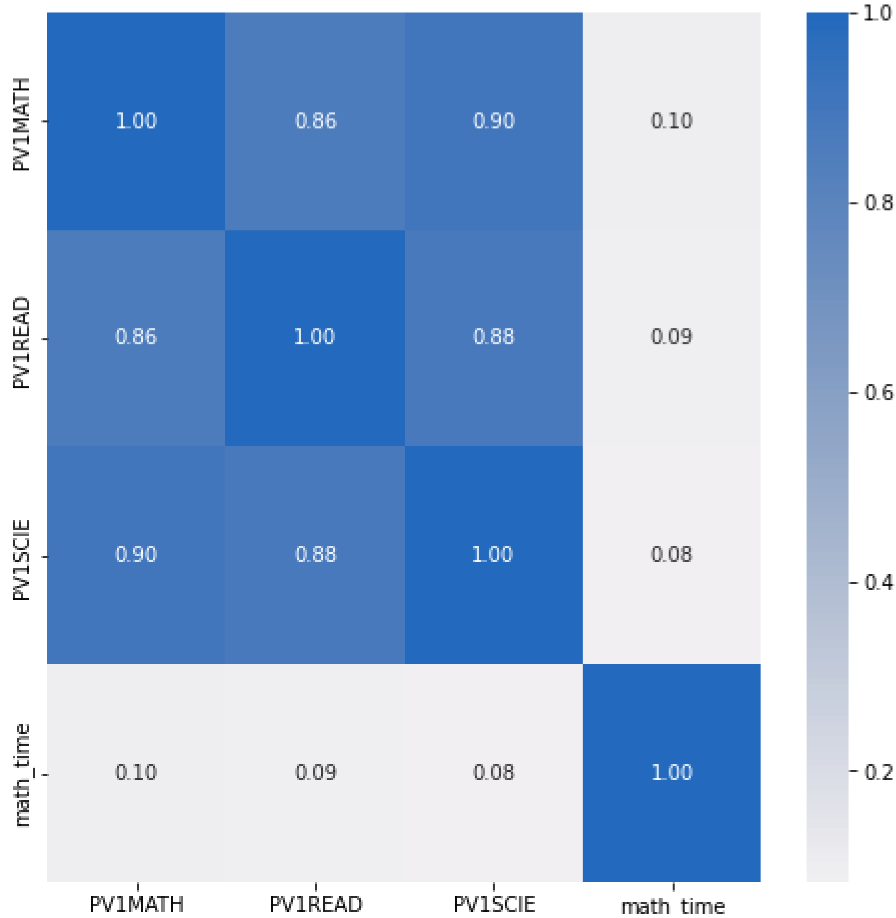
We see that while the mode is the same for all groups, the mean is going bigger as growth in students's awareness of exponenta.

Numerical variables

In [29]:
```
numeric_vars=['PV1MATH','PV1READ','PV1SCIE','math_time']
```

First of all, let's have a look on correlation table of our numerical variables.

In [107…
```
plt.figure(figsize=[8, 8])
sb.heatmap(pisa_data[numeric_vars].corr(), annot=True, fmt=".2f", cmap="vlag_r", center=
```

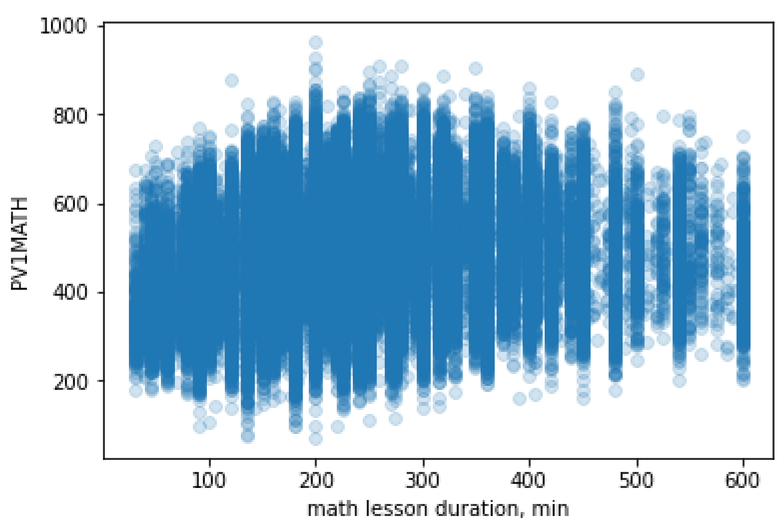


Two main conclusions:

- test results in different fields are correlated with each other
- surprisingly, math_time doesn't correlated even to results in math.

In [114…:
```
#let's look on scatter plot for math lessons duration and test results in Math
plt.scatter(data=pisa_data,y='PV1MATH',x='math_time', alpha=0.2)
plt.xlabel('math lesson duration, min')
plt.ylabel('PV1MATH')
```

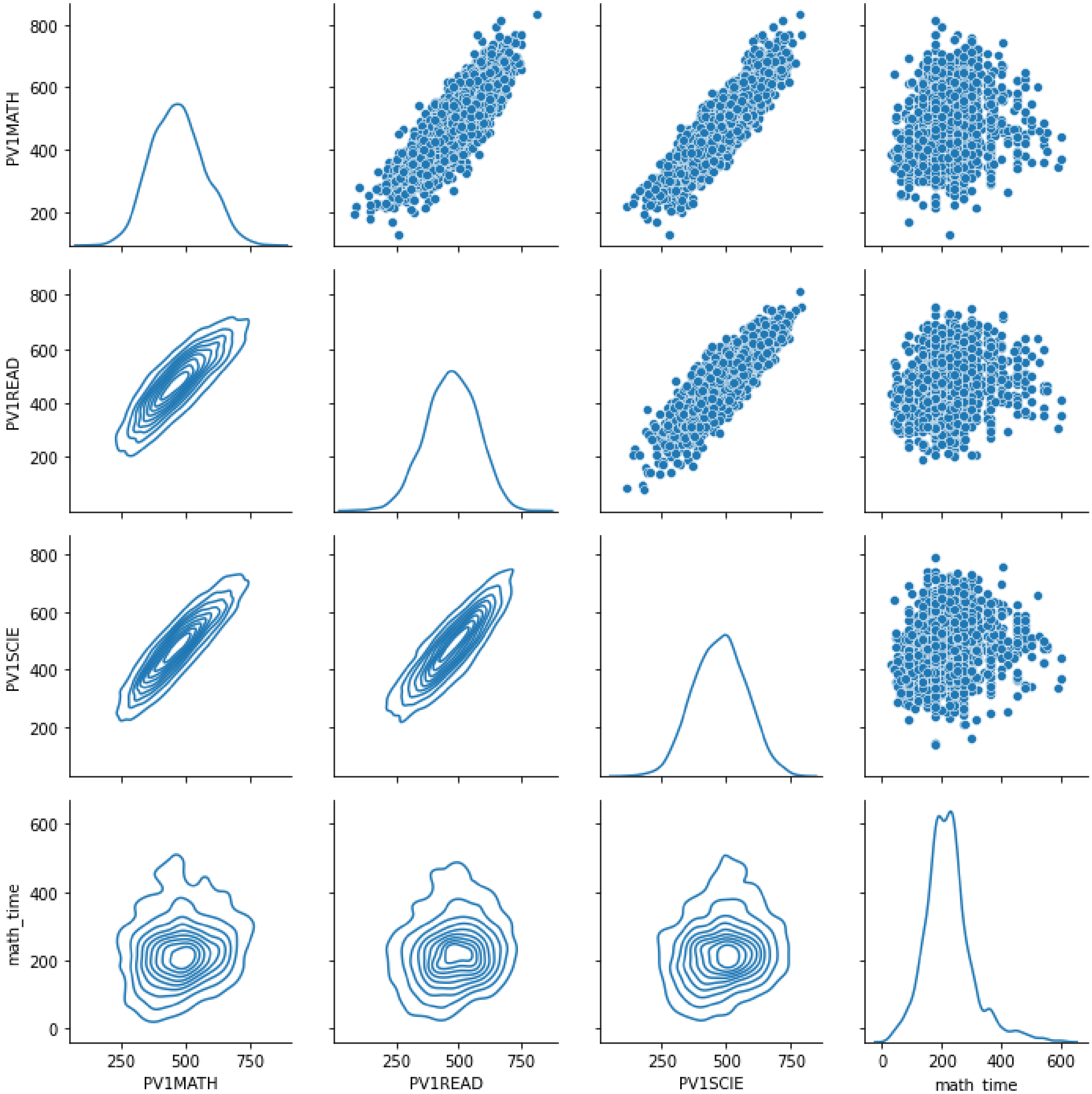
Out[114]:
```
Text(0, 0.5, 'PV1MATH')
```



In [31]:
```
# plot matrix: sample 3000 students so that plots are clearer and they render faster
print("total number of observations", pisa_data.shape)
data_samp = pisa_data.sample(n=3000, replace=False)
print("sample size", data_samp.shape)
```

```
g = sb.PairGrid(data=data_samp, vars=numeric_vars)
g.map_diag(sb.kdeplot)
g.map_upper(sb.scatterplot)
g.map_lower(sb.kdeplot);
```

```
total number of observations (482128, 7)
sample size (3000, 7)
```



Talk about some of the relationships you observed in this part of the investigation. How did the feature(s) of interest vary with other features in the dataset?

- as expected - higher math results are associated wuth deeper knowledge of exponenta function and with longer time of learning math (on average).
- results in three test are correlated with each other.
- we found that test results in science is significantly worse for whose students, who never have access to Internet or get access in the age 13+ (it means less than 2 years expirience in Internet - PISA is designed for 15-years old students).We are going to look deeper at this variable in next session

Did you observe any interesting relationships between the other features (not the main feature(s) of interest)?

- however, quantity of books at home behaves not so straightforward - according to test results in reading posessing more books doesn't lead you to higher results.
- interestingly - top valued otliyers in terms of duration of math lesson per week are in group "doesn't know exponenta". Frankly speaking, I think it is, first of all, question of

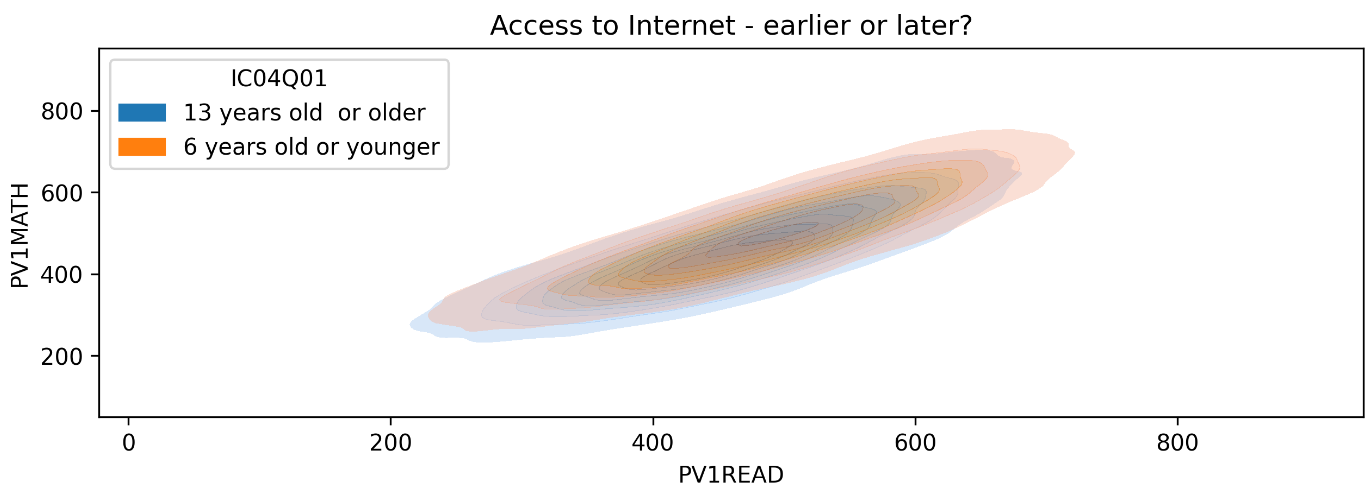
Multivariate Exploration

> Create plots of three or more variables to investigate your data even further. Make sure that your investigations are justified, and follow from your work in the previous sections.

We saw that results in test in Science corresponds with the age access to the Internet. let's compare result in Math and Reading in of two groups of studens - who get access to Internet at age 13+ and who at 6 or earlier. Does the Internet at the childhood provide some value for student?

In [72]:
```
pisa_data_13=pisa_data[(pisa_data['IC04Q01']=='13 years old  or older')]
pisa_data_6=pisa_data[(pisa_data['IC04Q01']=='6 years old or younger')]
df_6_13 = pd.concat([pisa_data_13, pisa_data_6], ignore_index=True)
df_6_13['IC04Q01'] = df_6_13['IC04Q01'].cat.remove_unused_categories()
```

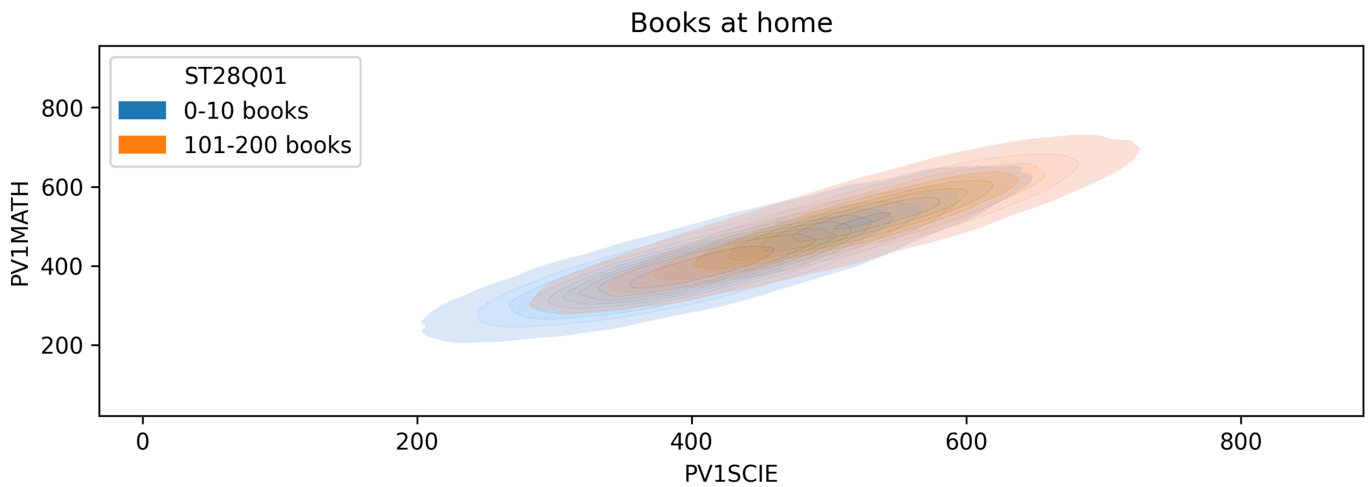
In [73]:
```
fig, ax = plt.subplots(1, 1, figsize=[10, 3], dpi=300)
sb.kdeplot(
    y="PV1MATH", x="PV1READ", hue='IC04Q01', data=df_6_13, alpha=0.4, fill=True, ax=ax
)
ax.set_title("Access to Internet - earlier or later?")
#ax.legend(title='Age of access to the Internet', labels=['6 years or younger','13+years
sb.move_legend(ax,loc='upper left')
plt.savefig("multivar.png")
```



We see that top results in math and reading (more than 600 points) are more likely to students who are internet users from their baby time. Now, let's do the same with 'ST28Q01' - quantity of books at home, we are going to look deeper on two catefories: '0-10 books' and '101-200 books'

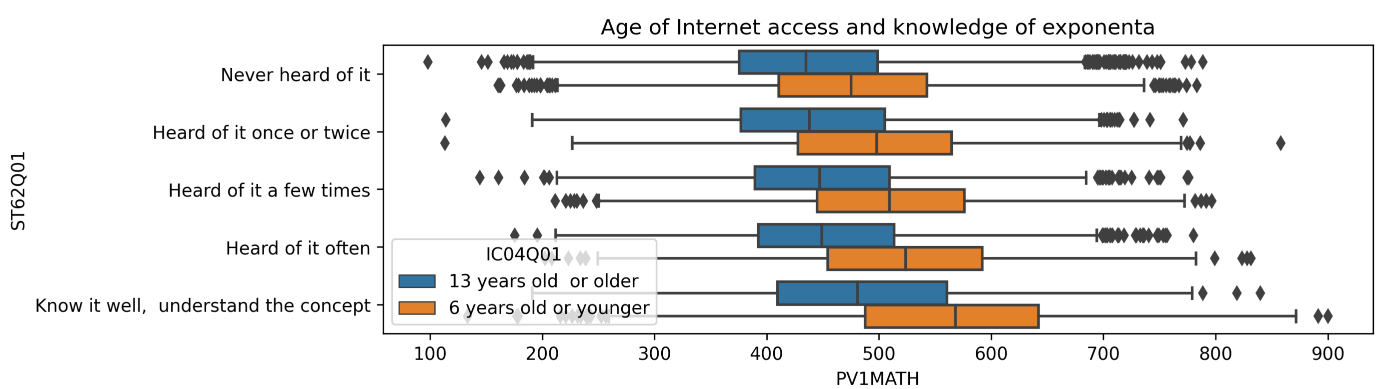
In [76]:
```
pisa_data_books_low=pisa_data[(pisa_data['ST28Q01']=='0-10 books')]
pisa_data_books_many=pisa_data[(pisa_data['ST28Q01']=='101-200 books')]
pisa_data_books = pd.concat([pisa_data_books_low, pisa_data_books_many], ignore_index=Tr
pisa_data_books['ST28Q01'] = pisa_data_books['ST28Q01'].cat.remove_unused_categories()
```

In [77]:
```
fig, ax = plt.subplots(1, 1, figsize=[10, 3], dpi=300)
sb.kdeplot(
    y="PV1MATH", x="PV1SCIE", hue='ST28Q01', data=pisa_data_books, alpha=0.4, fill=True,
)
ax.set_title("Books at home")
sb.move_legend(ax,loc='upper left')
plt.savefig("multivar2.png")
```



It seems clear - that students with 100+ books at home - outperform students with a few books at home. Now let's look what can "age of access" and "knowledge of exponenta" tell together about math results.

In [81]:
```
fig, ax = plt.subplots(1, 1, figsize=[10, 3], dpi=300)
sb.boxplot(data=df_6_13, y="ST62Q01", x="PV1MATH", hue="IC04Q01", ax=ax)
sb.move_legend(ax,loc='lower left')

ax.set_title("Age of Internet access and knowledge of exponenta");
```

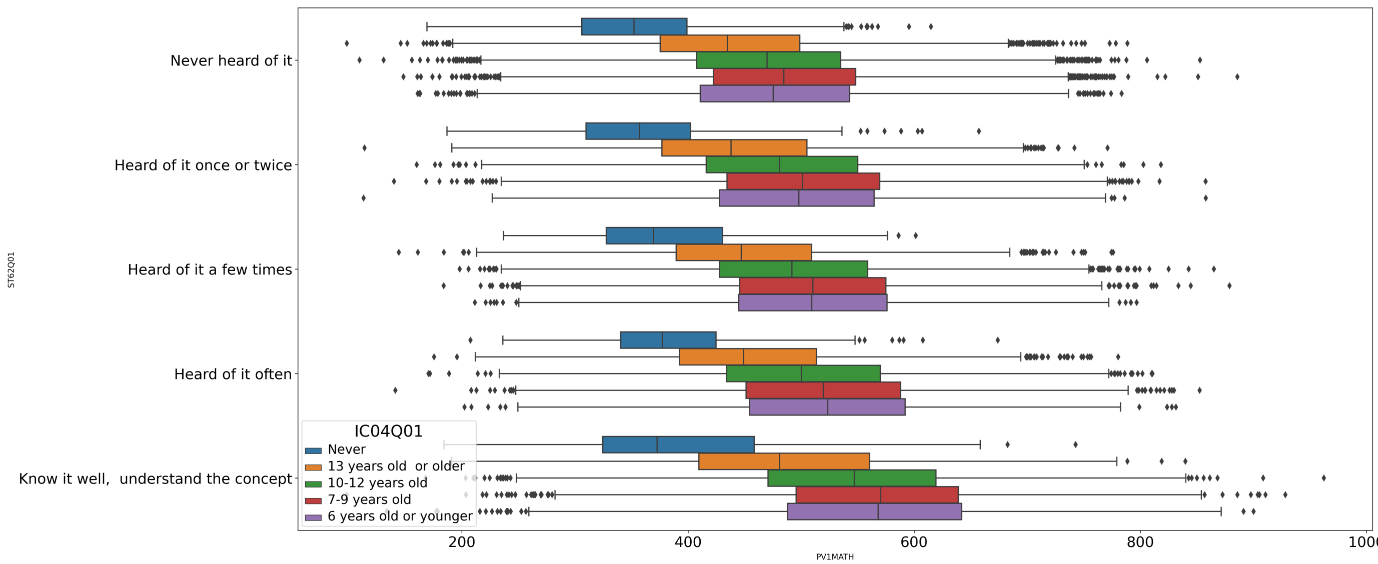


It is very intersting - it seems that "age of access" is more important in terms of math results than knowledge of exponenta. With the same knowledge - students with early access demonstrate significantly hidger results. However, let's check is there such relationship for all categories of "age of access".

In [102…
```
fig, ax = plt.subplots(1, 1, figsize=[24, 12], dpi=300)
sb.boxplot(data=pisa_data, y="ST62Q01", x="PV1MATH", hue="IC04Q01", ax=ax)
sb.move_legend(ax,loc='lower left')
ax.tick_params(axis='both', which='both', labelsize=18);
fig.suptitle("How math results depend on age of Internet access and knowledge of exponen
plt.setp(ax.get_legend().get_texts(), fontsize='16');
plt.setp(ax.get_legend().get_title(), fontsize='20');

#ax.set_title();
```

Out[102]:
```
[None, None]
```

How math results depend on age of Internet access and knowledge of exponenta



It seems cleat that two groups (in terms of age of access) are differ very much from the others - student who never get access to Internet or who get access in the age 13+

Talk about some of the relationships you observed in this part of the investigation. Were there features that strengthened each other in terms of looking at your feature(s) of interest?

It seems that age of access to Internet is very important in terms of PISA results. We can say, that if student didn't get access to the Internet up to the age 12 - he or she probably will have comparatively low results.

Were there any interesting or surprising interactions between features?

A lot of books at home doesn't add value - on contrary, results of students with 500+ books are slightly lower than results of student with 200-500 books at home

Conclusions

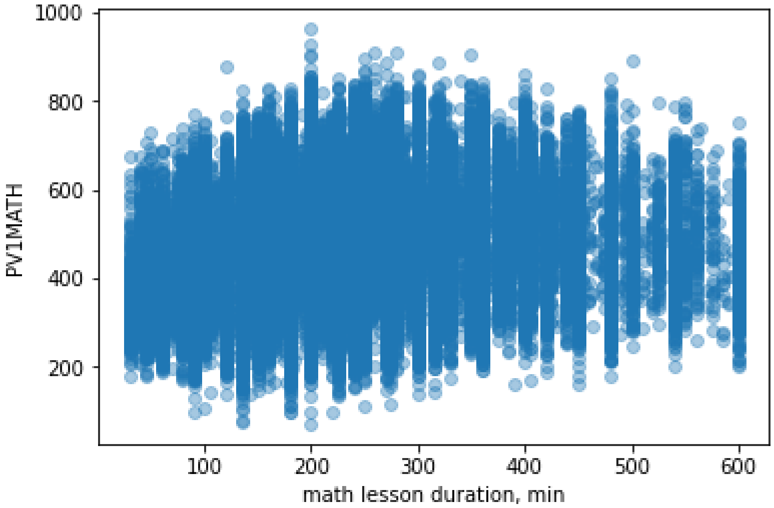
Some key points:

- All test results are normally distributed
- Dataset contains a lot of missing values. For about 50% of records answers weren't given for 30-40% of questions.
- Test results in math doesn't directly correlated to duration of math lessons.
- Student who didn't get access to the Internet before age 13 demonstrate significantly lower results. However, it possible is example of correlation and not of causality. Probably, there is some other reasons that influence both - learning (and, as result, PISA test outcomes) and age of access to the Internet.

In [112…

Out[112]:

```
Text(0, 0.5, 'PV1MATH')
```


1000
800
600
400
200
PV1MATH
100
200
300
400
500
600
math lesson duration, min

In [ ]: