Ex1.StdDev

January 16, 2024

1 1st exercise: Work with standard deviations for anomaly detection

- Course: AML
- Lecturer: Gernot Heisenberg
- Author of notebook: Finn Heydemann
- Date: 28.10.2023

GENERAL NOTE 1: Please make sure you are reading the entire notebook, since it contains a lot of information on your tasks (e.g. regarding the set of certain paramaters or a specific computational trick), and the written mark downs as well as comments contain a lot of information on how things work together as a whole.
GENERAL NOTE 2: * Please, when commenting source code, just use English language only. * When describing an observation please use English language, too * This applies to all exercises throughout this course.

1.0.1 DESCRIPTION:

This notebook allows you for getting into standard deviations as a common technique to detect anomalies when the data is normally distributed.

1.0.2 TASKS:

The tasks that you need to work on within this notebook are always indicated below as bullet points. If a task is more challenging and consists of several steps, this is indicated as well. Make sure you have worked down the task list and commented your doings. This should be done by using markdown. Make sure you don't forget to specify your name and your matriculation number in the notebook.

YOUR TASKS in this exercise are as follows: 1. import the notebook to Google Colab or use your local machine. 2. make sure you specified you name and your matriculation number in the header below my name and date. * set the date too and remove mine. 3. read the entire notebook carefully * add comments whereever you feel it necessary for better understanding * run the notebook for the first time. * understand the output 4. go and find three different data sets on the web * kaggle.com might be a good source (they also offer an API for data download) * make sure two of the three data sets are normally distributed * download one data set that is not normally distributed 5. visualize the data 6. compute the anomalies 7. visualize the anomalies 8. does the 0,3% rule apply? 9. what are differences between the normally distributed and the non-normally distributed data sets with respect to the outlier detection? 10. which statement can be made and which cannot? ———————————————————————————

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from numpy.random import randn
np.random.seed(1)
```

[2]:
```
# Function to Detection Outlier on one-dimentional datasets.
def find_anomalies(random_data):
    #define a list to accumlate anomalies
    anomalies = []

    # Set upper and lower limit to 3 standard deviation
    random_data_std = np.std(random_data)
    random_data_mean = np.mean(random_data)
    anomaly_cut_off = random_data_std * 3

    lower_limit  = random_data_mean - anomaly_cut_off
    upper_limit = random_data_mean + anomaly_cut_off

    print("lower limit=", round(lower_limit,8))
    print("upper limit=", round(upper_limit,8))

    # Generate outliers list
    for outlier in random_data:
        if outlier > upper_limit or outlier < lower_limit:
            anomalies.append(outlier)

    return anomalies
```

[3]:
```
# multiply and add by random numbers to get some real values
# randn generates samples from the normal distribution (important - see below)
data = np.random.randn(50000)  * 20 + 20
```

[4]:
```
anomalies = find_anomalies(data)
```

```
lower limit= -39.96482266
```

```
upper limit= 80.11636225
```

1.1 Result

These anomalies are exceeding the lower and upper 3rd scatter range. Thus, statistically spoken, they do belong to a population size of less than 0,3% of the entire data set! For sure, the above conclusion is true if and only if the data is normally distributed!

[5]:
```python
print(anomalies)
```

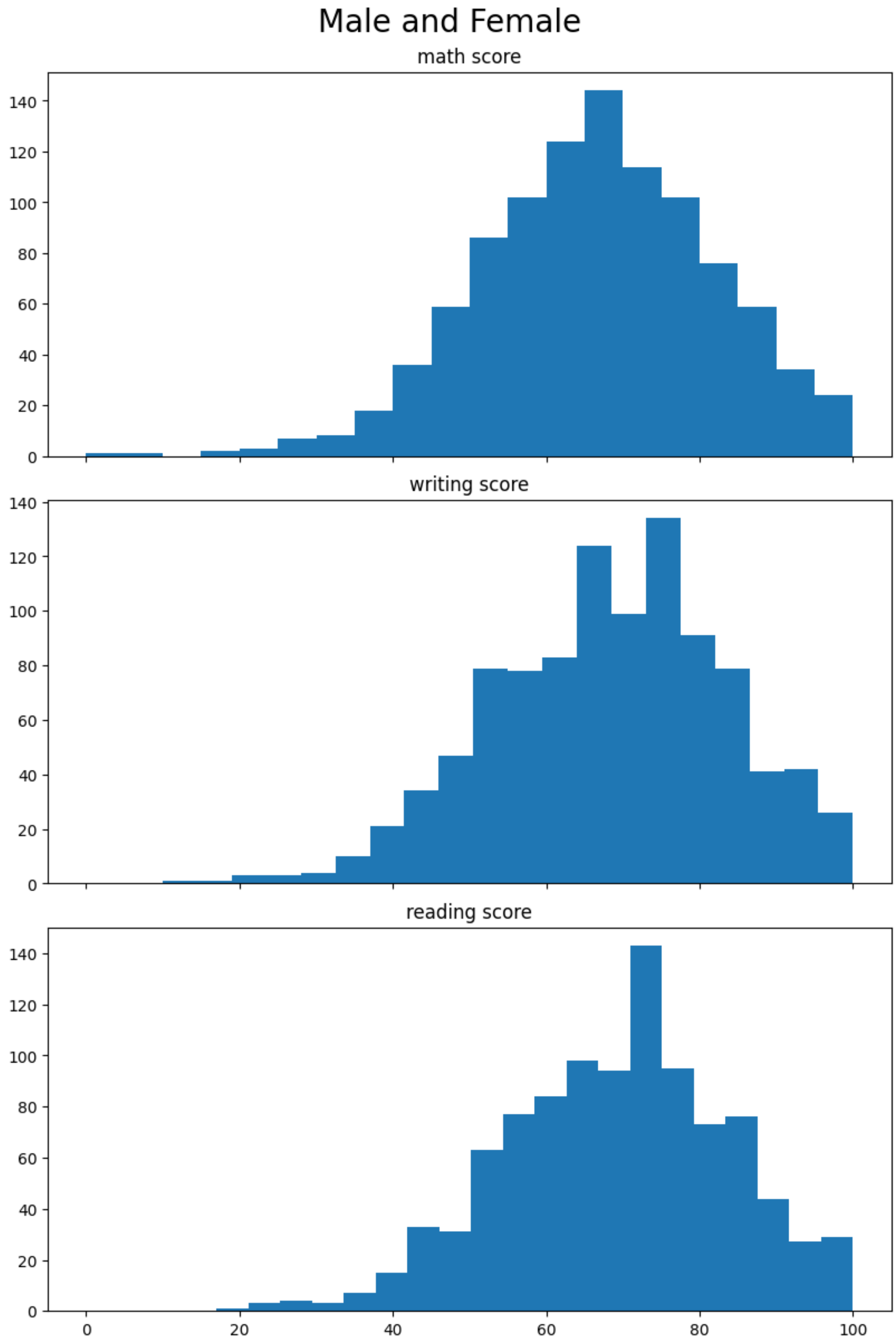
```
[80.61714224744061, 99.17205408075927, 86.4215751234073, -41.0752876085261,
-43.06714900382097, -40.32063970418597, 88.65326863591119, -41.28282712210201,
94.80497807409178, 82.70094680175814, 84.76686393504752, -45.06068469953237,
82.64059456281011, 80.25509136474466, -43.81232693648161, -45.21230191553079,
88.08604551497943, 82.369591815651, 85.9708104656157, 82.97968050439259,
-45.8971681339703, -41.11266480828613, 100.53698089094755, -44.38264211247683,
82.23783662033449, 92.26554014166295, 87.96231313004853, 91.21746641411097,
-53.128801985095905, -45.641575926212354, 91.21225295344836, -43.46923274683489,
82.13498657145581, 82.90737927394908, -46.216851244091984, -41.9565310384887,
-45.6065519410858, 80.5243698860871, -43.30420249353775, -42.25263324372853,
81.59215546151304, 103.36235355910189, 96.68762041821407, -40.586879558462776,
-48.71851620008829, -45.18412137279293, 81.54158108441533, 84.5587224070789,
-40.7688517191754, -44.08128900692253, -42.043297873881315, 87.71430311862296,
88.9391203157937, -40.52113314693459, -49.02805812510347, 86.72050502651298,
-44.452278374625166, -58.55028132123577, 87.83885146048503, -41.810065776968784,
-42.38237155225082, 91.19937116220078, 93.19531683513023, -41.0172829464469,
82.94710559091223, 86.8410257826826, -47.358945718270846, -47.27471768342224,
-44.05283959700523, 81.39224116690686, -64.66329594864382, -41.91924535511986,
86.98293402121955, -42.78324538163221, -43.44070544512959, 83.14254869231635,
80.29722046012112, 80.97664481662183, -49.135306899574985, 90.95359817188209,
85.7504914101956, 90.63353726783707, -52.25624570621119, -48.427273980448774,
92.45148786479089, -40.60136533430394, 90.66197938200516, -40.652000332079666,
85.47272253055641, -40.59878231029168, -53.957837173283735, -40.80036485428288,
82.4740392321337, 82.69560613707037, -46.18148126636443, 86.9908487614501,
80.1457297060347, 82.24721888631221, -45.832275955612076, 84.176266941649,
84.4193809675114, -48.12080165824392, -47.66016305888587, 83.03299706691035,
-46.34572895708007, -43.94127801481702, -44.888893484963376, 82.73490952329395,
83.57335544403156, 80.7727413101268, -52.50413207267792, -49.69745467233162,
80.22051938742219, -61.76977168084022, -41.57467213920625, -58.86284640187061,
-51.197383857166216, -56.11928395638613, -48.85042290718998, -52.64159591899764,
-40.858015925037485, 90.00512514161373, -44.150803760611865, 81.0757508416524,
88.42082610751223, -48.97515466922074, -41.87410601743178, 81.91272478059072,
84.69303765209298, -40.93984641793596, -52.615615036883995, -49.39834453038375,
84.48319182306491, 80.41393381216217, 89.04319371214211, 83.11545348720793,
86.3948526034166, -54.34260433053814, -53.884089553553295, -60.83360878272147,
-45.502951473453436, -42.84117089742673]
```

[6]:
```python
data = pd.read_csv("data/exercise_1/StudentsPerformance.csv")
```

[7]:
```
data.head()
```

[7]:
```
   gender race/ethnicity parental level of education         lunch  \
0  female        group B           bachelor's degree      standard
1  female        group C                some college      standard
2  female        group B             master's degree      standard
3    male        group A          associate's degree  free/reduced
4    male        group C                some college      standard

  test preparation course  math score  reading score  writing score
0                    none          72             72             74
1               completed          69             90             88
2                    none          90             95             93
3                    none          47             57             44
4                    none          76             78             75
```

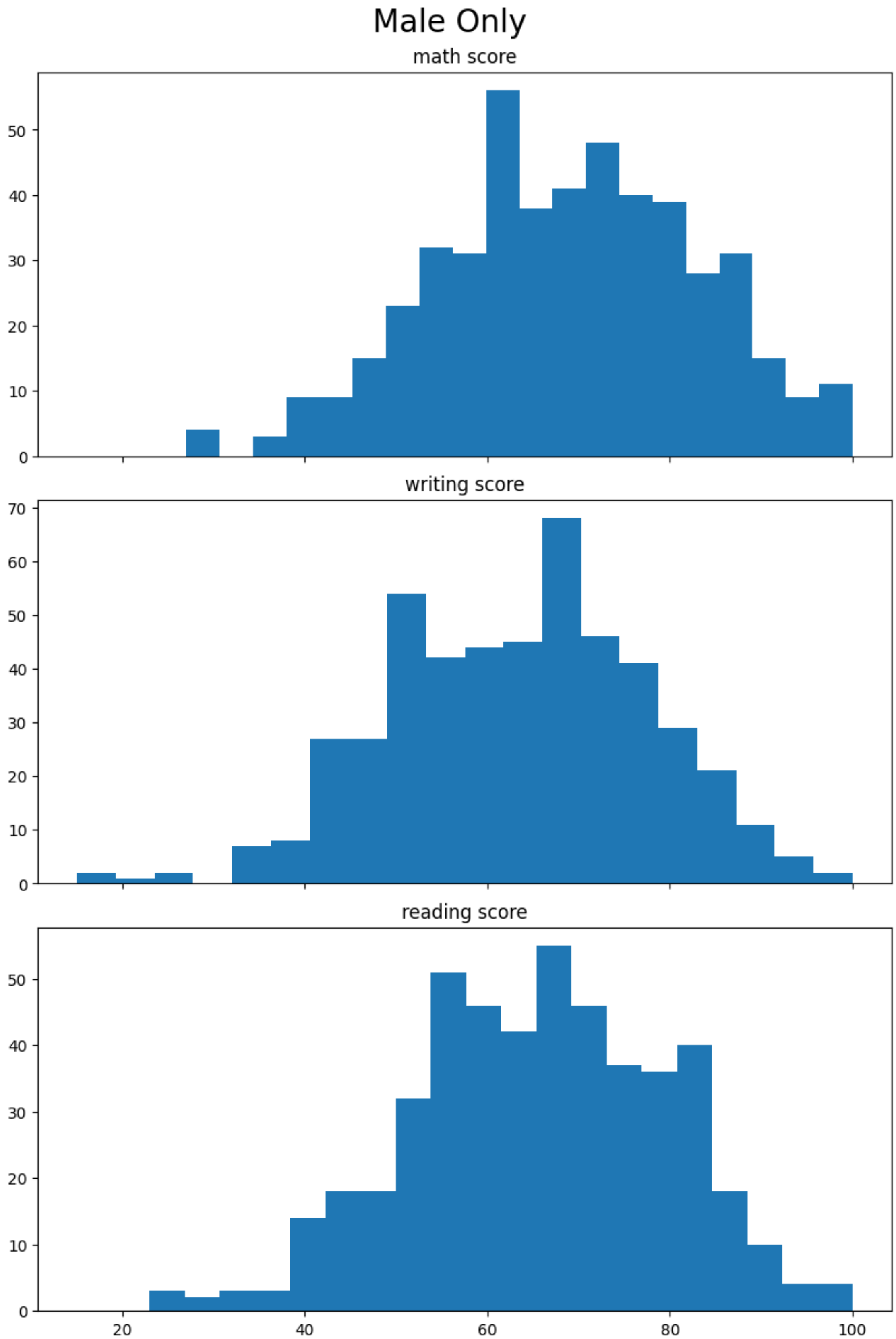
[8]:
```
import matplotlib.pyplot as plt

fig, axs = plt.subplots(3, figsize=(8, 12), layout='constrained', sharex=True)
fig.suptitle("Male and Female", fontsize=20)
for ax, key in zip(axs, ["math score", "writing score", "reading score"]):
    ax.hist(data[key], bins=20)
    ax.set_title(key)

plt.show()
```



Male and Female
math score
writing score
reading score
0
20
40
60
80
100
120
140

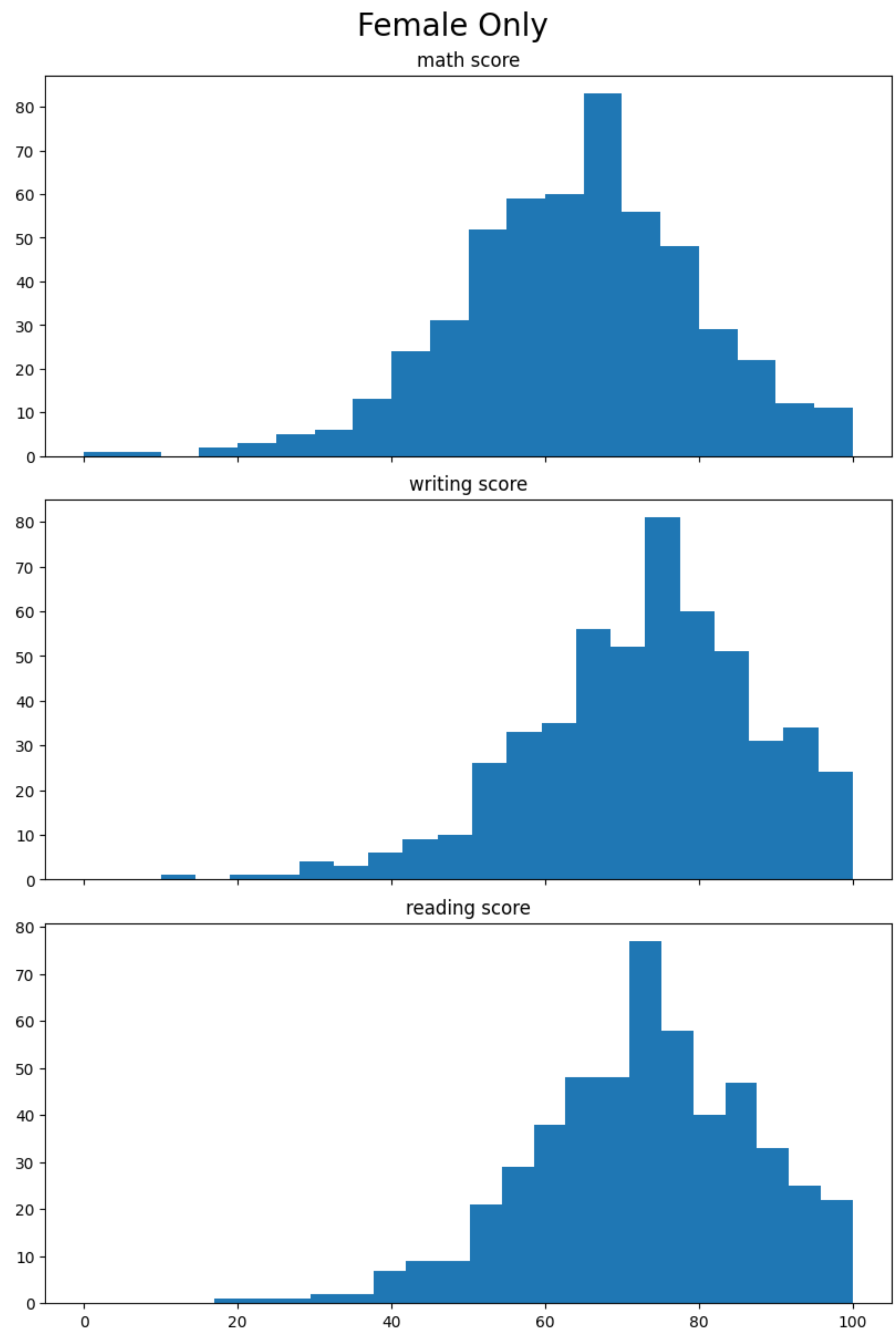
[9]:

```
fig, axs = plt.subplots(3, figsize=(8, 12), layout='constrained', sharex=True)
fig.suptitle("Male Only", fontsize=20)
for ax, key in zip(axs, ["math score", "writing score", "reading score"]):
    ax.hist(data[data["gender"] == "male"][key], bins=20)
    ax.set_title(key)

plt.show()
```


Male Only
math score
writing score
reading score
0
10
20
30
40
50
60
70
20
40
60
80
100

[10]:

```python
fig, axs = plt.subplots(3, figsize=(8, 12), layout='constrained', sharex=True)
fig.suptitle("Female Only", fontsize=20)
for ax, key in zip(axs, ["math score", "writing score", "reading score"]):
    ax.hist(data[data["gender"] == "female"][key], bins=20)
    ax.set_title(key)

plt.show()
```


Female Only
math score
writing score
reading score
0
10
20
30
40
50
60
70
80
0
20
40
60
80
100

[11]:
```
data["math score"].mean(), data["math score"].std()
```

[11]:
```
(66.089, 15.16308009600945)
```

[12]:
```
def anom_detection(data: np.ndarray or pd.Series):
    return data[np.logical_or(data < data.mean() - 3 * data.std(), data > data.
 ↪mean() + 3 * data.std())]

anom_detection(data["math score"])
```

[12]:
```
17     18
59      0
787    19
980     8
Name: math score, dtype: int64
```

[13]:
```
ind = anom_detection(data["math score"]).index
data.loc[ind]
```

[13]:
```
     gender race/ethnicity parental level of education         lunch  \
17   female        group B            some high school  free/reduced
59   female        group C            some high school  free/reduced
787  female        group B                some college      standard
980  female        group B                 high school  free/reduced

    test preparation course  math score  reading score  writing score
17                     none          18             32             28
59                     none           0             17             10
787                    none          19             38             32
980                    none           8             24             23
```

[14]:
```
data = pd.read_csv("data/exercise_1/city_temperature.csv")
```

```
/tmp/ipykernel_30670/3924103419.py:1: DtypeWarning: Columns (2) have mixed
types. Specify dtype option on import or set low_memory=False.
  data = pd.read_csv("data/exercise_1/city_temperature.csv")
```

[15]:
```
data.head()
```

[15]:
```
   Region  Country State     City  Month  Day  Year  AvgTemperature
0  Africa  Algeria   NaN  Algiers      1    1  1995            64.2
1  Africa  Algeria   NaN  Algiers      1    2  1995            49.4
2  Africa  Algeria   NaN  Algiers      1    3  1995            48.8
3  Africa  Algeria   NaN  Algiers      1    4  1995            46.4
4  Africa  Algeria   NaN  Algiers      1    5  1995            47.9
```

[16]:
```
data.drop(data[data["AvgTemperature"] == -99].index, inplace=True)
```

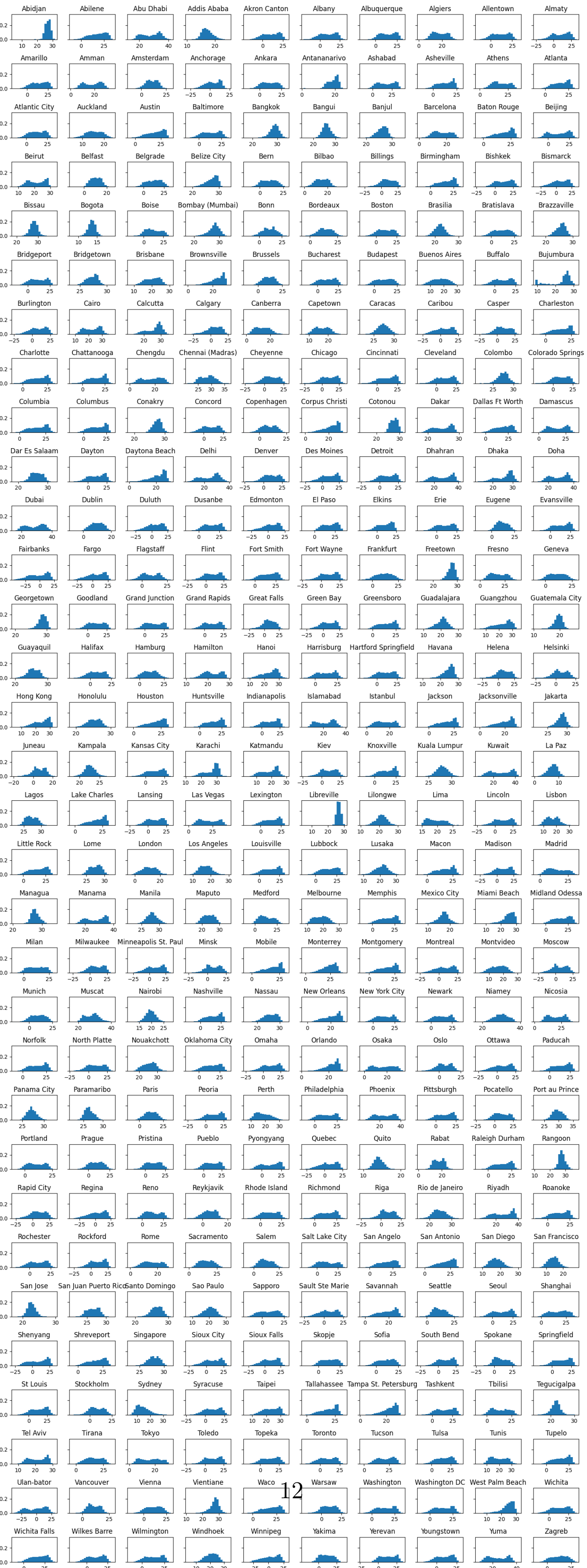
[17]:
```
data["AvgTemperature"] = (data["AvgTemperature"] - 32) * 5/9
```

[18]:
```
data["date"] = pd.to_datetime(data[["Year", "Month", "Day"]])
data = data.set_index("date").groupby(["City", pd.
 ↪Grouper(freq="D")])["AvgTemperature"].mean()
data = data.reset_index()
```

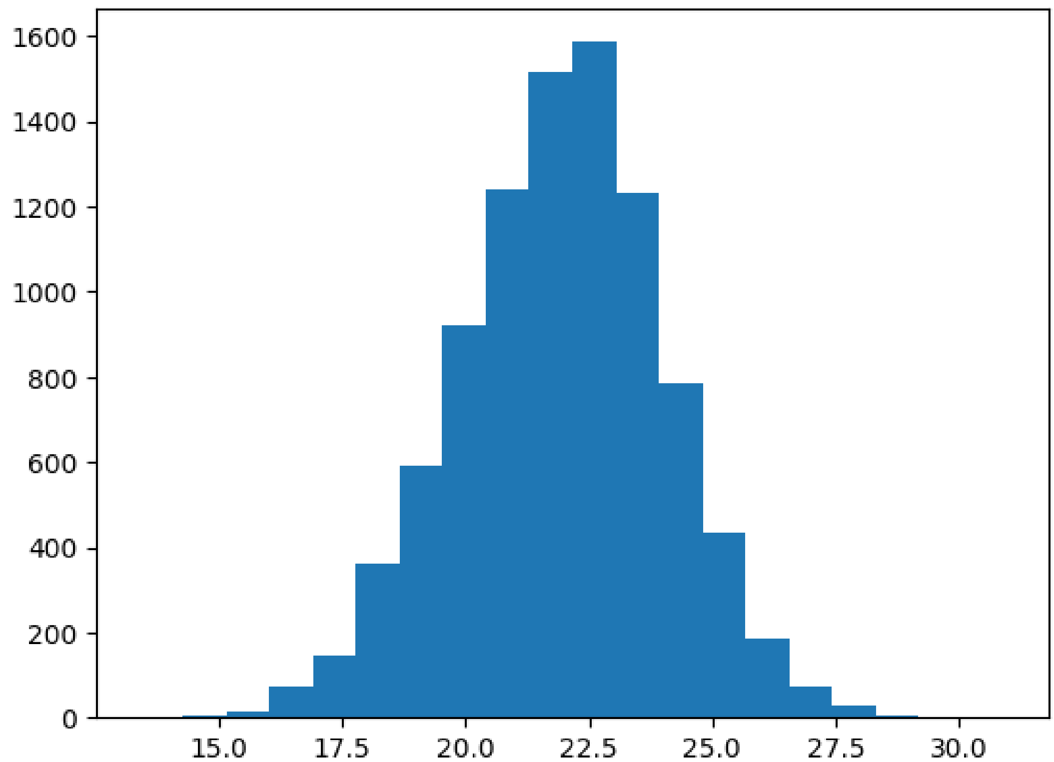
[19]:
```
data
```

[19]:
```
            City        date  AvgTemperature
0        Abidjan 1995-01-01       28.111111
1        Abidjan 1995-01-02       27.944444
2        Abidjan 1995-01-03       27.222222
3        Abidjan 1995-01-04       28.500000
4        Abidjan 1995-01-05       28.555556
…            …          …             …
2769402   Zurich 2020-05-09       19.500000
2769403   Zurich 2020-05-10       18.166667
2769404   Zurich 2020-05-11       11.111111
2769405   Zurich 2020-05-12        6.388889
2769406   Zurich 2020-05-13        7.000000

[2769407 rows x 3 columns]
```

[20]:
```
citys = np.unique(data["City"])
fig, axs = plt.subplots(len(citys) // 10, 10, sharey=True, figsize=(15, 40),␣
 ↪tight_layout=True)
for ax, city in zip(axs.flatten(), citys):
    p_data = data[data["City"] == city]["AvgTemperature"]
    weights = np.ones_like(p_data)/float(len(p_data))
    ax.hist(p_data, bins=20, weights=weights)
    ax.set_title(city)

plt.show()
```


Abidjan Abilene Abu Dhabi Addis Ababa Akron Canton Albany Albuquerque Algiers Allentown Almaty
Amarillo Amman Amsterdam Anchorage Ankara Antananarivo Ashabad Asheville Athens Atlanta
Atlantic City Auckland Austin Baltimore Bangkok Bangui Banjul Barcelona Baton Rouge Beijing
Beirut Belfast Belgrade Belize City Bern Bilbao Billings Birmingham Bishkek Bismarck
Bissau Bogota Boise Bombay (Mumbai) Bonn Bordeaux Boston Brasilia Bratislava Brazzaville
Bridgeport Bridgetown Brisbane Brownsville Brussels Bucharest Budapest Buenos Aires Buffalo Bujumbura
Burlington Cairo Calcutta Calgary Canberra Capetown Caracas Caribou Casper Charleston
Charlotte Chattanooga Chengdu Chennai (Madras) Cheyenne Chicago Cincinnati Cleveland Colombo Colorado Springs
Columbia Columbus Conakry Concord Copenhagen Corpus Christi Cotonou Dakar Dallas Ft Worth Damascus
Dar Es Salaam Dayton Daytona Beach Delhi Denver Des Moines Detroit Dhahran Dhaka Doha
Dubai Dublin Duluth Dusanbe Edmonton El Paso Elkins Erie Eugene Evansville
Fairbanks Fargo Flagstaff Flint Fort Smith Fort Wayne Frankfurt Freetown Fresno Geneva
Georgetown Goodland Grand Junction Grand Rapids Great Falls Green Bay Greensboro Guadalajara Guangzhou Guatemala City
Guayaquil Halifax Hamburg Hamilton Hanoi Harrisburg Hartford Springfield Havana Helena Helsinki
Hong Kong Honolulu Houston Huntsville Indianapolis Islamabad Istanbul Jackson Jacksonville Jakarta
Juneau Kampala Kansas City Karachi Katmandu Kiev Knoxville Kuala Lumpur Kuwait La Paz
Lagos Lake Charles Lansing Las Vegas Lexington Libreville Lilongwe Lima Lincoln Lisbon
Little Rock Lome London Los Angeles Louisville Lubbock Lusaka Macon Madison Madrid
Managua Manama Manila Maputo Medford Melbourne Memphis Mexico City Miami Beach Midland Odessa
Milan Milwaukee Minneapolis St. Paul Minsk Mobile Monterrey Montgomery Montreal Montvideo Moscow
Munich Muscat Nairobi Nashville Nassau New Orleans New York City Newark Niamey Nicosia
Norfolk North Platte Nouakchott Oklahoma City Omaha Orlando Osaka Oslo Ottawa Paducah
Panama City Paramaribo Paris Peoria Perth Philadelphia Phoenix Pittsburgh Pocatello Port au Prince
Portland Prague Pristina Pueblo Pyongyang Quebec Quito Rabat Raleigh Durham Rangoon
Rapid City Regina Reno Reykjavik Rhode Island Richmond Riga Rio de Janeiro Riyadh Roanoke
Rochester Rockford Rome Sacramento Salem Salt Lake City San Angelo San Antonio San Diego San Francisco
San Jose San Juan Puerto Rico Santo Domingo Sao Paulo Sapporo Sault Ste Marie Savannah Seattle Seoul Shanghai
Shenyang Shreveport Singapore Sioux City Sioux Falls Skopje Sofia South Bend Spokane Springfield
St Louis Stockholm Sydney Syracuse Taipei Tallahassee Tampa St. Petersburg Tashkent Tbilisi Tegucigalpa
Tel Aviv Tirana Tokyo Toledo Topeka Toronto Tucson Tulsa Tunis Tupelo
Ulan-bator Vancouver Vienna Vientiane Waco Warsaw Washington Washington DC West Palm Beach Wichita
Wichita Falls Wilkes Barre Wilmington Windhoek Winnipeg Yakima Yerevan Youngstown Yuma Zagreb

[21]:
```
plt.hist(data[data["City"] == "Brasilia"]["AvgTemperature"], bins=20)
plt.show()
```



[22]:
```
ind = anom_detection(data[data["City"] == "Brasilia"]["AvgTemperature"]).index
```

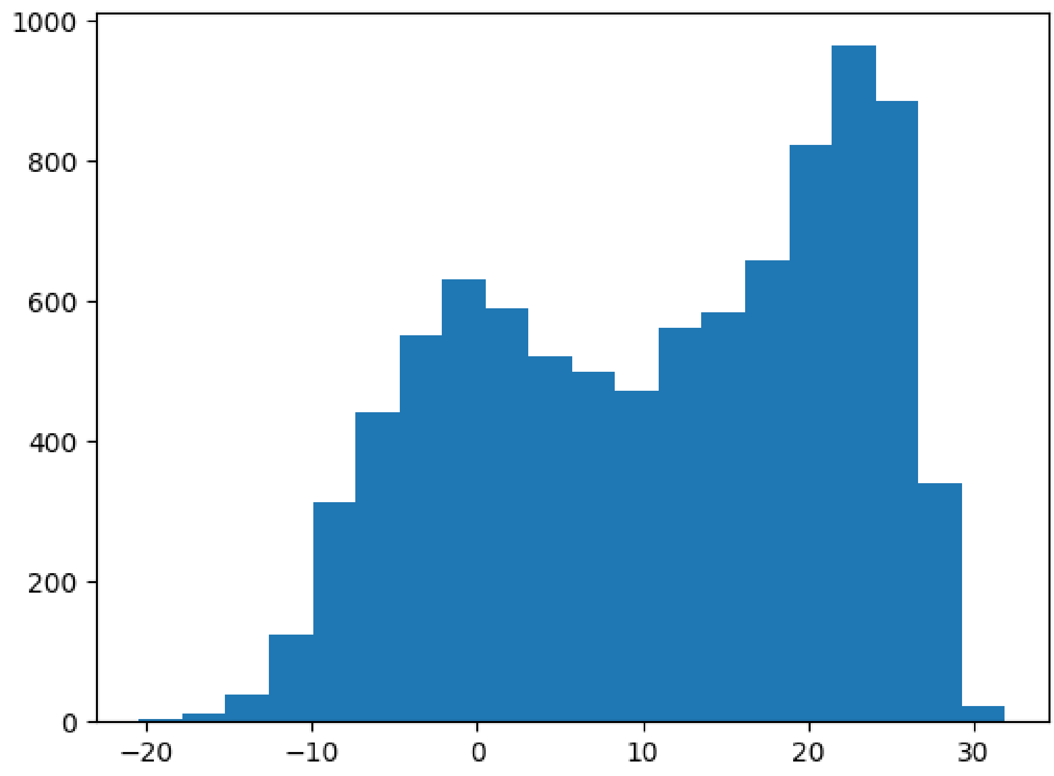
[23]:
```
data.loc[ind]
```

[23]:

	City	date	AvgTemperature
406288	Brasilia	1996-06-29	13.666667
406289	Brasilia	1996-06-30	14.333333
406633	Brasilia	1997-06-09	15.111111
406771	Brasilia	1997-10-25	28.611111
406986	Brasilia	1998-06-02	15.500000
407080	Brasilia	1998-09-06	30.000000
407223	Brasilia	1999-02-17	28.500000
408563	Brasilia	2002-10-26	30.944444
409198	Brasilia	2004-07-24	15.277778
409548	Brasilia	2005-07-09	15.388889
409549	Brasilia	2005-07-10	14.333333
409550	Brasilia	2005-07-11	14.611111

```
409908  Brasilia 2006-07-04       15.388889
409919  Brasilia 2006-07-15       14.888889
410642  Brasilia 2008-07-09       14.888889
410643  Brasilia 2008-07-10       15.444444
410647  Brasilia 2008-07-14       15.333333
412910  Brasilia 2014-09-29       28.388889
412929  Brasilia 2014-10-18       28.777778
413914  Brasilia 2017-07-04       15.111111
413915  Brasilia 2017-07-05       15.055556
413916  Brasilia 2017-07-06       15.555556
414017  Brasilia 2017-10-15       29.000000
414262  Brasilia 2018-06-18       13.388889
414712  Brasilia 2019-09-20       28.222222
414713  Brasilia 2019-09-21       29.555556
414742  Brasilia 2019-10-20       28.555556
414765  Brasilia 2019-11-12       28.500000
```

[24]:
```
plt.hist(data[data["City"] == "Pyongyang"]["AvgTemperature"], bins=20)
plt.show()
```



[25]:
```
anom_detection(data[data["City"] == "Pyongyang"]["AvgTemperature"])
```

[25]: Series([], Name: AvgTemperature, dtype: float64)

Obviously the anomaly detection does'nt work when the data is not normally distrubted. When checking the daily temperatures in Pyongyang no anomalies are detected. Whereas when the daily temperatures are evenly distributed, anomolies are detected. Same goes for the student performances.

```
[ ]:
```