wasm-js-energy-study-analysis

November 4, 2023

[1]:

```
pip install pandas matplotlib seaborn numpy scipy statsmodels scikits.bootstrap
```

```
Requirement already satisfied: pandas in /opt/conda/lib/python3.9/site-packages
(2.1.2)
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.9/site-
packages (3.8.1)
Requirement already satisfied: seaborn in /opt/conda/lib/python3.9/site-packages
(0.13.0)
Requirement already satisfied: numpy in /opt/conda/lib/python3.9/site-packages
(1.26.1)
Requirement already satisfied: scipy in /opt/conda/lib/python3.9/site-packages
(1.11.3)
Requirement already satisfied: statsmodels in /opt/conda/lib/python3.9/site-
packages (0.14.0)
Requirement already satisfied: scikits.bootstrap in
/opt/conda/lib/python3.9/site-packages (1.1.0)
Requirement already satisfied: tzdata>=2022.1 in /opt/conda/lib/python3.9/site-
packages (from pandas) (2023.3)
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/conda/lib/python3.9/site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.9/site-
packages (from pandas) (2022.1)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.9/site-
packages (from matplotlib) (21.3)
Requirement already satisfied: pyparsing>=2.3.1 in
/opt/conda/lib/python3.9/site-packages (from matplotlib) (3.0.7)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.9/site-
packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/opt/conda/lib/python3.9/site-packages (from matplotlib) (4.44.0)
Requirement already satisfied: pillow>=8 in /opt/conda/lib/python3.9/site-
packages (from matplotlib) (10.1.0)
Requirement already satisfied: contourpy>=1.0.1 in
/opt/conda/lib/python3.9/site-packages (from matplotlib) (1.1.1)
Requirement already satisfied: kiwisolver>=1.3.1 in
/opt/conda/lib/python3.9/site-packages (from matplotlib) (1.4.5)
Requirement already satisfied: importlib-resources>=3.2.0 in
/opt/conda/lib/python3.9/site-packages (from matplotlib) (5.6.0)
```

```
Requirement already satisfied: patsy>=0.5.2 in /opt/conda/lib/python3.9/site-
packages (from statsmodels) (0.5.3)
Requirement already satisfied: pyerf in /opt/conda/lib/python3.9/site-packages
(from scikits.bootstrap) (1.0.1)
Requirement already satisfied: typing-extensions in
/opt/conda/lib/python3.9/site-packages (from scikits.bootstrap) (4.8.0)
Requirement already satisfied: zipp>=3.1.0 in /opt/conda/lib/python3.9/site-
packages (from importlib-resources>=3.2.0->matplotlib) (3.7.0)
Requirement already satisfied: six in /opt/conda/lib/python3.9/site-packages
(from patsy>=0.5.2->statsmodels) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

[2]:

```python
import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy
from scipy import stats
import scipy
import helper.cliffs_delta as cliff
import statsmodels.api as sm
import itertools
import scikits.bootstrap as boot
```

1 Table of contents

2 Prepare Data

[3]:

```python
sns.set(rc = {'figure.figsize':(10,10)})
confidence_level = 0.95

df = pd.read_csv('./data/energy.csv',sep=';',quotechar='"',names=["browser",␣
 ↪"language", "algorithm", "device", "energy"])
df['energy_total'] = df['energy'].astype(float)
df['implementation'] = numpy.where(df['language'] == 'js', 'js', 'wasm')

## unique values in columns
browsers = numpy.sort(df['browser'].unique())
browserpairs = list(itertools.combinations(browsers, 2))
languages = df['language'].unique()
implementations = numpy.sort(df['implementation'].unique())
implementationpairs = list(itertools.combinations(implementations, 2))
```

```
languagepairs = list(itertools.combinations(languages, 2))
algorithms = df['algorithm'].unique()
devices = df['device'].unique()

df.head()
```

```
[3]:    browser language              algorithm    device    energy  energy_total  \
     0  firefox        c              fibonacci  SM-G991B  6.531442      6.531442
     1   chrome        c          humblenumbers  SM-G991B  5.787359      5.787359
     2   chrome       js  matrixmultiplication  SM-G991B  9.739240      9.739240
     3   chrome       js          seqnonsquares  SM-G991B  4.516185      4.516185
     4   chrome       js                nqueens  SM-G991B  5.935186      5.935186

       implementation
     0           wasm
     1           wasm
     2             js
     3             js
     4             js
```

2.1 Count samples

```
[4]: data = []

     for browser in browsers:
         for language in languages:
             for algorithm in algorithms:
                 for device in devices:
                     #print(browser,device,numpy.mean(df[(df['browser'] == browser)␣
       ↪& (df['language'] == language) & (df['algorithm'] == algorithm) &␣
       ↪(df['device'] == device)]['energy']))
                     data.append(
                         [browser, language, algorithm, device, df[(df['browser'] ==␣
       ↪browser) & (df['language'] == language) & (df['algorithm'] == algorithm) &␣
       ↪(df['device'] == device)]['energy_total'].count()]
                     )

     # Create the pandas DataFrame
     count = pd.DataFrame(data, columns = ['browser', 'language', 'algorithm',␣
       ↪'device', 'count'])
     print(count.to_string())
```

```
     browser language              algorithm    device  count
0     chrome        c              fibonacci  SM-G991B     30
1     chrome        c              fibonacci   Nexus 5     32
2     chrome        c          humblenumbers  SM-G991B     31
3     chrome        c          humblenumbers   Nexus 5     34
```

4	chrome	c	matrixmultiplication	SM-G991B	31
5	chrome	c	matrixmultiplication	Nexus 5	32
6	chrome	c	seqnonsquares	SM-G991B	31
7	chrome	c	seqnonsquares	Nexus 5	33
8	chrome	c	nqueens	SM-G991B	30
9	chrome	c	nqueens	Nexus 5	31
10	chrome	c	bubblesort	SM-G991B	32
11	chrome	c	bubblesort	Nexus 5	34
12	chrome	c	perfectnumbers	SM-G991B	31
13	chrome	c	perfectnumbers	Nexus 5	33
14	chrome	c	insertionsort	SM-G991B	31
15	chrome	c	insertionsort	Nexus 5	34
16	chrome	c	heapsort	SM-G991B	31
17	chrome	c	heapsort	Nexus 5	33
18	chrome	c	towersofhanoi	SM-G991B	32
19	chrome	c	towersofhanoi	Nexus 5	32
20	chrome	c	countingsort	SM-G991B	32
21	chrome	c	countingsort	Nexus 5	33
22	chrome	c	shellsort	SM-G991B	30
23	chrome	c	shellsort	Nexus 5	33
24	chrome	c	ackermann	SM-G991B	32
25	chrome	c	ackermann	Nexus 5	32
26	chrome	c	kmeanspp	SM-G991B	32
27	chrome	c	kmeanspp	Nexus 5	32
28	chrome	c	gnomesort	SM-G991B	32
29	chrome	c	gnomesort	Nexus 5	32
30	chrome	c	mergesort	SM-G991B	32
31	chrome	c	mergesort	Nexus 5	31
32	chrome	c	happynumbers	SM-G991B	30
33	chrome	c	happynumbers	Nexus 5	32
34	chrome	c	pancakesort	SM-G991B	30
35	chrome	c	pancakesort	Nexus 5	33
36	chrome	c	quicksort	SM-G991B	31
37	chrome	c	quicksort	Nexus 5	32
38	chrome	js	fibonacci	SM-G991B	32
39	chrome	js	fibonacci	Nexus 5	32
40	chrome	js	humblenumbers	SM-G991B	31
41	chrome	js	humblenumbers	Nexus 5	31
42	chrome	js	matrixmultiplication	SM-G991B	31
43	chrome	js	matrixmultiplication	Nexus 5	30
44	chrome	js	seqnonsquares	SM-G991B	32
45	chrome	js	seqnonsquares	Nexus 5	33
46	chrome	js	nqueens	SM-G991B	31
47	chrome	js	nqueens	Nexus 5	32
48	chrome	js	bubblesort	SM-G991B	31
49	chrome	js	bubblesort	Nexus 5	31
50	chrome	js	perfectnumbers	SM-G991B	31
51	chrome	js	perfectnumbers	Nexus 5	31

52	chrome	js	insertionsort	SM-G991B	31
53	chrome	js	insertionsort	Nexus 5	33
54	chrome	js	heapsort	SM-G991B	31
55	chrome	js	heapsort	Nexus 5	33
56	chrome	js	towersofhanoi	SM-G991B	32
57	chrome	js	towersofhanoi	Nexus 5	31
58	chrome	js	countingsort	SM-G991B	32
59	chrome	js	countingsort	Nexus 5	34
60	chrome	js	shellsort	SM-G991B	31
61	chrome	js	shellsort	Nexus 5	33
62	chrome	js	ackermann	SM-G991B	31
63	chrome	js	ackermann	Nexus 5	33
64	chrome	js	kmeanspp	SM-G991B	31
65	chrome	js	kmeanspp	Nexus 5	33
66	chrome	js	gnomesort	SM-G991B	32
67	chrome	js	gnomesort	Nexus 5	32
68	chrome	js	mergesort	SM-G991B	31
69	chrome	js	mergesort	Nexus 5	34
70	chrome	js	happynumbers	SM-G991B	31
71	chrome	js	happynumbers	Nexus 5	32
72	chrome	js	pancakesort	SM-G991B	32
73	chrome	js	pancakesort	Nexus 5	31
74	chrome	js	quicksort	SM-G991B	31
75	chrome	js	quicksort	Nexus 5	32
76	firefox	c	fibonacci	SM-G991B	31
77	firefox	c	fibonacci	Nexus 5	33
78	firefox	c	humblenumbers	SM-G991B	32
79	firefox	c	humblenumbers	Nexus 5	33
80	firefox	c	matrixmultiplication	SM-G991B	32
81	firefox	c	matrixmultiplication	Nexus 5	32
82	firefox	c	seqnonsquares	SM-G991B	32
83	firefox	c	seqnonsquares	Nexus 5	33
84	firefox	c	nqueens	SM-G991B	31
85	firefox	c	nqueens	Nexus 5	32
86	firefox	c	bubblesort	SM-G991B	31
87	firefox	c	bubblesort	Nexus 5	31
88	firefox	c	perfectnumbers	SM-G991B	32
89	firefox	c	perfectnumbers	Nexus 5	31
90	firefox	c	insertionsort	SM-G991B	31
91	firefox	c	insertionsort	Nexus 5	32
92	firefox	c	heapsort	SM-G991B	31
93	firefox	c	heapsort	Nexus 5	32
94	firefox	c	towersofhanoi	SM-G991B	32
95	firefox	c	towersofhanoi	Nexus 5	34
96	firefox	c	countingsort	SM-G991B	31
97	firefox	c	countingsort	Nexus 5	33
98	firefox	c	shellsort	SM-G991B	31
99	firefox	c	shellsort	Nexus 5	33

100	firefox	c	ackermann	SM-G991B	30
101	firefox	c	ackermann	Nexus 5	32
102	firefox	c	kmeanspp	SM-G991B	31
103	firefox	c	kmeanspp	Nexus 5	31
104	firefox	c	gnomesort	SM-G991B	31
105	firefox	c	gnomesort	Nexus 5	31
106	firefox	c	mergesort	SM-G991B	32
107	firefox	c	mergesort	Nexus 5	32
108	firefox	c	happynumbers	SM-G991B	31
109	firefox	c	happynumbers	Nexus 5	33
110	firefox	c	pancakesort	SM-G991B	32
111	firefox	c	pancakesort	Nexus 5	31
112	firefox	c	quicksort	SM-G991B	31
113	firefox	c	quicksort	Nexus 5	32
114	firefox	js	fibonacci	SM-G991B	31
115	firefox	js	fibonacci	Nexus 5	32
116	firefox	js	humblenumbers	SM-G991B	30
117	firefox	js	humblenumbers	Nexus 5	32
118	firefox	js	matrixmultiplication	SM-G991B	30
119	firefox	js	matrixmultiplication	Nexus 5	31
120	firefox	js	seqnonsquares	SM-G991B	32
121	firefox	js	seqnonsquares	Nexus 5	32
122	firefox	js	nqueens	SM-G991B	31
123	firefox	js	nqueens	Nexus 5	33
124	firefox	js	bubblesort	SM-G991B	31
125	firefox	js	bubblesort	Nexus 5	33
126	firefox	js	perfectnumbers	SM-G991B	32
127	firefox	js	perfectnumbers	Nexus 5	33
128	firefox	js	insertionsort	SM-G991B	32
129	firefox	js	insertionsort	Nexus 5	32
130	firefox	js	heapsort	SM-G991B	30
131	firefox	js	heapsort	Nexus 5	33
132	firefox	js	towersofhanoi	SM-G991B	31
133	firefox	js	towersofhanoi	Nexus 5	32
134	firefox	js	countingsort	SM-G991B	32
135	firefox	js	countingsort	Nexus 5	33
136	firefox	js	shellsort	SM-G991B	31
137	firefox	js	shellsort	Nexus 5	33
138	firefox	js	ackermann	SM-G991B	32
139	firefox	js	ackermann	Nexus 5	32
140	firefox	js	kmeanspp	SM-G991B	32
141	firefox	js	kmeanspp	Nexus 5	32
142	firefox	js	gnomesort	SM-G991B	30
143	firefox	js	gnomesort	Nexus 5	32
144	firefox	js	mergesort	SM-G991B	30
145	firefox	js	mergesort	Nexus 5	32
146	firefox	js	happynumbers	SM-G991B	31
147	firefox	js	happynumbers	Nexus 5	31

```
148   firefox        js           pancakesort  SM-G991B     31
149   firefox        js           pancakesort   Nexus 5     33
150   firefox        js             quicksort  SM-G991B     32
151   firefox        js             quicksort   Nexus 5     33
```

2.2 Total Energy

[5]:

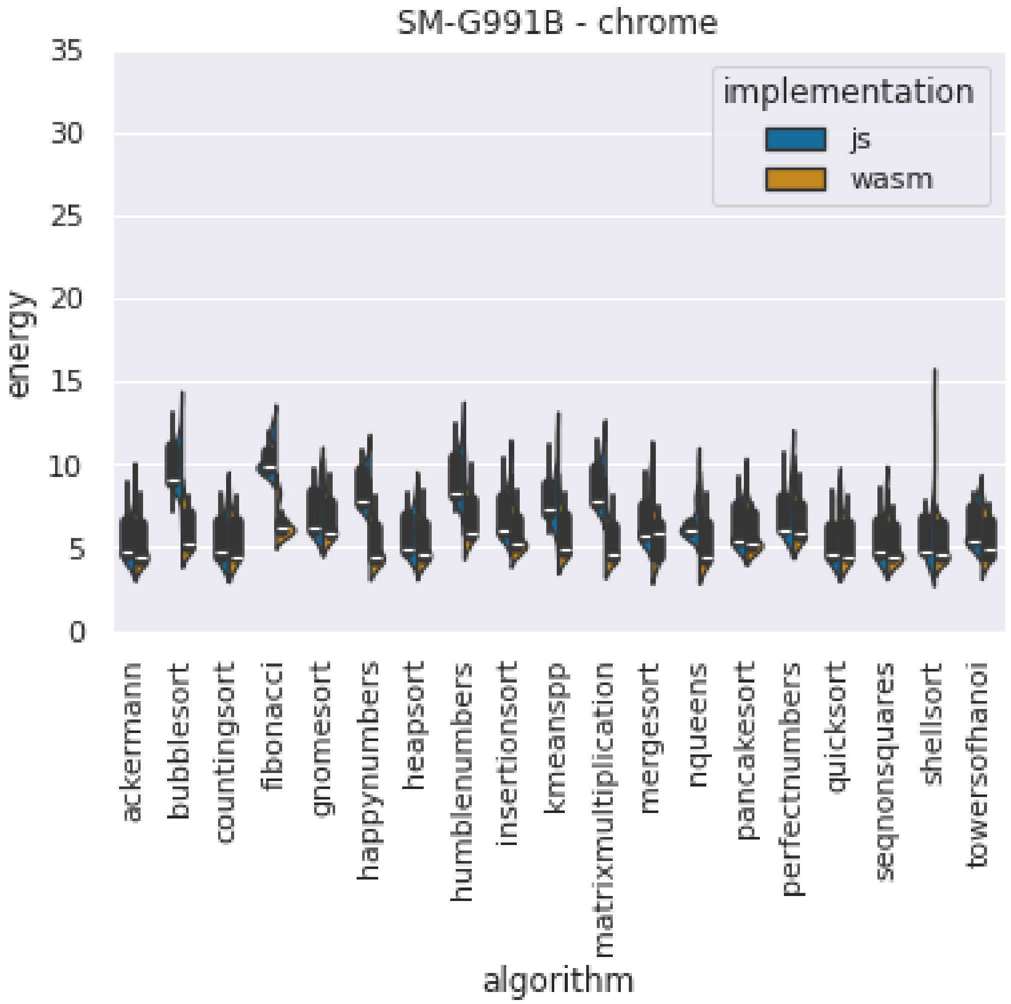
```
data = []
for device in devices:
    for implementation in implementations:
        x = df[(df['device'] == device) & (df['implementation'] ==␣
 ↪implementation)]
        sum = numpy.round(numpy.sum(x['energy']), 2)

        data.append(
            [device, implementation, sum]
        )

# Create the pandas DataFrame
stat = pd.DataFrame(data, columns = ['device', 'implementation', 'sum'])
print(stat.to_string())
```

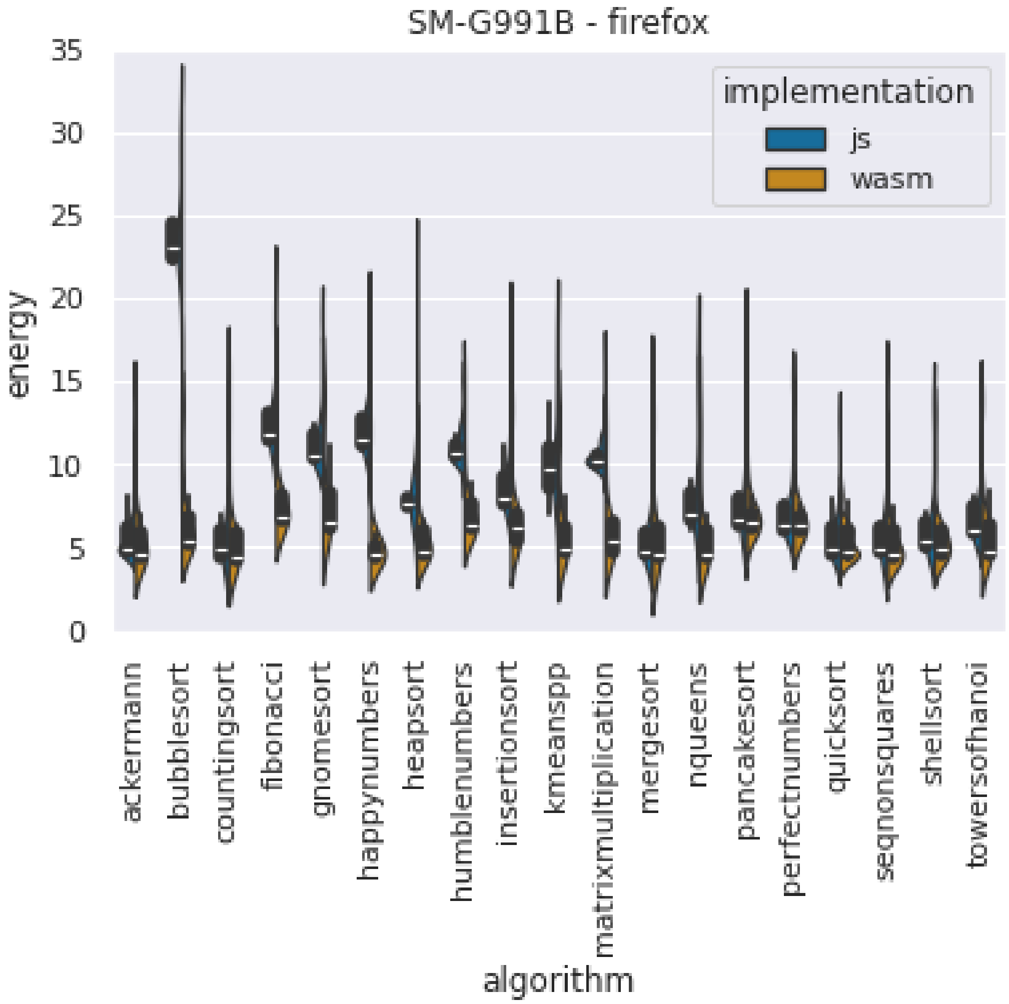
```
     device implementation       sum
0  SM-G991B             js   9638.40
1  SM-G991B           wasm   7110.95
2   Nexus 5             js  14852.36
3   Nexus 5           wasm  11826.85
```

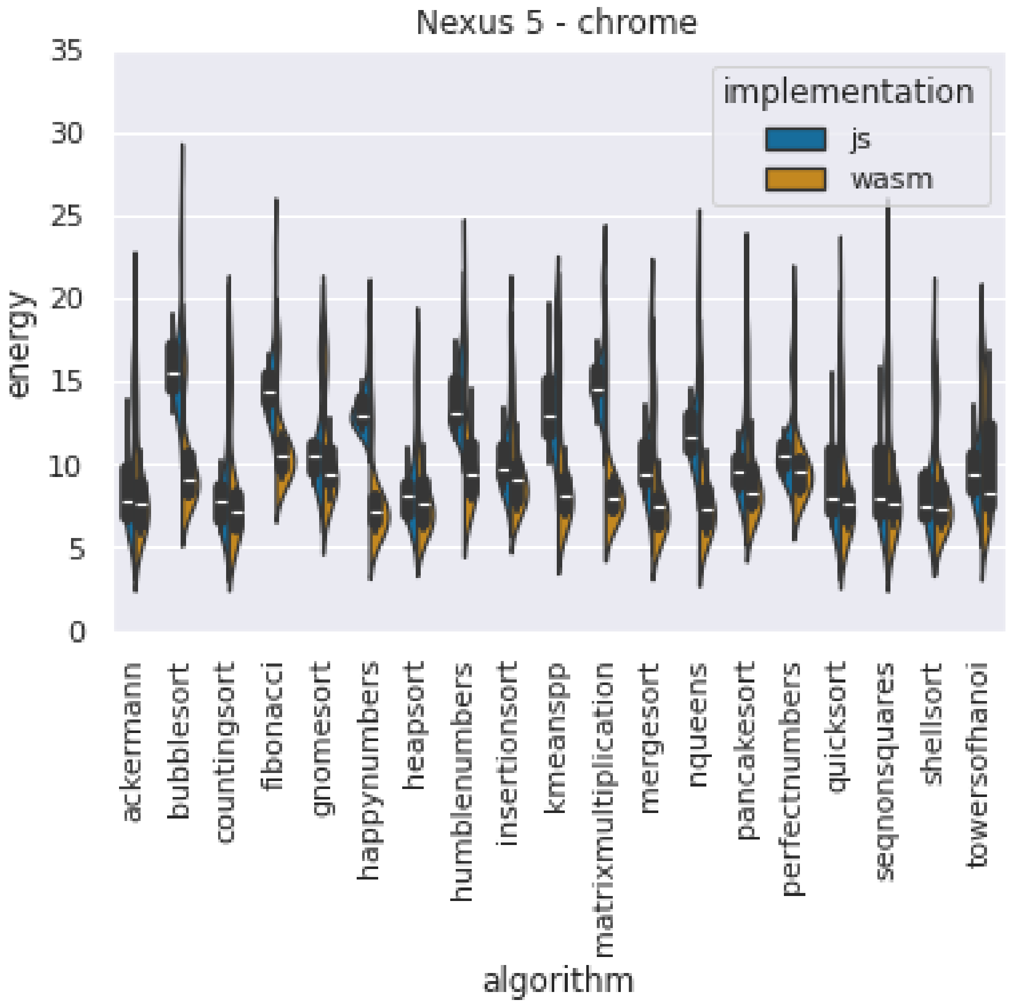
3 Violinplot (Algorithms)

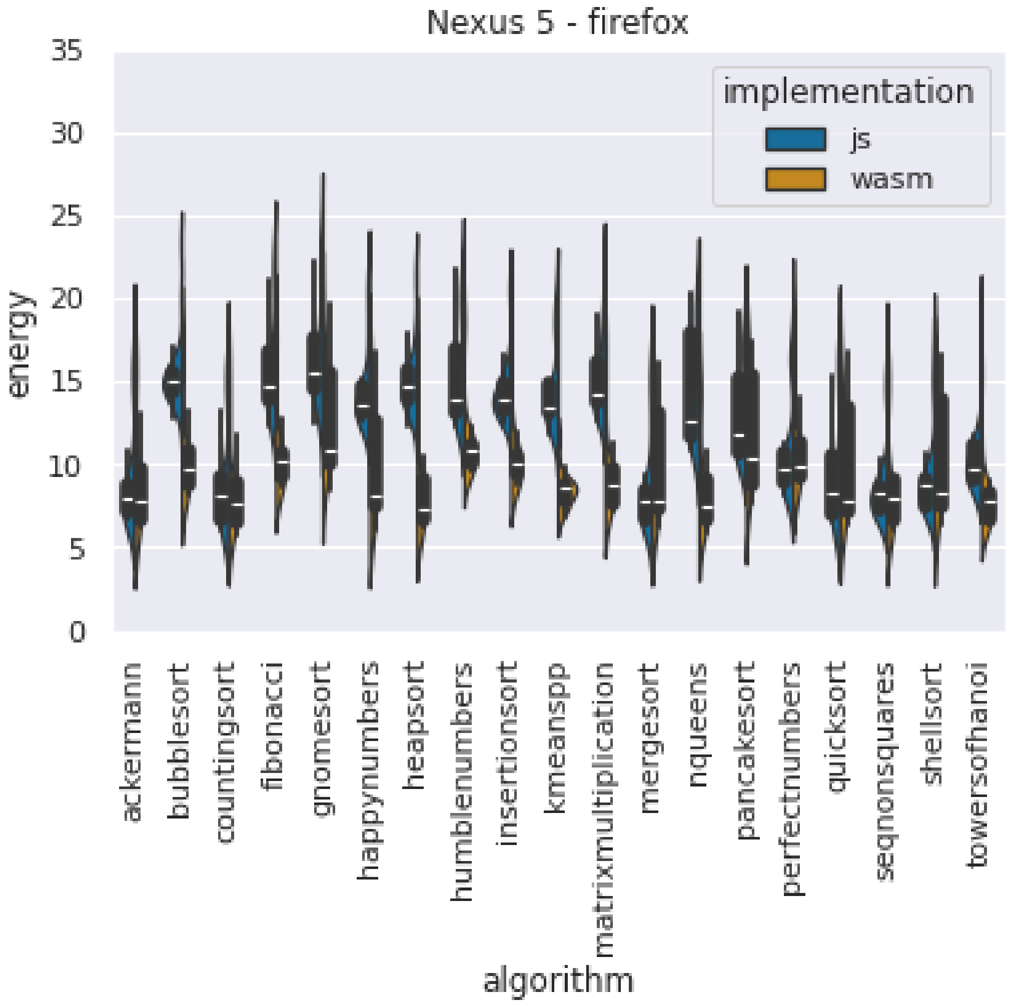
[6]:

```
for device in devices:
    for browser in browsers:
        data = df[(df['device'] == device) & (df['browser'] == browser)].
 ↪sort_values(by=['algorithm'])
        plt.figure()
        plt.xticks(rotation=90)
        plt.ylim(0, 35)
        sns.violinplot(x='algorithm', y='energy', hue='implementation',␣
 ↪hue_order=implementations, data=data, palette='colorblind', split=True).
 ↪set_title(device + " - " + browser)
```


SM-G991B - chrome
implementation
js
wasm
energy
35
30
25
20
15
10
5
0
ackermann
bubblesort
countingsort
fibonacci
gnomesort
happynumbers
heapsort
humblenumbers
insertionsort
kmeanspp
matrixmultiplication
mergesort
nqueens
pancakesort
perfectnumbers
quicksort
seqnonsquares
shellsort
towersofhanoi
algorithm


SM-G991B - firefox
implementation
js
wasm
energy
0
5
10
15
20
25
30
35
algorithm
ackermann
bubblesort
countingsort
fibonacci
gnomesort
happynumbers
heapsort
humblenumbers
insertionsort
kmeanspp
matrixmultiplication
mergesort
nqueens
pancakesort
perfectnumbers
quicksort
seqnonsquares
shellsort
towersofhanoi


Nexus 5 - chrome
energy
0
5
10
15
20
25
30
35
implementation
js
wasm
algorithm
ackermann
bubblesort
countingsort
fibonacci
gnomesort
happynumbers
heapsort
humblenumbers
insertionsort
kmeanspp
matrixmultiplication
mergesort
nqueens
pancakesort
perfectnumbers
quicksort
seqnonsquares
shellsort
towersofhanoi



4 RQ1: JavaScript vs. WebAssembly

4.1 Shapiro Wilk Test

[7]:
```
data = []
non_normal = 0

for implementation in implementations:
    energy = df[(df['implementation'] == implementation)]['energy']

    if len(energy) >= 3:
        shapiro_test = stats.shapiro(energy)

        non_normal += (1 if shapiro_test.pvalue <= 0.05 else 0)
        data.append(
            [implementation,
             shapiro_test.statistic,
             shapiro_test.pvalue
            ]
        )

# Create the pandas DataFrame
swt = pd.DataFrame(data, columns = ['implementation', 'w', 'p'])
#print(swt.to_string())
display(swt)

print("\n{} non-normally distributed samples".format(non_normal))
print("{} normally distributed samples".format(len(swt) - non_normal))
print("{:.2f}% non-normally distributed samples".format(non_normal/
 ↪len(swt)*100))
```

```
  implementation         w             p
0             js  0.927419  1.147179e-32
1           wasm  0.863110  1.110529e-41


2 non-normally distributed samples
0 normally distributed samples
100.00% non-normally distributed samples
```

4.2 Shapiro Wilk Test (By Device)

[8]:
```
data = []
non_normal = 0

for device in devices:
    for implementation in implementations:
        energy = df[(df['implementation'] == implementation) & (df['device'] ==␣
 ↪device)]['energy']

        if len(energy) >= 3:
            shapiro_test = stats.shapiro(energy)

            non_normal += (1 if shapiro_test.pvalue <= 0.05 else 0)

            data.append(
                [
                 device,
                 implementation,
                 shapiro_test.statistic,
                 shapiro_test.pvalue
                ]
            )

# Create the pandas DataFrame
swt = pd.DataFrame(data, columns = ['device', 'implementation', 'w', 'p'])
#print(swt.to_string())
display(swt)

print("\n{} non-normally distributed samples".format(non_normal))
print("{} normally distributed samples".format(len(swt) - non_normal))
print("{:.2f}% non-normally distributed samples".format(non_normal/
 ↪len(swt)*100))
```

```
    device implementation         w             p
0  SM-G991B             js  0.771749  1.073380e-37
1  SM-G991B           wasm  0.747010  3.835847e-39
2   Nexus 5             js  0.960188  8.760830e-18
3   Nexus 5           wasm  0.857640  6.419627e-32


4 non-normally distributed samples
0 normally distributed samples
100.00% non-normally distributed samples
```

4.3 Mann-Whitney-U-Test

[9]:

```
data = []

for implementationpair in implementationpairs:
    impl1_energy = df[(df['implementation'] == implementationpair[0])]['energy']
    impl2_energy = df[(df['implementation'] == implementationpair[1])]['energy']
    eff = cliff.cliffs_delta(impl1_energy, impl2_energy)

    u = stats.mannwhitneyu(impl1_energy, impl2_energy, alternative='two-sided')

    data.append(
        [
         implementationpair[0] + ' vs. ' + implementationpair[1],
         u.statistic,
         u.pvalue,
         eff[0],
         eff[1]
        ]
    )

# Create the pandas DataFrame
ut = pd.DataFrame(data, columns = ['implementation', 'u', 'p', 'eff', 'interp'])
display(ut)
```

```
interp = ut['interp'].value_counts()
interp = pd.DataFrame(interp, columns = ['interp', 'percent'])
interp['percent'] = (interp['interp'] / interp['interp'].sum()) * 100

display(interp)
```

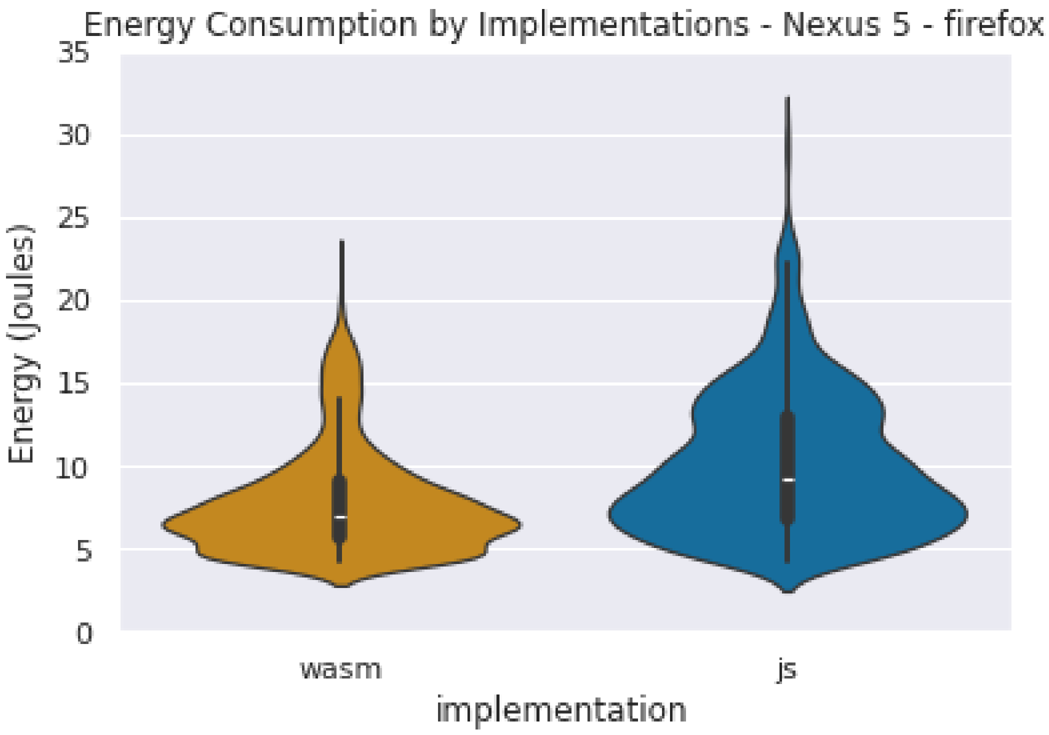
```
  implementation          u             p       eff  interp
0    js vs. wasm  3904820.0  1.173223e-93  0.341271  medium
```

```
Empty DataFrame
Columns: [interp, percent]
Index: []
```

4.4 Violinplot

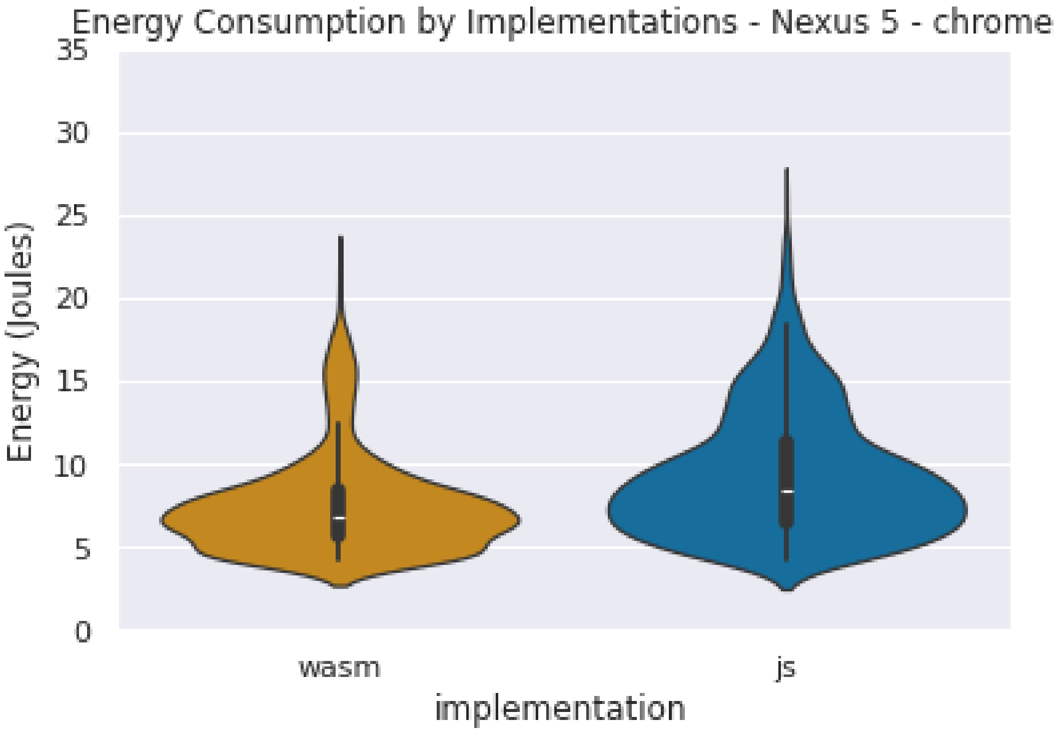
[10]:

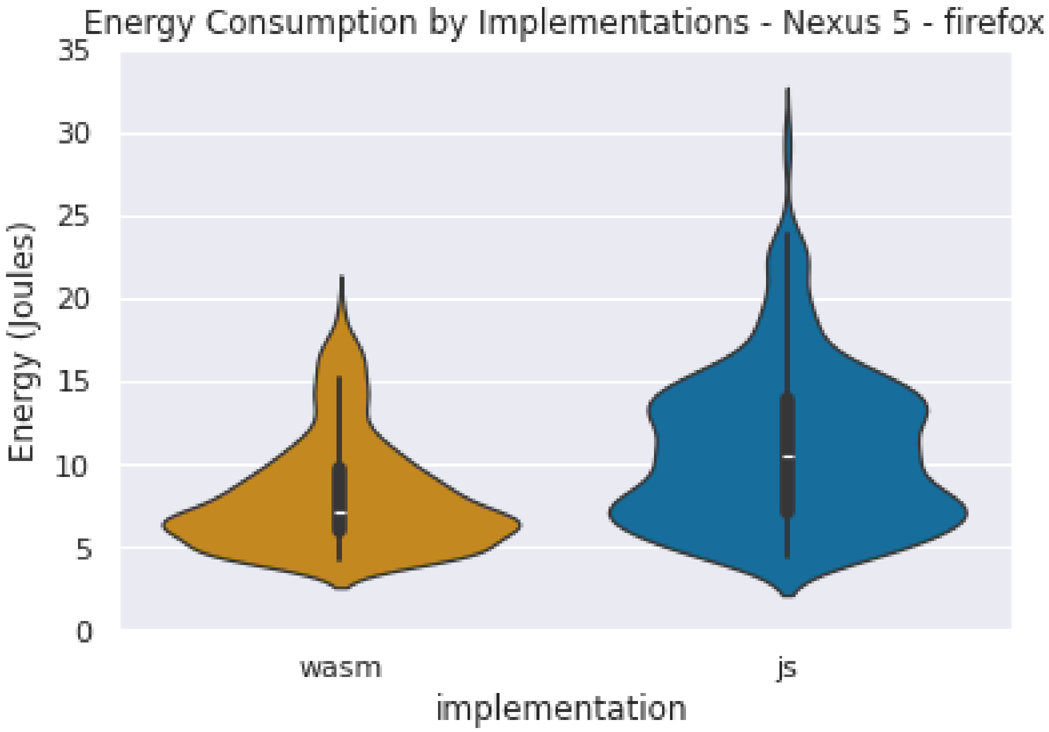
```
data = df
plt.ylim(0, 35)
vp = sns.violinplot(x='implementation', y='energy', hue='implementation',␣
 ↪hue_order=implementations, data=data, palette='colorblind' ,dodge=False)
vp.set_title("Energy Consumption by Implementations - " + device + " - " +␣
 ↪browser)
vp.set_ylabel("Energy (Joules)")
plt.show()
```



4.5 Violinplot (By Browser)

```
[11]: data = []
      for browser in browsers:
          data = df[(df['browser'] == browser)]
          plt.ylim(0, 35)
          vp = sns.violinplot(x='implementation', y='energy', hue='implementation',␣
       ↪hue_order=implementations, data=data, palette='colorblind' ,dodge=False)
          vp.set_title("Energy Consumption by Implementations - " + device + " - " +␣
       ↪browser)
          vp.set_ylabel("Energy (Joules)")
          plt.show()
```

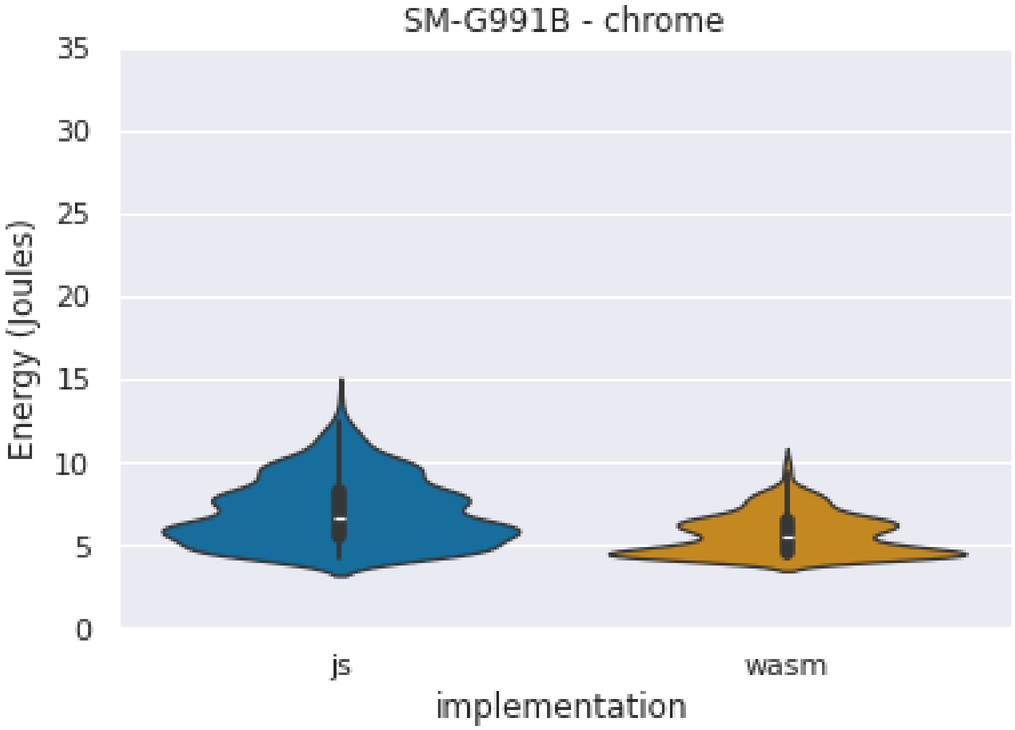


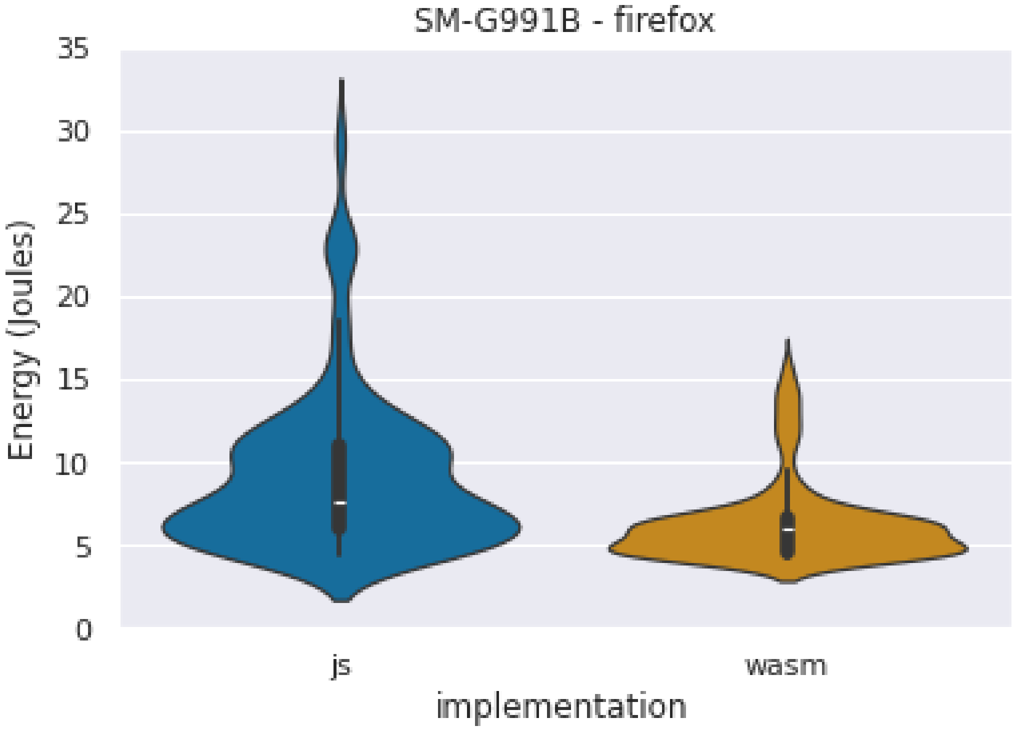


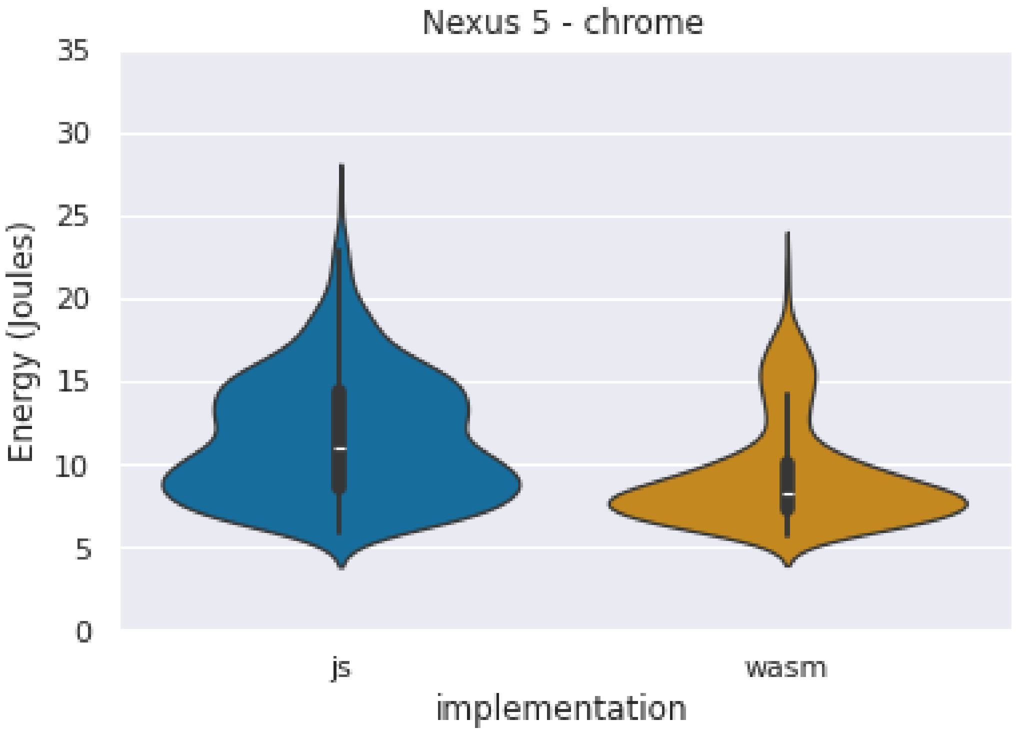
4.6 Violinplot (By Browser & By Device)

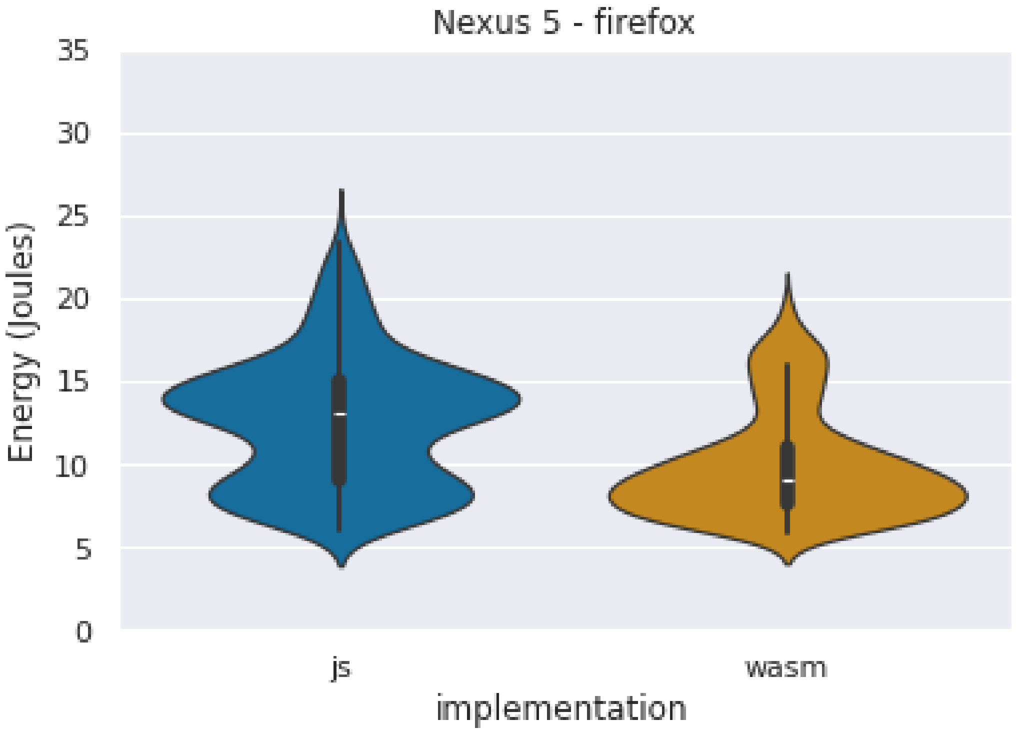
[12]:

```
data = []
for device in devices:
    for browser in browsers:
        data = df[(df['browser'] == browser) & (df['device'] == device)].
 ↪sort_values(by=['implementation'])
        plt.ylim(0, 35)
        vp = sns.violinplot(x='implementation', y='energy',␣
 ↪hue='implementation', hue_order=implementations, data=data,␣
 ↪palette='colorblind', dodge=False)
        vp.set_title(device + " - " + browser)
        vp.set_ylabel("Energy (Joules)")
        plt.show()
```

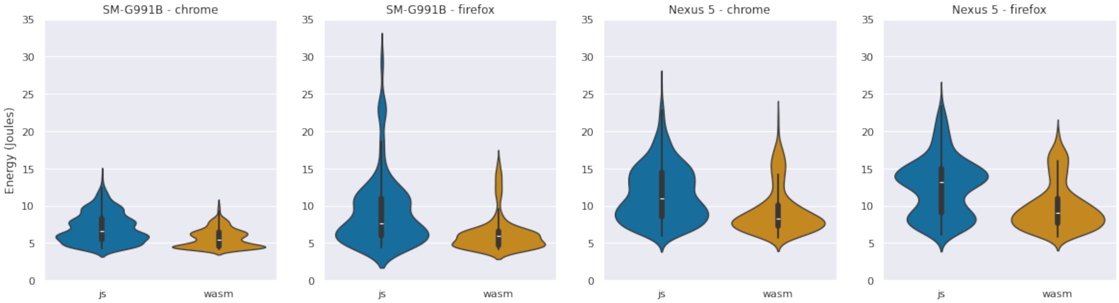

SM-G991B - chrome
Energy (Joules)
35
30
25
20
15
10
5
0
js
wasm
implementation


SM-G991B - firefox
Energy (Joules)
35
30
25
20
15
10
5
0
js
wasm
implementation


Nexus 5 - chrome
35
30
25
20
15
10
5
0
Energy (Joules)
js
wasm
implementation

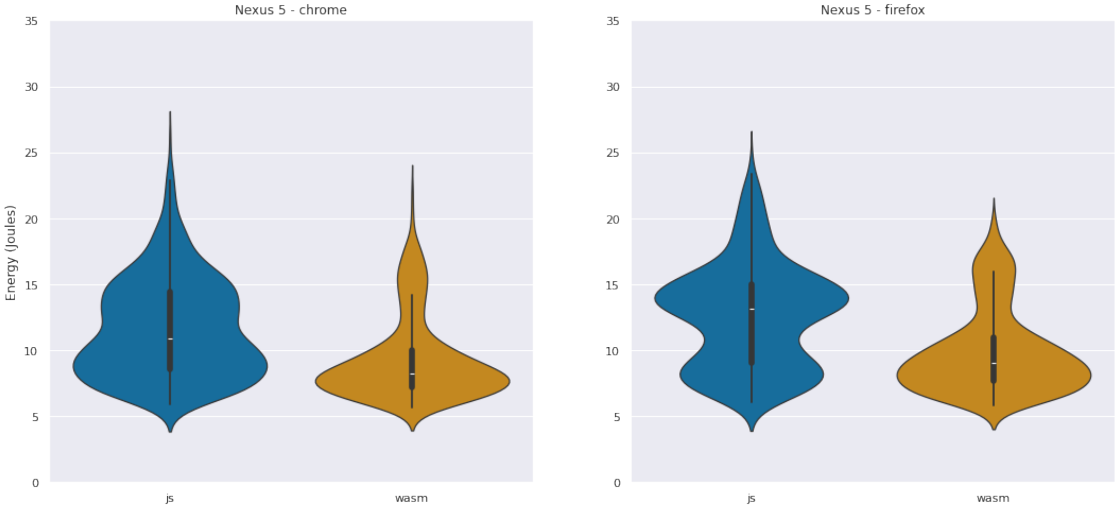

Nexus 5 - firefox
35
30
25
20
15
10
5
0
Energy (Joules)
js
wasm
implementation

[13]:
```python
data = []
index=0
fig, axes = plt.subplots(1, 4, figsize=(20, 5))

for device in devices:
    for browser in browsers:
        data = df[(df['browser'] == browser) & (df['device'] == device)].
 ↪sort_values(by=['implementation'])
        vp = sns.violinplot(x='implementation', y='energy',␣
 ↪hue='implementation', hue_order=implementations, data=data,␣
 ↪palette='colorblind', dodge=False, ax=axes[index])
        vp.set_title(device + " - " + browser)
        axes[index].set_ylim(0, 35)
        axes[index].set_xlabel("")
        if index == 0:
            axes[0].set_ylabel("Energy (Joules)")
        else:
            axes[index].set_ylabel("")
        index+=1
```



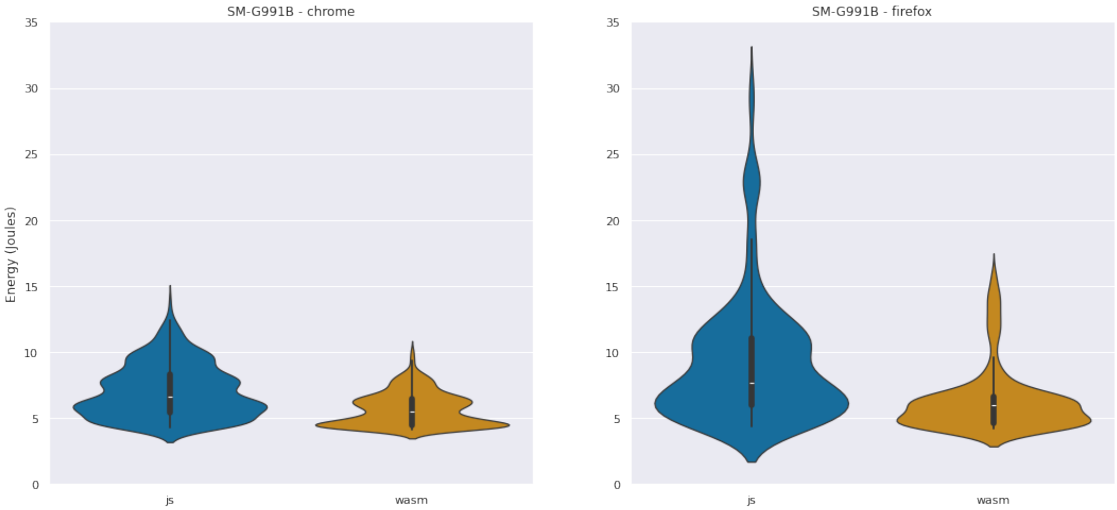
4.7 Violinplot (By Browser & Low End Device)

[14]:
```python
data = []

index=0
fig, axes = plt.subplots(1, 2, figsize=(18, 8))
for browser in browsers:
    data = df[(df['browser'] == browser) & (df['device'] == 'Nexus 5')].
 ↪sort_values(by=['implementation'])
    vp = sns.violinplot(x='implementation', y='energy', hue='implementation',␣
 ↪hue_order=implementations, data=data, palette='colorblind', dodge=False,␣
 ↪ax=axes[index])
    vp.set_title("Nexus 5 - " + browser)
    axes[index].set_ylim(0, 35)
    axes[index].set_xlabel("")
    if index == 0:
        axes[0].set_ylabel("Energy (Joules)")
    else:
        axes[index].set_ylabel("")
    index+=1
```



4.8 Violinplot (By Browser & High End Device)

[15]:
```
data = []

index=0
fig, axes = plt.subplots(1, 2, figsize=(18, 8))
for browser in browsers:
    data = df[(df['browser'] == browser) & (df['device'] == 'SM-G991B')].
 ↪sort_values(by=['implementation'])
    vp = sns.violinplot(x='implementation', y='energy', hue='implementation',␣
 ↪hue_order=implementations, data=data, palette='colorblind', dodge=False,␣
 ↪ax=axes[index])
    vp.set_title("SM-G991B - " + browser)
    axes[index].set_ylim(0, 35)
    axes[index].set_xlabel("")
    if index == 0:
        axes[0].set_ylabel("Energy (Joules)")
    else:
        axes[index].set_ylabel("")
    index+=1
        #plt.show()
```



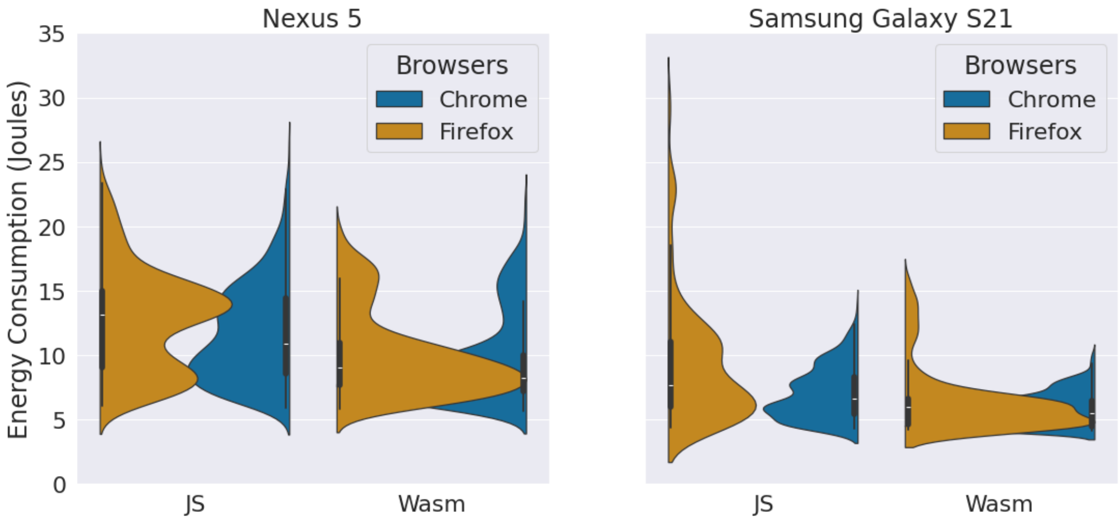
4.9 Violinplot (By Browser & Device Types)

[16]:

```python
## Prepare naming of values for final violinplot
df_violin = df.copy()
df_violin['implementation'] = df_violin['implementation'].replace(['wasm'], 'Wasm')
df_violin['implementation'] = df_violin['implementation'].replace(['js'], 'JS')
df_violin['browser'] = df_violin['browser'].replace(['chrome'], 'Chrome')
df_violin['browser'] = df_violin['browser'].replace(['firefox'], 'Firefox')
df_violin['device'] = df_violin['device'].replace(['SM-G991B'], 'Samsung Galaxy S21')
browsers_violin = numpy.sort(df_violin['browser'].unique())
devices_violin = numpy.sort(df_violin['device'].unique())
```

[17]:

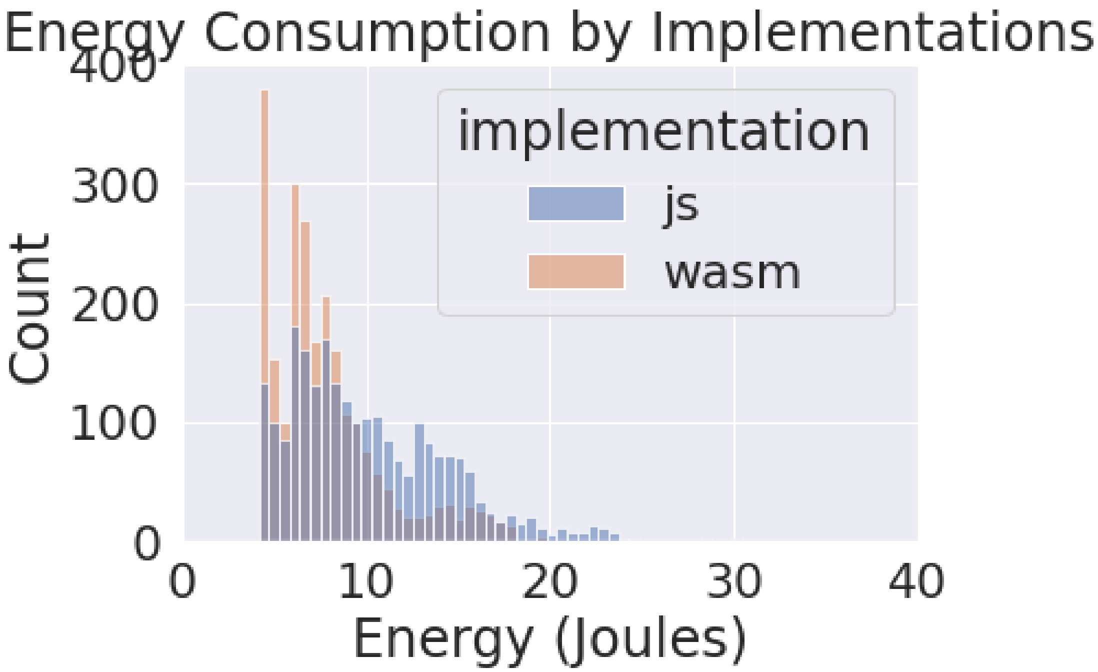
```python
sns.set(font_scale=2)
data = []
index=0
fig, axes = plt.subplots(1, 2, figsize=(18, 8))
for device in devices_violin:
    data = df_violin[(df_violin['device'] == device)].sort_values(by=['implementation'])
    vp = sns.violinplot(x='implementation', y='energy', hue='browser', hue_order=browsers_violin, data=data, palette='colorblind', dodge=False, ax=axes[index], split=True)
    vp.set_title(device)
    axes[index].legend(title="Browsers")
    axes[index].set_ylim(0, 35)
    axes[index].set_xlabel("")
    axes[index].set_ylabel("Energy Consumption (Joules)")
    if index == 0:
        axes[0].set_ylabel("Energy Consumption (Joules)")
    else:
        axes[index].set_ylabel("")
        axes[index].set_yticklabels([])
    index+=1
```



4.10 Histogram

[18]:
```
sns.histplot(data=df, x="energy", hue="implementation",␣
 ↪hue_order=implementations).set_title("Energy Consumption by Implementations")
plt.xlabel("Energy (Joules)")
plt.ylim(0, 400)
plt.xlim(0, 40)
```

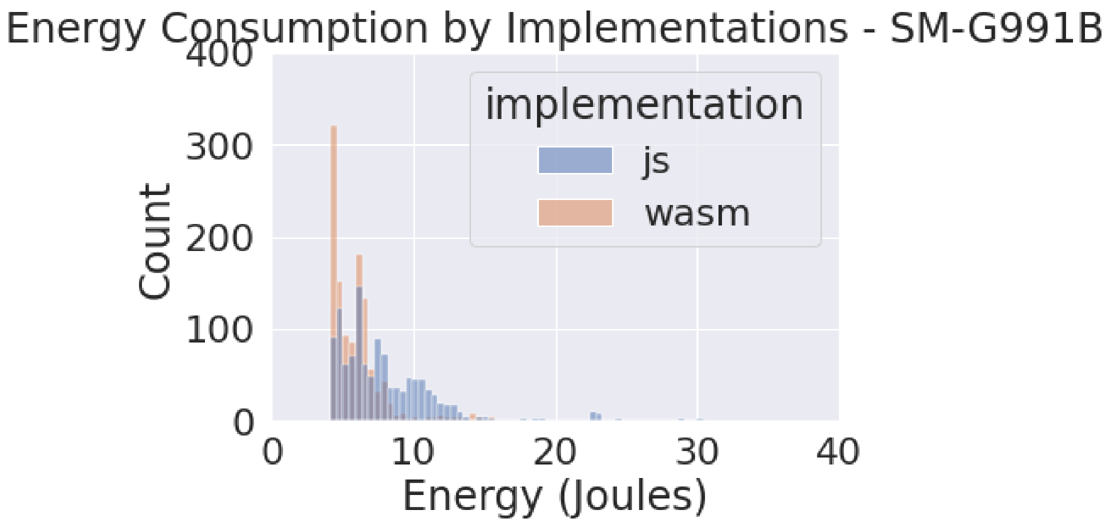
[18]:
```
(0.0, 40.0)
```

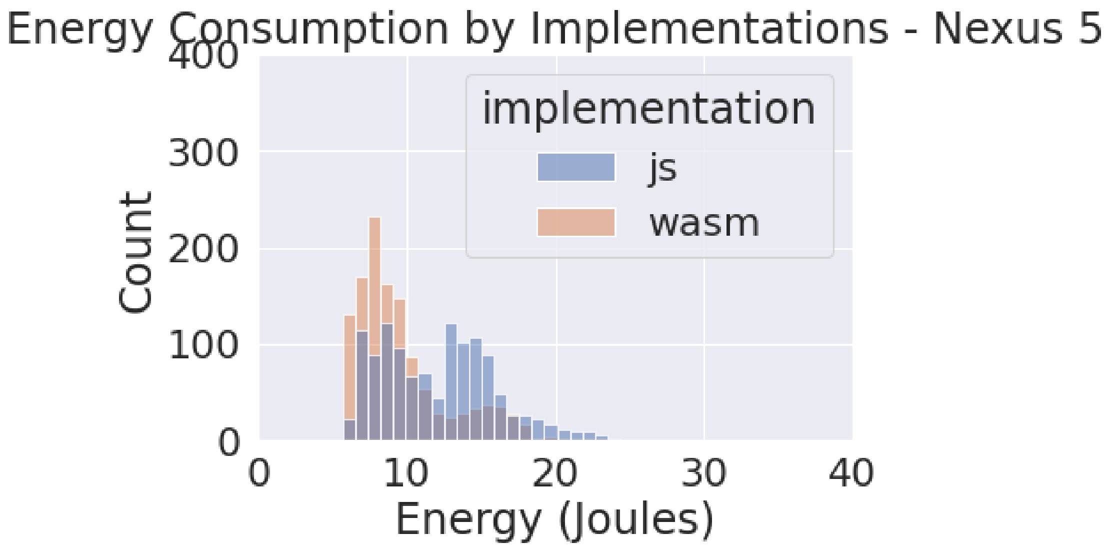


4.11 Histogram (By Device)

[19]:

```
data = []
for device in devices:
    data = df[(df['device'] == device)]
    plt.xlabel("Energy (Joules)")
    plt.ylim(0, 400)
    plt.xlim(0, 40)
    sns.histplot(data=data, x="energy", hue="implementation",␣
 ↪hue_order=implementations).set_title("Energy Consumption by Implementations␣
 ↪- " + device)
    plt.show()
```

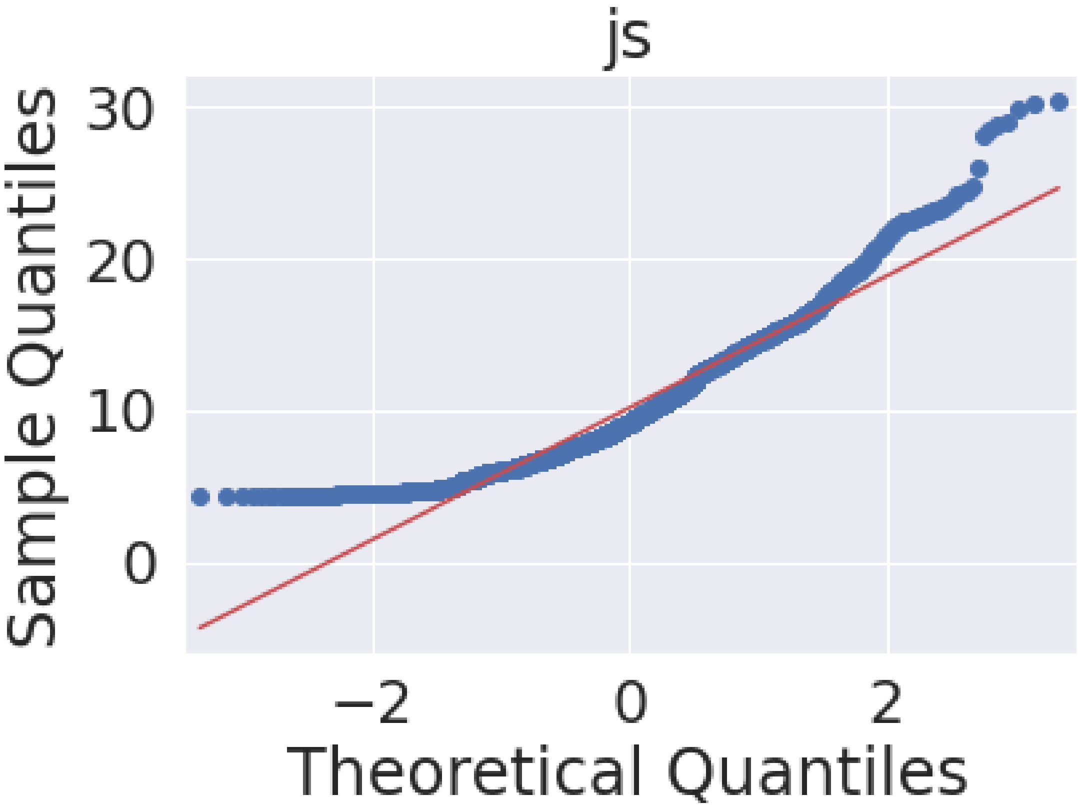


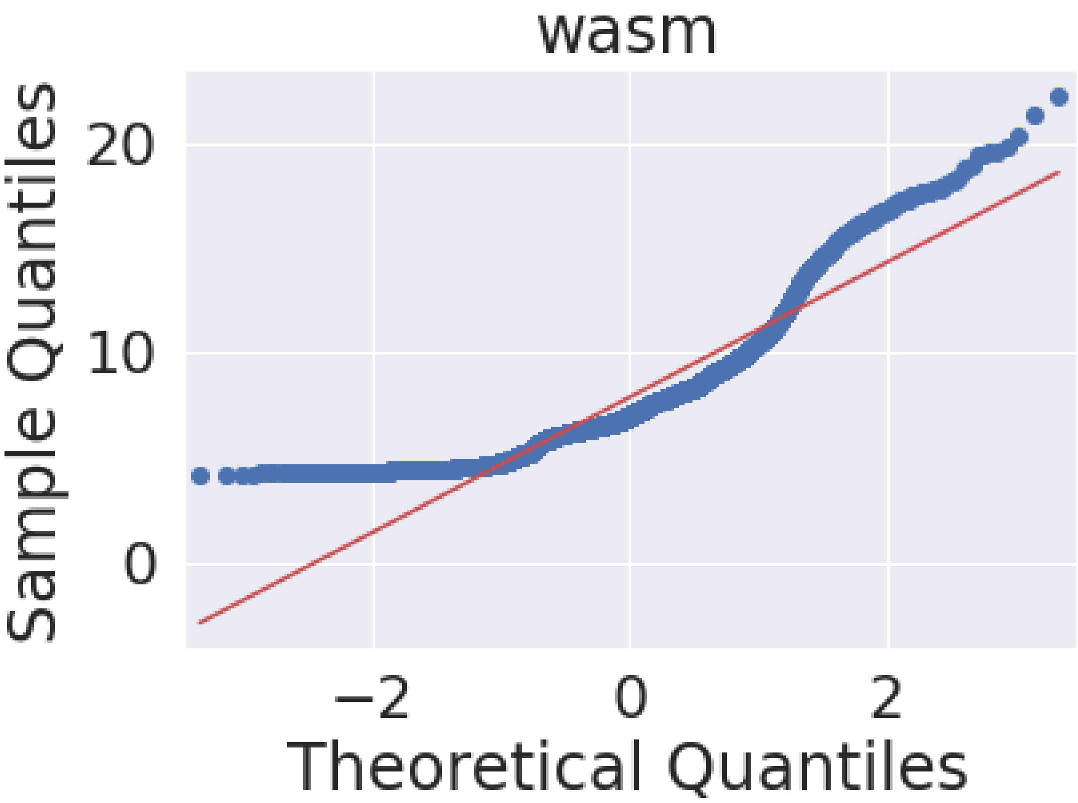


4.12 Q-Q-Plot

[20]:

```
data = []
for implementation in implementations:
    data = df[(df['implementation'] == implementation)]
    qq = sm.qqplot(data.energy, line='s')
    h = plt.title(implementation)
```

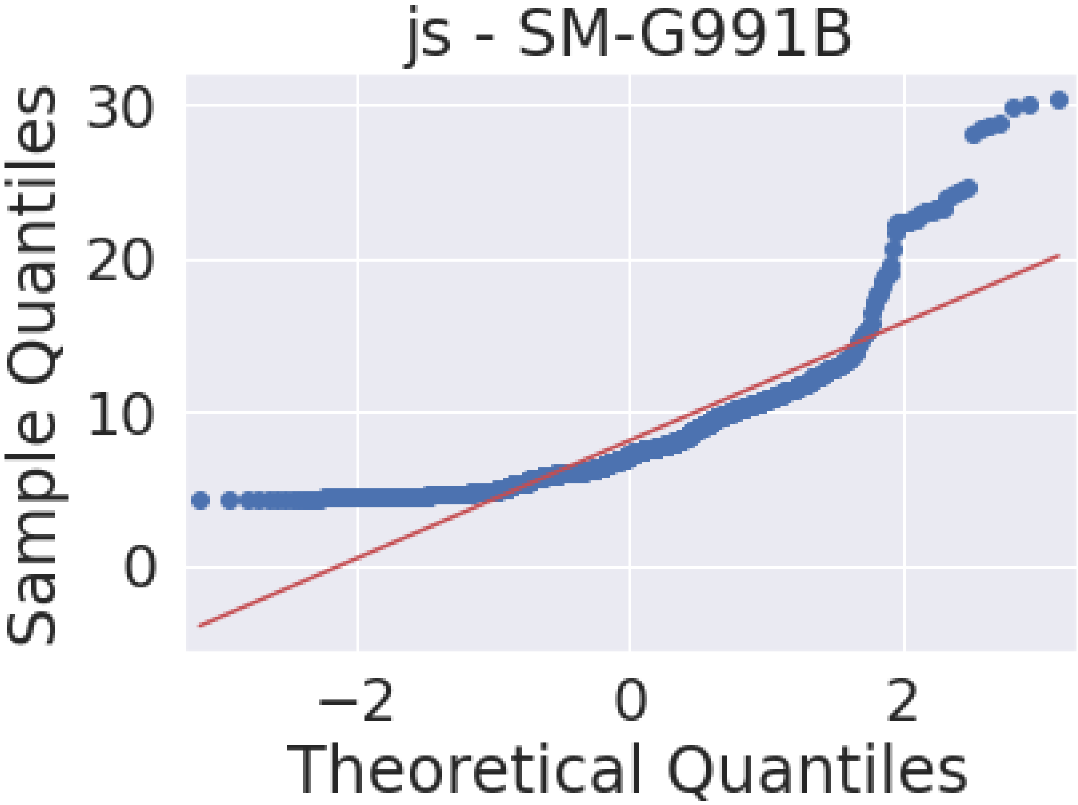

js
Sample Quantiles
30
20
10
0
−2
0
2
Theoretical Quantiles

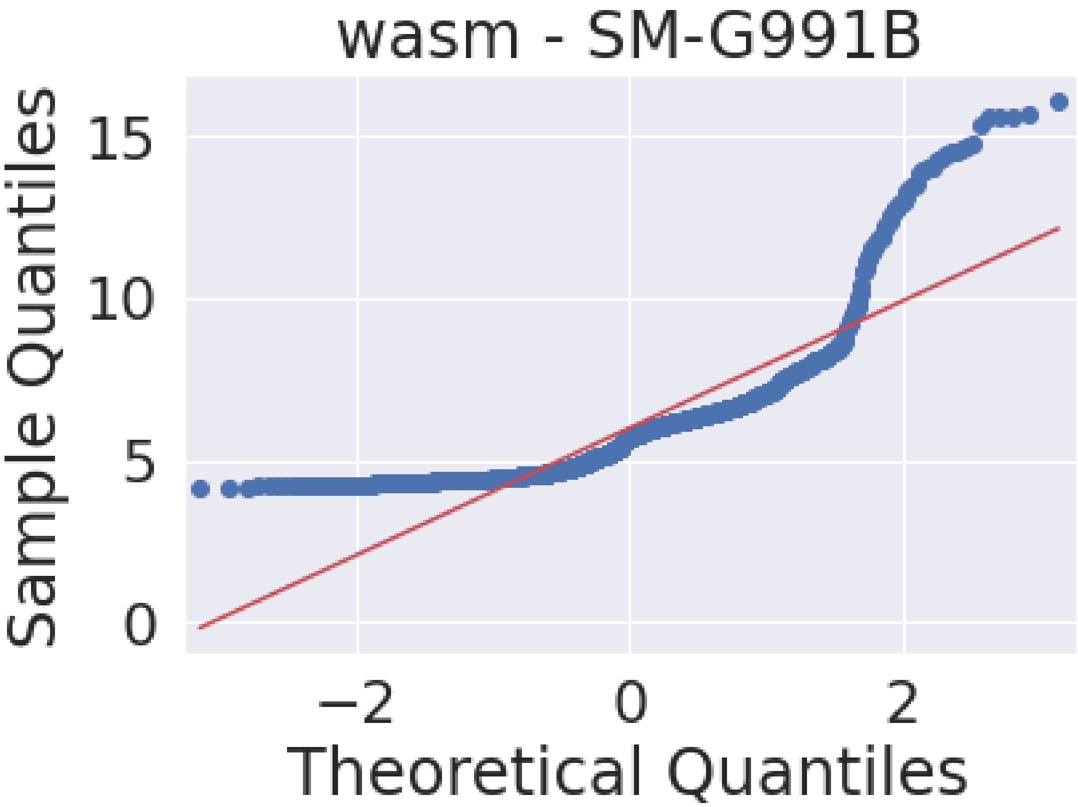


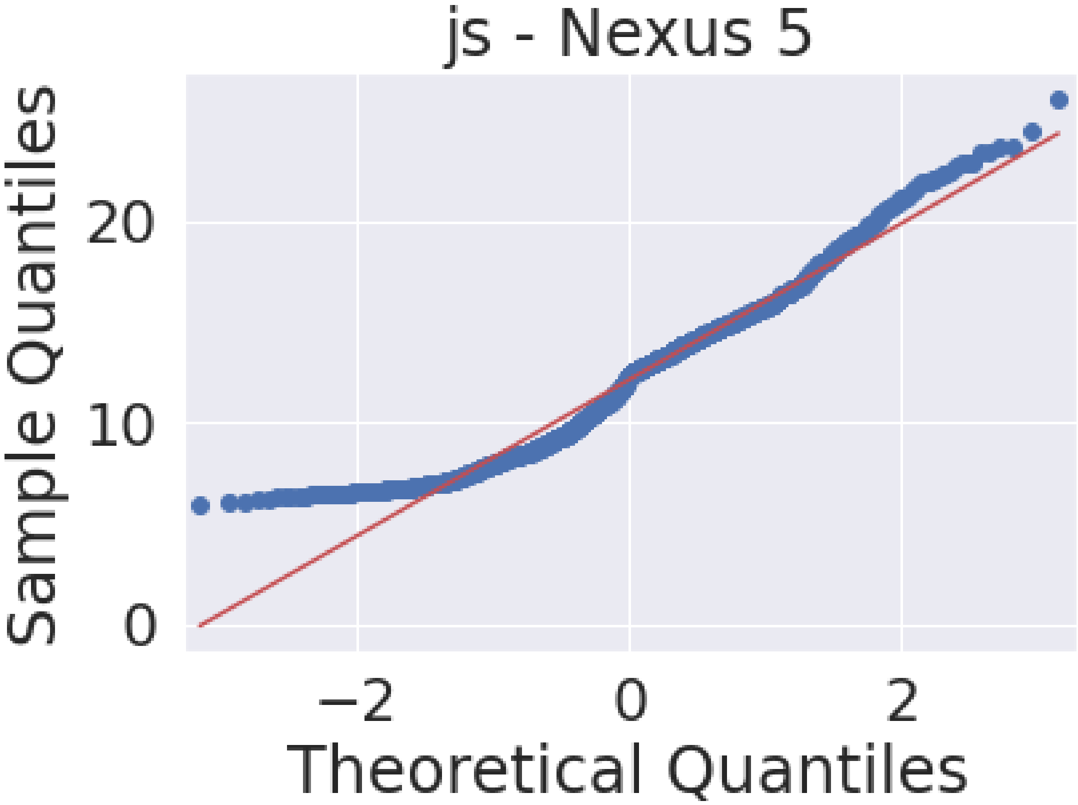
4.13 Q-Q-plot (By Device)

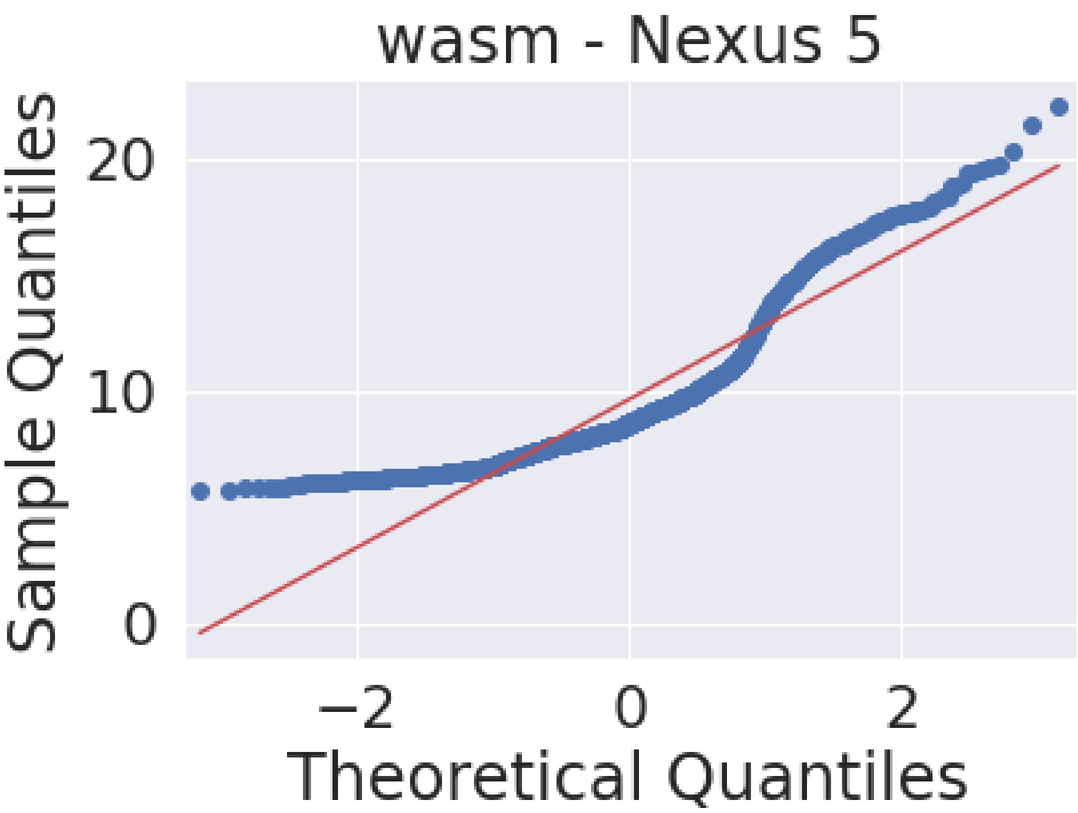
[21]:

```
data = []
for device in devices:
    for implementation in implementations:
        data = df[(df['implementation'] == implementation) & (df['device'] ==␣
 ↪device)]
        qq = sm.qqplot(data.energy, line='s')
        h = plt.title(implementation + " - " + device)
```


js - SM-G991B
Sample Quantiles
30
20
10
0
−2
0
2
Theoretical Quantiles


wasm - SM-G991B
Sample Quantiles
15
10
5
0
−2
0
2
Theoretical Quantiles


js - Nexus 5
Sample Quantiles
20
10
0
−2
0
2
Theoretical Quantiles



4.14 Mann Whitney U Test (same Browsers)

[22]:

```
data = []

for browser in browsers:
    for implementationpair in implementationpairs:
        impl1_energy = df[(df['implementation'] == implementationpair[0]) &␣
 ↪(df['browser'] == browser)]['energy']
        impl2_energy = df[(df['implementation'] == implementationpair[1]) &␣
 ↪(df['browser'] == browser)]['energy']
        eff = cliff.cliffs_delta(impl1_energy, impl2_energy)

        u = stats.mannwhitneyu(impl1_energy, impl2_energy,␣
 ↪alternative='two-sided')

        data.append(
            [
             browser,
             implementationpair[0] + ' vs. ' + implementationpair[1],
             u.statistic,
             u.pvalue,
             eff[0],
             eff[1]
            ]
        )

# Create the pandas DataFrame
ut = pd.DataFrame(data, columns = ['browser', 'implementation', 'u', 'p',␣
 ↪'eff', 'interp'])
display(ut)

interp = ut['interp'].value_counts()
interp = pd.DataFrame(interp, columns = ['interp', 'percent'])
interp['percent'] = (interp['interp'] / interp['interp'].sum()) * 100

display(interp)
```

```
   browser implementation         u             p       eff  interp
0   chrome    js vs. wasm  960697.0  1.168028e-41  0.317780   small
1  firefox    js vs. wasm  998597.0  5.082446e-57  0.374314  medium
```

```
Empty DataFrame
Columns: [interp, percent]
Index: []
```

4.15 Mann Whitney U Test (same Browsers - by Device)

[23]:
```
data = []

for device in devices:
    for browser in browsers:
        for implementationpair in implementationpairs:
            impl1_energy = df[(df['implementation'] == implementationpair[0]) &␣
 ↪(df['browser'] == browser) & (df['device'] == device)]['energy']
            impl2_energy = df[(df['implementation'] == implementationpair[1]) &␣
 ↪(df['browser'] == browser) & (df['device'] == device)]['energy']
            eff = cliff.cliffs_delta(impl1_energy, impl2_energy)

            u = stats.mannwhitneyu(impl1_energy, impl2_energy,␣
 ↪alternative='two-sided')

            data.append(
                [
                 device,
                 browser,
                 implementationpair[0] + ' vs. ' + implementationpair[1],
                 u.statistic,
                 u.pvalue,
                 eff[0],
                 eff[1]
                ]
            )

# Create the pandas DataFrame
ut = pd.DataFrame(data, columns = ['device', 'browser', 'implementation', 'u',␣
 ↪'p', 'eff', 'interp'])
display(ut)

interp = ut['interp'].value_counts()
interp = pd.DataFrame(interp, columns = ['interp', 'percent'])
interp['percent'] = (interp['interp'] / interp['interp'].sum()) * 100

display(interp)
```

```
     device  browser implementation         u             p       eff  interp
0  SM-G991B   chrome    js vs. wasm  250266.0  1.591990e-36  0.423401  medium
1  SM-G991B  firefox    js vs. wasm  256565.0  1.160505e-42  0.459227  medium
2   Nexus 5   chrome    js vs. wasm  263410.0  3.872448e-33  0.395187  medium
3   Nexus 5  firefox    js vs. wasm  261860.0  3.617754e-33  0.396013  medium
```

```
Empty DataFrame
Columns: [interp, percent]
Index: []
```

4.16 Mann Whitney U Test (Cross Browsers)

[24]:
```
data = []

for pairswitch in [[0,1],[1,0]]:
    for implementationpair in implementationpairs:
        for browserpair in browserpairs:
            browser1_energy = df[(df['browser'] == browserpair[0]) &␣
 ↪(df['implementation'] == implementationpair[pairswitch[0]])]['energy']
            browser2_energy = df[(df['browser'] == browserpair[1]) &␣
 ↪(df['implementation'] == implementationpair[pairswitch[1]])]['energy']
            eff = cliff.cliffs_delta(browser1_energy, browser2_energy)

            u = stats.mannwhitneyu(browser1_energy, browser2_energy,␣
 ↪alternative='two-sided')

            data.append(
                [
                 browserpair[0] + ' vs. ' + browserpair[1],
                 implementationpair[pairswitch[0]] + ' vs. ' +␣
 ↪implementationpair[pairswitch[1]],
                 u.statistic,
                 u.pvalue,
                 eff[0],
                 eff[1]
                ]
            )

# Create the pandas DataFrame
ut = pd.DataFrame(data, columns = ['browser', 'implementation', 'u', 'p',␣
 ↪'eff', 'interp'])
display(ut)

interp = ut['interp'].value_counts()
interp = pd.DataFrame(interp, columns = ['interp', 'percent'])
interp['percent'] = (interp['interp'] / interp['interp'].sum()) * 100

display(interp)
```

```
              browser implementation         u             p       eff  interp
0  chrome vs. firefox    js vs. wasm  880131.0  3.841190e-19  0.210271   small
1  chrome vs. firefox    wasm vs. js  391450.0  3.216940e-86 -0.462606  medium
```

```
Empty DataFrame
Columns: [interp, percent]
Index: []
```

4.17 Mann Whitney U Test (Cross Browsers - By Device)

[25]:

```
data = []

for device in devices:
    for pairswitch in [[0,1],[1,0]]:
        for implementationpair in implementationpairs:
            for browserpair in browserpairs:
                browser1_energy = df[(df['browser'] == browserpair[0]) &␣
 ↪(df['implementation'] == implementationpair[pairswitch[0]]) & (df['device']␣
 ↪== device)]['energy']
                browser2_energy = df[(df['browser'] == browserpair[1]) &␣
 ↪(df['implementation'] == implementationpair[pairswitch[1]]) & (df['device']␣
 ↪== device)]['energy']
                eff = cliff.cliffs_delta(browser1_energy, browser2_energy)

                u = stats.mannwhitneyu(browser1_energy, browser2_energy,␣
 ↪alternative='two-sided')
```

```
                data.append(
                    [
                     device,
                     browserpair[0] + ' vs. ' + browserpair[1],
                     implementationpair[pairswitch[0]] + ' vs. ' +␣
 ↪implementationpair[pairswitch[1]],
                     u.statistic,
                     u.pvalue,
                     eff[0],
                     eff[1]
                    ]
                )

# Create the pandas DataFrame
ut = pd.DataFrame(data, columns = ['device', 'browser', 'implementation', 'u',␣
 ↪'p', 'eff', 'interp'])
display(ut)

interp = ut['interp'].value_counts()
interp = pd.DataFrame(interp, columns = ['interp', 'percent'])
interp['percent'] = (interp['interp'] / interp['interp'].sum()) * 100

display(interp)
```

```
     device             browser implementation         u             p  \
0  SM-G991B  chrome vs. firefox    js vs. wasm  227749.0  1.136373e-17
1  SM-G991B  chrome vs. firefox    wasm vs. js   74651.0  4.165707e-65
2   Nexus 5  chrome vs. firefox    js vs. wasm  239728.0  7.751432e-18
3   Nexus 5  chrome vs. firefox    wasm vs. js   97397.0  1.789286e-49

        eff interp
0  0.286627  small
1 -0.572545  large
2  0.284300  small
3 -0.486644  large
```

```
Empty DataFrame
Columns: [interp, percent]
Index: []
```

4.18 Descriptive Statistics

[26]:

```
data = []
for implementation in implementations:
    x = df[(df['implementation'] == implementation)]
    mean = numpy.round(numpy.mean(x['energy']), 2)
    median = numpy.round(numpy.median(x['energy']), 2)
    min = numpy.round(numpy.amin(x['energy']), 2)
    max = numpy.round(numpy.amax(x['energy']), 2)
    std = numpy.round(numpy.std(x['energy']), 2)
    sem = numpy.round(stats.sem(x['energy']), 2)
    q1 = numpy.round(numpy.quantile(x['energy'], 0.25), 2)
    q3 = numpy.round(numpy.quantile(x['energy'], 0.75), 2)

    data.append(
        [implementation, mean, std, min, q1, median, q3, max, sem]
    )

# Create the pandas DataFrame
stat = pd.DataFrame(data, columns = ['implementation', 'mean', 'std', 'min',␣
 ↪'q1', 'median', 'q3', 'max', 'sem'])
# display(stat)
print(stat.to_string())

# Alternative of pandas: x['energy'].describe()
```

```
  implementation   mean  std  min    q1 median     q3    max  sem
0             js  10.16  4.34  4.30  6.77   9.19  12.96  30.46  0.09
1           wasm   7.84  3.22  4.16  5.75   6.97   9.04  22.29  0.07
```

4.19 Descriptive Statistics Difference

[27]:

```
data = []

for implementationpair in implementationpairs:
    implementation1 = stat[(stat['implementation'] == implementationpair[1])]
    implementation2 = stat[(stat['implementation'] == implementationpair[0])]

    mean_diff = implementation1.iloc[0]['mean']-implementation2.iloc[0]['mean']
    median_diff = implementation1.iloc[0]['median']-implementation2.
 ↪iloc[0]['median']
    min_diff = implementation1.iloc[0]['min']-implementation2.iloc[0]['min']
    max_diff = implementation1.iloc[0]['max']-implementation2.iloc[0]['max']
    std_diff = implementation1.iloc[0]['std']-implementation2.iloc[0]['std']
    sem_diff = implementation1.iloc[0]['sem']-implementation2.iloc[0]['sem']
    q1_diff = implementation1.iloc[0]['q1']-implementation2.iloc[0]['q1']
    q3_diff = implementation1.iloc[0]['q3']-implementation2.iloc[0]['q3']

    data.append(
        [implementationpair[1] + ' vs. ' + implementationpair[0],
         numpy.round(mean_diff, 2),
         numpy.round(mean_diff/implementation2.iloc[0]['mean']*100, 2),
         numpy.round(median_diff, 2),
         numpy.round(median_diff/implementation2.iloc[0]['median']*100, 2),
         numpy.round(min_diff, 2),
         numpy.round(min_diff/implementation2.iloc[0]['min']*100, 2),
         numpy.round(max_diff, 2),
         numpy.round(max_diff/implementation2.iloc[0]['max']*100, 2),
         numpy.round(std_diff, 2),
         numpy.round(std_diff/implementation2.iloc[0]['std']*100, 2),
         numpy.round(sem_diff, 2),
         numpy.round(sem_diff/implementation2.iloc[0]['sem']*100, 2),
         numpy.round(q1_diff, 2),
         numpy.round(q1_diff/implementation2.iloc[0]['q1']*100, 2),
         numpy.round(q3_diff, 2),
         numpy.round(q3_diff/implementation2.iloc[0]['q3']*100, 2),
        ]
    )

# Create the pandas DataFrame
ut = pd.DataFrame(data, columns =␣
 ↪['rq','mean_diff','mean_diff%','median_diff','median_diff%','min_diff','min_diff%','max_dif
display(ut)
#print(ut.to_string())
```

```
           rq  mean_diff  mean_diff%  median_diff  median_diff%  min_diff  \
0  wasm vs. js      -2.32      -22.83        -2.22        -24.16     -0.14

   min_diff%  max_diff  max_diff%  std_diff  std_diff%  sem_diff  sem_diff%  \
0      -3.26     -8.17     -26.82     -1.12     -25.81     -0.02     -22.22

   q1_diff  q1_diff%  q3_diff  q3_diff%
0    -1.02    -15.07    -3.92    -30.25
```

4.20 Descriptive Statistics (By Browser)

[28]:
```
data = []
for implementation in implementations:
    for browser in browsers:
        x = df[(df['implementation'] == implementation) & (df['browser'] ==␣
 ↪browser)]
        mean = numpy.round(numpy.mean(x['energy']), 2)
        median = numpy.round(numpy.median(x['energy']), 2)
        min = numpy.round(numpy.amin(x['energy']), 2)
        max = numpy.round(numpy.amax(x['energy']), 2)
        std = numpy.round(numpy.std(x['energy']), 2)
        sem = numpy.round(stats.sem(x['energy']), 2)
        q1 = numpy.round(numpy.quantile(x['energy'], 0.25), 2)
        q3 = numpy.round(numpy.quantile(x['energy'], 0.75), 2)

        data.append(
            [implementation, browser, mean, std, min, q1, median,  q3, max, sem]
        )

# Create the pandas DataFrame
stat = pd.DataFrame(data, columns = ['implementation', 'browser', 'mean',␣
 ↪'std', 'min', 'q1', 'median', 'q3', 'max', 'sem'])
# display(stat)
print(stat.to_string())

# Alternative of pandas: x['energy'].describe()
```

```
  implementation  browser   mean   std   min    q1  median     q3    max   sem
0             js   chrome   9.37  3.79  4.30  6.52    8.47  11.28  26.04  0.11
1             js  firefox  10.94  4.70  4.39  7.11   10.43  13.89  30.46  0.14
2           wasm   chrome   7.55  3.04  4.16  5.64    6.85   8.37  22.29  0.09
3           wasm  firefox   8.14  3.36  4.22  5.96    7.09   9.67  19.78  0.10
```

4.21 Descriptive Statistics Difference (By Browser)

[29]:

```
data = []

for implementationpair in implementationpairs:
    for browser in browsers:
        implementation1 = stat[(stat['implementation'] ==␣
 ↪implementationpair[1]) & (stat['browser'] == browser)]
        implementation2 = stat[(stat['implementation'] ==␣
 ↪implementationpair[0]) & (stat['browser'] == browser)]

        mean_diff = implementation1.iloc[0]['mean']-implementation2.
 ↪iloc[0]['mean']
        median_diff = implementation1.iloc[0]['median']-implementation2.
 ↪iloc[0]['median']
        min_diff = implementation1.iloc[0]['min']-implementation2.iloc[0]['min']
        max_diff = implementation1.iloc[0]['max']-implementation2.iloc[0]['max']
        std_diff = implementation1.iloc[0]['std']-implementation2.iloc[0]['std']
        sem_diff = implementation1.iloc[0]['sem']-implementation2.iloc[0]['sem']
        q1_diff = implementation1.iloc[0]['q1']-implementation2.iloc[0]['q1']
        q3_diff = implementation1.iloc[0]['q3']-implementation2.iloc[0]['q3']

        data.append(
            [implementationpair[1] + ' vs. ' + implementationpair[0] + ' ' +␣
 ↪browser,
             numpy.round(mean_diff, 2),
             numpy.round(mean_diff/implementation2.iloc[0]['mean']*100, 2),
             numpy.round(median_diff, 2),
             numpy.round(median_diff/implementation2.iloc[0]['median']*100, 2),
             numpy.round(min_diff, 2),
                numpy.round(min_diff/implementation2.iloc[0]['min']*100, 2),
                numpy.round(max_diff, 2),
                numpy.round(max_diff/implementation2.iloc[0]['max']*100, 2),
                numpy.round(std_diff, 2),
                numpy.round(std_diff/implementation2.iloc[0]['std']*100, 2),
                numpy.round(sem_diff, 2),
                numpy.round(sem_diff/implementation2.iloc[0]['sem']*100, 2),
                numpy.round(q1_diff, 2),
                numpy.round(q1_diff/implementation2.iloc[0]['q1']*100, 2),
                numpy.round(q3_diff, 2),
                numpy.round(q3_diff/implementation2.iloc[0]['q3']*100, 2),
            ]
        )

# Create the pandas DataFrame
ut = pd.DataFrame(data, columns =␣
 ↪['rq','mean_diff','mean_diff%','median_diff','median_diff%','min_diff','min_diff%','max_dif
display(ut)
#print(ut.to_string())
```

```
                   rq  mean_diff  mean_diff%  median_diff  median_diff%  \
0   wasm vs. js chrome      -1.82      -19.42        -1.62         -19.13
1  wasm vs. js firefox      -2.80      -25.59        -3.34         -32.02

   min_diff  min_diff%  max_diff  max_diff%  std_diff  std_diff%  sem_diff  \
0     -0.14      -3.26     -3.75     -14.40     -0.75     -19.79     -0.02
1     -0.17      -3.87    -10.68     -35.06     -1.34     -28.51     -0.04

   sem_diff%  q1_diff  q1_diff%  q3_diff  q3_diff%
0     -18.18    -0.88    -13.50    -2.91    -25.80
1     -28.57    -1.15    -16.17    -4.22    -30.38
```

4.22 Descriptive Statistics Difference (Cross Browser)

```
[30]: data = []

for pairswitch in [[0,1],[1,0]]:
    for implementationpair in implementationpairs:
        for browserpair in browserpairs:
            implementation1 = stat[(stat['browser'] ==␣
 ↪browserpair[pairswitch[1]]) & (stat['implementation'] ==␣
 ↪implementationpair[1])]
            implementation2 = stat[(stat['browser'] ==␣
 ↪browserpair[pairswitch[0]]) & (stat['implementation'] ==␣
 ↪implementationpair[0])]

#for implementationpair in implementationpairs:
```

```
#    for browser in browsers:
#        implementation1 = stat[(stat['implementation'] ==␣
 ↪implementationpair[0]) & (stat['browser'] == browser)]
#        implementation2 = stat[(stat['implementation'] ==␣
 ↪implementationpair[1]) & (stat['browser'] == browser)]

        mean_diff = implementation1.iloc[0]['mean']-implementation2.
 ↪iloc[0]['mean']
        median_diff = implementation1.iloc[0]['median']-implementation2.
 ↪iloc[0]['median']
        min_diff = implementation1.iloc[0]['min']-implementation2.iloc[0]['min']
        max_diff = implementation1.iloc[0]['max']-implementation2.iloc[0]['max']
        std_diff = implementation1.iloc[0]['std']-implementation2.iloc[0]['std']
        sem_diff = implementation1.iloc[0]['sem']-implementation2.iloc[0]['sem']
        q1_diff = implementation1.iloc[0]['q1']-implementation2.iloc[0]['q1']
        q3_diff = implementation1.iloc[0]['q3']-implementation2.iloc[0]['q3']

        data.append(
            [
             browserpair[pairswitch[1]] + ' ' + implementationpair[1] + ' vs. '␣
 ↪+ browserpair[pairswitch[0]] + ' ' + implementationpair[0],
             numpy.round(mean_diff, 2),
             numpy.round(mean_diff/implementation2.iloc[0]['mean']*100, 2),
             numpy.round(median_diff, 2),
             numpy.round(median_diff/implementation2.iloc[0]['median']*100, 2),
             numpy.round(min_diff, 2),
             numpy.round(min_diff/implementation2.iloc[0]['min']*100, 2),
             numpy.round(max_diff, 2),
             numpy.round(max_diff/implementation2.iloc[0]['max']*100, 2),
             numpy.round(std_diff, 2),
             numpy.round(std_diff/implementation2.iloc[0]['std']*100, 2),
             numpy.round(sem_diff, 2),
             numpy.round(sem_diff/implementation2.iloc[0]['sem']*100, 2),
             numpy.round(q1_diff, 2),
             numpy.round(q1_diff/implementation2.iloc[0]['q1']*100, 2),
             numpy.round(q3_diff, 2),
             numpy.round(q3_diff/implementation2.iloc[0]['q3']*100, 2),
            ]
        )

# Create the pandas DataFrame
ut = pd.DataFrame(data, columns =␣
 ↪['rq','mean_diff','mean_diff%','median_diff','median_diff%','min_diff','min_diff%','max_dif
display(ut)
#print(ut.to_string())
```

```
                          rq  mean_diff  mean_diff%  median_diff  \
0  firefox wasm vs. chrome js      -1.23      -13.13         -1.38
1  chrome wasm vs. firefox js      -3.39      -30.99         -3.58

   median_diff%  min_diff  min_diff%  max_diff  max_diff%  std_diff  \
0        -16.29     -0.08      -1.86     -6.26     -24.04     -0.43
1        -34.32     -0.23      -5.24     -8.17     -26.82     -1.66

   std_diff%  sem_diff  sem_diff%  q1_diff  q1_diff%  q3_diff  q3_diff%
0     -11.35     -0.01      -9.09    -0.56     -8.59    -1.61    -14.27
1     -35.32     -0.05     -35.71    -1.47    -20.68    -5.52    -39.74
```

4.23 Descriptive Statistics (By Browser & By Device)

[31]:

```
data = []
for device in devices:
    for implementation in implementations:
        for browser in browsers:
            x = df[(df['implementation'] == implementation) & (df['browser'] ==␣
↪browser) & (df['device'] == device)]
            mean = numpy.round(numpy.mean(x['energy']), 2)
            median = numpy.round(numpy.median(x['energy']), 2)
            min = numpy.round(numpy.amin(x['energy']), 2)
            max = numpy.round(numpy.amax(x['energy']), 2)
            std = numpy.round(numpy.std(x['energy']), 2)
            sem = numpy.round(stats.sem(x['energy']), 2)
            q1 = numpy.round(numpy.quantile(x['energy'], 0.25), 2)
            q3 = numpy.round(numpy.quantile(x['energy'], 0.75), 2)

            data.append(
                [implementation, device, browser, mean, std, min, q1, median,␣
↪q3, max, sem]
            )

# Create the pandas DataFrame
stat = pd.DataFrame(data, columns = ['implementation', 'device', 'browser',␣
↪'mean', 'std', 'min', 'q1', 'median', 'q3', 'max', 'sem'])
# display(stat)
print(stat.to_string())

# Alternative of pandas: x['energy'].describe()
```

```
  implementation    device  browser   mean   std   min    q1  median     q3
max   sem
0             js  SM-G991B   chrome   7.05  2.00  4.30  5.46    6.62   8.29
13.96  0.08
1             js  SM-G991B  firefox   9.21  4.82  4.39  5.99    7.67  11.03
30.46  0.20
```

```
2           wasm  SM-G991B   chrome   5.67  1.25  4.16  4.49   5.47   6.46
10.14  0.05
3           wasm  SM-G991B  firefox   6.32  2.43  4.22  4.63   5.96   6.62
16.11  0.10
4             js   Nexus 5   chrome  11.64  3.75  5.92  8.53  10.93  14.44
26.04  0.15
5             js   Nexus 5  firefox  12.61  3.91  6.06  9.03  13.13  15.01
24.44  0.16
6           wasm   Nexus 5   chrome   9.34  3.16  5.67  7.23   8.24  10.02
22.29  0.13
7           wasm   Nexus 5  firefox   9.91  3.19  5.79  7.63   9.04  10.99
19.78  0.13
```

4.24 Descriptive Statistics Difference (By Browser & By Device)

[32]:
```
data = []

for implementationpair in implementationpairs:
    for device in devices:
        for browser in browsers:
            implementation1 = stat[(stat['implementation'] ==␣
 ↪implementationpair[1]) & (stat['browser'] == browser) & (stat['device'] ==␣
 ↪device)]
            implementation2 = stat[(stat['implementation'] ==␣
 ↪implementationpair[0]) & (stat['browser'] == browser) & (stat['device'] ==␣
 ↪device)]

            mean_diff = implementation1.iloc[0]['mean']-implementation2.
 ↪iloc[0]['mean']
            median_diff = implementation1.iloc[0]['median']-implementation2.
 ↪iloc[0]['median']
            min_diff = implementation1.iloc[0]['min']-implementation2.
 ↪iloc[0]['min']
            max_diff = implementation1.iloc[0]['max']-implementation2.
 ↪iloc[0]['max']
            std_diff = implementation1.iloc[0]['std']-implementation2.
 ↪iloc[0]['std']
            sem_diff = implementation1.iloc[0]['sem']-implementation2.
 ↪iloc[0]['sem']
            q1_diff = implementation1.iloc[0]['q1']-implementation2.
 ↪iloc[0]['q1']
            q3_diff = implementation1.iloc[0]['q3']-implementation2.
 ↪iloc[0]['q3']

            data.append(
                [implementationpair[1] + ' vs. ' + implementationpair[0] + ' '␣
 ↪+ browser,
                 device,
                 numpy.round(mean_diff, 2),
                 numpy.round(mean_diff/implementation2.iloc[0]['mean']*100, 2),
                 numpy.round(median_diff, 2),
                 numpy.round(median_diff/implementation2.iloc[0]['median']*100,␣
 ↪2),
                 numpy.round(min_diff, 2),
                 numpy.round(min_diff/implementation2.iloc[0]['min']*100, 2),
                 numpy.round(max_diff, 2),
                 numpy.round(max_diff/implementation2.iloc[0]['max']*100, 2),
                 numpy.round(std_diff, 2),
                 numpy.round(std_diff/implementation2.iloc[0]['std']*100, 2),
                 numpy.round(sem_diff, 2),
                 numpy.round(sem_diff/implementation2.iloc[0]['sem']*100, 2),
                 numpy.round(q1_diff, 2),
                 numpy.round(q1_diff/implementation2.iloc[0]['q1']*100, 2),
                 numpy.round(q3_diff, 2),
                 numpy.round(q3_diff/implementation2.iloc[0]['q3']*100, 2),
                ]
            )

# Create the pandas DataFrame
ut = pd.DataFrame(data, columns =␣
 ↪['rq','device','mean_diff','mean_diff%','median_diff','median_diff%','min_diff','min_diff%'
display(ut)
#print(ut.to_string())
```

```
                   rq   device  mean_diff  mean_diff%  median_diff  \
0   wasm vs. js chrome  SM-G991B      -1.38      -19.57        -1.15
1  wasm vs. js firefox  SM-G991B      -2.89      -31.38        -1.71
2   wasm vs. js chrome   Nexus 5      -2.30      -19.76        -2.69
3  wasm vs. js firefox   Nexus 5      -2.70      -21.41        -4.09

   median_diff%  min_diff  min_diff%  max_diff  max_diff%  std_diff  \
0        -17.37     -0.14      -3.26     -3.82     -27.36     -0.75
1        -22.29     -0.17      -3.87    -14.35     -47.11     -2.39
2        -24.61     -0.25      -4.22     -3.75     -14.40     -0.59
3        -31.15     -0.27      -4.46     -4.66     -19.07     -0.72

   std_diff%  sem_diff  sem_diff%  q1_diff  q1_diff%  q3_diff  q3_diff%
0     -37.50     -0.03     -37.50    -0.97    -17.77    -1.83    -22.07
1     -49.59     -0.10     -50.00    -1.36    -22.70    -4.41    -39.98
2     -15.73     -0.02     -13.33    -1.30    -15.24    -4.42    -30.61
3     -18.41     -0.03     -18.75    -1.40    -15.50    -4.02    -26.78
```

4.25 Descriptive Statistics Difference (Cross Browser & By Device)

```
[33]: data = []

for pairswitch in [[0,1],[1,0]]:
    for device in devices:
        for implementationpair in implementationpairs:
            for browserpair in browserpairs:
                implementation1 = stat[(stat['browser'] ==␣
 ↪browserpair[pairswitch[1]]) & (stat['implementation'] ==␣
 ↪implementationpair[1]) & (stat['device'] == device)]
                implementation2 = stat[(stat['browser'] ==␣
 ↪browserpair[pairswitch[0]]) & (stat['implementation'] ==␣
 ↪implementationpair[0]) & (stat['device'] == device)]

    #for implementationpair in implementationpairs:
    #    for browser in browsers:
    #        implementation1 = stat[(stat['implementation'] ==␣
 ↪implementationpair[0]) & (stat['browser'] == browser)]
    #        implementation2 = stat[(stat['implementation'] ==␣
 ↪implementationpair[1]) & (stat['browser'] == browser)]

            mean_diff = implementation1.iloc[0]['mean']-implementation2.
 ↪iloc[0]['mean']
            median_diff = implementation1.iloc[0]['median']-implementation2.
 ↪iloc[0]['median']
            min_diff = implementation1.iloc[0]['min']-implementation2.
 ↪iloc[0]['min']
            max_diff = implementation1.iloc[0]['max']-implementation2.
 ↪iloc[0]['max']
            std_diff = implementation1.iloc[0]['std']-implementation2.
 ↪iloc[0]['std']
            sem_diff = implementation1.iloc[0]['sem']-implementation2.
 ↪iloc[0]['sem']
            q1_diff = implementation1.iloc[0]['q1']-implementation2.
 ↪iloc[0]['q1']
            q3_diff = implementation1.iloc[0]['q3']-implementation2.
 ↪iloc[0]['q3']

            data.append(
                [
                 browserpair[pairswitch[1]] + ' ' + implementationpair[1] + '␣
 ↪vs. ' + browserpair[pairswitch[0]] + ' ' + implementationpair[0],
                 device,
                 numpy.round(mean_diff, 2),
                 numpy.round(mean_diff/implementation2.iloc[0]['mean']*100, 2),
                 numpy.round(median_diff, 2),
                 numpy.round(median_diff/implementation2.iloc[0]['median']*100,␣
 ↪2),
                 numpy.round(min_diff, 2),
                 numpy.round(min_diff/implementation2.iloc[0]['min']*100, 2),
                 numpy.round(max_diff, 2),
                 numpy.round(max_diff/implementation2.iloc[0]['max']*100, 2),
                 numpy.round(std_diff, 2),
                 numpy.round(std_diff/implementation2.iloc[0]['std']*100, 2),
                 numpy.round(sem_diff, 2),
                 numpy.round(sem_diff/implementation2.iloc[0]['sem']*100, 2),
                 numpy.round(q1_diff, 2),
                 numpy.round(q1_diff/implementation2.iloc[0]['q1']*100, 2),
                 numpy.round(q3_diff, 2),
                 numpy.round(q3_diff/implementation2.iloc[0]['q3']*100, 2),
                ]
            )

# Create the pandas DataFrame
ut = pd.DataFrame(data, columns =␣
 ↪['rq','device','mean_diff','mean_diff%','median_diff','median_diff%','min_diff','min_diff%'
display(ut)
#print(ut.to_string())
```

```
                           rq    device  mean_diff  mean_diff%  median_diff  \
0  firefox wasm vs. chrome js  SM-G991B      -0.73      -10.35        -0.66
1  firefox wasm vs. chrome js   Nexus 5      -1.73      -14.86        -1.89
2  chrome wasm vs. firefox js  SM-G991B      -3.54      -38.44        -2.20
3  chrome wasm vs. firefox js   Nexus 5      -3.27      -25.93        -4.89

   median_diff%  min_diff  min_diff%  max_diff  max_diff%  std_diff  \
0         -9.97     -0.08      -1.86      2.15      15.40      0.43
1        -17.29     -0.13      -2.20     -6.26     -24.04     -0.56
2        -28.68     -0.23      -5.24    -20.32     -66.71     -3.57
3        -37.24     -0.39      -6.44     -2.15      -8.80     -0.75

   std_diff%  sem_diff  sem_diff%  q1_diff  q1_diff%  q3_diff  q3_diff%
0      21.50      0.02      25.00    -0.83    -15.20    -1.67    -20.14
1     -14.93     -0.02     -13.33    -0.90    -10.55    -3.45    -23.89
2     -74.07     -0.15     -75.00    -1.50    -25.04    -4.57    -41.43
3     -19.18     -0.03     -18.75    -1.80    -19.93    -4.99    -33.24
```

5 RQ2: JS Energy Browser

5.1 Shapiro Wilk Test

[34]:

```
data = []
non_normal = 0

for browser in browsers:
    energy = df[(df['browser'] == browser) & (df['implementation'] ==␣
 ↪'js')]['energy']

    if len(energy) >= 3:
        shapiro_test = stats.shapiro(energy)

        non_normal += (1 if shapiro_test.pvalue <= 0.05 else 0)

        data.append(
            [browser, 'js',
             shapiro_test.statistic,
             shapiro_test.pvalue
            ]
        )

# Create the pandas DataFrame
swt = pd.DataFrame(data, columns = ['browser', 'implementation', 'w', 'p'])
#print(swt.to_string())
display(swt)

print("\n{} non-normally distributed samples".format(non_normal))
print("{} normally distributed samples".format(len(swt) - non_normal))
print("{:.2f}% non-normally distributed samples".format(non_normal/
 ↪len(swt)*100))
```

```
   browser implementation         w             p
0   chrome             js  0.923120  2.485691e-24
1  firefox             js  0.939191  8.412692e-22


2 non-normally distributed samples
0 normally distributed samples
100.00% non-normally distributed samples
```

5.2 Shapiro Wilk Test (By Device)

[35]:

```
data = []
non_normal = 0

for device in devices:
    for browser in browsers:
        energy = df[(df['browser'] == browser) & (df['implementation'] == 'js')␣
 ↪& (df['device'] == device)]['energy']

        if len(energy) >= 3:
            shapiro_test = stats.shapiro(energy)

            non_normal += (1 if shapiro_test.pvalue <= 0.05 else 0)

            data.append(
                [
                 device,
                 browser, 'js',
                 shapiro_test.statistic,
                 shapiro_test.pvalue
                ]
            )

# Create the pandas DataFrame
swt = pd.DataFrame(data, columns = ['device', 'browser', 'implementation', 'w',␣
 ↪'p'])
#print(swt.to_string())
display(swt)

print("\n{} non-normally distributed samples".format(non_normal))
print("{} normally distributed samples".format(len(swt) - non_normal))
print("{:.2f}% non-normally distributed samples".format(non_normal/
 ↪len(swt)*100))
```

```
     device  browser implementation         w             p
0  SM-G991B   chrome             js  0.943418  2.788160e-14
1  SM-G991B  firefox             js  0.808114  6.720060e-26
2   Nexus 5   chrome             js  0.949549  1.371225e-13
3   Nexus 5  firefox             js  0.963830  3.718362e-11


4 non-normally distributed samples
0 normally distributed samples
100.00% non-normally distributed samples
```

5.3 Mann Whitney U Test

[36]:

```
data = []

for browserpair in browserpairs:
    browser1_energy = df[(df['browser'] == browserpair[0]) &␣
 ↪(df['implementation'] == 'js')]['energy']
```

```
    browser2_energy = df[(df['browser'] == browserpair[1]) &␣
↪(df['implementation'] == 'js')]['energy']
    eff = cliff.cliffs_delta(browser1_energy, browser2_energy)

    u = stats.mannwhitneyu(browser1_energy, browser2_energy,␣
↪alternative='two-sided')

    data.append(
        [
         browserpair[0] + ' vs. ' + browserpair[1],
         u.statistic,
         u.pvalue,
         eff[0],
         eff[1]
        ]
    )

# Create the pandas DataFrame
ut = pd.DataFrame(data, columns = ['rq', 'u', 'p', 'eff', 'interp'])
display(ut)

interp = ut['interp'].value_counts()
interp = pd.DataFrame(interp, columns = ['interp', 'percent'])
interp['percent'] = (interp['interp'] / interp['interp'].sum()) * 100

display(interp)
```

```
                  rq         u             p       eff interp
0  chrome vs. firefox  586526.0  2.471443e-16 -0.192797  small
```

```
Empty DataFrame
Columns: [interp, percent]
Index: []
```

5.4 Mann Whitney U Test (By Device)

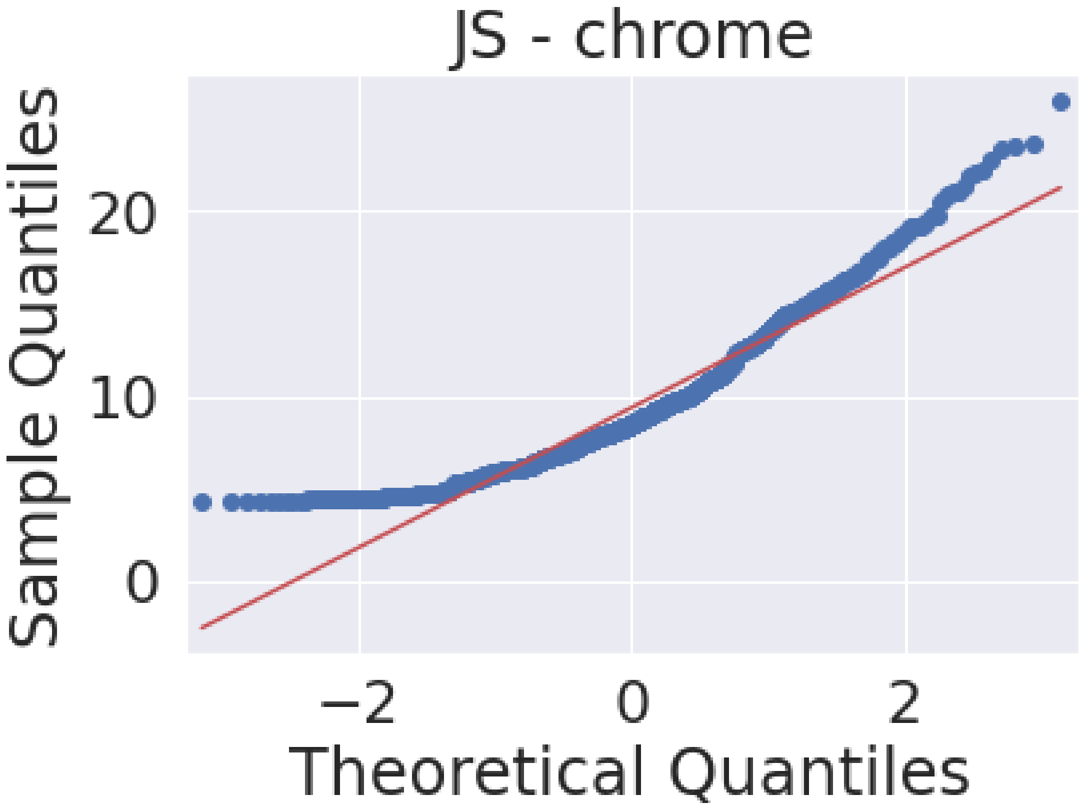
[37]:
```
data = []

for device in devices:
    for browserpair in browserpairs:
        browser1_energy = df[(df['browser'] == browserpair[0]) &␣
↪(df['implementation'] == 'js') & (df['device'] == device)]['energy']
        browser2_energy = df[(df['browser'] == browserpair[1]) &␣
↪(df['implementation'] == 'js') & (df['device'] == device)]['energy']
        eff = cliff.cliffs_delta(browser1_energy, browser2_energy)
        u = stats.mannwhitneyu(browser1_energy, browser2_energy,␣
↪alternative='two-sided')

        data.append(
            [
             device,
             browserpair[0] + ' vs. ' + browserpair[1],
             u.statistic,
             u.pvalue,
             eff[0],
             eff[1]
            ]
        )

# Create the pandas DataFrame
ut = pd.DataFrame(data, columns = ['device', 'rq', 'u', 'p', 'eff', 'interp'])
display(ut)

interp = ut['interp'].value_counts()
interp = pd.DataFrame(interp, columns = ['interp', 'percent'])
interp['percent'] = (interp['interp'] / interp['interp'].sum()) * 100

display(interp)
```

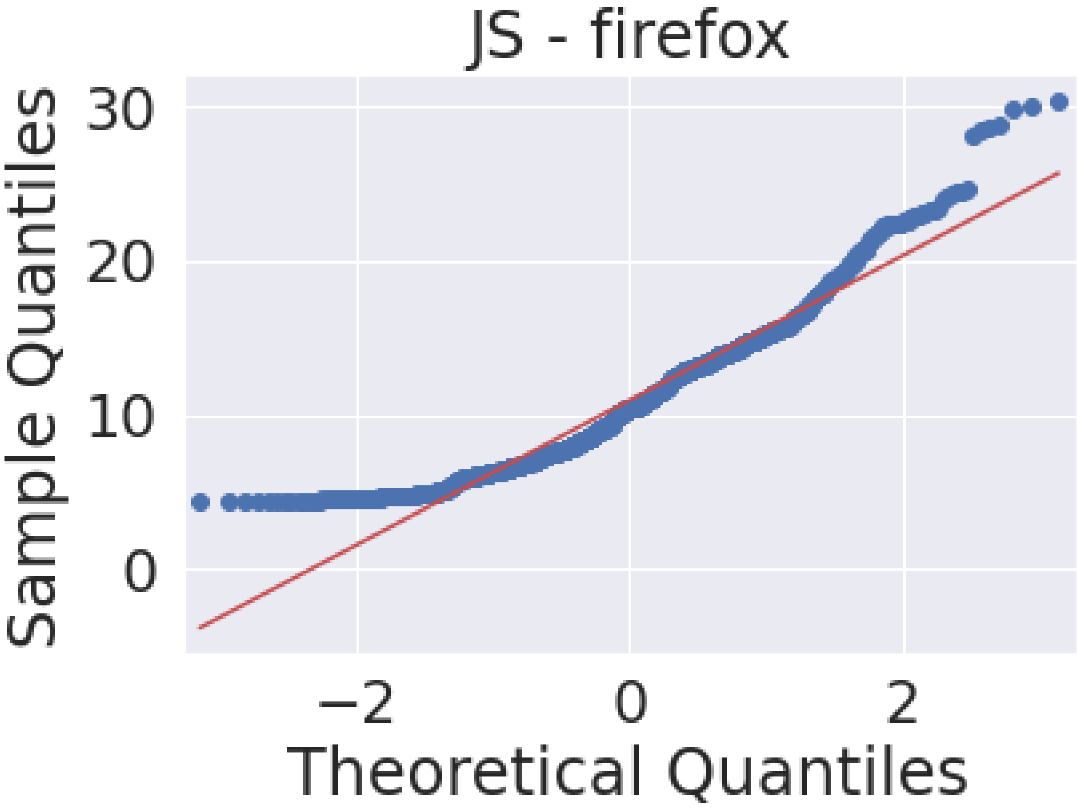
```
     device                  rq         u             p       eff      interp
0  SM-G991B  chrome vs. firefox  130114.0  9.184663e-15 -0.259970       small
1   Nexus 5  chrome vs. firefox  160175.0  9.598055e-06 -0.146084  negligible
```

```
Empty DataFrame
Columns: [interp, percent]
Index: []
```

5.5 Q-Q-Plot

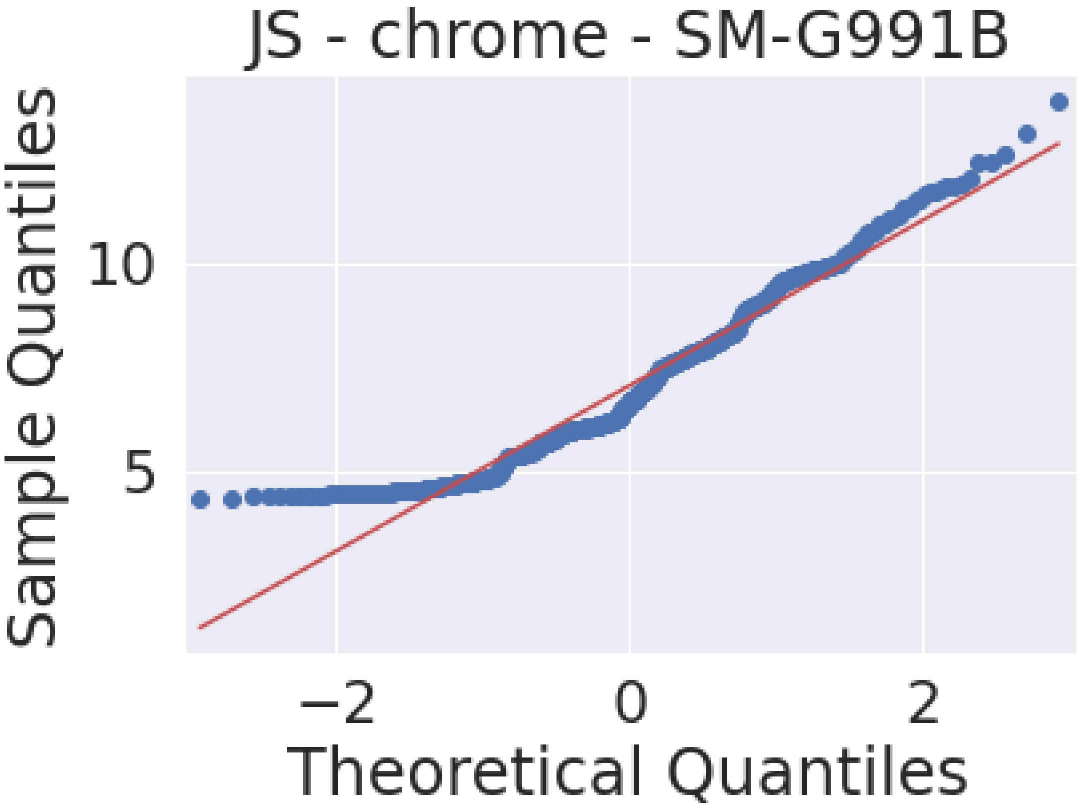
[38]:
```
for browser in browsers:
    data = df[(df['implementation'] == 'js') & (df['browser'] == browser)]
    qq = sm.qqplot(data.energy, line='s')
    h = plt.title('JS - ' + browser)
```

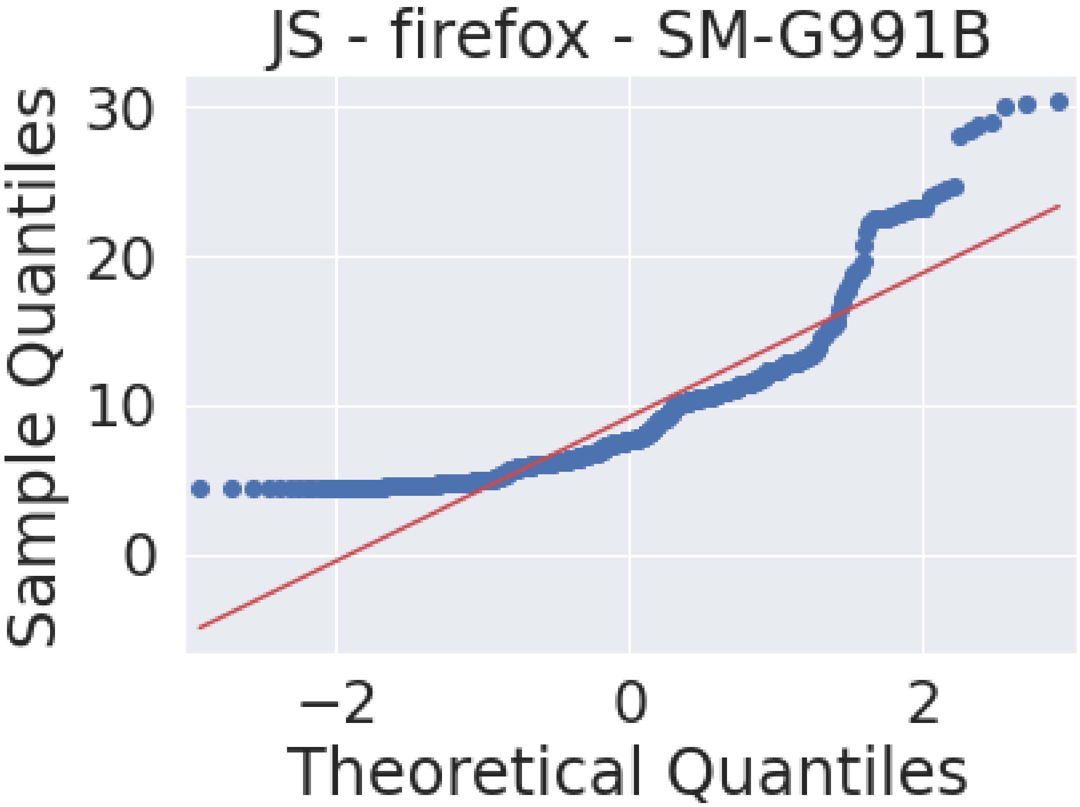

JS - chrome
Sample Quantiles
0
10
20
−2
0
2
Theoretical Quantiles

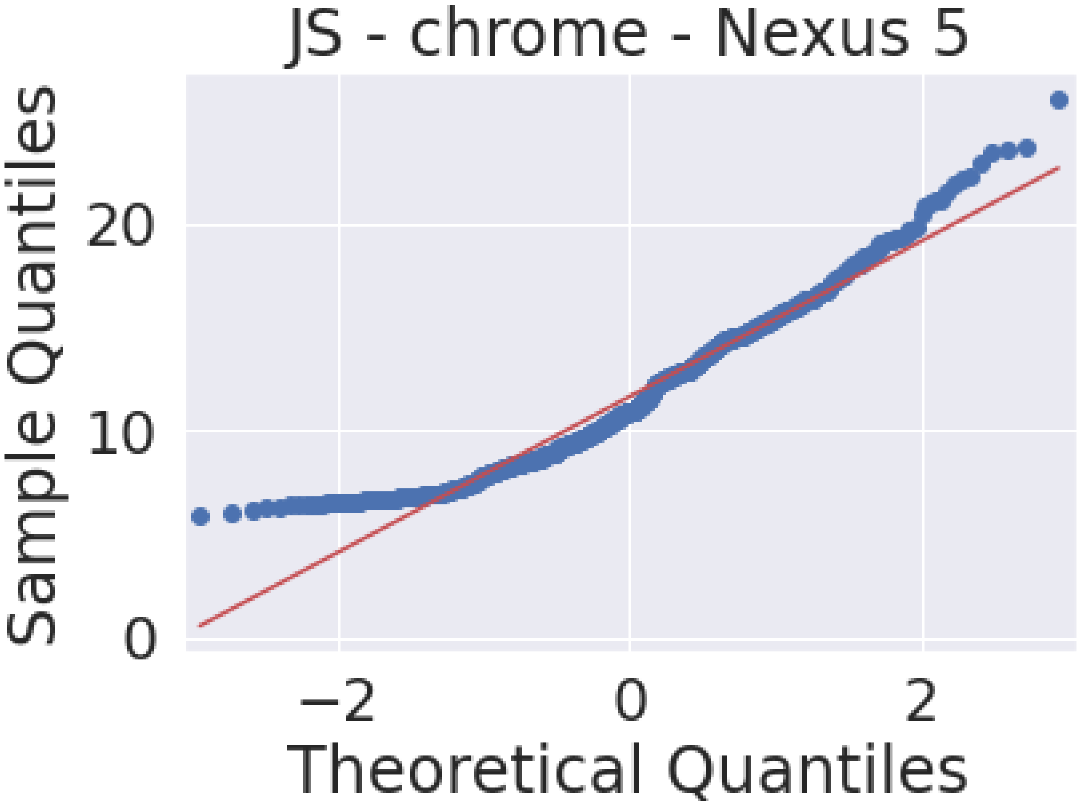


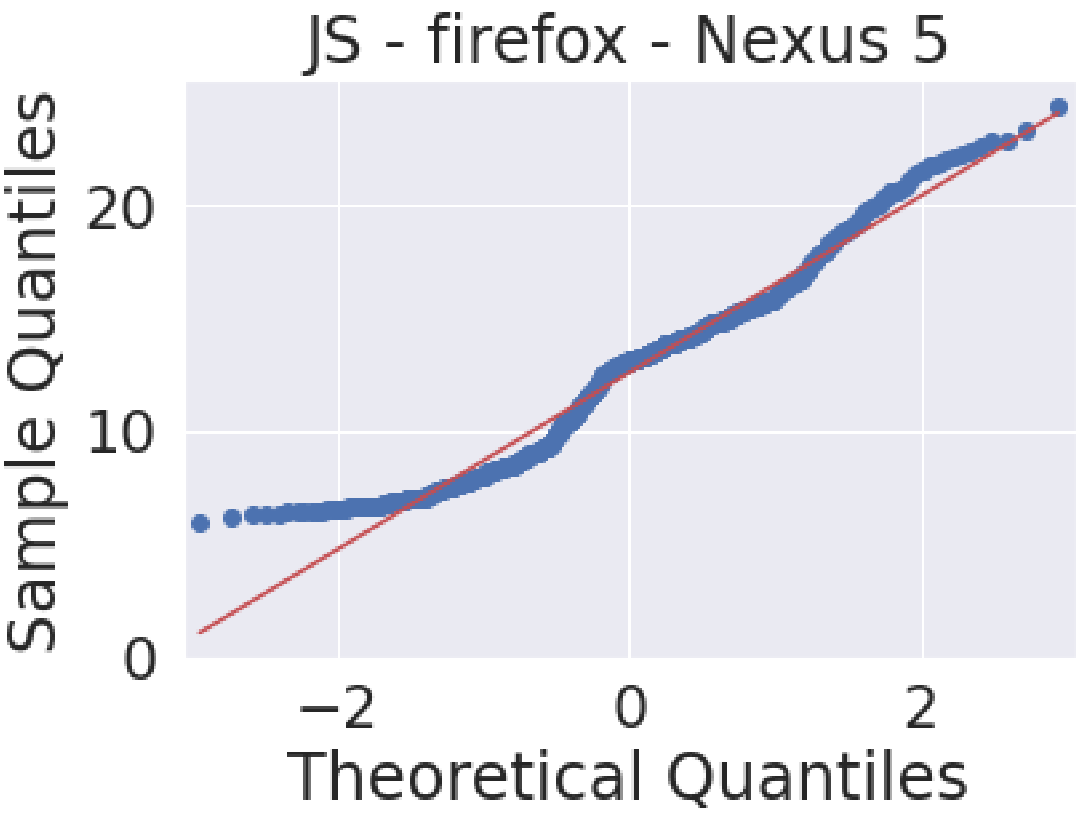
5.6 Q-Q-Plot (By Device)

[39]:
```
for device in devices:
    for browser in browsers:
        data = df[(df['implementation'] == 'js') & (df['browser'] == browser) &␣
 ↪(df['device'] == device)]
        qq = sm.qqplot(data.energy, line='s')
        h = plt.title('JS - ' + browser + ' - ' + device)
```


JS - chrome - SM-G991B
Sample Quantiles
10
5
−2
0
2
Theoretical Quantiles


JS - firefox - SM-G991B
Sample Quantiles
30
20
10
0
−2
0
2
Theoretical Quantiles

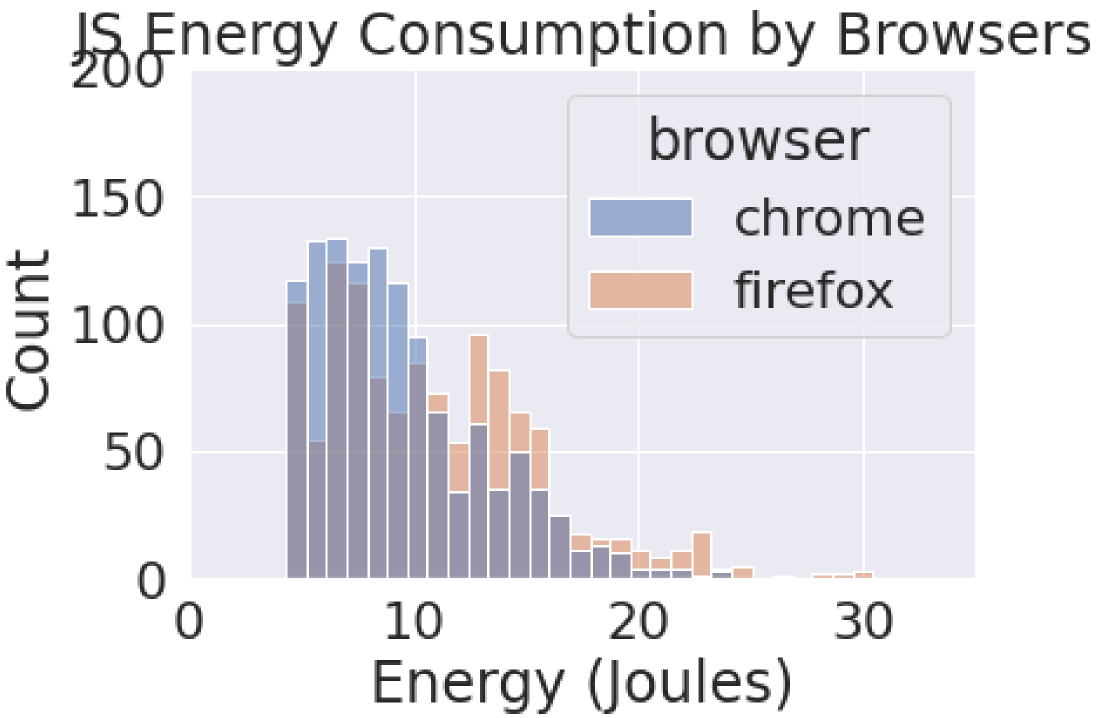

JS - chrome - Nexus 5
Sample Quantiles
0
10
20
−2
0
2
Theoretical Quantiles



5.7 Histogram

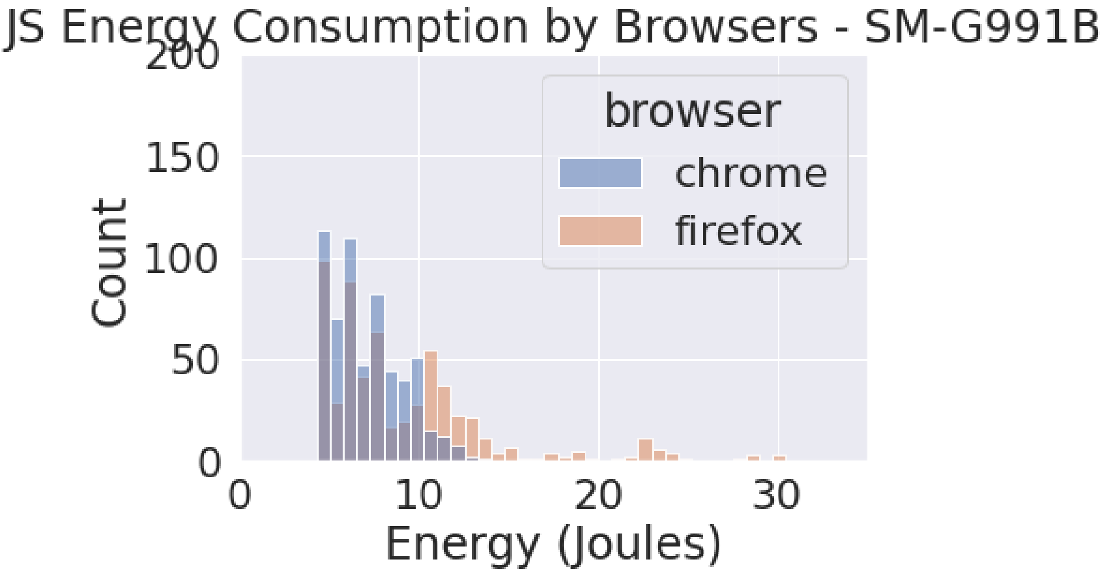
[40]:
```
data = df[(df['implementation'] == 'js')]
sns.histplot(data=data, x="energy", hue="browser", hue_order=browsers).
  ↪set_title("JS Energy Consumption by Browsers")
plt.xlabel("Energy (Joules)")
plt.ylim(0, 200)
plt.xlim(0, 35)
```

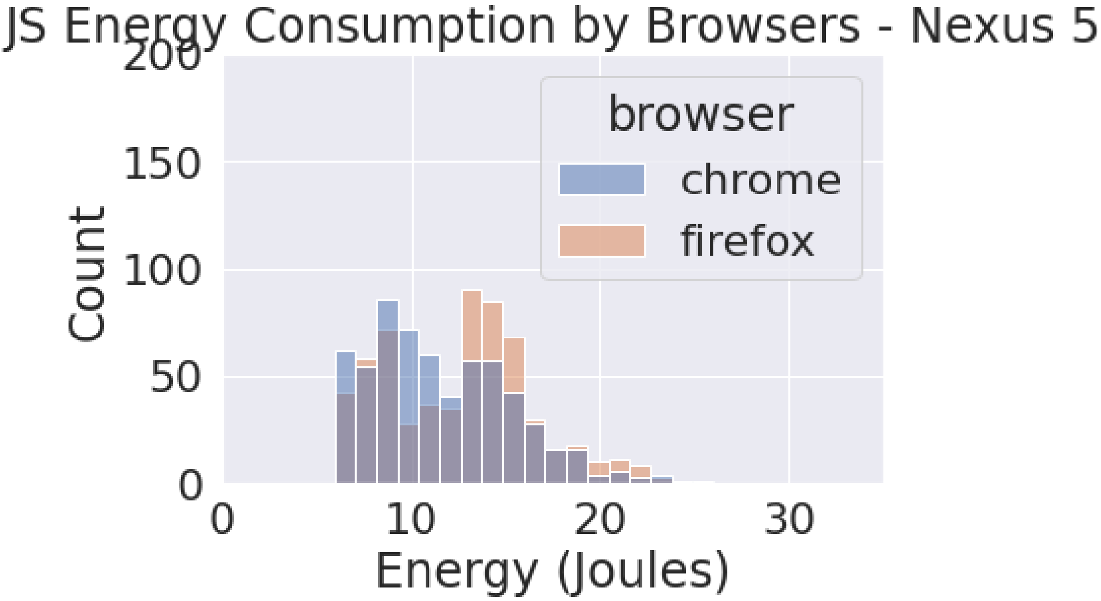
[40]:
```
(0.0, 35.0)
```



5.8 Histogramm (By Device)

[41]:
```
data = []
for device in devices:
    data = df[(df['implementation'] == 'js') & (df['device'] == device)]
    sns.histplot(data=data, x="energy", hue="browser", hue_order=browsers).
  ↪set_title("JS Energy Consumption by Browsers" + " - " + device)
    plt.xlabel("Energy (Joules)")
    plt.ylim(0, 200)
    plt.xlim(0, 35)
    plt.show()
```





5.9 Descriptive Statistics

[42]:

```
data = []
for browser in browsers:
    x = df[(df['browser'] == browser) & (df['implementation'] == 'js')]
    mean = numpy.round(numpy.mean(x['energy']), 2)
    median = numpy.round(numpy.median(x['energy']), 2)
    min = numpy.round(numpy.amin(x['energy']), 2)
    max = numpy.round(numpy.amax(x['energy']), 2)
    std = numpy.round(numpy.std(x['energy']), 2)
    sem = numpy.round(stats.sem(x['energy']), 2)
    q1 = numpy.round(numpy.quantile(x['energy'], 0.25), 2)
    q3 = numpy.round(numpy.quantile(x['energy'], 0.75), 2)

    data.append(
        [browser, mean, std, min, q1, median, q3, max, sem]
    )

# Create the pandas DataFrame
stat = pd.DataFrame(data, columns = ['browser', 'mean', 'std', 'min', 'q1',␣
 ↪'median', 'q3', 'max', 'sem'])
display(stat)
#print(stat.to_string())

# Alternative of pandas: x['energy'].describe()
```

```
   browser   mean   std   min    q1  median     q3    max   sem
0   chrome   9.37  3.79  4.30  6.52    8.47  11.28  26.04  0.11
1  firefox  10.94  4.70  4.39  7.11   10.43  13.89  30.46  0.14
```

5.10 Descriptive Statistics Difference

[43]:
```
data = []

for browserpair in browserpairs:
    browser1 = stat[(stat['browser'] == browserpair[0])]
    browser2 = stat[(stat['browser'] == browserpair[1])]

    mean_diff = browser1.iloc[0]['mean']-browser2.iloc[0]['mean']
    median_diff = browser1.iloc[0]['median']-browser2.iloc[0]['median']
    min_diff = browser1.iloc[0]['min']-browser2.iloc[0]['min']
    max_diff = browser1.iloc[0]['max']-browser2.iloc[0]['max']
    std_diff = browser1.iloc[0]['std']-browser2.iloc[0]['std']
    sem_diff = browser1.iloc[0]['sem']-browser2.iloc[0]['sem']
    q1_diff = browser1.iloc[0]['q1']-browser2.iloc[0]['q1']
    q3_diff = browser1.iloc[0]['q3']-browser2.iloc[0]['q3']

    data.append(
        [browserpair[0] + ' vs. ' + browserpair[1],
         numpy.round(mean_diff, 2),
         numpy.round(mean_diff/browser2.iloc[0]['mean']*100, 2),
         numpy.round(median_diff, 2),
         numpy.round(median_diff/browser2.iloc[0]['median']*100, 2),
         numpy.round(min_diff, 2),
         numpy.round(min_diff/browser2.iloc[0]['min']*100, 2),
         numpy.round(max_diff, 2),
         numpy.round(max_diff/browser2.iloc[0]['max']*100, 2),
         numpy.round(std_diff, 2),
         numpy.round(std_diff/browser2.iloc[0]['std']*100, 2),
         numpy.round(sem_diff, 2),
         numpy.round(sem_diff/browser2.iloc[0]['sem']*100, 2),
         numpy.round(q1_diff, 2),
         numpy.round(q1_diff/browser2.iloc[0]['q1']*100, 2),
         numpy.round(q3_diff, 2),
         numpy.round(q3_diff/browser2.iloc[0]['q3']*100, 2),
        ]
    )

# Create the pandas DataFrame
ut = pd.DataFrame(data, columns =␣
 ↪['rq','mean_diff','mean_diff%','median_diff','median_diff%','min_diff','min_diff%','max_dif
display(ut)
#print(ut.to_string())
```

```
                  rq  mean_diff  mean_diff%  median_diff  median_diff%  \
0  chrome vs. firefox      -1.57      -14.35        -1.96        -18.79   

   min_diff  min_diff%  max_diff  max_diff%  std_diff  std_diff%  sem_diff  \
0     -0.09      -2.05     -4.42     -14.51     -0.91     -19.36     -0.03   

   sem_diff%  q1_diff  q1_diff%  q3_diff  q3_diff%  
0     -21.43    -0.59      -8.3    -2.61    -18.79  
```

5.11 Descriptive Statistics (By Device)

[44]:

```
data = []
for device in devices:
    for browser in browsers:
        x = df[(df['browser'] == browser) & (df['implementation'] == 'js') &␣
 ↪(df['device'] == device)]
        mean = numpy.round(numpy.mean(x['energy']), 2)
        median = numpy.round(numpy.median(x['energy']), 2)
        min = numpy.round(numpy.amin(x['energy']), 2)
        max = numpy.round(numpy.amax(x['energy']), 2)
        std = numpy.round(numpy.std(x['energy']), 2)
        sem = numpy.round(stats.sem(x['energy']), 2)
        q1 = numpy.round(numpy.quantile(x['energy'], 0.25), 2)
        q3 = numpy.round(numpy.quantile(x['energy'], 0.75), 2)

        data.append(
            [device, browser, mean, std, min, q1, median, q3, max, sem]
        )

# Create the pandas DataFrame
stat = pd.DataFrame(data, columns = ['device', 'browser', 'mean', 'std', 'min',␣
 ↪'q1', 'median', 'q3', 'max', 'sem'])
display(stat)
#print(stat.to_string())

# Alternative of pandas: x['energy'].describe()
```

```
     device  browser   mean   std   min    q1  median     q3    max   sem
0  SM-G991B   chrome   7.05  2.00  4.30  5.46    6.62   8.29  13.96  0.08
1  SM-G991B  firefox   9.21  4.82  4.39  5.99    7.67  11.03  30.46  0.20
2   Nexus 5   chrome  11.64  3.75  5.92  8.53   10.93  14.44  26.04  0.15
3   Nexus 5  firefox  12.61  3.91  6.06  9.03   13.13  15.01  24.44  0.16
```

5.12 Descriptive Statistics Difference (By Device)

[45]:

```
data = []

for device in devices:
    for browserpair in browserpairs:
        browser1 = stat[(stat['browser'] == browserpair[0]) & (stat['device']␣
 ↪== device)]
        browser2 = stat[(stat['browser'] == browserpair[1]) & (stat['device']␣
 ↪== device)]

        mean_diff = browser1.iloc[0]['mean']-browser2.iloc[0]['mean']
        median_diff = browser1.iloc[0]['median']-browser2.iloc[0]['median']
        min_diff = browser1.iloc[0]['min']-browser2.iloc[0]['min']
        max_diff = browser1.iloc[0]['max']-browser2.iloc[0]['max']
        std_diff = browser1.iloc[0]['std']-browser2.iloc[0]['std']
        sem_diff = browser1.iloc[0]['sem']-browser2.iloc[0]['sem']
        q1_diff = browser1.iloc[0]['q1']-browser2.iloc[0]['q1']
        q3_diff = browser1.iloc[0]['q3']-browser2.iloc[0]['q3']

        data.append(
            [
             device,
             browserpair[0] + ' vs. ' + browserpair[1],
             numpy.round(mean_diff, 2),
             numpy.round(mean_diff/browser2.iloc[0]['mean']*100, 2),
             numpy.round(median_diff, 2),
             numpy.round(median_diff/browser2.iloc[0]['median']*100, 2),
             numpy.round(min_diff, 2),
             numpy.round(min_diff/browser2.iloc[0]['min']*100, 2),
             numpy.round(max_diff, 2),
             numpy.round(max_diff/browser2.iloc[0]['max']*100, 2),
             numpy.round(std_diff, 2),
             numpy.round(std_diff/browser2.iloc[0]['std']*100, 2),
             numpy.round(sem_diff, 2),
             numpy.round(sem_diff/browser2.iloc[0]['sem']*100, 2),
             numpy.round(q1_diff, 2),
             numpy.round(q1_diff/browser2.iloc[0]['q1']*100, 2),
             numpy.round(q3_diff, 2),
             numpy.round(q3_diff/browser2.iloc[0]['q3']*100, 2),
            ]
        )

# Create the pandas DataFrame
ut = pd.DataFrame(data, columns = ['device',␣
 ↪'rq','mean_diff','mean_diff%','median_diff','median_diff%','min_diff','min_diff%','max_diff
display(ut)
#print(ut.to_string())
```

```
    device                 rq  mean_diff  mean_diff%  median_diff  \
0  SM-G991B  chrome vs. firefox      -2.16      -23.45        -1.05
1   Nexus 5  chrome vs. firefox      -0.97       -7.69        -2.20

   median_diff%  min_diff  min_diff%  max_diff  max_diff%  std_diff  \
0        -13.69     -0.09      -2.05     -16.5     -54.17     -2.82
1        -16.76     -0.14      -2.31       1.6       6.55     -0.16

   std_diff%  sem_diff  sem_diff%  q1_diff  q1_diff%  q3_diff  q3_diff%
0     -58.51     -0.12     -60.00    -0.53     -8.85    -2.74    -24.84
1      -4.09     -0.01      -6.25    -0.50     -5.54    -0.57     -3.80
```

6 RQ2: WASM Energy Browser

6.1 Shapiro Wilk Test

[46]:

```
data = []
non_normal = 0

for browser in browsers:
    energy = df[(df['browser'] == browser) & (df['implementation'] ==␣
 ↪'wasm')]['energy']

    if len(energy) >= 3:
        shapiro_test = stats.shapiro(energy)

        non_normal += (1 if shapiro_test.pvalue <= 0.05 else 0)

        data.append(
                [browser, 'wasm',
                 shapiro_test.statistic,
                 shapiro_test.pvalue
                ]
           )

# Create the pandas DataFrame
swt = pd.DataFrame(data, columns = ['browser', 'implementation', 'w', 'p'])
#print(swt.to_string())
display(swt)

print("\n{} non-normally distributed samples".format(non_normal))
print("{} normally distributed samples".format(len(swt) - non_normal))
print("{:.2f}% non-normally distributed samples".format(non_normal/
 ↪len(swt)*100))
```

```
   browser implementation         w             p
0   chrome           wasm  0.837385  2.093366e-33
1  firefox           wasm  0.882936  3.217827e-29
```

```
2 non-normally distributed samples
0 normally distributed samples
100.00% non-normally distributed samples
```

6.2 Shapiro Wilk Test (By Device)

[47]:

```
data = []
non_normal = 0

for device in devices:
    for browser in browsers:
        energy = df[(df['browser'] == browser) & (df['implementation'] ==␣
 ↪'wasm') & (df['device'] == device)]['energy']

        if len(energy) >= 3:
            shapiro_test = stats.shapiro(energy)

            non_normal += (1 if shapiro_test.pvalue <= 0.05 else 0)

            data.append(
                [
                 device,
                 browser, 'wasm',
                 shapiro_test.statistic,
                 shapiro_test.pvalue
                ]
            )

# Create the pandas DataFrame
swt = pd.DataFrame(data, columns = ['device', 'browser', 'implementation', 'w',␣
 ↪'p'])
#print(swt.to_string())
display(swt)

print("\n{} non-normally distributed samples".format(non_normal))
print("{} normally distributed samples".format(len(swt) - non_normal))
print("{:.2f}% non-normally distributed samples".format(non_normal/
 ↪len(swt)*100))
```

```
     device  browser implementation         w             p
0  SM-G991B   chrome           wasm  0.911131  4.198296e-18
1  SM-G991B  firefox           wasm  0.723657  3.320376e-30
2   Nexus 5   chrome           wasm  0.828244  3.247344e-25
3   Nexus 5  firefox           wasm  0.879442  2.271143e-21
```

```
4 non-normally distributed samples
0 normally distributed samples
100.00% non-normally distributed samples
```

6.3 Mann Whitney U Test

[48]:

```
data = []

for browserpair in browserpairs:
    browser1_energy = df[(df['browser'] == browserpair[0]) &␣
 ↪(df['implementation'] == 'wasm')]['energy']
    browser2_energy = df[(df['browser'] == browserpair[1]) &␣
 ↪(df['implementation'] == 'wasm')]['energy']
    eff = cliff.cliffs_delta(browser1_energy, browser2_energy)

    u = stats.mannwhitneyu(browser1_energy, browser2_energy,␣
 ↪alternative='two-sided')

    data.append(
        [browserpair[0] + ' vs. ' + browserpair[1],
         u.statistic,
         u.pvalue,
         eff[0],
         eff[1]
        ]
    )
```

```
# Create the pandas DataFrame
ut = pd.DataFrame(data, columns = ['rq','u', 'p', 'eff', 'interp'])
display(ut)

interp = ut['interp'].value_counts()
interp = pd.DataFrame(interp, columns = ['interp', 'percent'])
interp['percent'] = (interp['interp'] / interp['interp'].sum()) * 100

display(interp)
```

```
                   rq         u         p       eff      interp
0  chrome vs. firefox  655827.0  0.000019 -0.100408  negligible
```

```
Empty DataFrame
Columns: [interp, percent]
Index: []
```

6.4 Mann Whitney U Test (By Device)

[49]:

```
data = []

for device in devices:
    for browserpair in browserpairs:
        browser1_energy = df[(df['browser'] == browserpair[0]) &␣
 ↪(df['implementation'] == 'wasm') & (df['device'] == device)]['energy']
        browser2_energy = df[(df['browser'] == browserpair[1]) &␣
 ↪(df['implementation'] == 'wasm') & (df['device'] == device)]['energy']
        eff = cliff.cliffs_delta(browser1_energy, browser2_energy)

        u = stats.mannwhitneyu(browser1_energy, browser2_energy,␣
 ↪alternative='two-sided')

        data.append(
            [
             device,
             browserpair[0] + ' vs. ' + browserpair[1],
             u.statistic,
             u.pvalue,
             eff[0],
             eff[1]
            ]
        )

# Create the pandas DataFrame
ut = pd.DataFrame(data, columns = ['device', 'rq','u', 'p', 'eff', 'interp'])
display(ut)
```

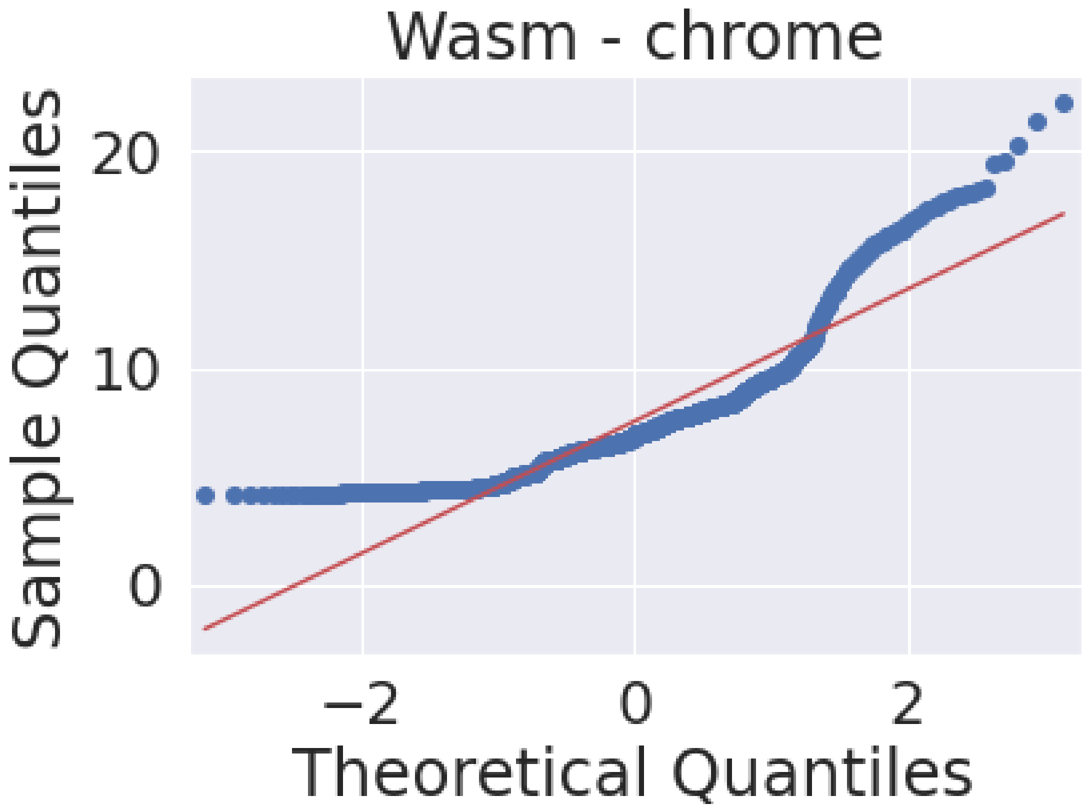
```
interp = ut['interp'].value_counts()
interp = pd.DataFrame(interp, columns = ['interp', 'percent'])
interp['percent'] = (interp['interp'] / interp['interp'].sum()) * 100

display(interp)
```

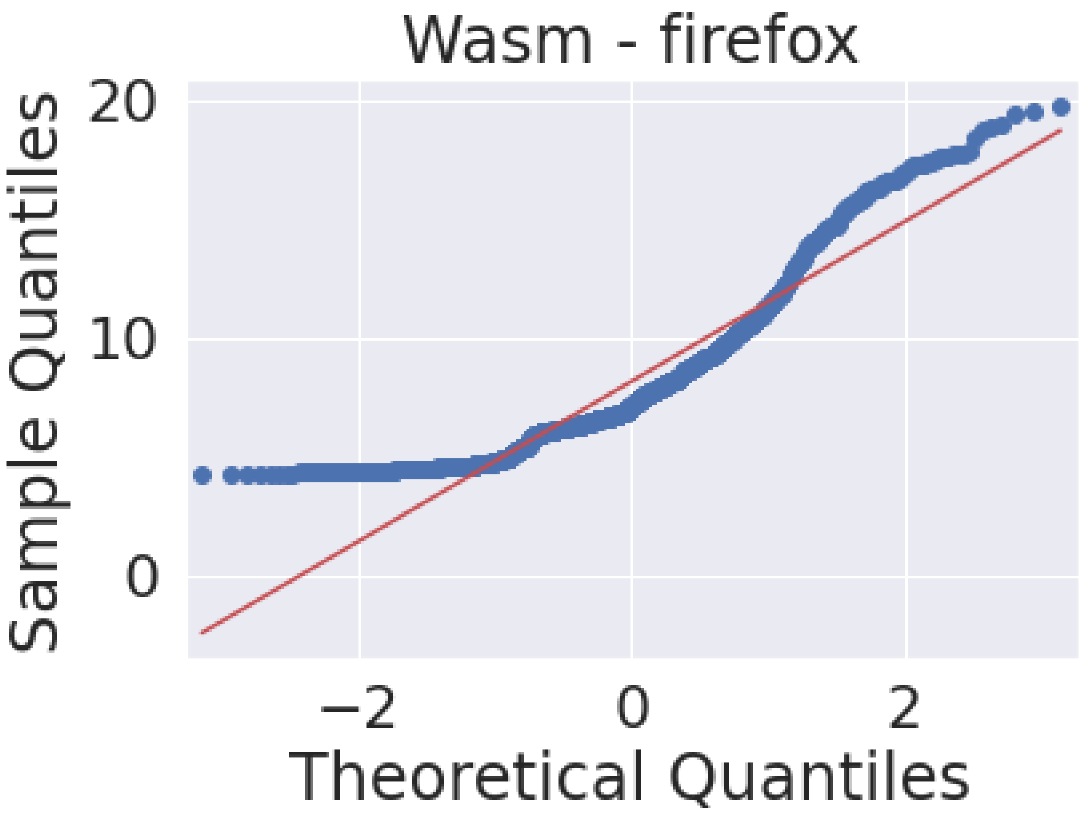
```
     device                  rq         u         p       eff      interp
0  SM-G991B  chrome vs. firefox  154952.0  0.000402 -0.118702  negligible
1   Nexus 5  chrome vs. firefox  162482.0  0.000023 -0.139392  negligible
```

```
Empty DataFrame
Columns: [interp, percent]
Index: []
```

6.5 Q-Q-Plot

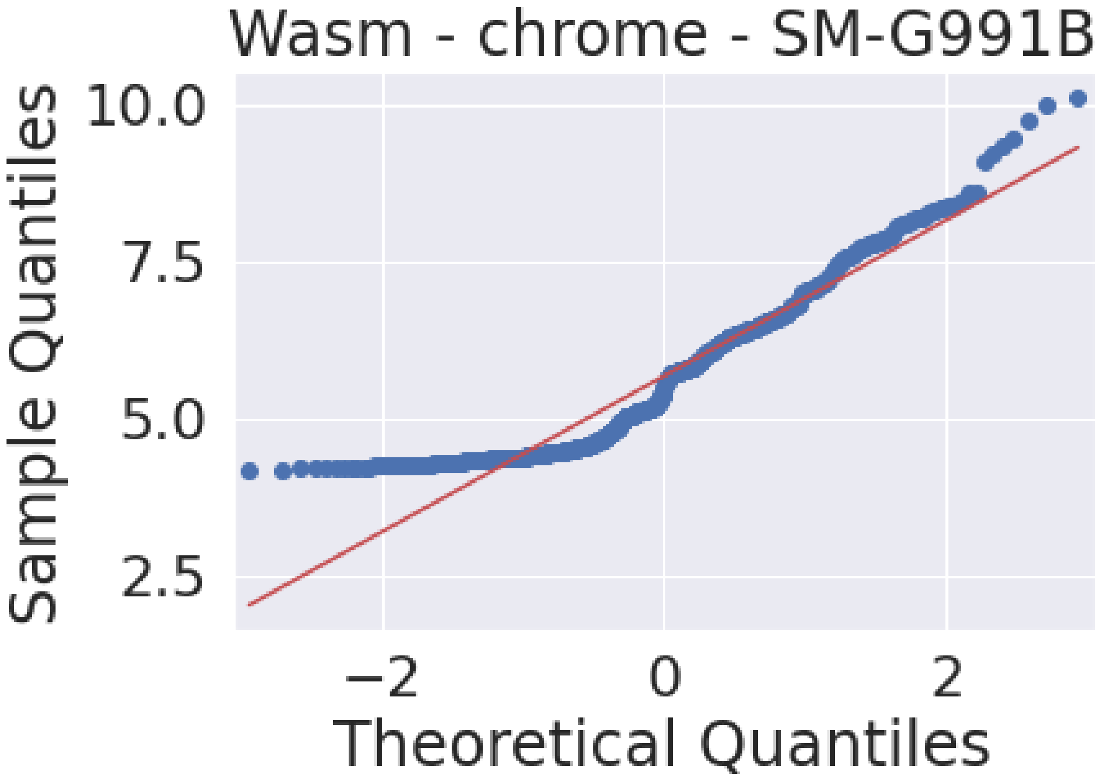
```
[50]: for browser in browsers:
          data = df[(df['implementation'] == 'wasm') & (df['browser'] == browser)]
          qq = sm.qqplot(data.energy, line='s')
          h = plt.title('Wasm - ' + browser)
```

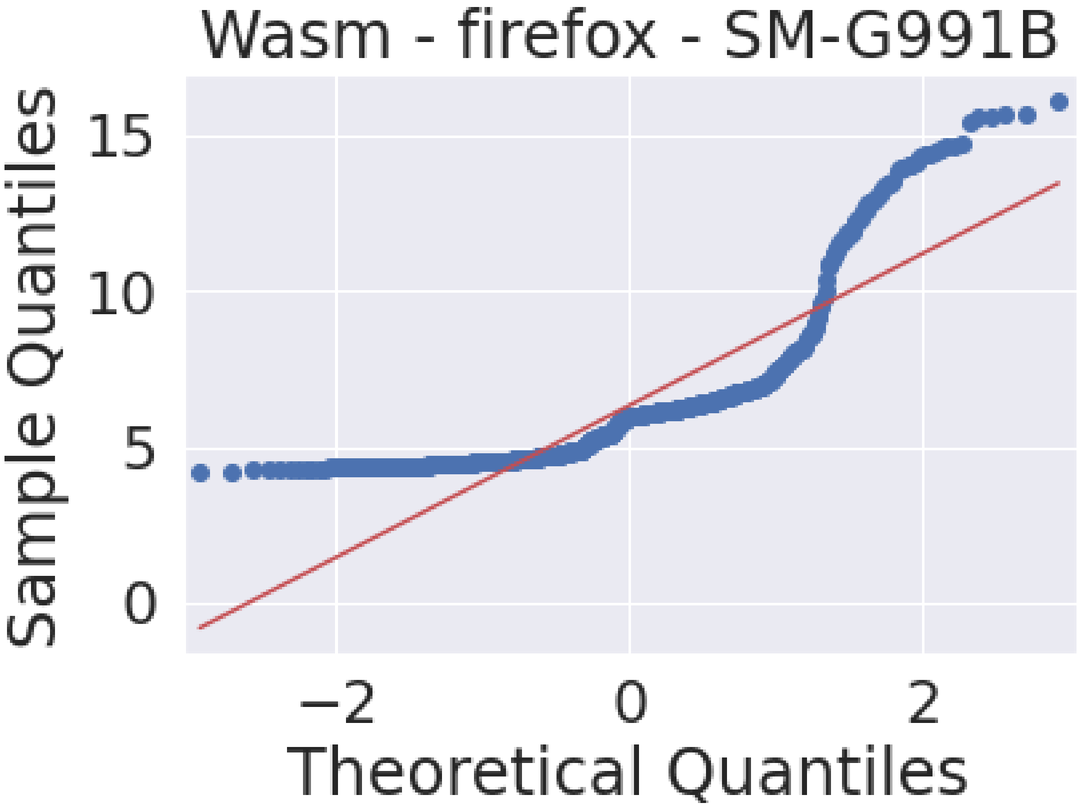


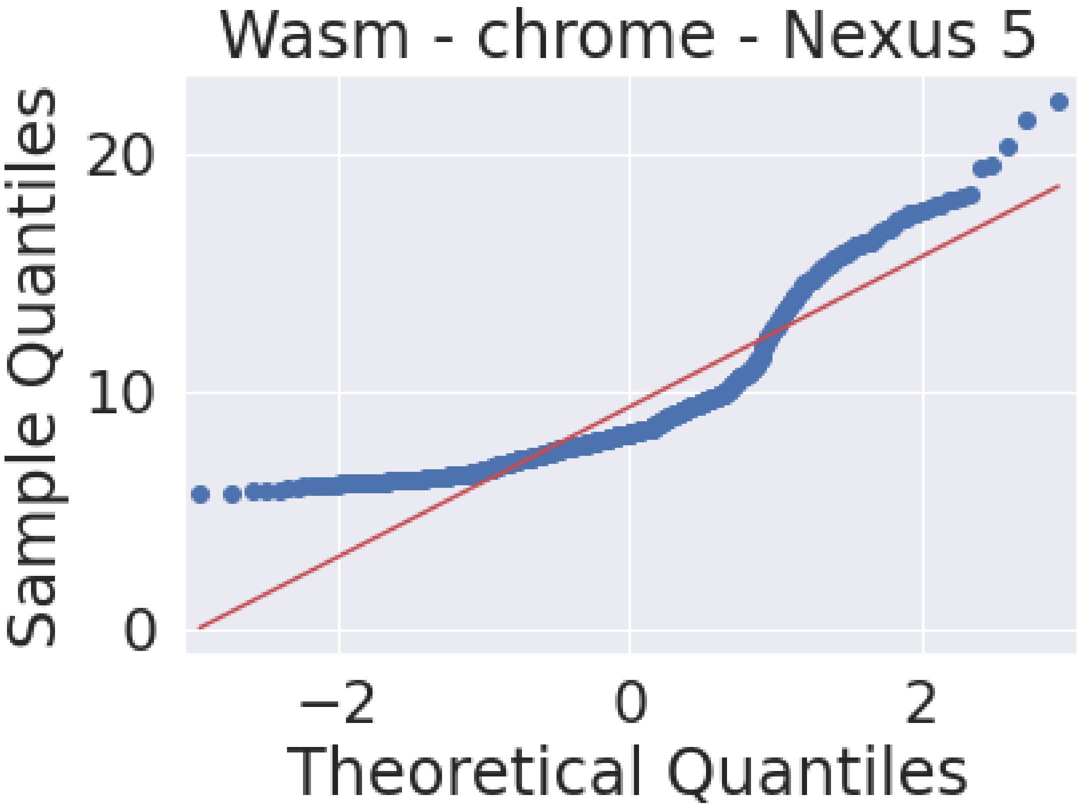


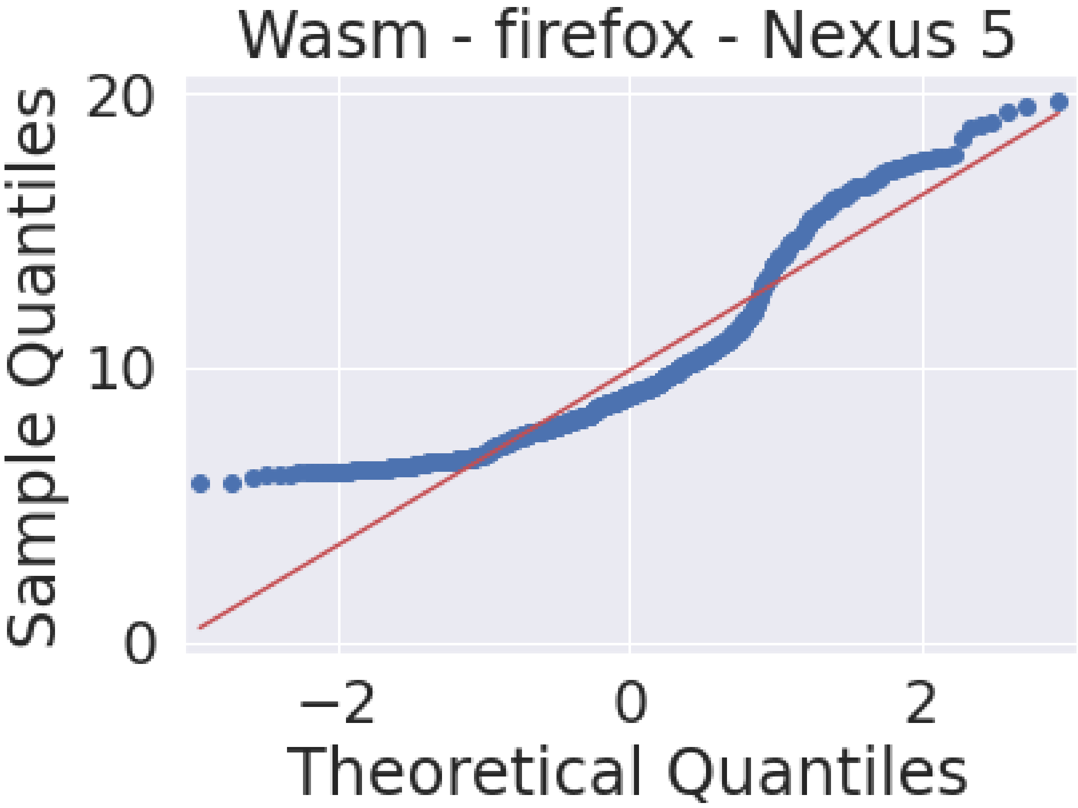
6.6 Q-Q-Plot (By Device)

```
[51]: for device in devices:
          for browser in browsers:
              data = df[(df['implementation'] == 'wasm') & (df['browser'] == browser)␣
       ↪& (df['device'] == device)]
              qq = sm.qqplot(data.energy, line='s')
              h = plt.title('Wasm - ' + browser + ' - ' + device)
```


Wasm - chrome - SM-G991B
Sample Quantiles
10.0
7.5
5.0
2.5
−2
0
2
Theoretical Quantiles


Wasm - firefox - SM-G991B
Sample Quantiles
15
10
5
0
−2
0
2
Theoretical Quantiles

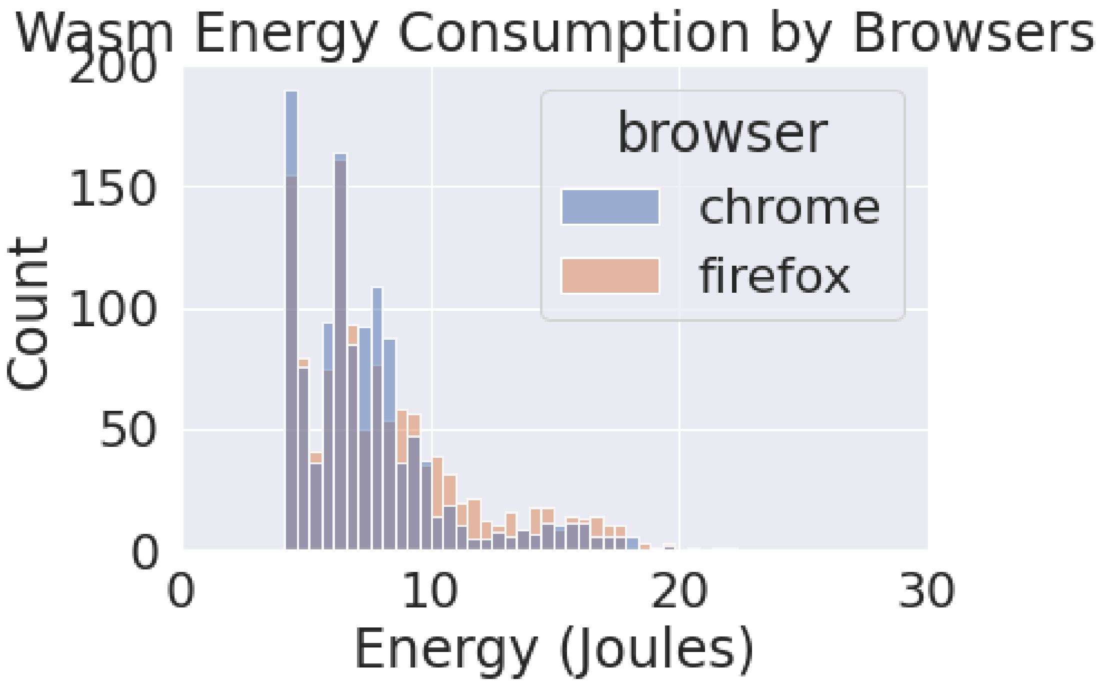

Wasm - chrome - Nexus 5
Sample Quantiles
20
10
0
−2
0
2
Theoretical Quantiles



6.7 Histogram

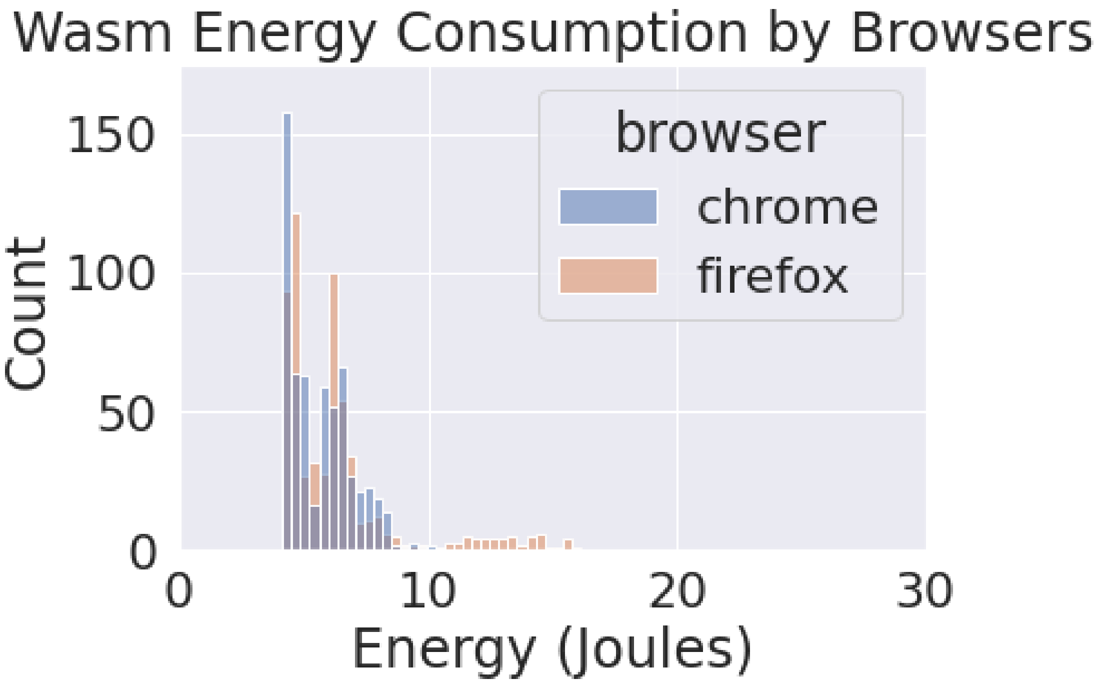
```
[52]: data = df[(df['implementation'] == 'wasm')]
      sns.histplot(data=data, x="energy", hue="browser", hue_order=browsers).
        ↪set_title("Wasm Energy Consumption by Browsers")
      plt.xlabel("Energy (Joules)")
      plt.ylim(0, 200)
      plt.xlim(0, 30)
```

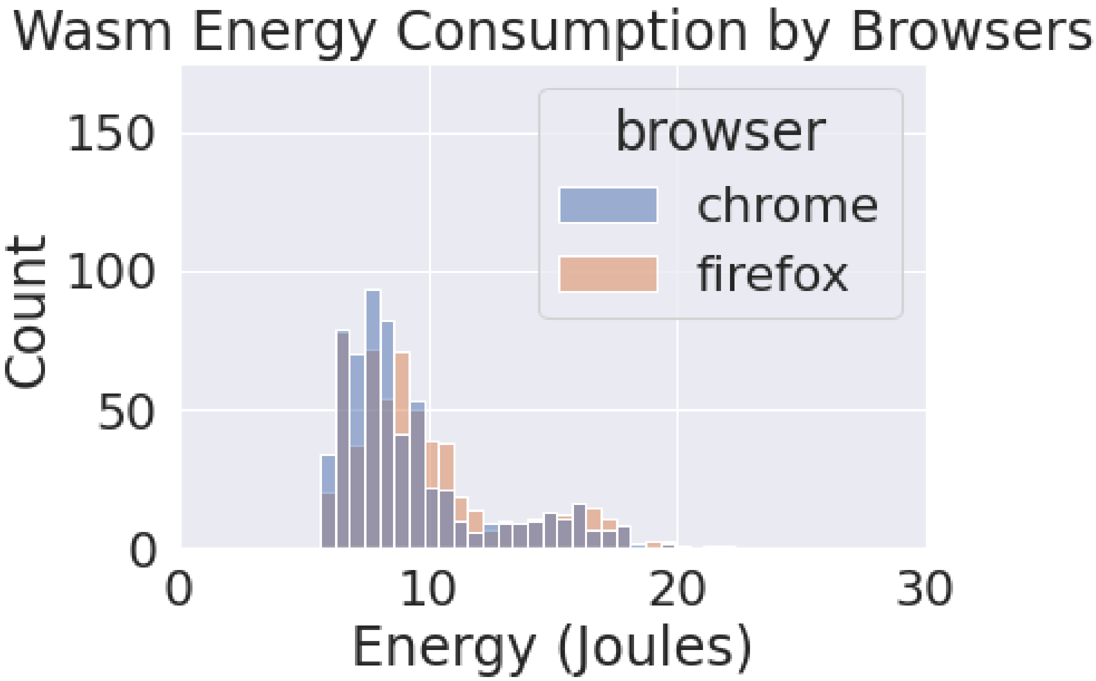
```
[52]: (0.0, 30.0)
```



6.8 Histogram (By Device)

```
[53]: data = []
      for device in devices:
          data = df[(df['implementation'] == 'wasm') & (df['device'] == device)]
          sns.histplot(data=data, x="energy", hue="browser", hue_order=browsers).
        ↪set_title("Wasm Energy Consumption by Browsers")
          plt.xlabel("Energy (Joules)")
          plt.ylim(0, 175)
          plt.xlim(0, 30)
          plt.show()
```


Wasm Energy Consumption by Browsers
browser
chrome
firefox
Count
150
100
50
0
0
10
20
30
Energy (Joules)


Wasm Energy Consumption by Browsers
browser
chrome
firefox
Count
150
100
50
0
0
10
20
30
Energy (Joules)

6.9 Descriptive Statistics

[54]:

```
data = []
for browser in browsers:
    x = df[(df['browser'] == browser) & (df['implementation'] == 'wasm')]
    mean = numpy.round(numpy.mean(x['energy']), 2)
    median = numpy.round(numpy.median(x['energy']), 2)
    min = numpy.round(numpy.amin(x['energy']), 2)
    max = numpy.round(numpy.amax(x['energy']), 2)
    std = numpy.round(numpy.std(x['energy']), 2)
    sem = numpy.round(stats.sem(x['energy']), 2)
    q1 = numpy.round(numpy.quantile(x['energy'], 0.25), 2)
    q3 = numpy.round(numpy.quantile(x['energy'], 0.75), 2)

    data.append(
        [browser, mean, std, min, q1, median, q3, max, sem]
    )

# Create the pandas DataFrame
stat = pd.DataFrame(data, columns = ['browser', 'mean', 'std', 'min', 'q1',␣
 ↪'median', 'q3', 'max', 'sem'])
# display(stat)
print(stat.to_string())

# Alternative of pandas: x['energy'].describe()
```

```
   browser  mean   std   min    q1  median    q3    max   sem
0   chrome  7.55  3.04  4.16  5.64    6.85  8.37  22.29  0.09
1  firefox  8.14  3.36  4.22  5.96    7.09  9.67  19.78  0.10
```

6.10 Descriptive Statistics Difference

[55]:

```
data = []

for browserpair in browserpairs:
    browser1 = stat[(stat['browser'] == browserpair[0])]
    browser2 = stat[(stat['browser'] == browserpair[1])]

    mean_diff = browser1.iloc[0]['mean']-browser2.iloc[0]['mean']
    median_diff = browser1.iloc[0]['median']-browser2.iloc[0]['median']
    min_diff = browser1.iloc[0]['min']-browser2.iloc[0]['min']
    max_diff = browser1.iloc[0]['max']-browser2.iloc[0]['max']
    std_diff = browser1.iloc[0]['std']-browser2.iloc[0]['std']
    sem_diff = browser1.iloc[0]['sem']-browser2.iloc[0]['sem']
    q1_diff = browser1.iloc[0]['q1']-browser2.iloc[0]['q1']
    q3_diff = browser1.iloc[0]['q3']-browser2.iloc[0]['q3']
    data.append(
        [browserpair[0] + ' vs. ' + browserpair[1],
         numpy.round(mean_diff, 2),
         numpy.round(mean_diff/browser2.iloc[0]['mean']*100, 2),
         numpy.round(median_diff, 2),
         numpy.round(median_diff/browser2.iloc[0]['median']*100, 2),
         numpy.round(min_diff, 2),
         numpy.round(min_diff/browser2.iloc[0]['min']*100, 2),
         numpy.round(max_diff, 2),
         numpy.round(max_diff/browser2.iloc[0]['max']*100, 2),
         numpy.round(std_diff, 2),
         numpy.round(std_diff/browser2.iloc[0]['std']*100, 2),
         numpy.round(sem_diff, 2),
         numpy.round(sem_diff/browser2.iloc[0]['sem']*100, 2),
         numpy.round(q1_diff, 2),
         numpy.round(q1_diff/browser2.iloc[0]['q1']*100, 2),
         numpy.round(q3_diff, 2),
         numpy.round(q3_diff/browser2.iloc[0]['q3']*100, 2),
        ]
    )

# Create the pandas DataFrame
ut = pd.DataFrame(data, columns =␣
 ↪['rq','mean_diff','mean_diff%','median_diff','median_diff%','min_diff','min_diff%','max_dif
display(ut)
#print(ut.to_string())
```

```
                      rq  mean_diff  mean_diff%  median_diff  median_diff%  \
0  chrome vs. firefox      -0.59       -7.25        -0.24         -3.39

   min_diff  min_diff%  max_diff  max_diff%  std_diff  std_diff%  sem_diff  \
0     -0.06      -1.42      2.51      12.69     -0.32      -9.52     -0.01

   sem_diff%  q1_diff  q1_diff%  q3_diff  q3_diff%
0      -10.0    -0.32     -5.37     -1.3    -13.44
```

6.11 Descriptive Statistics (By Device)

[56]:
```
data = []

for device in devices:
    for browser in browsers:
        x = df[(df['browser'] == browser) & (df['implementation'] == 'wasm') &␣
 ↪(df['device'] == device)]
        mean = numpy.round(numpy.mean(x['energy']), 2)
        median = numpy.round(numpy.median(x['energy']), 2)
        min = numpy.round(numpy.amin(x['energy']), 2)
        max = numpy.round(numpy.amax(x['energy']), 2)
        std = numpy.round(numpy.std(x['energy']), 2)
        sem = numpy.round(stats.sem(x['energy']), 2)
        q1 = numpy.round(numpy.quantile(x['energy'], 0.25), 2)
        q3 = numpy.round(numpy.quantile(x['energy'], 0.75), 2)

        data.append(
            [device, browser, mean, std, min, q1, median, q3, max, sem]
        )

# Create the pandas DataFrame
stat = pd.DataFrame(data, columns = ['device', 'browser', 'mean', 'std', 'min',␣
 ↪'q1', 'median', 'q3', 'max', 'sem'])
# display(stat)
print(stat.to_string())

# Alternative of pandas: x['energy'].describe()
```

```
     device  browser  mean   std   min    q1  median     q3    max   sem
0  SM-G991B   chrome  5.67  1.25  4.16  4.49    5.47   6.46  10.14  0.05
1  SM-G991B  firefox  6.32  2.43  4.22  4.63    5.96   6.62  16.11  0.10
2   Nexus 5   chrome  9.34  3.16  5.67  7.23    8.24  10.02  22.29  0.13
3   Nexus 5  firefox  9.91  3.19  5.79  7.63    9.04  10.99  19.78  0.13
```

6.12 Descriptive Statistics Difference (By Device)

[57]:

```
data = []

for device in devices:
    for browserpair in browserpairs:
        browser1 = stat[(stat['browser'] == browserpair[0]) & (stat['device']␣
 ↪== device)]
        browser2 = stat[(stat['browser'] == browserpair[1]) & (stat['device']␣
 ↪== device)]

        mean_diff = browser1.iloc[0]['mean']-browser2.iloc[0]['mean']
        median_diff = browser1.iloc[0]['median']-browser2.iloc[0]['median']
        min_diff = browser1.iloc[0]['min']-browser2.iloc[0]['min']
        max_diff = browser1.iloc[0]['max']-browser2.iloc[0]['max']
        std_diff = browser1.iloc[0]['std']-browser2.iloc[0]['std']
        sem_diff = browser1.iloc[0]['sem']-browser2.iloc[0]['sem']
        q1_diff = browser1.iloc[0]['q1']-browser2.iloc[0]['q1']
        q3_diff = browser1.iloc[0]['q3']-browser2.iloc[0]['q3']

        data.append(
            [
             device,
             browserpair[0] + ' vs. ' + browserpair[1],
             numpy.round(mean_diff, 2),
             numpy.round(mean_diff/browser2.iloc[0]['mean']*100, 2),
             numpy.round(median_diff, 2),
             numpy.round(median_diff/browser2.iloc[0]['median']*100, 2),
             numpy.round(min_diff, 2),
             numpy.round(min_diff/browser2.iloc[0]['min']*100, 2),
             numpy.round(max_diff, 2),
             numpy.round(max_diff/browser2.iloc[0]['max']*100, 2),
             numpy.round(std_diff, 2),
             numpy.round(std_diff/browser2.iloc[0]['std']*100, 2),
             numpy.round(sem_diff, 2),
             numpy.round(sem_diff/browser2.iloc[0]['sem']*100, 2),
             numpy.round(q1_diff, 2),
             numpy.round(q1_diff/browser2.iloc[0]['q1']*100, 2),
             numpy.round(q3_diff, 2),
             numpy.round(q3_diff/browser2.iloc[0]['q3']*100, 2),
            ]
        )

# Create the pandas DataFrame
ut = pd.DataFrame(data, columns = ['device',␣
 ↪'rq','mean_diff','mean_diff%','median_diff','median_diff%','min_diff','min_diff%','max_diff
display(ut)
#print(ut.to_string())
```

```
    device                   rq  mean_diff  mean_diff%  median_diff  \
0  SM-G991B  chrome vs. firefox      -0.65      -10.28        -0.49
1   Nexus 5  chrome vs. firefox      -0.57       -5.75        -0.80

   median_diff%  min_diff  min_diff%  max_diff  max_diff%  std_diff  \
0         -8.22     -0.06      -1.42     -5.97     -37.06     -1.18
1         -8.85     -0.12      -2.07      2.51      12.69     -0.03

   std_diff%  sem_diff  sem_diff%  q1_diff  q1_diff%  q3_diff  q3_diff%
0     -48.56     -0.05      -50.0    -0.14     -3.02    -0.16     -2.42
1      -0.94      0.00        0.0    -0.40     -5.24    -0.97     -8.83
```

[ ]: