# Executing experiments

Tuesday, November 16, 2021 1:45 PM

BasicAer has 3 backends:

1. qasm_simulator - to execute quantum experiments
2. statevector_simulator - to get state of the qubits
3. unitary_simulator - to get unitary

## Executing Experiments

```
from qiskit import BasicAer
backend = BasicAer.get_backend(simulator_name)
job = execute(qc, backend, shots, coupling_map)
result = job.result()
counts = result.get_counts()
```

Important args for execute:

1. experiment (QuantumCircuit or Schedule (pulse))
2. Backend (get properties using backend.properties() (given))
   if coupling map not provided, backend configuration is used
   else overrride backend config
3. CouplingMap - list / CouplingMap instance
   adjacency matrix to show qubit connectivity (i.e allowed CNOT configs)
   [[0,1],[0,2],[1,2]]
4. initial_layout - dict / list
   Initial position of virtual qubits on physical qubits
   ↓
   used if it is compatible with coupling constraint
   final layout may not be same as initial layout (transpiler may permute qubits)
   Allowed types - qiskit.transpiler.layout
   - dict
     virtual → physical { qr[0]: 0, qr[1]: 3, qr[2]: 5 }
     physical → virtual { 0: qr[0], 3: qr[1], 5: qr[2] }
   - list
     virtual to physical - [0, 3, 5]
     physical to virtual - [qr[0], None, None, qr[1], None, qr[2]]
                            0                  3             5
5. seed_transpiler - random seed for stochastic parts of the transpiler
6. Optimization_level - 0 - No Optimization
   1 - Light Optimization
   2 - Heavy Optimization (At the expense of transpilation)
   Default - 1
7. shots - Number of repetitions for each circuit. Default - 1024
8. seed_simulator - Random seed to control sampling when backend is simulator

Experiments can also executed by run() method

transpile → qc = transpile(qc, backend)

job = backend.run(qc, shots=1024)

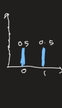
Unlike qiskit.execute(), run() method does not transpile schedule/circuit

## Return Experiment Results:

### Histogram:

```
from qiskit.visualization import plot_histogram
qc = QuantumCircuit(1)
qc.h(0)
qc.measure_all()
```

```
backend = BasicAer.get_backend('qasm_simulator')
job = execute(qc, backend, shots=1024)
result = job.result()
counts = result.get_counts()
plot_histogram(counts)
```




plot_histogram:

- data - dict / list(dict)
- figsize - default = (7,5)
- color - list of colors for bars
- number_to_keep - # of terms to plot, rest is made into single bar called rest
- sort - default - asc
  - asc → 000, 001, ...
  - desc → 111, 110, ...
  - value → sort by counts/probabilities
  - value_desc → descending order of probabilities
  - hamming
- legend - label for data
- bar_labels - T/F → display probabilities of bar if T
- title

### Statevector

```
backend = BasicAer.get_backend('statevector_simulator')
result = execute(qc, backend).result()
sv = result.get_statevector()
```

### Unitary

```
backend = BasicAer.get_backend('unitary_simulator')
result = execute(qc, backend).result()
U = result.get_unitary()
```

## Monitoring job status

```
from qiskit.tools import job_monitor
job_monitor(job)
```

Checking version - qiskit.__version__

If you want to check hardware backend info such as connectivity

```
from qiskit.tools.jupyter
%qiskit_backend_overview
```

## Accessing Aer Backends:

```
from qiskit import Aer
list of Aer Backends: Aer.backends()
```

- aer_simulator
- aer_simulator_statevector
- aer_simulator_density_matrix
- aer_simulator_stabilizer
- aer_simulator_matrix_product_state
- aer_simulator_extended_stabilizer
- aer_simulator_unitary
- aer_simulator_superop
- statevector_simulator
- qasm_simulator
- unitary_simulator
- pulse_simulator

Accessing Aer Backend - Aer.get_backend('backend_name')