Create Expectations

INTRODUCTION TO DATA QUALITY WITH GREAT EXPECTATIONS

Davina Moossazadeh
Data Scientist

datacamp

Expectations

Expectation - A verifiable assertion about data

- Column Expectations
- Shape and schema Expectations
  - **schema** - the blueprint of a dataset's structure

[1] https://docs.greatexpectations.io/docs/reference/learn/terms/expectation

Expectations

Expectation - A verifiable assertion about data

- Column Expectations
- **Shape and schema Expectations**
  - **schema** - the blueprint of a dataset's structure

[1] https://docs.greatexpectations.io/docs/reference/learn/terms/expectation

The Renewable Power Generation dataset

	Time	Energy delta[Wh]	GHI	temp	pressure	humidity	wind_speed	rain_1h	snow_1h	clouds_all
0	2021-10-26 00:30:00	0	0.0	3.5	1018	94	2.8	0.00	0.0	63
1	2020-04-21 22:30:00	0	0.0	6.3	1028	71	4.9	0.00	0.0	5
2	2021-08-05 02:15:00	0	0.0	11.5	1013	94	2.7	0.00	0.0	89
3	2017-08-28 10:15:00	4253	166.0	18.6	1020	71	2.7	0.13	0.0	37
4	2021-10-01 10:45:00	3553	113.2	14.7	1023	61	5.0	0.00	0.0	99
...	...	...	...	...	...	...	...	...	...	...
118061	2017-01-26 00:45:00	0	0.0	0.0	1034	100	0.8	0.00	0.0	93
118062	2019-10-16 23:00:00	0	0.0	11.0	1011	97	4.6	0.00	0.0	10
118063	2019-11-03 20:15:00	0	0.0	9.0	991	98	1.6	0.00	0.0	95
118064	2019-02-01 05:45:00	0	0.0	-1.7	1000	95	3.9	0.00	0.0	26
118065	2021-09-15 14:00:00	227	19.4	19.2	1012	71	3.7	0.00	0.0	100

118066 rows × 17 columns

[1] https://www.kaggle.com/datasets/pythonafroz/renewable-power-generation-and-weather-conditions

Creating an Expectation

```
gx.expectations.Expect...(...)
```

GX classes (Expectations): `PascalCase`

GX functions / methods: `snake_case`

[1] https://docs.greatexpectations.io/docs/core/define_expectations/create_an_expectation/

Creating an Expectation

```
row_count_expectation = gx.expectations.ExpectTableRowCountToEqual(
    value=118000
)
validation_results = batch.validate(
    expect=row_count_expectation
)
```

[1] https://docs.greatexpectations.io/docs/core/define_expectations/create_an_expectation/
https://docs.greatexpectations.io/docs/core/define_expectations/test_an_expectation/

Assessing an Expectation

```
print(validation_results)
```


```
{
    "success": false,
    "expectation_config": {
        "type": "expect_table_row_count_to_equal",
        "kwargs": {"batch_id": "my_pandas_datasource-my_dataframe_asset", "value": 118000},
        "meta": {},
        "rendered_content": [{"name": "atomic.prescriptive.summary", "value": {"schema": {"type": "com.supercond
    },
    "result": {"observed_value": 118066},
    "meta": {},
    "exception_info": {"raised_exception": false, "exception_traceback": null, "exception_message": null},
    "rendered_content": [{"name": "atomic.diagnostic.observed_value", "value": {"schema": {"type": "com.supercon
}
```


[1] https://docs.greatexpectations.io/docs/core/run_validations/run_a_validation_definition/

Assessing an Expectation

```
print(validation_results.describe())
```


```
{
    "expectation_type": "expect_table_row_count_to_equal",
    "success": false,
    "kwargs": {
        "batch_id": "my_pandas_datasource-my_dataframe_asset",
        "value": 118000
    },
    "result": {
        "observed_value": 118066
    }
}
```

Assessing an Expectation

```
print(validation_results.success)
```

```
False
```

```
print(validation_results["success"])
```

```
False
```

Assessing an Expectation

```
print(validation_results.result)
```

```
{'observed_value': 118066}
```

```
print(validation_results["result"])
```

```
{'observed_value': 118066}
```

Other common Expectations

Shape Expectations:

```
ExpectTableRowCountToEqual(value: int)
ExpectTableRowCountToBeBetween(
    min_value: int, max_value: int
)
ExpectTableColumnCountToEqual(
    value: int
)
ExpectTableColumnCountToBeBetween(
    min_value: int, max_value: int
)
```

Column name Expectations:

```
ExpectTableColumnsToMatchSet(
    column_set: set
)
ExpectColumnToExist(column: str)
```

Cheat sheet

Create an Expectation:

```
gx.expectations.Expect...(...)
```

Validate an Expectation:

```
validation_results = batch.validate(
    expect=expectation
)
```

Create a row count Expectation:

```
expectation = gx.expectations. \
ExpectTableRowCountToEqual(
    value: int
)
```

Check Validation Results:

```
validation_results.describe()
validation_results.success
validation_results.result
```

Let's practice!

INTRODUCTION TO DATA QUALITY WITH GREAT EXPECTATIONS

Schema Expectations

INTRODUCTION TO DATA QUALITY WITH GREAT EXPECTATIONS

Davina Moossazadeh
Data Scientist

datacamp

Shape and schema Expectations

Expectation - A verifiable assertion about data

- Column Expectations
- **Shape and schema Expectations**
  - **schema** - the blueprint of a dataset's structure

[1] https://docs.greatexpectations.io/docs/reference/learn/terms/expectation

Row count

```
row_count_expectation = gx.expectations.ExpectTableRowCountToEqual(
    value=118000
)
validation_results = batch.validate(expect=row_count_expectation)
```

```
print(validation_results.success)
```

```
False
```

```
print(validation_results.result)
```

```
{'observed_value': 118066}
```

Row count range

Use `ExpectTableRowCountToBeBetween` to define a range for **row** counts:

```
row_count_expectation = gx.expectations.ExpectTableRowCountToBeBetween(
    min_value=117000, max_value=119000
)
validation_results = batch.validate(expect=row_count_expectation)
```

```
print(validation_results.success)
```

```
True
```

```
print(validation_results.result)
```

```
{'observed_value': 118066}
```

Column count

```
col_count_expectation = gx.expectations.ExpectTableColumnCountToEqual(
    value=15
)
validation_results = batch.validate(expect=col_count_expectation)
```

```
print(validation_results.success)
```

```
False
```

```
print(validation_result.result)
```

```
{'observed_value': 18}
```

Column count range

Use `ExpectTableColumnCountToBeBetween` to define a range for **column** counts:

```
col_count_expectation = gx.expectations.ExpectTableColumnCountToBeBetween(
    min_value=14, max_value=18
)
validation_results = batch.validate(expect=col_count_expectation)
```

```
print(validation_results.success)
```

```
True
```

```
print(validation_result.result)
```

```
{'observed_value': 18}
```

Column name sets

Use `ExpectTableColumnsToMatchSet` to validate column names against a set:

```
expected_cols = ['clouds_all', 'snow_1h', 'rain_1h', 'wind_speed', 'humidity',
'pressure', 'temp', 'GHI', 'Energy delta[Wh]', 'Time', 'Time']
col_names_expectation = gx.expectations.ExpectTableColumnsToMatchSet(
    column_set=expected_cols
)
print(col_names_expectation.success, col_names_expectation.result)
```

```
True
{'observed_value': ['Time', 'Energy delta[Wh]', 'GHI', 'temp', 'pressure',
'humidity', 'wind_speed', 'rain_1h', 'snow_1h', 'clouds_all'}
```

Individual column names

Use `ExpectColumnToExist` to check if a specific column is present in the dataset:

```
col_name_expectation = gx.expectations.ExpectColumnToExist(column="not_a_column")
validation_result = batch.validate(expect=col_name_expectation)
print(validation_result.success)
```

```
False
```

```
col_name_expectation = gx.expectations.ExpectColumnToExist(column="GHI")
validation_result = batch.validate(expect=col_name_expectation)
print(validation_result.success)
```

```
True
```

Cheat sheet

Shape Expectations:

```
ExpectTableRowCountToEqual(value: int)
ExpectTableRowCountToBeBetween(
    min_value: int, max_value: int
)
ExpectTableColumnCountToEqual(
    value: int
)
ExpectTableColumnCountToBeBetween(
    min_value: int, max_value: int
)
```

Column name Expectations:

```
ExpectTableColumnsToMatchSet(
    column_set: set
)
ExpectColumnToExist(column: str)
```

Let's practice!

INTRODUCTION TO DATA QUALITY WITH GREAT EXPECTATIONS

Create a Suite of Expectations

INTRODUCTION TO DATA QUALITY WITH GREAT EXPECTATIONS

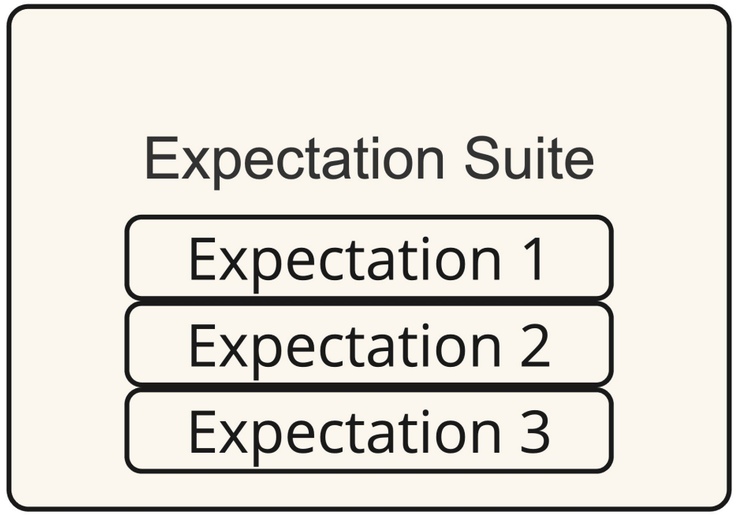
Davina Moossazadeh
Data Scientist

datacamp

Expectation Suites

Expectation - A verifiable assertion about data

Expectation Suite - A group of Expectations describing the same set of data



[1] https://docs.greatexpectations.io/docs/core/define_expectations/organize_expectation_suites/

Creating an Expectation Suite

Create an Expectation Suite named `"my_suite"` with the `ExpectationSuite` class:

```
suite = gx.ExpectationSuite(
    name="my_suite"
)
```

[1] https://docs.greatexpectations.io/docs/core/define_expectations/organize_expectation_suites/

Creating an Expectation Suite

```
print(suite)
```

```
{
  "name": "my_suite",
  "id": "a8118858-32e4-4d7d-b548-9cd7d5048958",
  "expectations": [],
  "meta": {
    "great_expectations_version": "1.2.4"
  },
  "notes": null
}
```

Adding Expectations

Add Expectations to a Suite using `.add_expectation()` (one Expectation at a time):

```
expectation = gx.expectations.ExpectTableRowCountToEqual(
    value=118000
)
```

```
suite.add_expectation(
    expectation=expectation
)
```

Adding Expectations

```
print(suite)
```


```
{ "name": "my_suite",
  "id": "a8118858-32e4-4d7d-b548-9cd7d5048958",
  "expectations": [
      {"type": "expect_table_row_count_to_equal",
       "kwargs": {"value": "118000"},
       "meta": {},
       "id": "3f1f21db-2146-417a-876e-43e11b635665"}
  ],
  "meta": {"great_expectations_version": "1.2.4"},
  "notes": null }
```

Viewing an Expectation Suite

```
print(suite.expectations)
print(suite["expectations"])
```

```
[
    ExpectTableRowCountToEqual(
        id='3f1f21db-2146-417a-876e-43e11b635665',
        meta=None, notes=None, result_format=<ResultFormat.BASIC: 'BASIC'>,
        description=None, catch_exceptions=False, rendered_content=None,
        batch_id=None, row_condition=None, condition_parser=None,
        value=118000
    ),
]
```

Viewing an Expectation Suite

```
print(suite.name)
print(suite["name"])
```

```
"my_suite"
```

```
print(suite.meta)
print(suite["meta"])
```

```
{"great_expectations_version": "1.2.4"}
```

```
print(suite.id)
print(suite["id"])
```

```
"a8118858-32e4-4d7d-b548-9cd7d5048958"
```

```
print(suite.notes)
print(suite["notes"])
```

```
None
```

Validating an Expectation Suite

Validate an Expectation Suite using `batch.validate()`, setting the `expect` parameter to the Suite:

```
validation_results = batch.validate(
    expect=suite
)
```

```
print(validation_results)
```

Validating an Expectation Suite


```
{ "success": false,
  "results": [{
      "success": false,
      "expectation_config": {
          "type": "expect_table_row_count_to_equal", "kwargs": {"batch_id": "my_datasource-my_dataframe_asset", "value": 118000}, "meta": {},
      },
      "result": {"observed_value": 118066},
      "meta": {},
      "exception_info": {"raised_exception": false, "exception_traceback": null, "exception_message": null},
      "rendered_content": [{"name": "atomic.diagnostic.observed_value", "value": {"schema": {"type": "com.superconductive.rendered.string"},
  }],
  "suite_name": "my_suite",
  "suite_parameters": {},
  "statistics": {"evaluated_expectations": 1, "successful_expectations": 0, "unsuccessful_expectations": 1, "success_percent": 0.0},
  "meta": {
    "great_expectations_version": "1.2.4",
    "batch_spec": {"batch_data": "PandasDataFrame"},
    "batch_markers": {"ge_load_time": "20241118T192902.403204Z", "pandas_data_fingerprint": "7d6363a614af65df638bdb6c053b44d3"},
    "active_batch_definition": {"datasource_name": "my_datasource", "data_connector_name": "fluent", "data_asset_name": "my_dataframe_asset",
  },
  "id": null }
```

Validating an Expectation Suite

```
print(validation_results.success)
```

```
False
```

```
print(validation_results.describe())
```

Expectation Suite Validation Results


```
{ "success": false,
  "statistics": {
    "evaluated_expectations": 1, "successful_expectations": 0,
    "unsuccessful_expectations": 1, "success_percent": 0.0
  },
  "expectations": [{
    "expectation_type": "expect_table_row_count_to_equal",
    "success": false,
    "kwargs": {"batch_id": "my_datasource-my_dataframe_asset", "value": 118000},
    "result": {"observed_value": 118066}},
  ],
  "result_url": null
}
```

Cheat sheet

Create Expectation Suite:

```
suite = gx.ExpectationSuite(name: str)
```

Add Expectation to Suite:

```
suite.add_expectation(expectation)
```

Access Suite's Expectations:

```
suite.expectations
```

Validate Expectation Suite:

```
validation_results = batch.validate(
    expect=suite
)
```

Check Validation Results:

```
validation_results.success
validation_results.describe()
```

Let's practice!

INTRODUCTION TO DATA QUALITY WITH GREAT EXPECTATIONS

Validate Expectation Suites

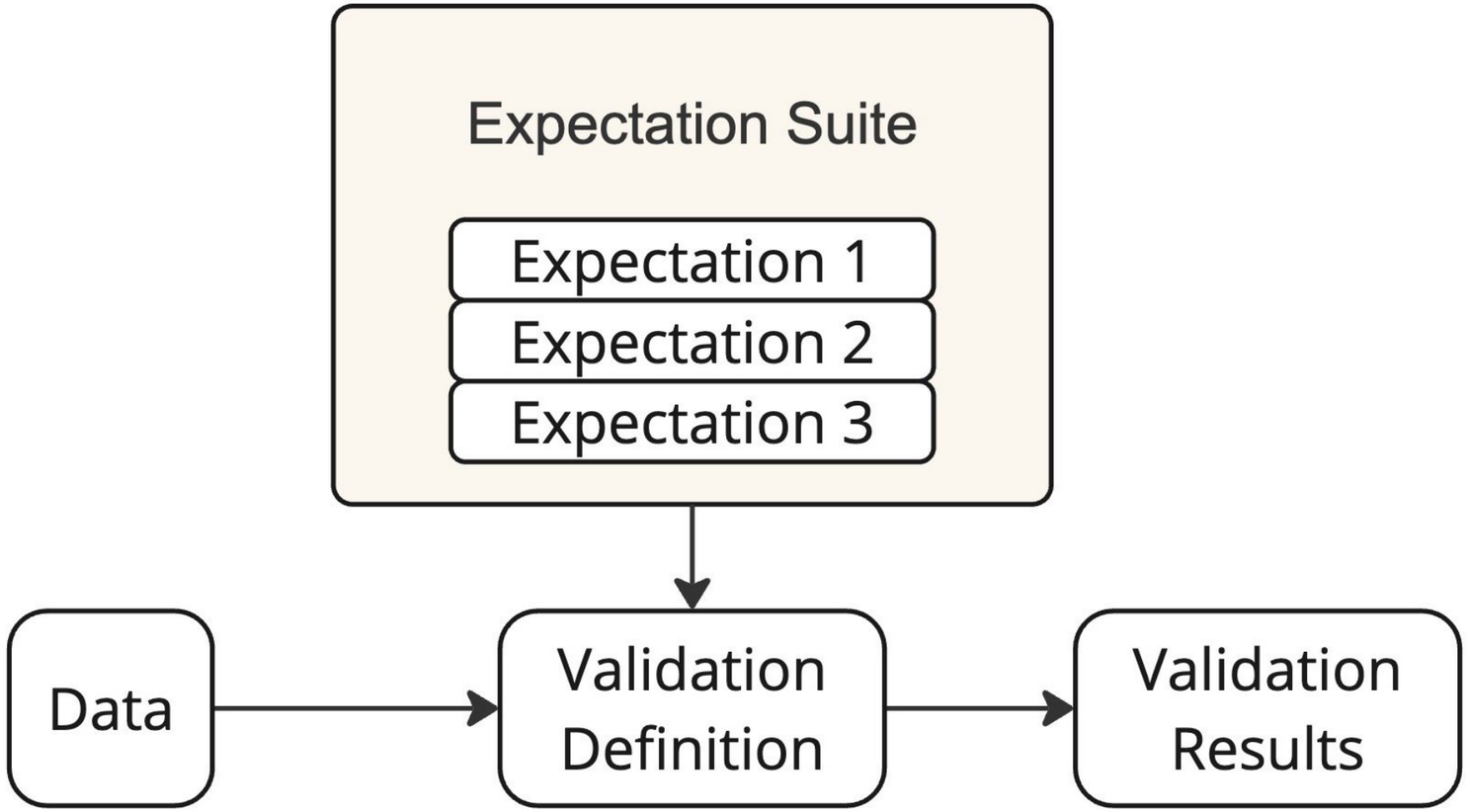
INTRODUCTION TO DATA QUALITY WITH GREAT EXPECTATIONS

Davina Moossazadeh
Data Scientist

datacamp

Validation Definitions

Validation Definition - A reference that links an Expectation Suite to data that it describes



[1] https://docs.greatexpectations.io/docs/core/run_validations/create_a_validation_definition/

Creating a Validation Definition

Create a Validation Definition with the `ValidationDefinition` class:

```
validation_definition = gx.ValidationDefinition(
    name="my_validation_definition",
    data=batch_definition,
    suite=suite,
)
```

```
print(validation_definition)
```

Viewing a Validation Definition

```
name='my_validation_definition'
data=BatchDefinition(
    id='1fcb36d6-fac6-4b9a-8ba6-a659978fd59e',
    name='my_batch_definition',
    partitioner=None
)
suite={
  "name": "my_suite",
  "id": "0a123b9c-e370-4b18-b703-785dde88732d",
  "expectations": [],
  "meta": {"great_expectations_version": "1.2.4"},
  "notes": null
}
id=None
```

Viewing a Validation Definition

```
print(validation_definition.name)
```

```
'my_validation_definition'
```

```
print(validation_definition.data)
```

```
id='1fcb36d6-fac6-4b9a-8ba6-a659978fd59e'
name='my_batch_definition'
partitioner=None
```

```
print(validation_definition.suite)
```

```
{
  "name": "my_suite",
  "id": "0a123b9c-e370-4b18-b703-785dde88732d",
  "expectations": [],
  "meta": {"great_expectations_version": "1.2.4"},
  "notes": null
}
```

```
print(validation_definition.id)
```

```
None
```

Viewing a Validation Definition

```
print(validation_definition.data_source)
```

```
assets:
  - batch_definitions:
      - name: my_batch_definition
        partitioner: null
    batch_metadata: {}
    id: 83682084-3bc4-4898-a807-fadc0f911415
    name: 'my_dataframe_asset'
    type: dataframe
id: f71d275e-a5b2-402e-a53c-8dad6975cce5
name: 'my_pandas_data_source'
type: pandas
```

Running a Validation Definition

Run a Validation using the Validation Definition's `.run()` method, passing the DataFrame via `batch_parameters`:

```
validation_results = validation_definition.run(
    batch_parameters={"dataframe": dataframe}
)
```

Validation Definition errors

Note the error:

```
ValidationDefinitionRelatedResourcesFreshnessError:
ExpectationSuite 'my_suite' must be added to the DataContext before it can be
updated. Please call `context.suites.add(<SUITE_OBJECT>)`, then try your action
again.
```

Running a Validation Definition before adding the Expectation Suite to the Data Context raises an error

Adding an Expectation Suite

Add an Expectation Suite to the Data Context using `.suites.add()`:

```
suite = context.suites.add(
    suite=suite
)
```

Assessing a Validation Definition

```
validation_results = validation_definition.run(
    batch_parameters={"dataframe": dataframe}
)
```

```
Calculating Metrics: 0/0 [00:00<?, ?it/s]
```

```
print(validation_results.success)
```

```
False
```

```
print(validation_results.describe())
```

Assessing a Validation Definition


```
{ "success": false,
  "statistics": {
    "evaluated_expectations": 1, "successful_expectations": 0,
    "unsuccessful_expectations": 1, "success_percent": 0.0
  },
  "expectations": [{
    "expectation_type": "expect_table_row_count_to_equal",
    "success": false,
    "kwargs": {"batch_id": ""my_datasource-my_dataframe_asset", "value": 118000},
    "result": {"observed_value": 11866}}
  ],
  "result_url": "https://app.greatexpectations.io/organizations/my_org/data-assets/*
}
```

A note about GX

- Great Expectations offers multiple workflows for similar tasks
  - e.g., Batch Definitions vs. Validation Definitions
- This course provides a broad understanding, but alternative GX implementations may use different approaches

Cheat sheet

Add Expectation Suite to Data Context:

```
context.suites.add(suite)
```

Create Validation Definition:

```
validation_definition = \
gx.ValidationDefinition(
  name: str,
  data=batch_definition,
  suite=suite
)
```

Run Validation:

```
validation_results = \
validation_definition.run(
  batch_parameters={"dataframe": dataframe}
)
```

Check Validation Results:

```
validation_results.success
validation_results.describe()
```

Let's practice!

INTRODUCTION TO DATA QUALITY WITH GREAT EXPECTATIONS