

HCP Analyzing Data with Statistics 1

Analyzing Data with Statistics

Date of Publish: 2018-12-21



<https://docs.hortonworks.com/>

Contents

Analyzing Data Using Statistical and Mathematical Functions.....	3
Approximation Statistics.....	3
Mathematical Functions.....	3
Distributional Statistics.....	4
Statistical Outlier Detection.....	5
Outlier Analysis.....	6
Median Absolution Deviation.....	6
Example.....	6

Analyzing Data Using Statistical and Mathematical Functions

HCP provides a variety of advanced analytics that use statistics and advanced mathematical functions. Capturing the statistical snapshots in a scalable way can open up doors for more advanced analytics such as outlier analysis. These analytics provide a robust set of statistical functions and statistical-based algorithms in the form of Stellar functions. These functions can be used from everywhere where Stellar is used.

Approximation Statistics

Data scientists frequently want to find anomalies in numerical data. To that end, HCP has some simple approximation statistics.

Table 1: Approximation Statistics

Function	Description	Input	Returns
HLLP_ADD	Add value to the HyperLogLogPlus estimator set.	<ul style="list-style-type: none"> hyperLogLogPlus - The hllp estimator to add a value to value+ - Value to add to the set. Takes a single item or a list. 	The HyperLogLogPlus set with a new value added
HLLP_CARDINALITY	Returns HyperLogLogPlus-estimated cardinality for this set.	<ul style="list-style-type: none"> hyperLogLogPlus - The hllp set 	Long value representing the cardinality for this set
HLLP_INIT	Initializes the HyperLogLogPlus estimator set. p must be a value between 4 and sp and sp must be less than 32 and greater than 4.	<ul style="list-style-type: none"> p - The precision value for the normal set sp - The precision value for the sparse set. If p is set, but sp is 0 or not specified, the sparse set will be disabled. 	A new HyperLogLogPlus set
HLLP_MERGE	Merge hllp sets together. The resulting estimator is initialized with p and sp precision values from the first provided hllp estimator set.	<ul style="list-style-type: none"> hllp - List of hllp estimators to merge. Takes a single hllp set or a list. 	True if the filter might contain the value and false otherwise

Mathematical Functions

Data scientists frequently want to find anomalies in numerical data. To that end, HCP has some mathematical functions.

Table 2: Mathematical Functions

Function	Description	Input	Returns
ABS	Returns the absolute value of a number.	<ul style="list-style-type: none"> number - The number to take the absolute value of 	The absolute value of the number passed in
BIN	Computes the bin that the value is in given a set of bounds	<ul style="list-style-type: none"> value -the value to bin bounds - A list of value bounds (excluding min and max) in sorted order 	Which bin N the value falls in such that $\text{bound}(N-1) < \text{value} \leq \text{bound}(N)$. No min and max bounds are provided, so values smaller than the 0'th bound go in the 0'th bin, and values greater than the last bound go in the M'th bin.

Distributional Statistics

Data scientists frequently want to find anomalies in numerical data. To that end, HCP has some simple distributional statistics.

Table 3: Distributional Statistics

Function	Description	Input	Returns
STATS_ADD	Adds one or more input values to those that are used to calculate the summary statistics.	<ul style="list-style-type: none"> stats - The Stellar statistics object. If null, then a new one is initialized. value+ - One or more numbers to add 	A Stellar statistics object
STATS_BIN	Computes the bin that the value is in based on the statistical distribution.	<ul style="list-style-type: none"> stats - The Stellar statistics object value - The value to bin bounds? - A list of percentile bin bounds (excluding min and max) or a string representing a known and common set of bins. For convenience, we have provided QUARTILE, QUINTILE, and DECILE which you can pass in as a string arg. If this argument is omitted, then we assume a Quartile bin split. 	Which bin N the value falls in such that $\text{bound}(N-1) < \text{value} \leq \text{bound}(N)$. No min and max bounds are provided, so values smaller than the 0'th bound go in the 0'th bin, and values greater than the last bound go in the M'th bin.
STATS_COUNT	Calculates the count of the values accumulated (or in the window if a window is used).	<ul style="list-style-type: none"> stats - The Stellar statistics object 	The count of the values in the window or NaN if the statistics object is null
STATS_GEOMETRIC_MEAN	Calculates the geometric mean of the accumulated values (or in the window if a window is used).	<ul style="list-style-type: none"> stats - The Stellar statistics object 	The geometric mean of the values in the window or NaN if the statistics object is null.
STATS_INIT	Initializes a statistics object	<ul style="list-style-type: none"> window size - The number of input data values to maintain in a rolling window in memory. If window_size is equal to 0, then no rolling window is maintained. Using no rolling window is less memory intensive, but cannot calculate certain statistics like percentiles and kurtosis. 	A Stellar statistics object
STATS_KURTOSIS	Calculates the kurtosis of the accumulated values (or in the window if a window is used).	<ul style="list-style-type: none"> stats - The Stellar statistics object 	The kurtosis of the values in the window or NaN if the statistics object is null
STATS_MAX	Calculates the maximum of the accumulated values (or in the window if a window is used).	<ul style="list-style-type: none"> stats - The Stellar statistics object 	The maximum of the accumulated values in the window or NaN if the statistics object is null.
STATS_MEAN	Calculates the mean of the accumulated values (or in the window if a window is used).	<ul style="list-style-type: none"> stats - The Stellar statistics object 	The mean of the values in the window or NaN if the statistics object is null.
STATS_MERGE	Merges statistics objects	<ul style="list-style-type: none"> statistics - A list of statistics objects 	A Stellar statistics object
STATS_MIN	Calculates the minimum of the accumulated values (or in the window if a window is used).	<ul style="list-style-type: none"> stats - The Stellar statistics object 	The minimum of the accumulated values in the window or NaN if the statistics object is null.

Function	Description	Input	Returns
STATS_PERCENTILE	Computes the p'th percentile of the accumulated values (or in the window if a window is used).	<ul style="list-style-type: none"> stats - The Stellar statistics object p - A double where $0 \leq p < 1$ representing the percentile 	The p'th percentile of the data or NaN if the statistics object is null
STATS_POPULATION_VARIANCE	Calculates the population variance of the accumulated values (or in the window if a window is used).	<ul style="list-style-type: none"> stats - The Stellar statistics object 	The population variance of the values in the window or NaN if the statistics object is null.
STATS_QUADRATIC_MEAN	Calculates the quadratic mean of the accumulated values (or in the window if a window is used).	<ul style="list-style-type: none"> stats - The Stellar statistics object 	The quadratic mean of the values in the window or NaN if the statistics object is null.
STATS_SD	Calculates the standard deviation of the accumulated values (or in the window if a window is used).	<ul style="list-style-type: none"> stats - The Stellar statistics object 	The standard deviation of the values in the window or NaN if the statistics object is null.
STATS_SKEWNESS	Calculates the skewness of the accumulated values (or in the window if a window is used).	<ul style="list-style-type: none"> stats - The Stellar statistics object 	The skewness of the values in the window or NaN if the statistics object is null.
STATS_SUM	Calculates the sum of the accumulated values (or in the window if a window is used).	<ul style="list-style-type: none"> stats - The Stellar statistics object 	The sum of the values in the window or NaN if the statistics object is null.
STATS_SUM_LOGS	Calculates the sum of the (natural) log of the accumulated values (or in the window if a window is used).	<ul style="list-style-type: none"> stats - The Stellar statistics object 	The sum of the (natural) log of the values in the window or NaN if the statistics object is null.
STATS_SUM_SQUARES	Calculates the sum of the squares of the accumulated values (or in the window if a window is used).	<ul style="list-style-type: none"> stats - The Stellar statistics object 	The sum of the squares of the values in the window or NaN if the statistics object is null.
STATS_VARIANCE	Calculates the variance of the accumulated values (or in the window if a window is used).	<ul style="list-style-type: none"> stats - The Stellar statistics object 	The variance of the values in the window or NaN if the statistics object is null.

Statistical Outlier Detection

Data scientists frequently want to find anomalies in numerical data. To that end, HCP has some simple statistical outlier detectors.

Table 4: Statistical Outlier Detection

Function	Description	Input	Returns
OUTLIER_MAD_STATE_MERGE	Update the statistical state required to compute the Median Absolute Deviation.	<ul style="list-style-type: none"> state - A list of Median Absolute Deviation States to merge. Generally these are states across time. currentState? - The current state (optional) 	The Median Absolute Deviation state
OUTLIER_MAD_ADD	Add a piece of data to the state	<ul style="list-style-type: none"> state - The MAD state value - The numeric value to add 	The MAD state
OUTLIER_MAD_SCORE	Get the modified z-score normalized by the MAD: $\text{scale} * x_i - \text{median}(X) / \text{MAD}$.	<ul style="list-style-type: none"> state - The MAD state value - The numeric value to score scale? - Optionally the scale to use when computing the modified z-score. Default is 0.6745. 	The modified z-score

Outlier Analysis

Data scientists frequently want to find anomalies in numerical data. To that end, HCP has some simple statistical anomaly detectors.

Median Absolution Deviation

Much has been written about this robust estimator. See the first page of *Alternatives to the Median Absolute Deviation* for coverage of the good and the bad of median absolute deviation (MAD). The usage, however is fairly straightforward.

- Gather the statistical state required to compute the MAD.
 - The distribution of the values of a univariate random variable over time.
 - The distribution of the absolute deviations of the values from the median.
- Use this statistical state to score unseen values. The higher the score, the more unlike the previously seen data the value is.

There are a couple of issues which make MAD hard to compute. First, the statistical state requires computing median, which can be computationally expensive to compute exactly. To get around this, we use the `OnlineStatisticalProvider` to compute a sketch rather than the exact median. Secondly, the statistical state for seasonal data should be limited to a fixed, trailing window. We do this by ensuring that the MAD state is mergeable and able to be queried from within the Profiler.

Example

To illustrate how to use the MAD functionality we will create a dummy data stream of Gaussian noise. This example will also explain how to use the profiler to tag messages as outliers or not.

Data Generator

We can create a simple python script to generate a stream of Gaussian noise at the frequency of one message per second as a python script which should be saved at `~/rand_gen.py`.

```
#!/usr/bin/python
import random
import sys
import time
def main():
    mu = float(sys.argv[1])
    sigma = float(sys.argv[2])
    freq_s = int(sys.argv[3])
    while True:
        print str(random.gauss(mu, sigma))
        sys.stdout.flush()
        time.sleep(freq_s)

if __name__ == '__main__':
    main()
```

This script will take the following as arguments:

- The mean of the data generated
- The standard deviation of the data generated
- The frequency (in seconds) of the data generated

If, however, you'd like to test a longer tailed distribution, like the student t-distribution and have numpy installed, you can use the following as `~/rand_gen.py`:

```
#!/usr/bin/python
import random
```

```
import sys
import time
import numpy as np

def main():
    df = float(sys.argv[1])
    freq_s = int(sys.argv[2])
    while True:
        print str(np.random.standard_t(df))
        sys.stdout.flush()
        time.sleep(freq_s)

if __name__ == '__main__':
    main()
```

This script will take the following as arguments:

- The degrees of freedom for the distribution
- The frequency (in seconds) of the data generated

The Parser

We will create a parser that will take the single numbers in and create a message with a field called value in them using the CSVParser.

Add the following file to \$METRON_HOME/config/zookeeper/parsers/mad.json:

```
{
  "parserClassName" : "org.apache.metron.parsers.csv.CSVParser"
  , "sensorTopic" : "mad"
  , "parserConfig" : {
    "columns" : {
      "value_str" : 0
    }
  }
  , "fieldTransformations" : [
    {
      "transformation" : "STELLAR"
      , "output" : [ "value" ]
      , "config" : {
        "value" : "TO_DOUBLE(value_str)"
      }
    }
  ]
}
```

Enrichment and Threat Intelligence

We will set a threat triage level of 10 if a message generates an outlier score of more than 3.5. This cutoff will depend on your data and should be adjusted based on the assumed underlying distribution. Note that under the assumptions of normality, MAD will act as a robust estimator of the standard deviation, so the cutoff should be considered the number of standard deviations away.

For other distributions, there are other interpretations which will make sense in the context of measuring the "degree different".

Create the following in \$METRON_HOME/config/zookeeper/enrichments/mad.json:

```
{
  "enrichment": {
    "fieldMap": {
      "stellar": {
        "config": {
          "parser_score" : "OUTLIER_MAD_SCORE(OUTLIER_MAD_STATE_MERGE("
```

```

PROFILE_GET( 'sketchy_mad', 'global', PROFILE_FIXED(10, 'MINUTES')) ),
  value)"
    },
    "is_alert" : "if parser_score > 3.5 then true else is_alert"
  }
}
},
"fieldToTypeMap": { }
},
"threatIntel": {
  "fieldMap": { },
  "fieldToTypeMap": { },
  "triageConfig" : {
    "riskLevelRules" : [
      {
        "rule" : "parser_score > 3.5",
        "score" : 10
      }
    ],
    "aggregator" : "MAX"
  }
}
}
}

```

Index

We also need an indexing configuration.

Create the following in `$METRON_HOME/config/zookeeper/enrichments/mad.json`:

```

{
  "hdfs" : {
    "index": "mad",
    "batchSize": 1,
    "enabled" : true
  },
  "elasticsearch" : {
    "index": "mad",
    "batchSize": 1,
    "enabled" : true
  }
}

```

The Profiler

We can set up the profiler to track the MAD statistical state required to compute MAD. For the purposes of this demonstration, we will configure the profiler to capture statistics on the minute mark. We will capture a global statistical state for the value field and we will look back for a 5 minute window when computing the median.

Create the following file at `$METRON_HOME/config/zookeeper/profiler.json`:

```

{
  "profiles": [
    {
      "profile": "sketchy_mad",
      "foreach": "'global'",
      "onlyif": "true",
      "init" : {
        "s": "OUTLIER_MAD_STATE_MERGE(PROFILE_GET('sketchy_mad',
'global', PROFILE_FIXED(5, 'MINUTES')))"
      },
      "update": {
        "s": "OUTLIER_MAD_ADD(s, value)"
      },
      "result": "s"
    }
  ]
}

```



```
}  
  ]  
}
```

Adjust `$METRON_HOME/config/zookeeper/global.json` to adjust the capture duration:

```
"profiler.client.period.duration" : "1",  
"profiler.client.period.duration.units" : "MINUTES"
```

Adjust `$METRON_HOME/config/profiler.properties` to adjust the capture duration by changing `profiler.period.duration=15` to `profiler.period.duration=1`

Execute the Flow

When you have finished configuring the MAD functionality, you will need to execute the flow.

Procedure

1. Install the elasticsearch head plugin by executing: `/usr/share/elasticsearch/bin/plugin install mobz/elasticsearch-head`.
2. Stop all other parser topologies via `monit`.
3. Create the mad Kafka topic by executing: `/usr/hdp/current/kafka-broker/bin/kafka-topics.sh --zookeeper node1:2181 --create --topic mad --partitions 1 --replication-factor 1`.
4. Push the modified configs by executing: `$METRON_HOME/bin/zk_load_configs.sh --mode PUSH -z node1:2181 -i $METRON_HOME/config/zookeeper/`.
5. Start the profiler by executing: `$METRON_HOME/bin/start_profiler_topology.sh`.
6. Start the parser topology by executing: `$METRON_HOME/bin/start_parser_topology.sh -k node1:6667 -z node1:2181 -s mad`.
7. Ensure that the enrichment and indexing topologies are started. If not, then start those via `monit` or by hand.
8. Generate data into kafka by executing the following for at least 10 minutes: `~/rand_gen.py 0 1 1 | /usr/hdp/current/kafka-broker/bin/kafka-console-producer.sh --broker-list node1:6667 --topic mad`. Note: If you chose to use the `t-distribution` script above, you would adjust the parameters of the `rand_gen.py` script accordingly.
9. Stop the above with `ctrl-c` and send an obvious outlier into kafka: `echo "1000" | /usr/hdp/current/kafka-broker/bin/kafka-console-producer.sh --broker-list node1:6667 --topic mad`.

You should be able to find the outlier via the elasticsearch head plugin by searching for the messages where `is_alert` is true.