PromQL for Mere Mortals

Ian Billett
Software Engineer
ianbillett@improbable.io
/billett_
Date 07/11/19

IMPROBABLE

Agenda

- Intro
  PromQL is important
- Time Series
  Series-ly cool
- Vector Selectors
  Range vs Instant
- Gauges & Counters
  How en-gauge-ing
- Operators & Functions
  Functionally operational
- Demo!
  Pray to the demo gods

IMPROBABLE

PromQL is Important

- Queries
- Alerts
- Dashboards

IMPROBABLE

...but also intimidating

```
label_replace((sum by(job, env, cluster,
instance) (exthttp_requests_total{ code=~"^5..$",
err_category!="Transient", handler!="/health",
job="prometheus"} ) or on(job, env, cluster,
instance) sum by(job, env, cluster, instance)
((up{ job="prometheus"} == 1) * 0)), "code",
"5xx", "job", ".*")
```

IMPROBABLE

Time Series

```
identifier -> (t0, v0), (t1, v1), (t2, v2), (t3, v3), ....
```

```
http_requests_total{path="/status", method="GET"}
```

```
{__name__="http_requests_total", path="/status",
                                 method="GET"}
```

IMPROBABLE

PromQL

Four data types:

- Strings ✔️
- Scalars ✔️
- Instant Vectors 🤷‍♀️
- Range Vectors 🤷‍♀️

IMPROBABLE

Some context...

- SQL ✔️
- JQL ✔️ 😳
- PromQL ... ?

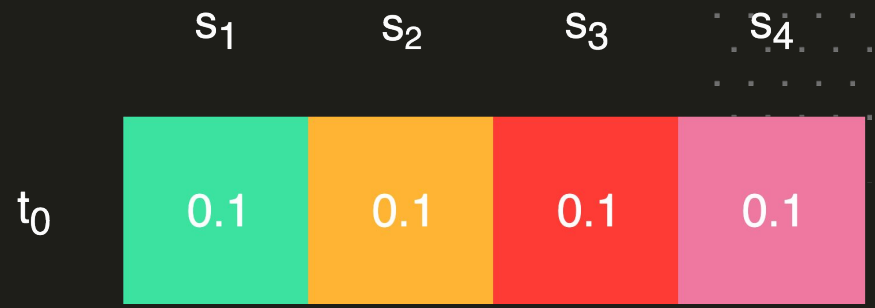
IMPROBABLE

Instant Vectors

Instant vector selectors allow the selection of a set of time series and a single sample value for each at a given timestamp (instant): in the simplest form, only a metric name is specified. This results in an instant vector containing elements for all time series that have this metric name.

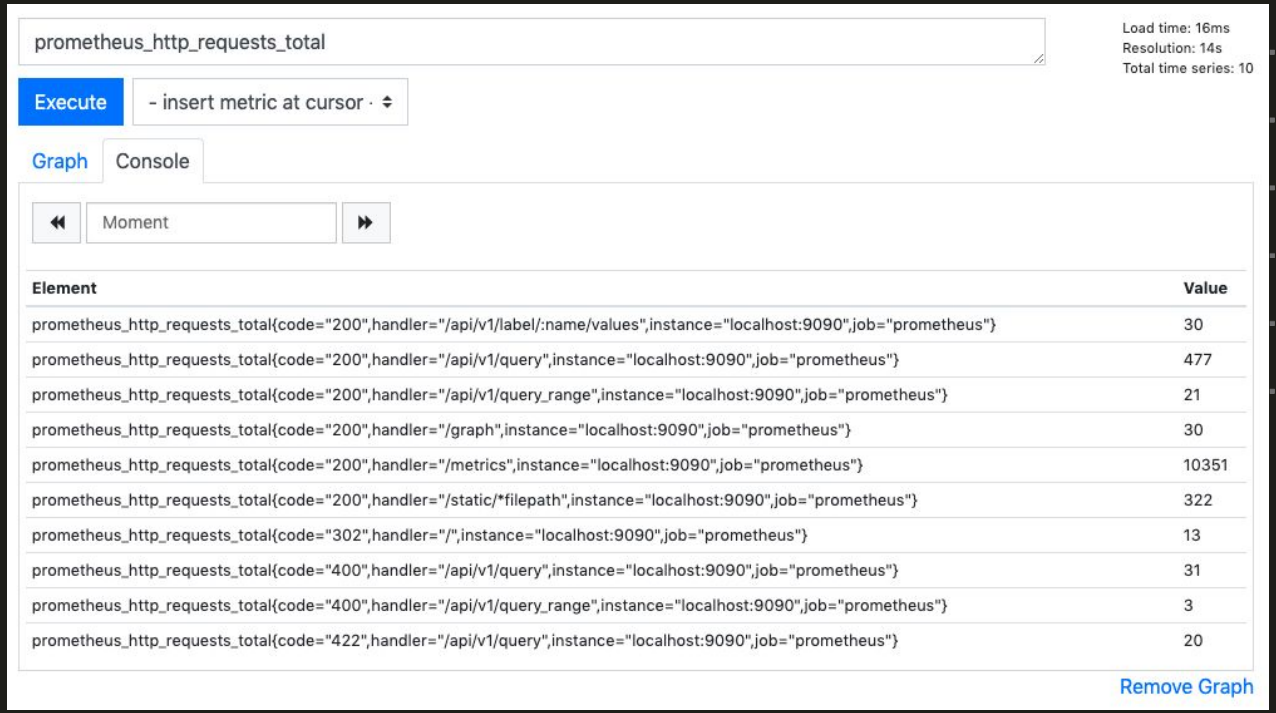
IMPROBABLE

Instant Vectors



IMPROBABLE

Instant Vectors



Element	Value
prometheus_http_requests_total{code="200",handler="/api/v1/label/:name/values",instance="localhost:9090",job="prometheus"}	30
prometheus_http_requests_total{code="200",handler="/api/v1/query",instance="localhost:9090",job="prometheus"}	477
prometheus_http_requests_total{code="200",handler="/api/v1/query_range",instance="localhost:9090",job="prometheus"}	21
prometheus_http_requests_total{code="200",handler="/graph",instance="localhost:9090",job="prometheus"}	30
prometheus_http_requests_total{code="200",handler="/metrics",instance="localhost:9090",job="prometheus"}	10351
prometheus_http_requests_total{code="200",handler="/static/*filepath",instance="localhost:9090",job="prometheus"}	322
prometheus_http_requests_total{code="302",handler="/",instance="localhost:9090",job="prometheus"}	13
prometheus_http_requests_total{code="400",handler="/api/v1/query",instance="localhost:9090",job="prometheus"}	31
prometheus_http_requests_total{code="400",handler="/api/v1/query_range",instance="localhost:9090",job="prometheus"}	3
prometheus_http_requests_total{code="422",handler="/api/v1/query",instance="localhost:9090",job="prometheus"}	20

IMPROBABLE

Range Vectors

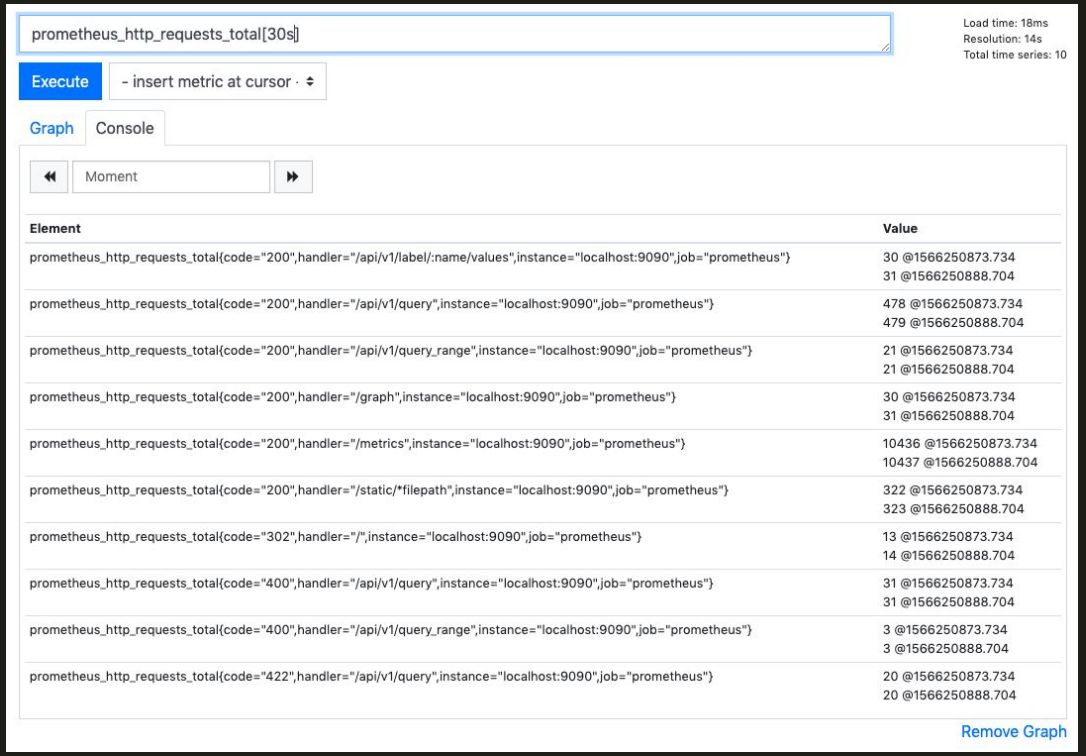
Range vector literals work like instant vector literals, except that they select a range of samples back from the current instant. Syntactically, a range duration is appended in square brackets (`[]`) at the end of a vector selector to specify how far back in time values should be fetched for each resulting range vector element.

IMPROBABLE

Range Vectors

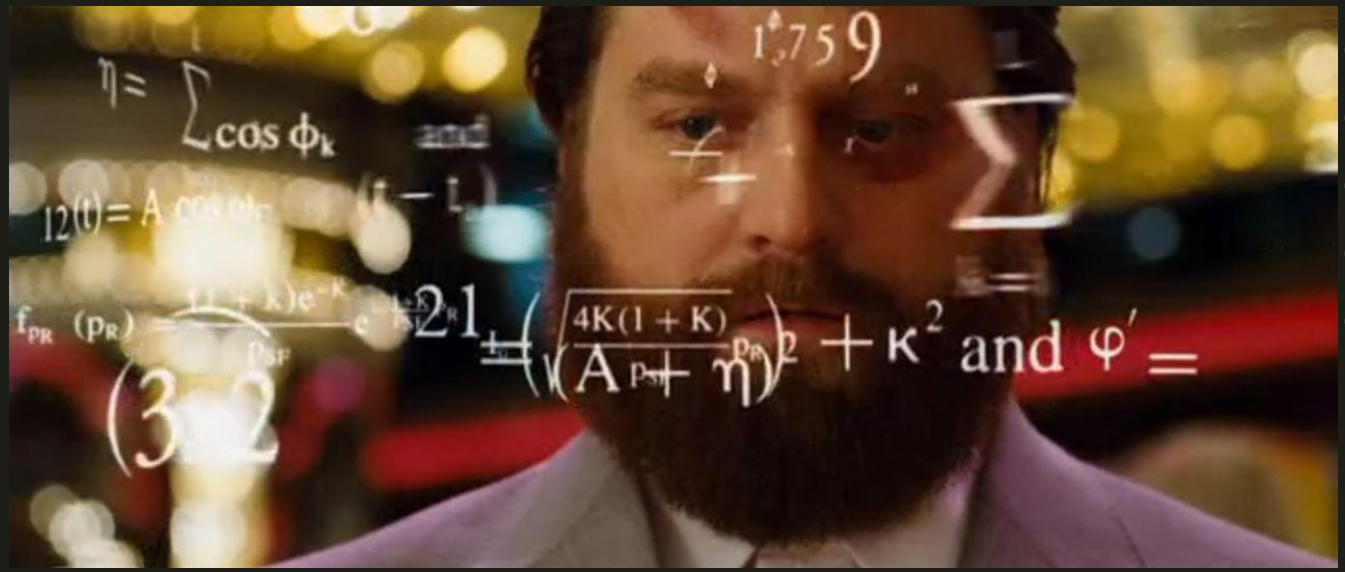
IMPROBABLE

Range Vectors



IMPROBABLE

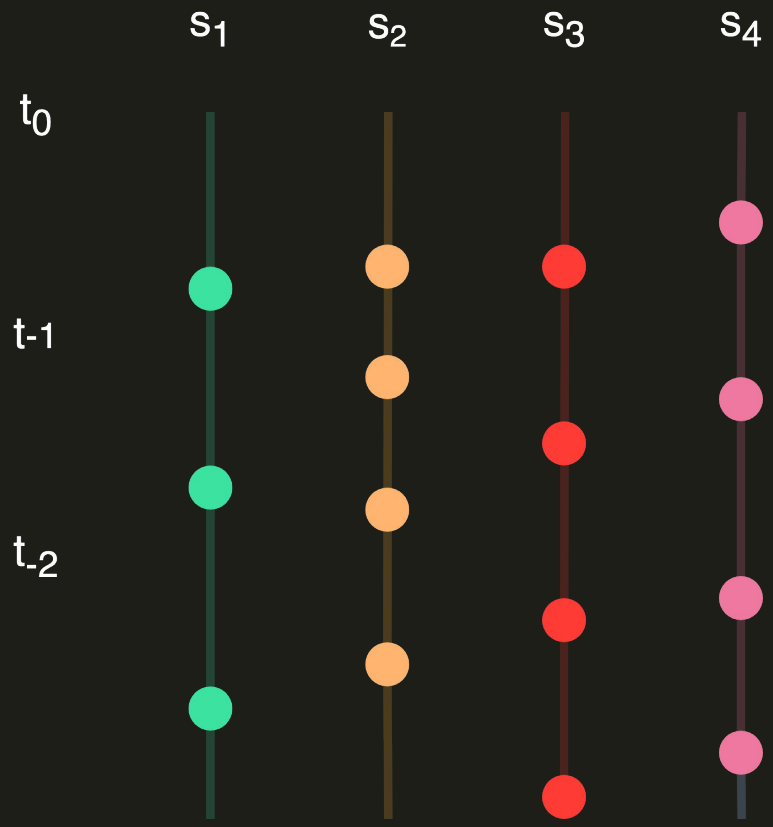
Are they really that different?



Yes

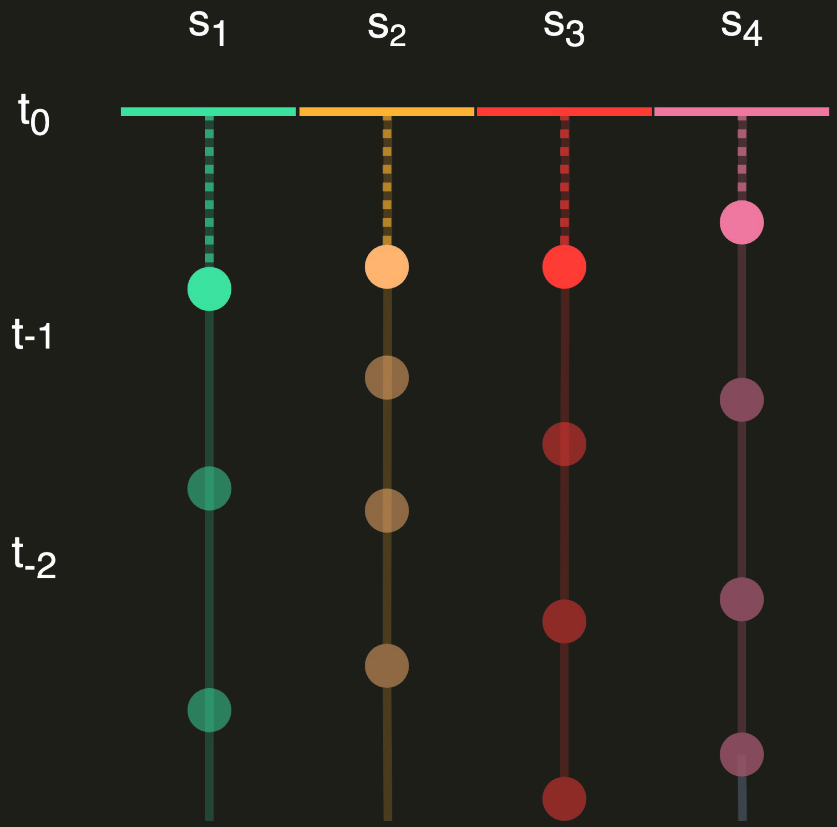
IMPROBABLE

Data Model



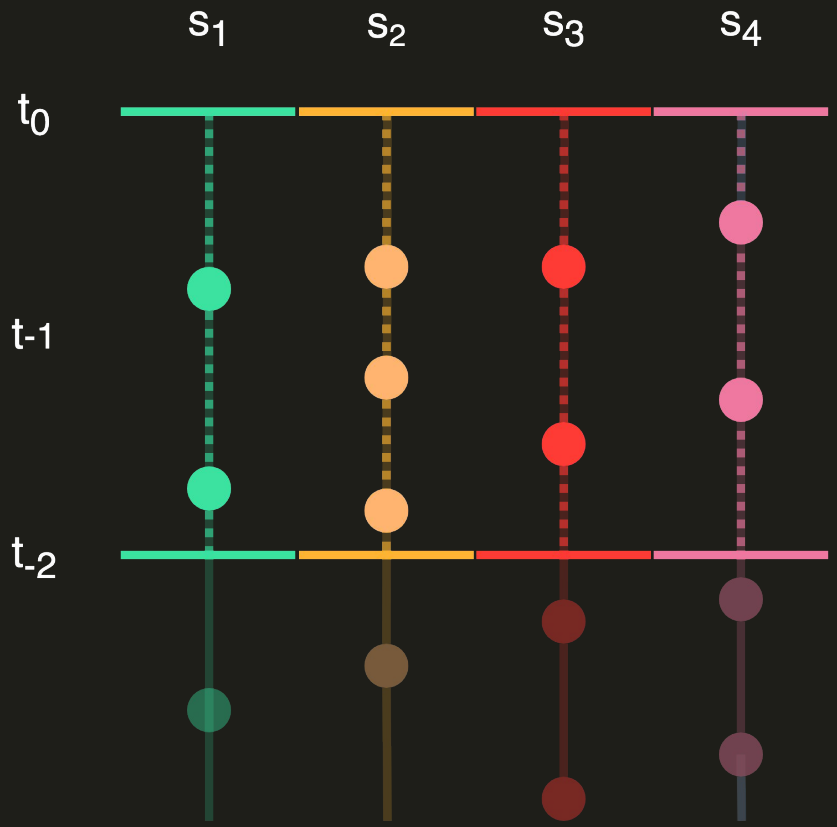
IMPROBABLE

Instant Vector



IMPROBABLE

Range Vector



IMPROBABLE

Instant Vector

One value per time series guaranteed.

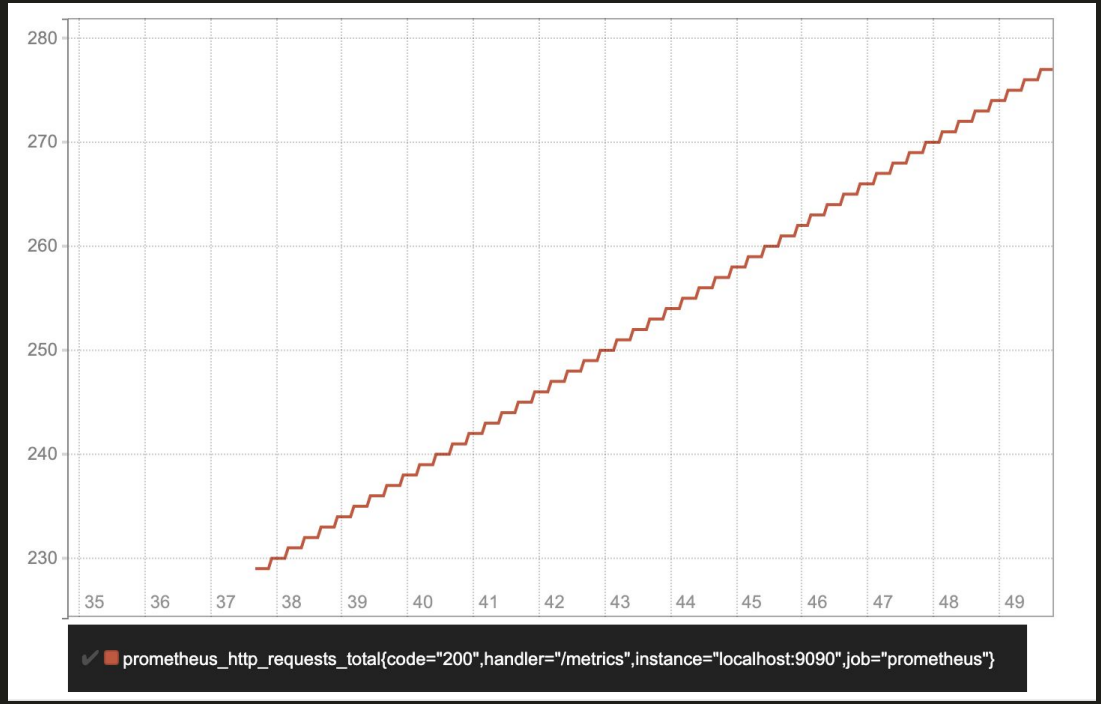
Range Vector

Any number of values between two timestamps.

IMPROBABLE

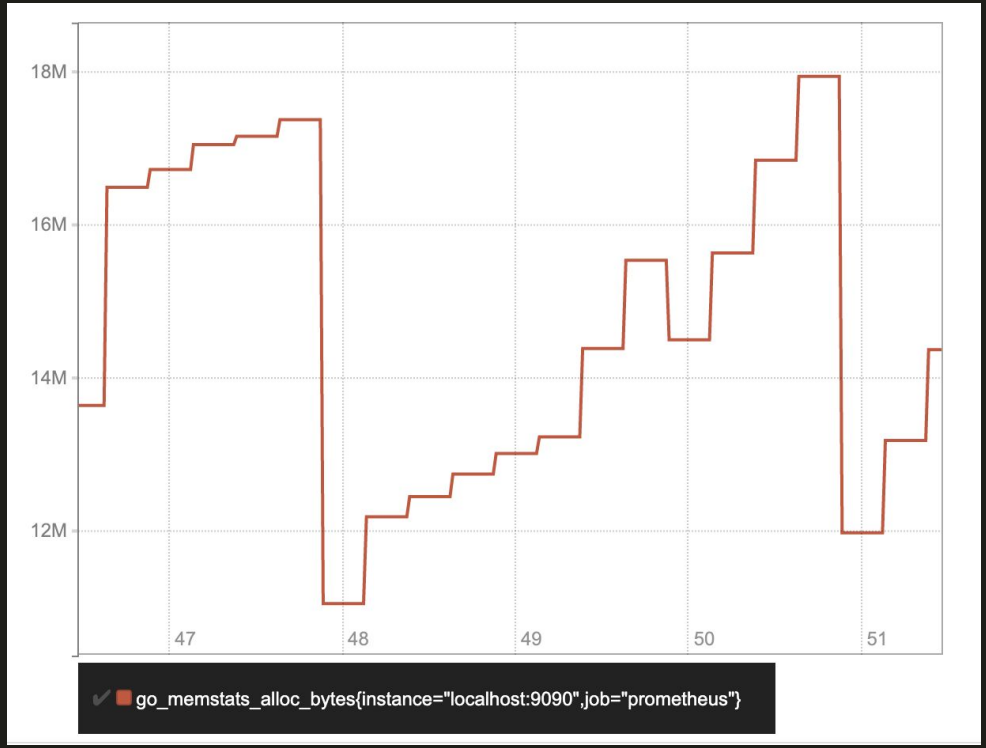
Counters & Gauges

Counters



IMPROBABLE

Gauges



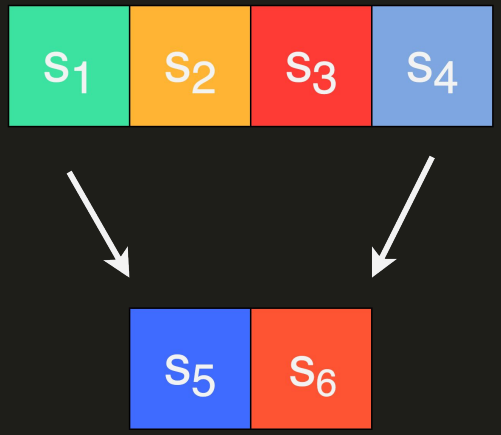
IMPROBABLE

IMPROBABLE

Aggregation Operators

Input: Instant Vector

Output: Instant Vector



- sum()
- min()
- max()
- stddev()
- stdvar()

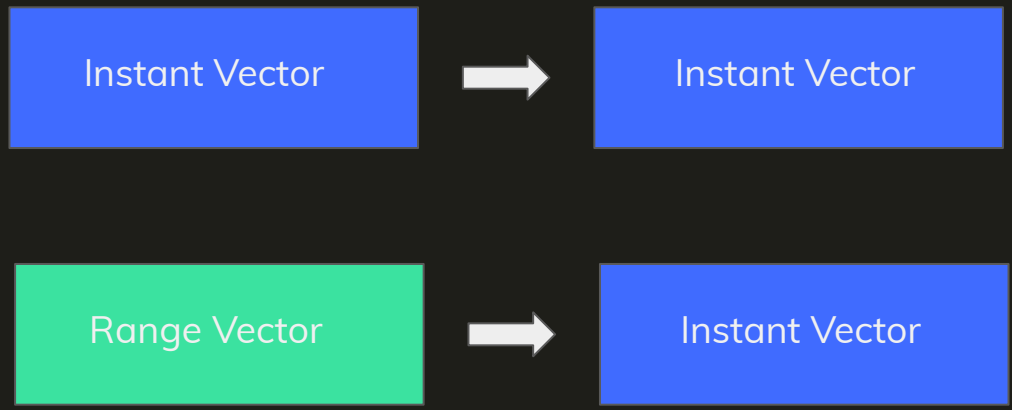
IMPROBABLE

Binary Operators

- Arithmetic Binary Operators: +, -, /, *, ^, %,
- Comparison Binary Operators: !=, ==, <, <=, >, >=
- Set Binary Operators: and, or, unless

IMPROBABLE

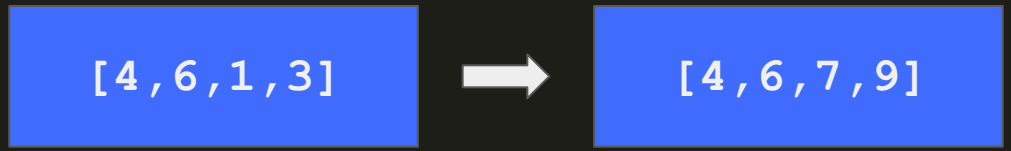
Functions



IMPROBABLE

rate(v range-vector)

```
rate(http_requests_total[10m])
```



IMPROBABLE

Demo time!

Gotchas

- Tell your aggregation operators about the labels you care about!
- Never compare raw counters - use rate()
- Be careful with label sets when using binary operators.

IMPROBABLE

Resources

- https://www.robustperception.io Blog
- Prometheus Up & Running Book
- https://prometheus.io/docs/prometheus/ documentation

IMPROBABLE

↳ www.improbable.io