Model-Based Approximate Query Processing

Semester Project Presentation

Futong Liu

Introduction

Background

- **Motivation**: In large-scale databases, it's expensive to compute the exact result. However, in many applications, an approximate answer is enough. For example, interactive visualisation of a dataset, IoT datasets.

- **AQP**: a framework that can timely and approximately answer a query within a fixed time bound and error bound.

- A common **template** SQL statement in AQP:

```
SELECT agg_func(y)
FROM Table
WHERE x BETWEEN lb AND ub
```

`agg_func`: COUNT, AVG, SUM, etc.
Column x: numerical or at least ordinal
Column y: numerical

Sampling-Based AQP

- **Online Sampling**
  - + supports ad-hoc queries
  - - expensive
  - - inaccurate or absent on rare populations
- **Offline Sampling** (eg. Stratified Sampling)
  - Biased sampling
  - Requires predictable queries

Model-Based AQP

Non-Parametric Density Estimation

Histogram

- not smooth

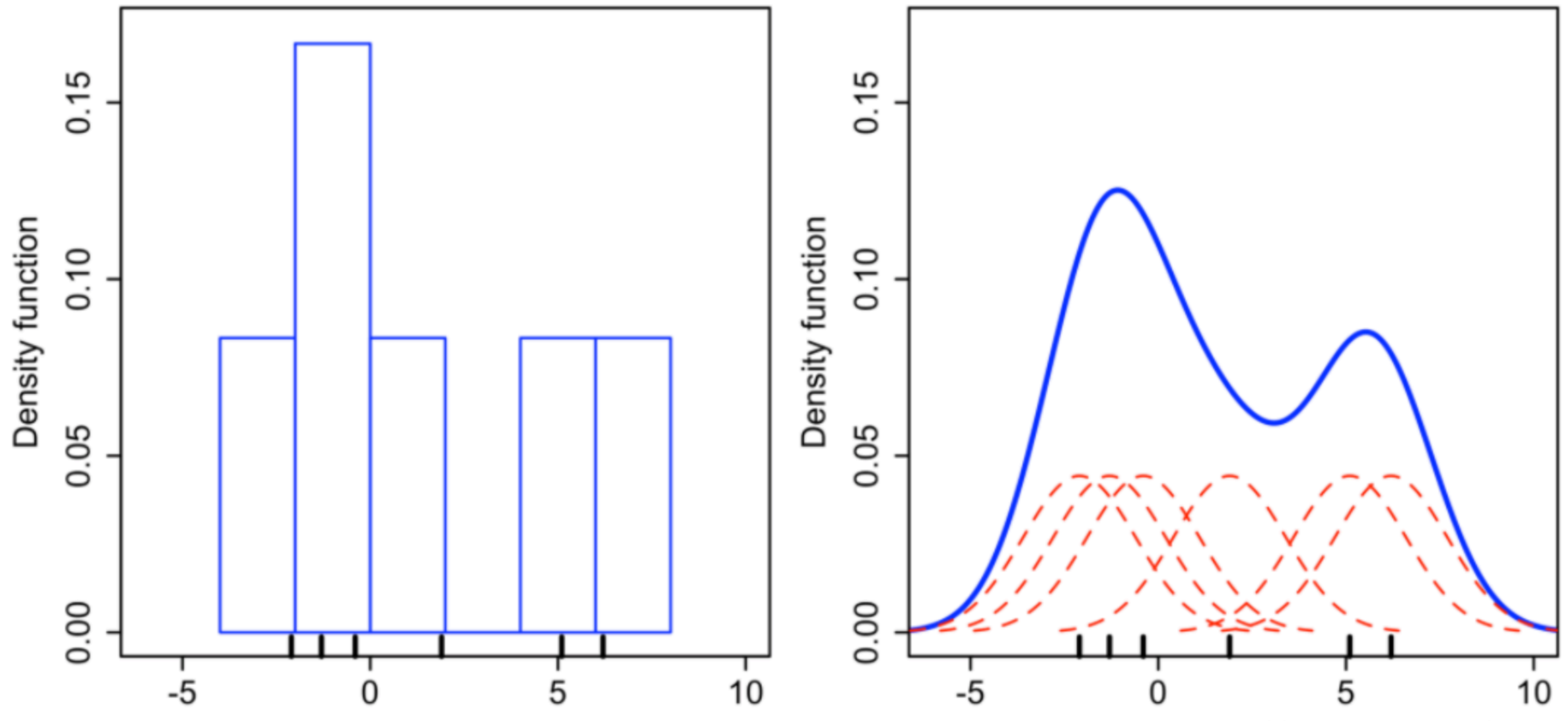
- depends on the size of bins significantly

- difficult to describe a multi-dimensional density function.

Kernel Density Estimation (KDE)

+ Smooth, Continuous

+ Depends on kernel type and kernel bandwidth only.



Model-Based AQP

Queries on One Single Column

```
SELECT agg_func(x)
FROM Table
WHERE x BETWEEN lb AND ub
```

$f(x)$: estimated density function from KDE

N : Number of records of column x

$$COUNT(x) = N \times \int_{lb}^{ub} f(x)dx$$

$$AVG(x) = \frac{\int_{lb}^{ub} xf(x)dx}{\int_{lb}^{ub} f(x)dx}$$

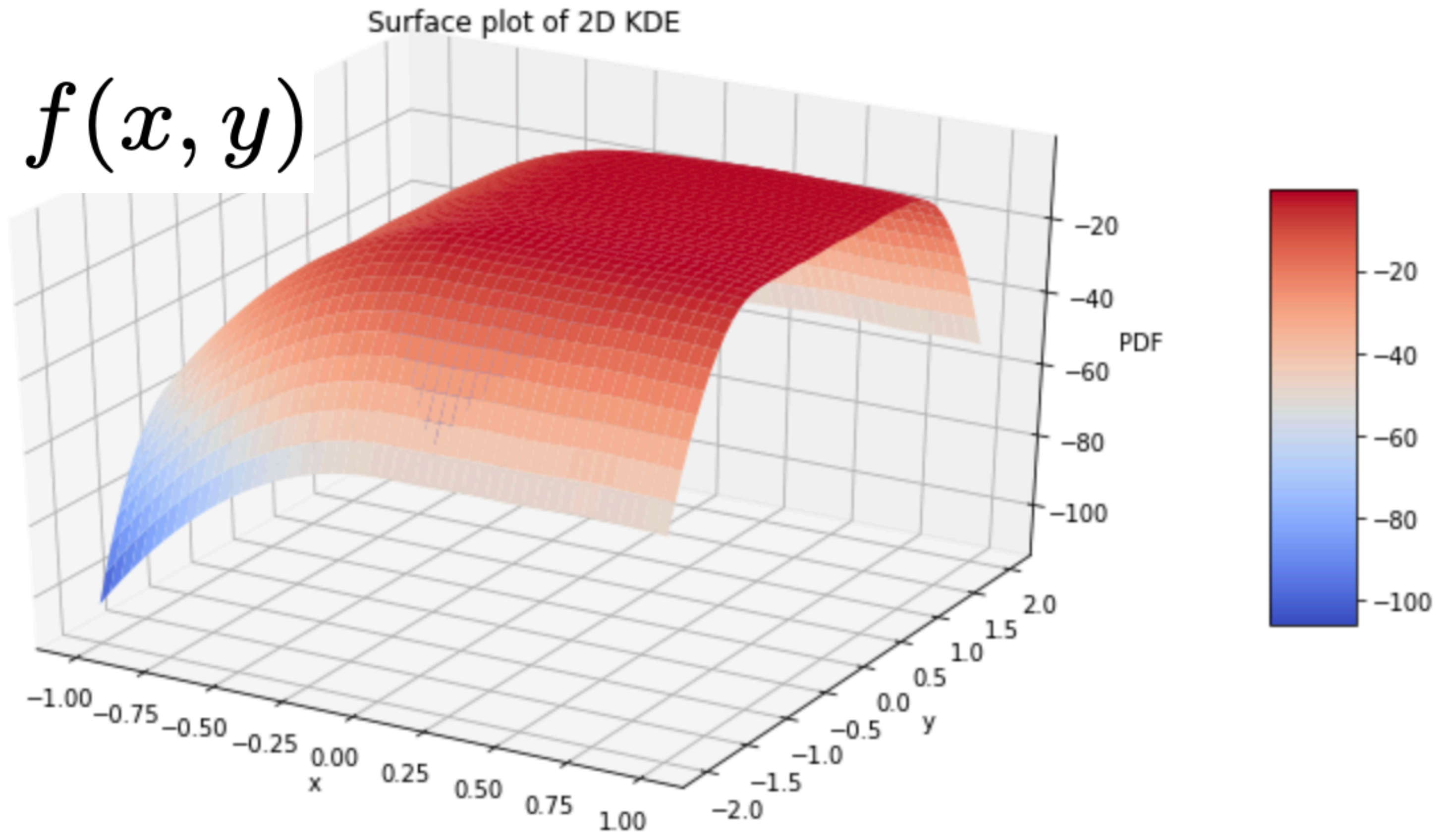
$$SUM(x) = COUNT(x) \times AVG(x)$$

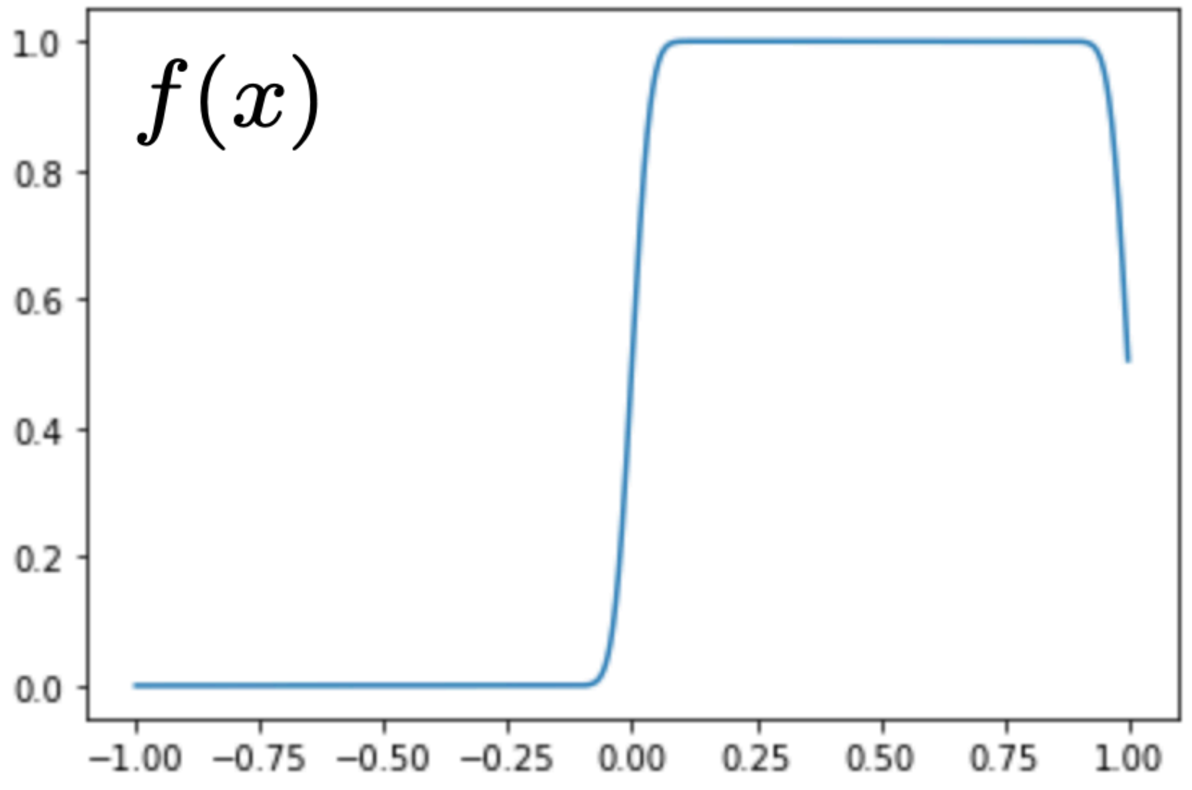
Model-Based AQP

Queries on Two Columns

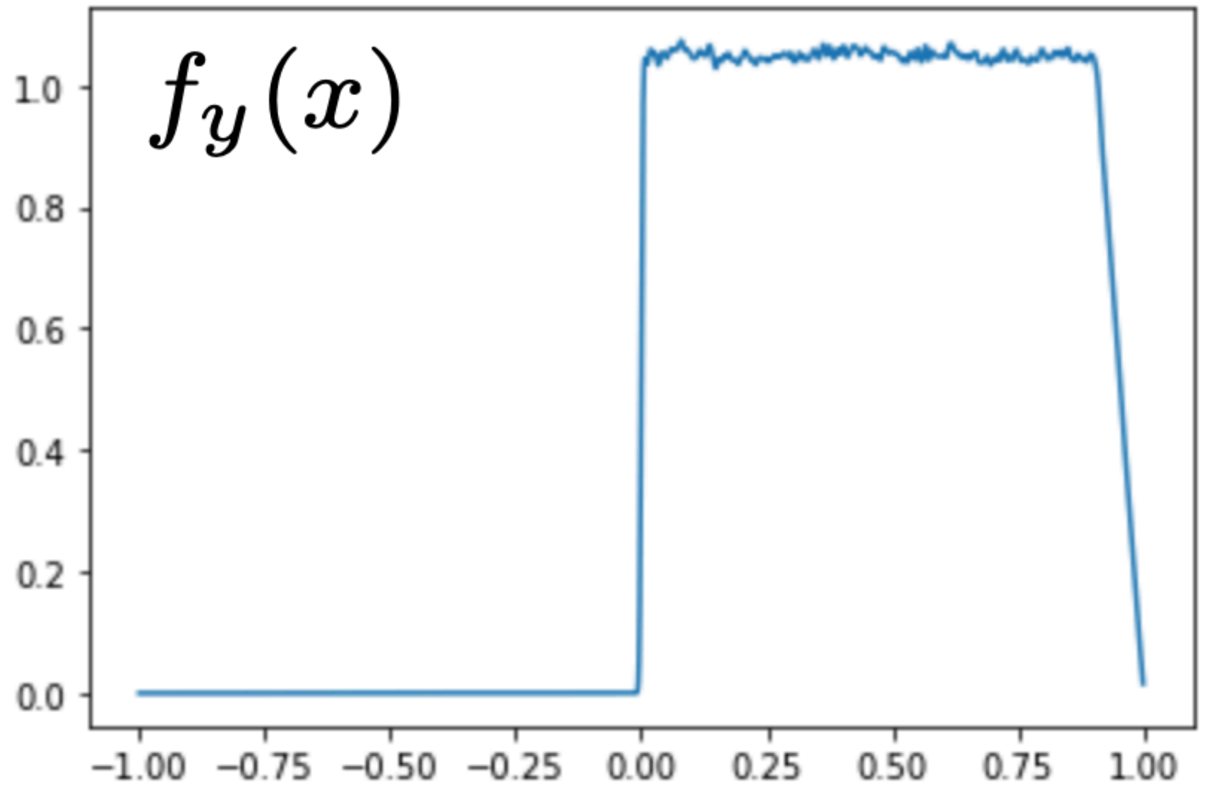
```
SELECT agg_func(y)
FROM Table
WHERE x BETWEEN lb AND ub
```

Two Dimensional KDE (2D-KDE)	Regression-Based KDE*	Weighted KDE
$COUNT(y) = N \times \int_{-\infty}^{\infty} \int_{lb}^{ub} f(x,y)dxdy$	$COUNT(y) = COUNT(x) = N \times \int_{lb}^{ub} f(x)dx$	$COUNT(y) = COUNT(x) = N \times \int_{lb}^{ub} f_x(x)dx$
$AVG(y) = \frac{\int_{-\infty}^{\infty} \int_{lb}^{ub} yf(x,y)dxdy}{\int_{-\infty}^{\infty} \int_{lb}^{ub} f(x,y)dxdy}$	$AVG(y) \approx E[R(x)] = \frac{\int_{lb}^{ub} R(x)f(x)dx}{\int_{lb}^{ub} f(x)dx}$	$AVG(y) = \frac{SUM(y)}{COUNT(y)}$
$SUM(y) = COUNT(y) \times AVG(y)$	$SUM(y) = COUNT(y) \times AVG(y)$	$SUM(y) = S \times \int_{lb}^{ub} f_y(x)dx$





$R(x)$: corresponding y at a given x



* Qingzhi Ma and Peter Triantafillou. “DBEst: Revisiting Approximate Query Processing Engines with Machine Learning Models”.

Model-Based AQP

Queries on Multiple Columns

```
SELECT AVG(z) FROM T
WHERE x BETWEEN lbx AND ubx
AND y BETWEEN lby AND uby
```

Two Dimensional KDE (2D-KDE)	Regression-Based KDE	Weighted KDE
$AVG(y) = \frac{\int_{-\infty}^{\infty}\int_{lby}^{uby}\int_{lbx}^{ubx} z f(x,y,z)dxdydz}{\int_{-\infty}^{\infty}\int_{lby}^{uby}\int_{lbx}^{ubx} f(x,y,z)dxdydz}$	$AVG(y) = \frac{\int_{lby}^{uby}\int_{lbx}^{ubx} R(x,y)f(x,y)dxdy}{\int_{lby}^{uby}\int_{lbx}^{ubx} f(x,y)dxdy}$	$SUM(y) = S \times \int_{lby}^{uby}\int_{lbx}^{ubx} f_z(x,y)dxdy$

Supporting “GROUP BY”

Experiment

Setup and Benchmark

- **KDE** model**:** Scipy KDE
- **Regression** Model: KNN regressor
- TPC-H Benchmark, specifically Q1, Q5, Q6, Q10
- Tested locally, so a scale factor of 1 is used. Experiments are run with PySpark locally on a MacBook with a 2.3 GHz Quad-Core processor, 16 GB memory and 256GB disk

Experiment

Query 1: Comparison of the three model-based methods

```
select
        l_returnflag,
        l_linestatus,
        sum(l_quantity) as sum_qty,
        sum(l_extendedprice) as sum_base_price,
        sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
        sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
        avg(l_quantity) as avg_qty,
        avg(l_extendedprice) as avg_price,
        avg(l_discount) as avg_disc,
        count(*) as count_order
from
        lineitem
where
        l_shipdate <= date '1998-12-01' - interval ':1' day (3)
group by
        l_returnflag,
        l_linestatus
order by
        l_returnflag,
        l_linestatus;
```

*This is not run on the full datase, and the result is inferred from subset runs.

Experiment

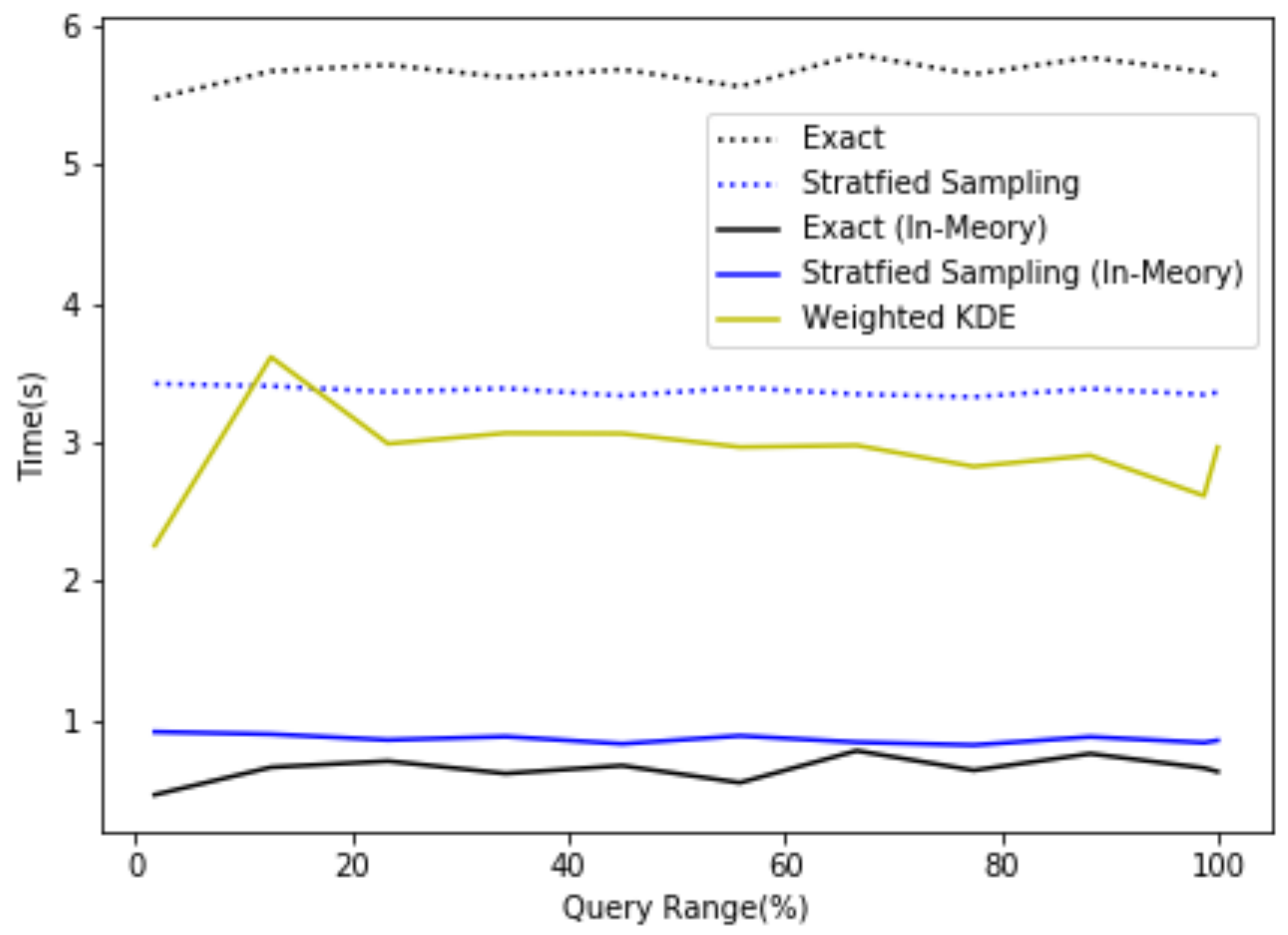
Query 1: Comparison of the three model-based methods

Metrics	Exact result	Stratified Sampling	2D-KDE*	Regression-Based KDE	Weighted-KDE
Construction time	0s	121.61s	800s	11.20s	5.37s
Execution time	5.62s	3.52s	200min	21.30s	2.51s
Space	1458.805MB	2.876MB	600MB	0.556MB	0.364MB
Relative Error	0%	1.57%	0.32%	1.89%	0.06%
STD	0%	0.67%	0.19%	1.38%	0.05%

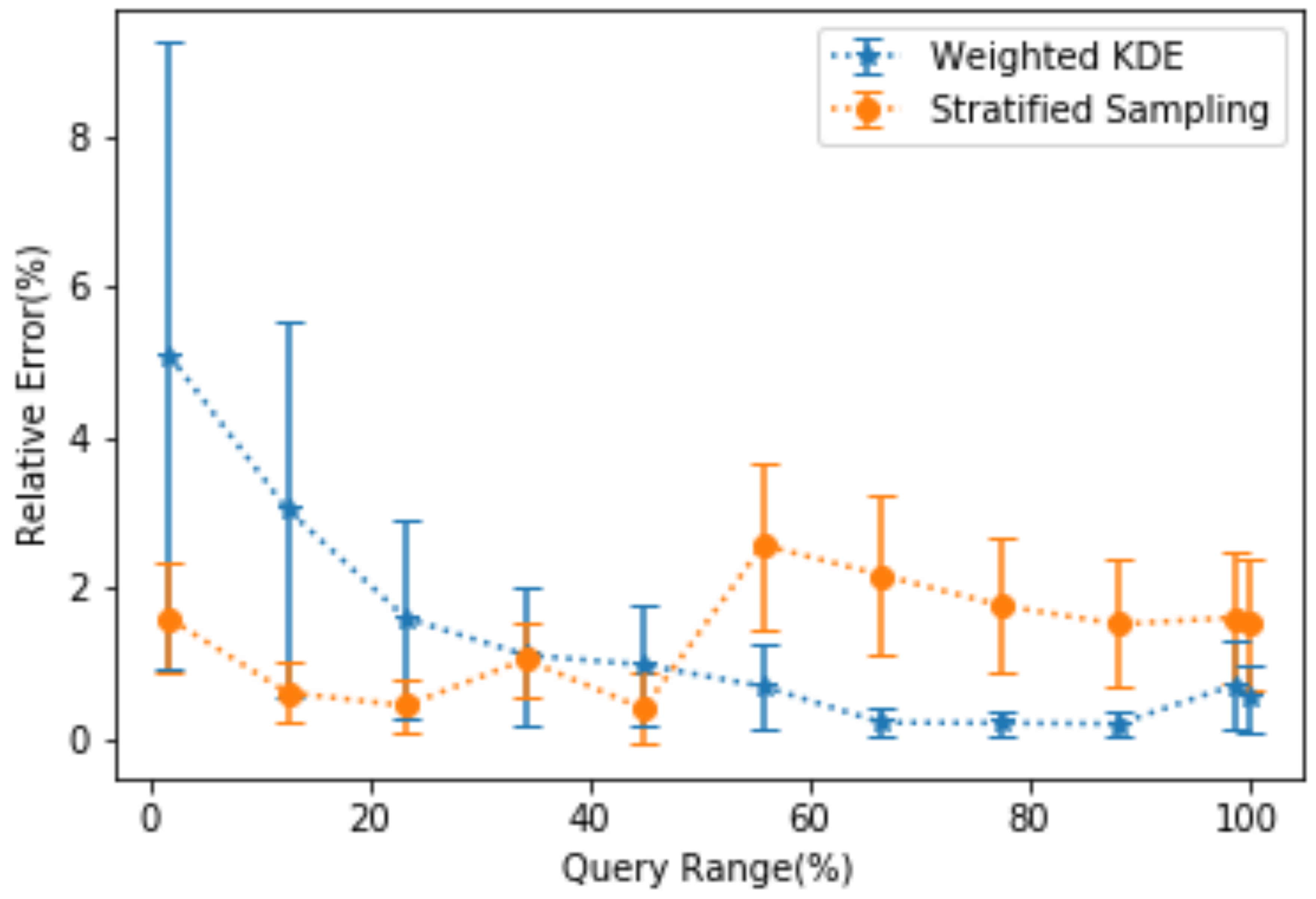
*This is not run on the full datase, and the result is inferred from subset runs.

Experiment

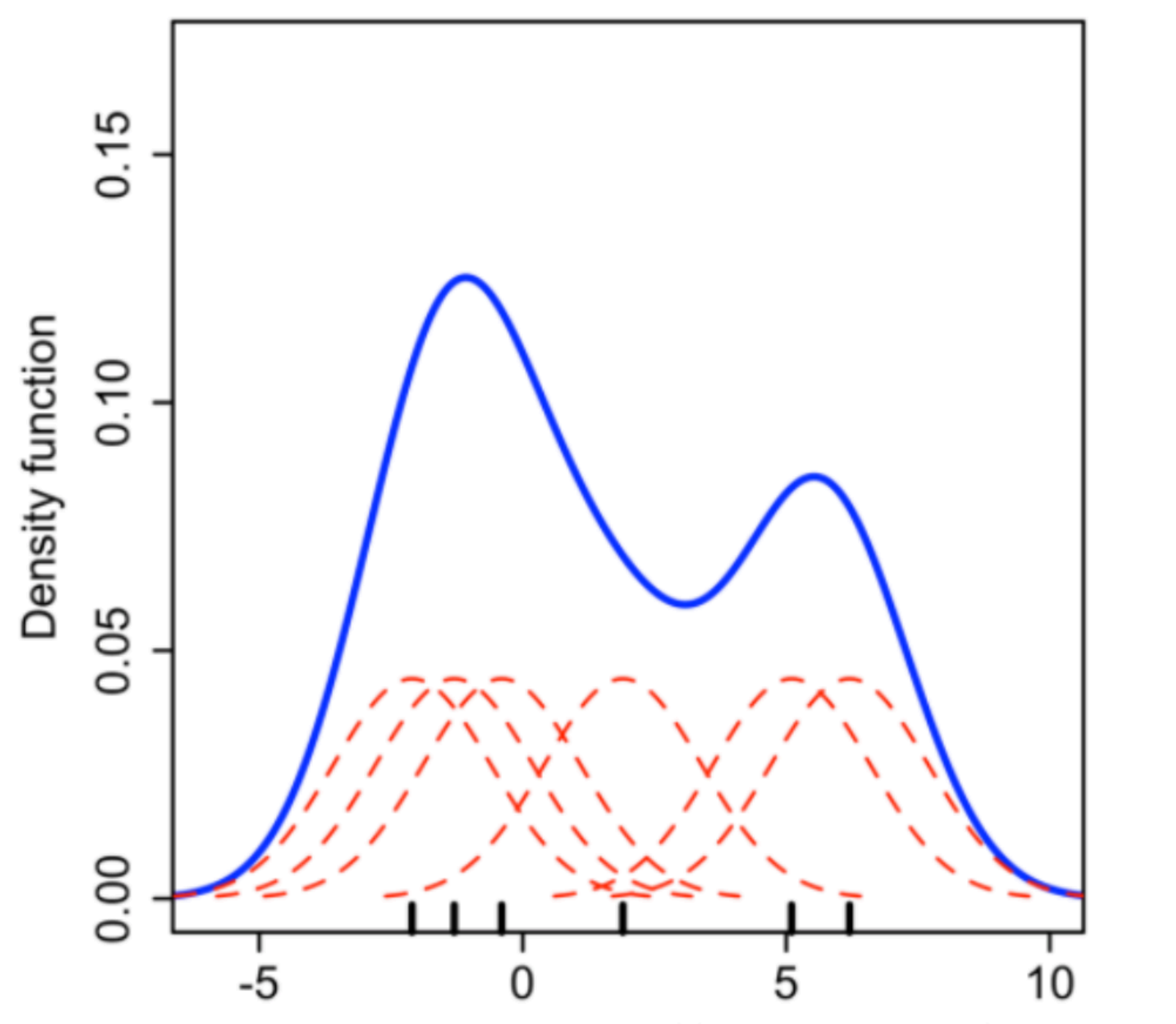
Query 1: Effect of Predicate Selectivity



Execution Time and Query Range

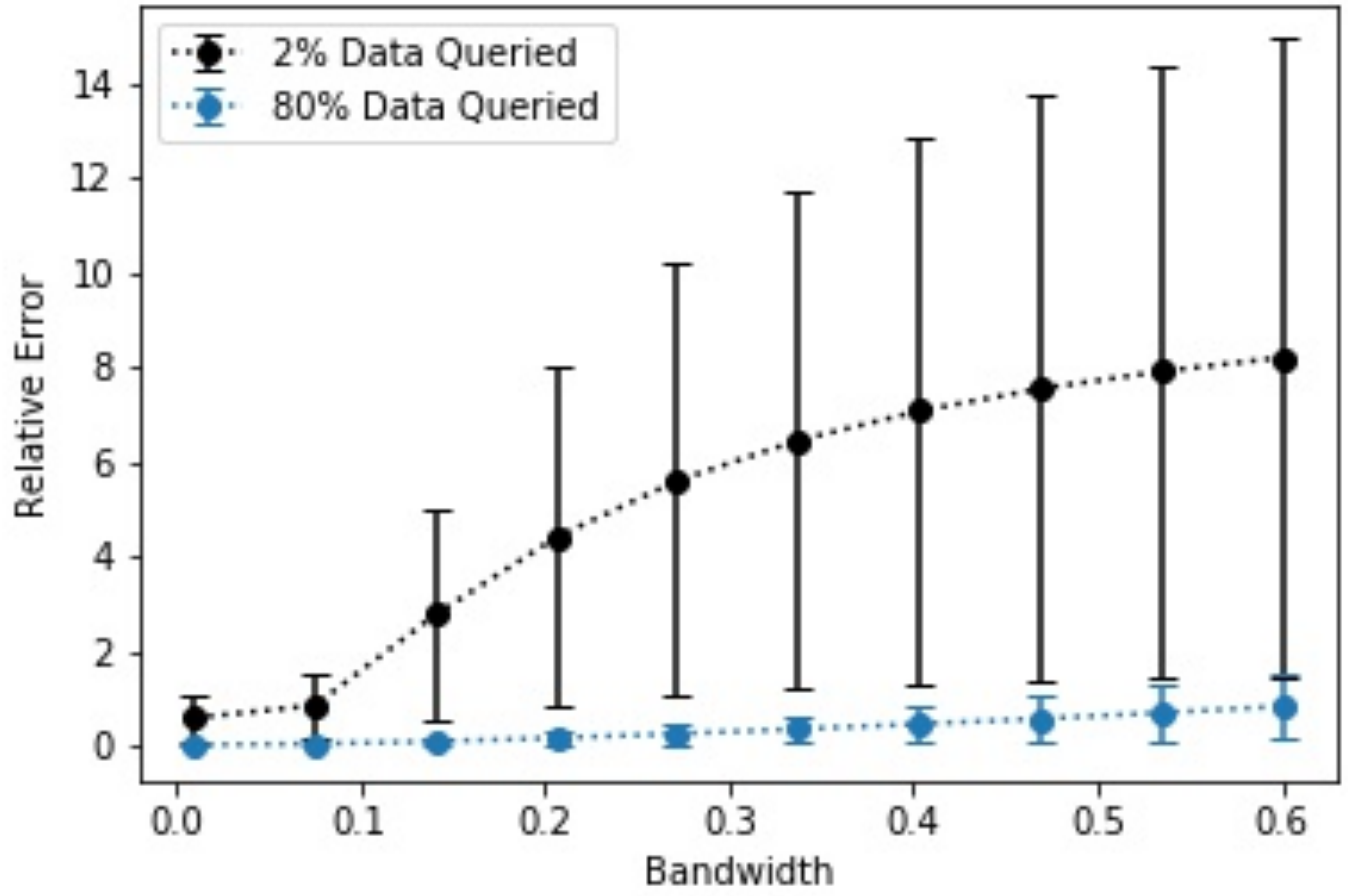


Relative Error and Query Range



Experiment

Query 1: Effect of Kernel Bandwidth



Relative Error and Bandwidth

Experiment

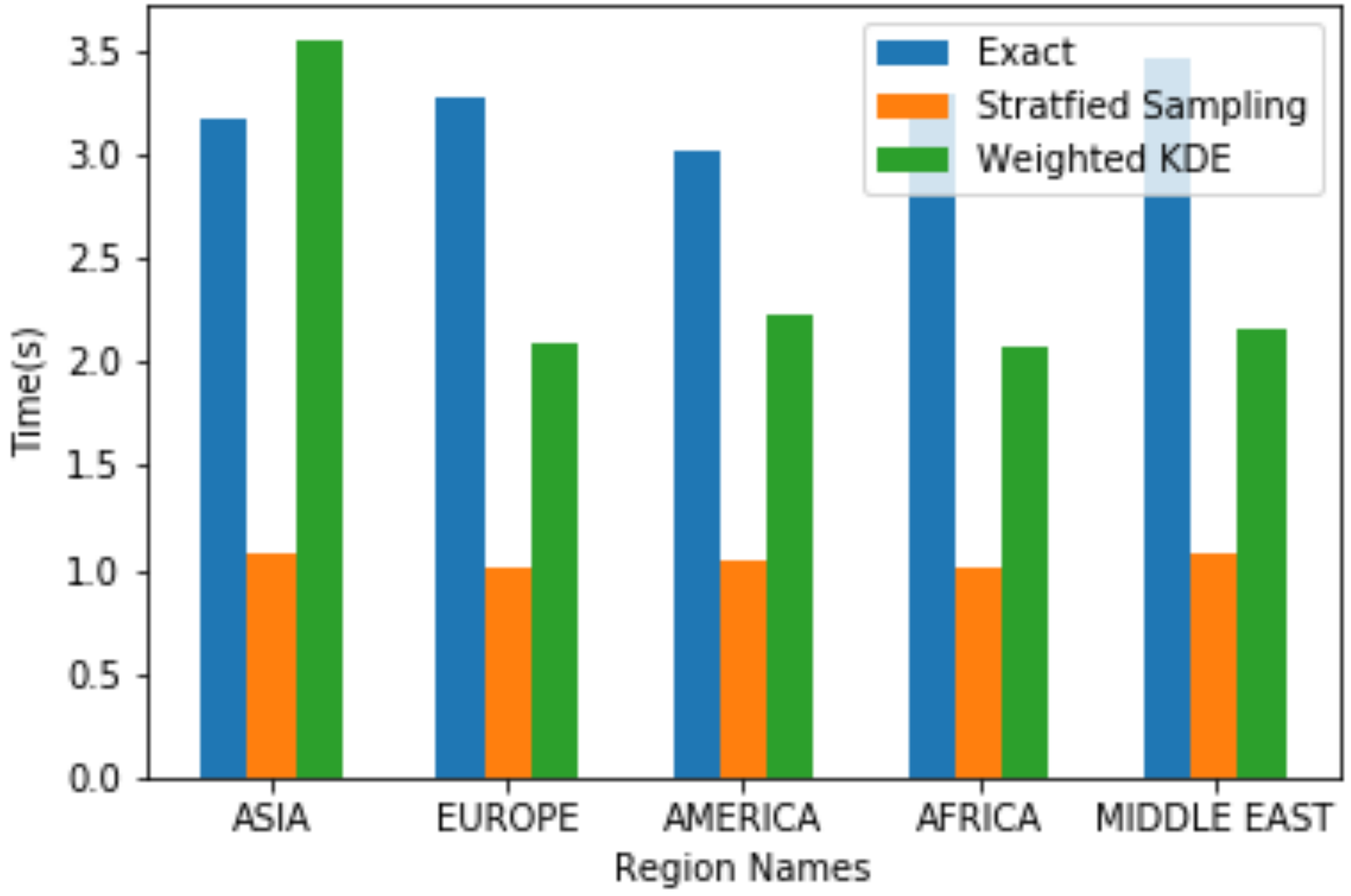
Query 5: Richness of Model

```
select
   n_name,
   sum(l_extendedprice * (1 - l_discount)) as revenue
from
   customer,
   orders,
   lineitem,
   supplier,
   nation,
   region
where
   c_custkey = o_custkey
   and l_orderkey = o_orderkey
   and l_suppkey = s_suppkey
   and c_nationkey = s_nationkey
   and s_nationkey = n_nationkey
   and n_regionkey = r_regionkey
   and r_name = ':1'
   and o_orderdate >= date ':2'
   and o_orderdate < date ':2' + interval '1' year
group by
   n_name
order by
   revenue desc;
```

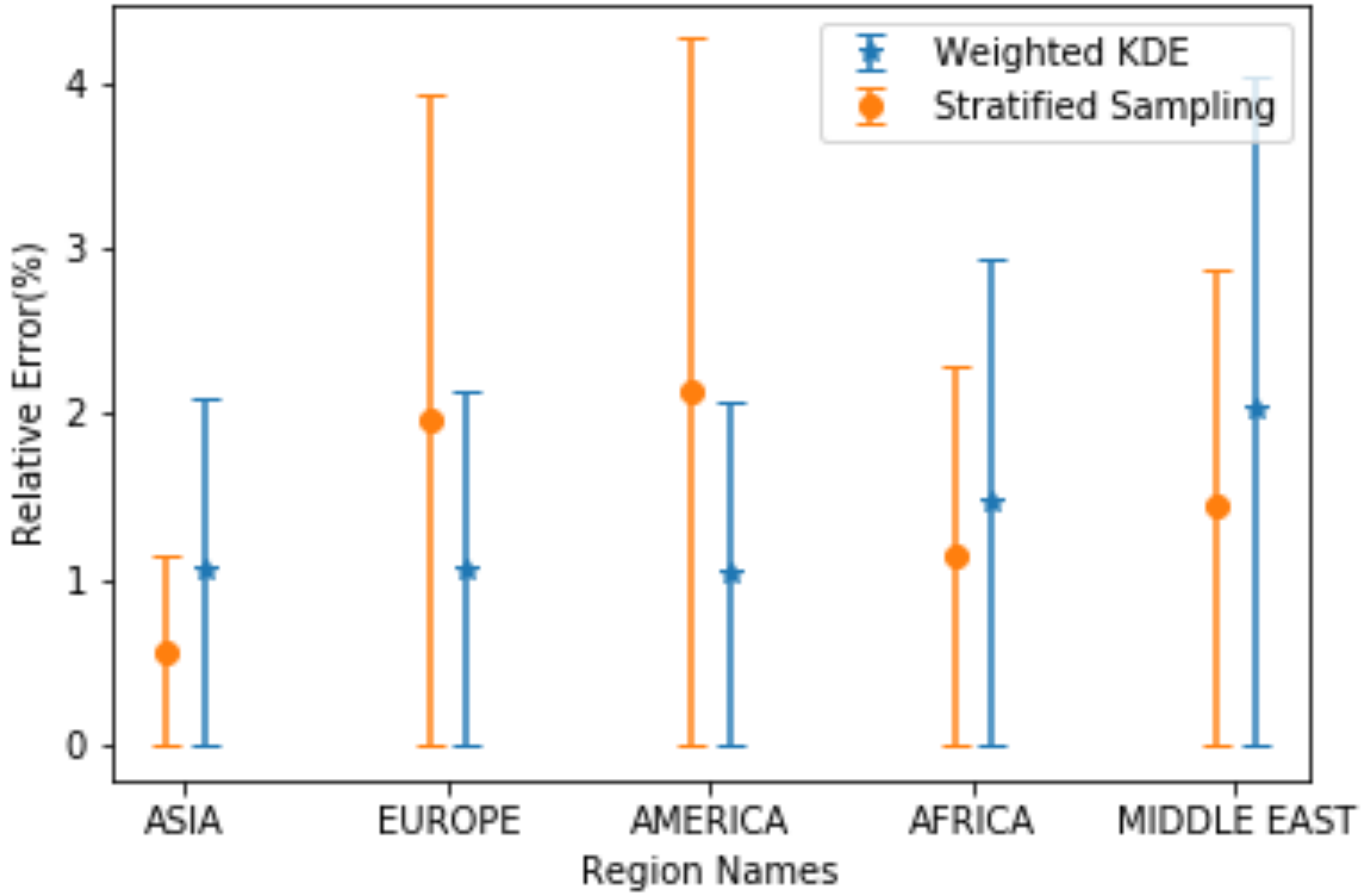
Experiment

Query 5: Richness of Model

Region Name	STDDEV of Countries's Count
AMERICA	228.194435
EUROPE	251.674393
ASIA	476.581263
MIDDLE EAST	578.440835
AFRICA	677.098442



Execution Time and r_name

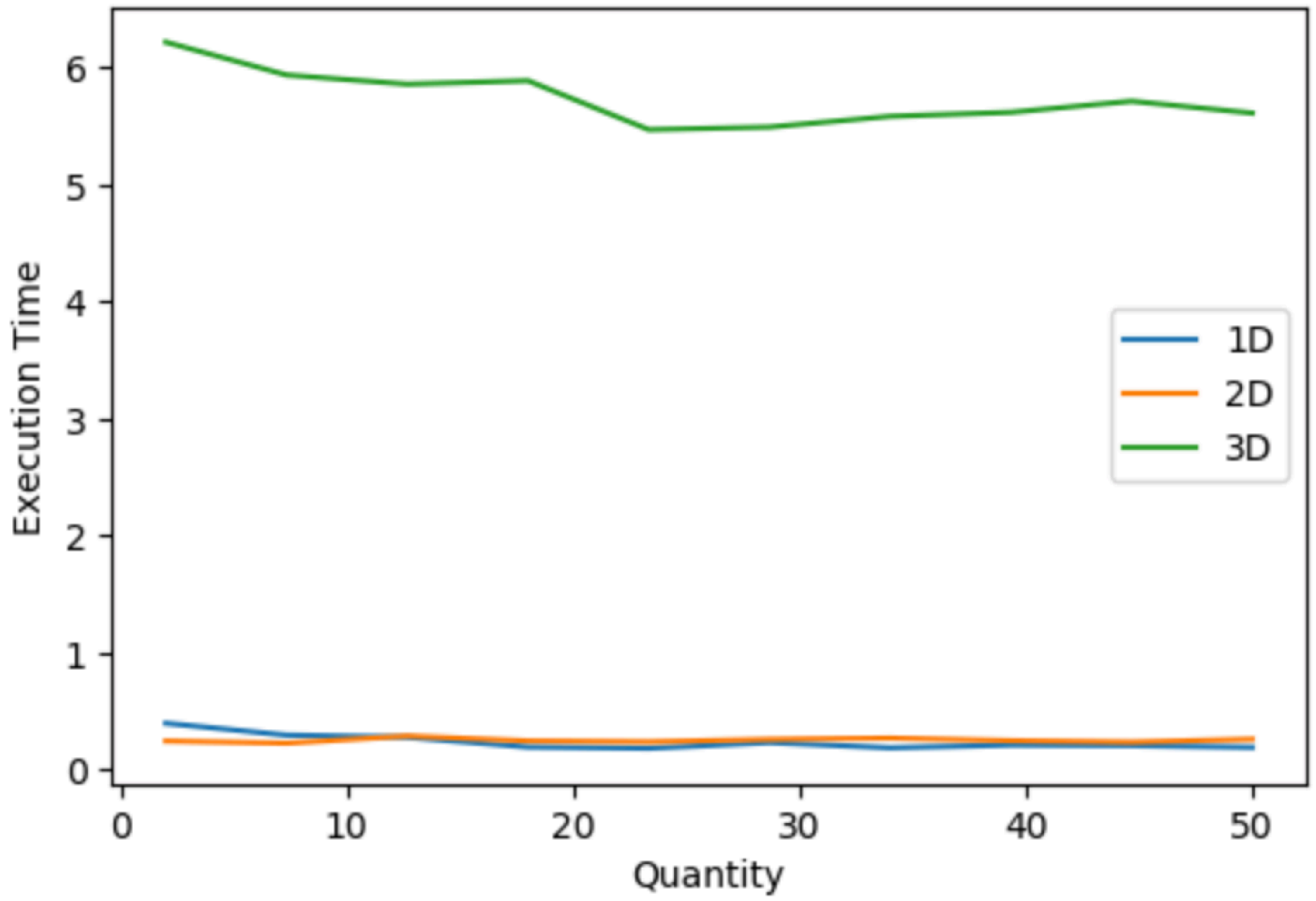


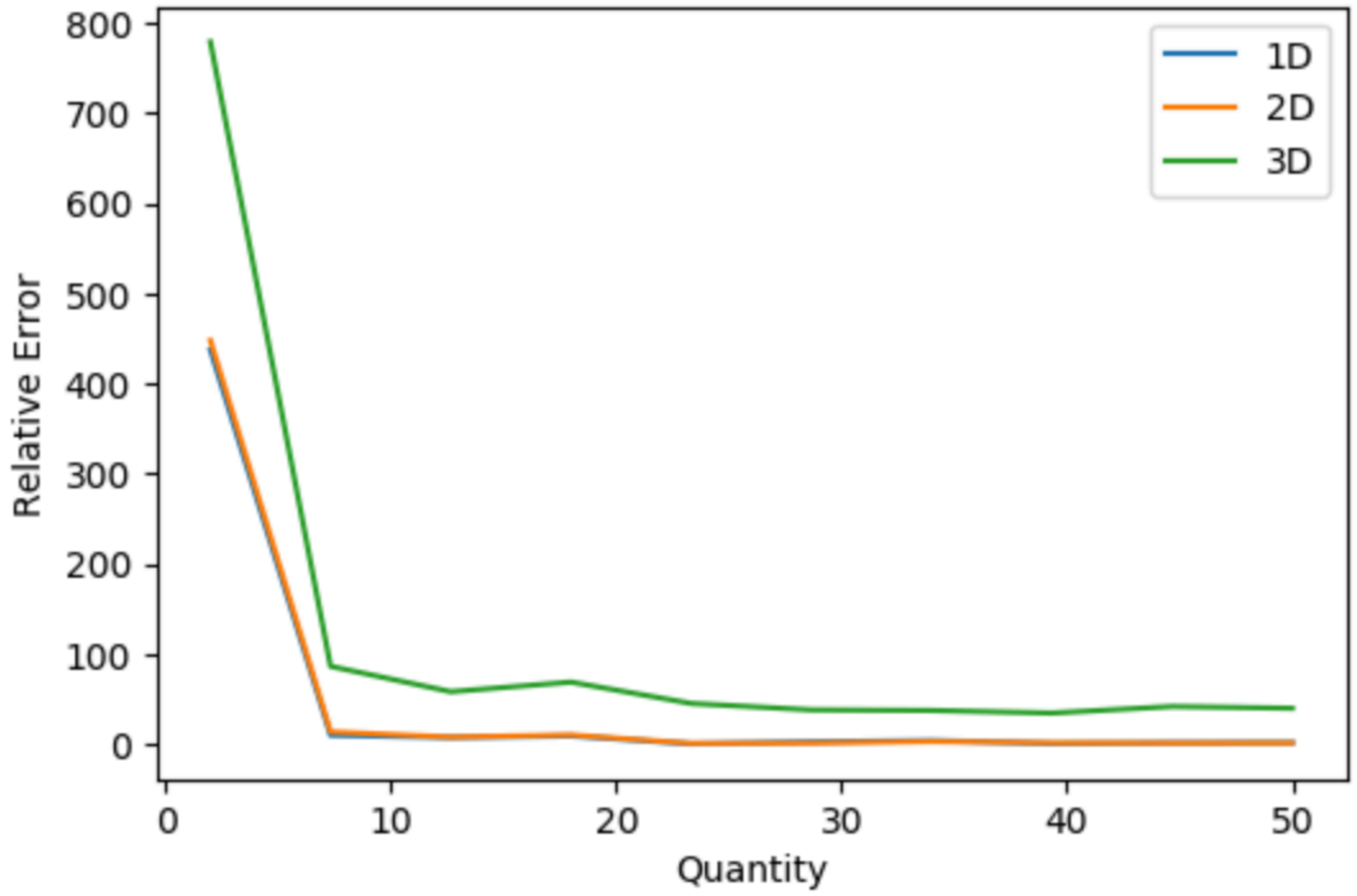
Relative Error and r_name

Experiment

Query 6*: Curse Of Dimensionality

1D: l_quantity
2D: l_quantity, l_shipdate
3D: l_quantity, l_shipdate, l_discount



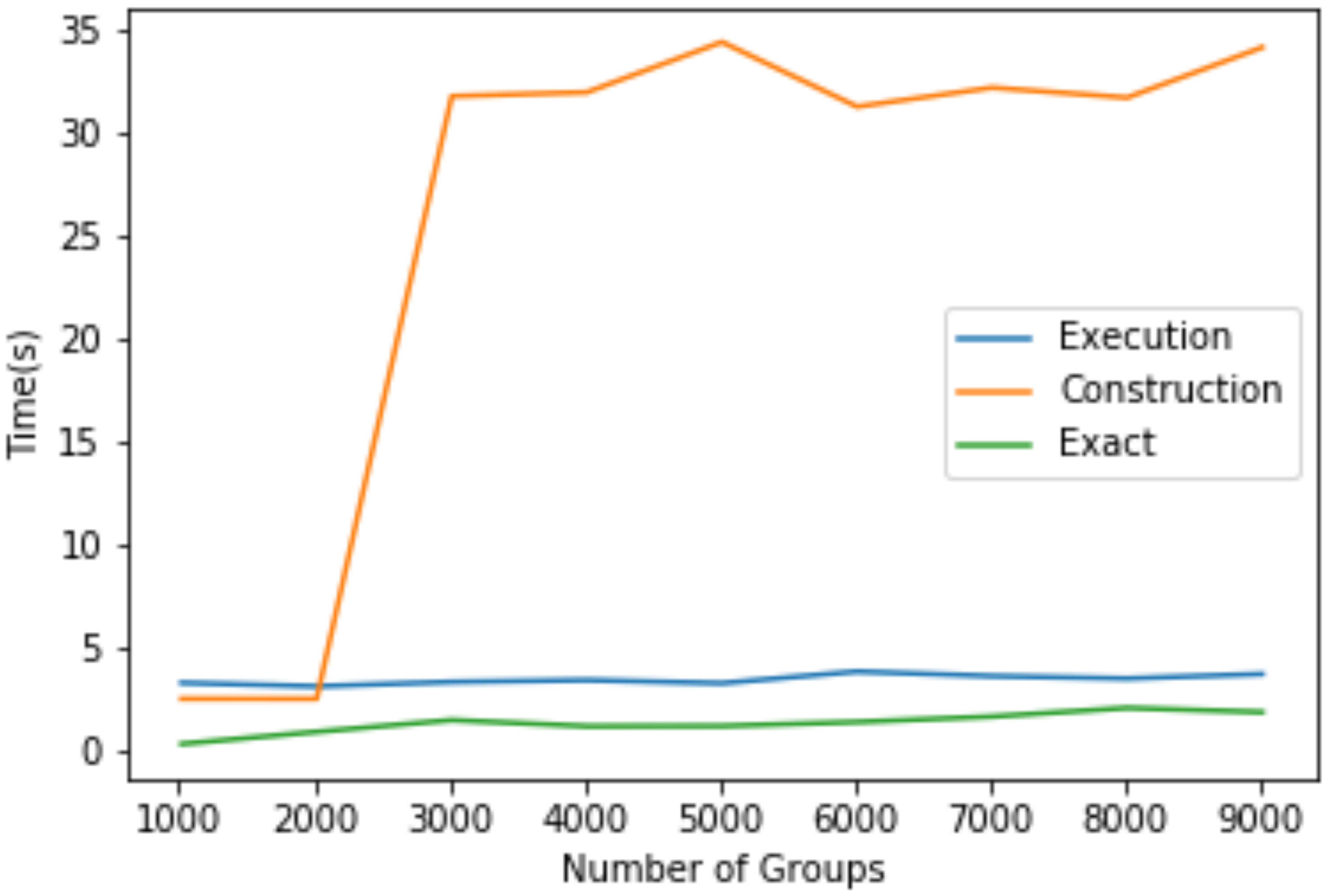


*This query is not run on the full dataset

Experiment

Query 10*: Effect of “GROUP BY”

```
select
   c_custkey,
   c_name,
   sum(l_extendedprice * (1 - l_discount)) as revenue,
   c_acctbal,
   n_name,
   c_address,
   c_phone,
   c_comment
from
   customer,
   orders,
   lineitem,
   nation
where
   c_custkey = o_custkey
   and l_orderkey = o_orderkey
   and o_orderdate >= date ':1'
   and o_orderdate < date ':1' + interval '3' month
   and l_returnflag = 'R'
   and c_nationkey = n_nationkey
group by
   c_custkey,
   c_name,
   c_acctbal,
   c_phone,
   n_name,
   c_address,
   c_comment
order by
   revenue desc;
```

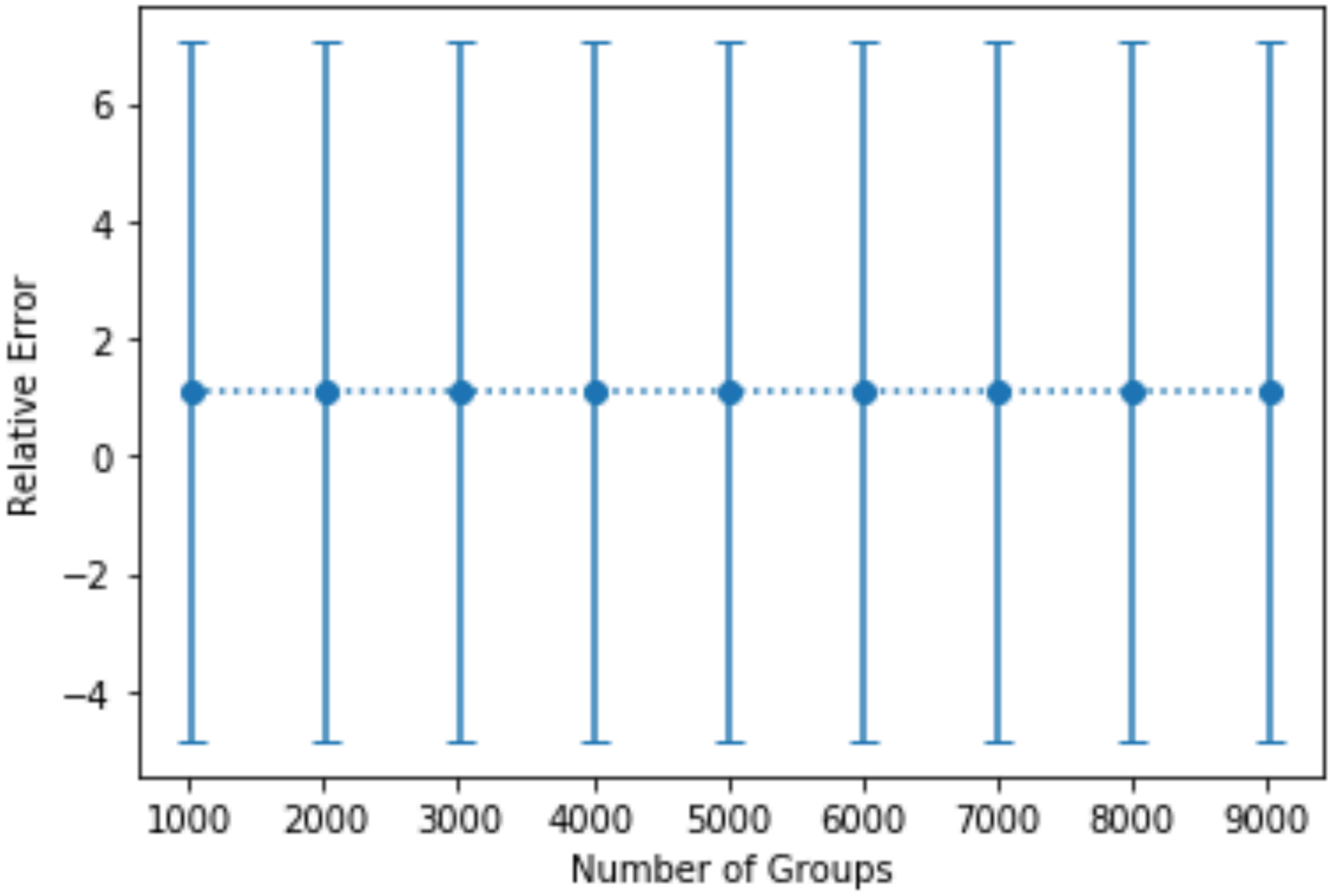


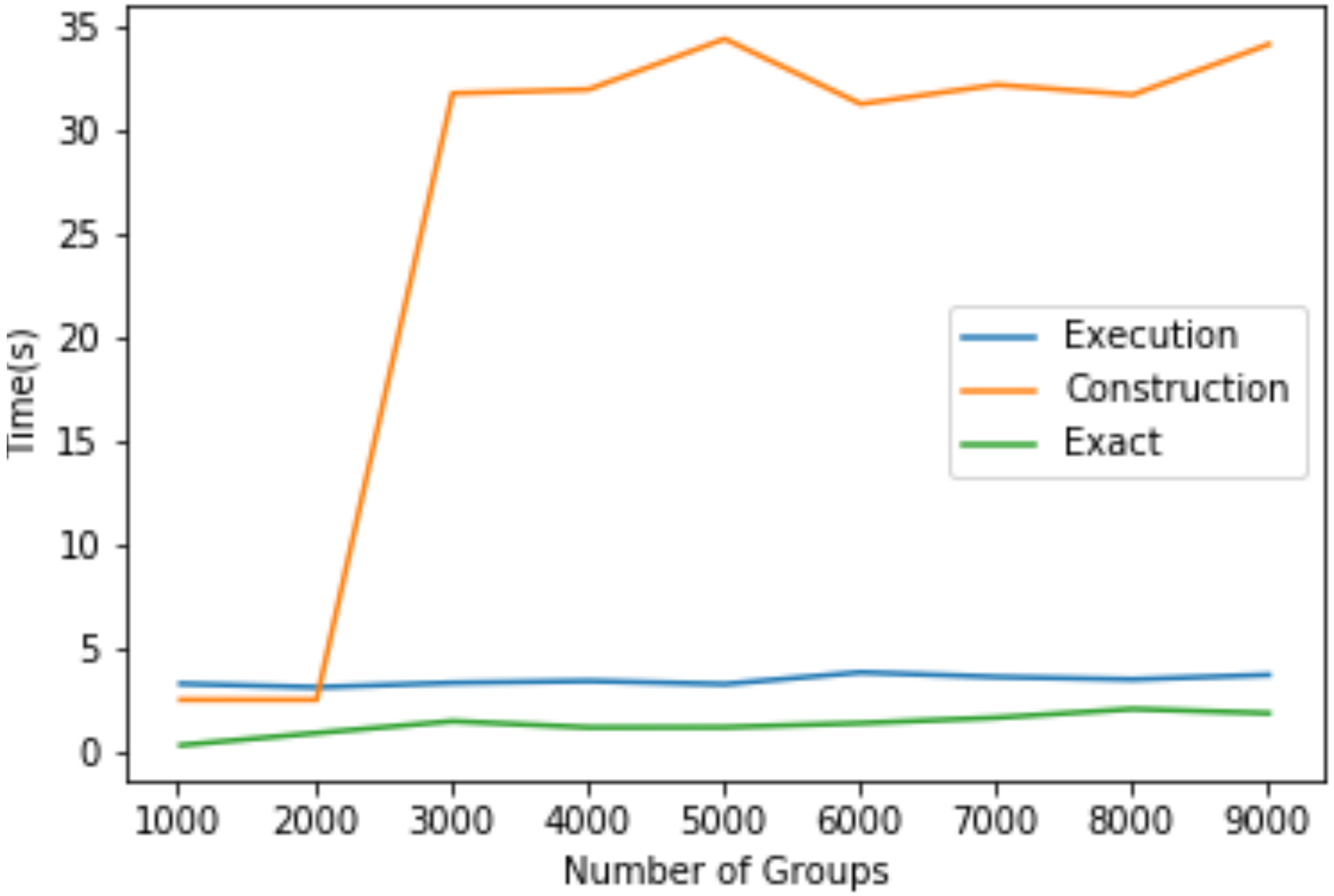
*This query is not run on the full dataset

Experiment

Query 10*: Effect of “GROUP BY”

```
select
   c_custkey,
```






```
   n_name,
   c_address,
   c_comment
order by
   revenue desc;
```

*This query is not run on the full dataset

Summary

- Limitations
  - The KDE-based method doesn't support ad-hoc queries.
  - The Model needs to be retrained if there are new records added.
  - It does not support the pre-definition of estimation error and confidence interval.
  - ….
- Conclusion
  - Weighted-KDE is fast to construct, uses small memory space, responses quickly and yields accurate results, while it also suffers from difficulties in some scenarios.
  - The use case plays an important role on selecting a more suitable AQP scheme.

Thanks

Appendix
KDE Package Selection

	Scikit-learn	Scipy
Kernel Type	gaussian, tophat, epanechnikov, exponential and etc.	Gaussian kernels only
Multidimensional?	Yes	Yes
Weighted KDE?	Yes	Yes
Reset the bandwidth after the model is built?	No	Yes
Automatic bandwidth determination?	No	Yes