fn make_private_group_by

Michael Shoemate

This proof resides in **"contrib"** because it has not completed the vetting process.

Proves soundness of `fn make_private_group_by` in `mod.rs` at commit 0db9c6036 (outdated[1]).

Vetting History

- Pull Request #512

1 Hoare Triple

Precondition

To ensure the correctness of the output, we require the following preconditions:

- Type `MS` must have trait `DatasetMetric`.
- Type `MI` must have trait `UnboundedMetric`.
- Type `MO` must have trait `ApproximateMeasure`.

Pseudocode

```python
def make_private_group_by(
    input_domain,
    input_metric,
    output_measure,
    plan,
    global_scale,
    threshold,
):
    input_, keys, aggs, expr_threshold = match_group_by(plan)

    t_prior = input_.make_stable(input_domain, input_metric)
    middle_domain, middle_metric = t_prior.output_space()

    grouping_columns = match_grouping_columns(keys)

    margin = middle_domain \
        .margins \
        .get(grouping_columns, Margin.default())

    # prepare a join measurement over all expressions
    expr_domain = ExprDomain(
        middle_domain,
```

[1] See new changes with `git diff 0db9c6036..76a98c7f rust/src/measurements/make_private_lazyframe/group_by/mod.rs`

```
        ExprContext.Aggregate(grouping_columns),
    )

    m_exprs = make_basic_composition([
        make_private_expr(
             expr_domain,
             PartitionDistance(middle_metric),
             output_measure,
             expr,
             global_scale,
        ) for expr in aggs
    ])

    # reconcile information about the threshold
    dp_exprs = m_exprs.invoke((input_, all()))

    if threshold is not None and expr_threshold is not None:
        raise "two thresholds set"

    if margin.public_info is not None:
        threshold = None
    elif expr_threshold is not None:
        name, threshold = expr_threshold
        plugin = find_len_expr(dp_exprs, name.as_str())[1]
        threshold = name, plugin, threshold
    elif threshold is not None:
        name, plugin = find_len_expr(dp_exprs, None)
        threshold = name, plugin, threshold
    else:
        raise f"The key set of {grouping_columns} is unknown and cannot be released."

    # prepare supporting elements
    def function(arg):
        output = DslPlan.GroupBy(
            input=arg,
            keys=keys,
            aggs=m_exprs((arg, all())),
            apply=None,
            maintain_order=false,
        )

        if threshold is not None:
            name, _, threshold_value = threshold
            output = DslPlan.Filter(
                input=output,
                predicate=col(name).gt(lit(threshold_value)),
            )
        return output

    def privacy_map(d_in):
        l0 = margin.max_influenced_partitions.unwrap_or(d_in).min(d_in)
        li = margin.max_partition_contributions.unwrap_or(d_in).min(d_in)
        l1 = l0.inf_mul(li).min(d_in)

        d_out = m_exprs.map((l0, l1, li))

        if threshold is not None:
            _, plugin, threshold_value = threshold
            if li >= threshold_value:
                raise f"Threshold must be greater than {li}."
            d_instability = f64.inf_cast(threshold_value.inf_sub(li))
            delta_single = integrate_discrete_noise_tail(plugin.distribution, plugin.scale, d_instability)
            delta_joint = 1 - (1 - delta_single).inf_powi(l0)
```

```python
            d_out = MO.add_delta(d_out, delta_joint)
        elif margin.public_info is None:
            raise "keys must be public if threshold is unknown"

        return d_out

    m_group_by_agg = Measurement(
        middle_domain,
        function,
        middle_metric,
        output_measure,
        privacy_map,
    )

    return t_prior >> m_group_by_agg
```

Postconditions

For every setting of the input parameters (`input_domain, input_metric, output_measure, plan, global_scale, threshold`) to `make_private_group_by` such that the given preconditions hold, `make_private_group_by` raises an exception (at compile time or run time) or returns a valid measurement. A valid measurement has the following property:

1. (Privacy guarantee). For every pair of elements x, x' in `input_domain` and for every pair (`d_in`, `d_out`), where `d_in` has the associated type for `input_metric` and `d_out` has the associated type for `output_measure`, if x, x' are `d_in`-close under `input_metric`, `privacy_map`(`d_in`) does not raise an exception, and `privacy_map`(`d_in`) $\leq$ `d_out`, then `function`(x), `function`(x') are `d_out`-close under `output_measure`.

2 Proof

We now prove the postcondition of `make_private_group_by`.

Proof. Start by establishing proof properties of the following three variable.

- By the postcondition of `StableDslPlan.make_stable`, `t_prior` is a valid transformation.
- By the postcondition of `match_grouping_columns`, `grouping_columns` holds the names of the grouping columns. `margin` denotes what is considered public information about the key set, pulled from descriptors in the input domain. By the postcondition of `make_basic_composition`,
- `m_exprs` is a valid measurement that prepares a batch of expressions that, when executed, satisfies the privacy guarantee of `m_exprs`.

In the case where grouping keys are considered public, no thresholding is performed. In the setting when grouping keys are considered private information, it is necessary to know if a suitable release for the data set length is made that can be used to filter sensitive keys.

The function returns a DslPlan that applies each expression from `m_exprs` to `arg` grouped by `keys`.

The measurement for each expression expects data set distances in terms of a triple:

- L^0: the greatest number of partitions that can be influenced by any one individual. This is no greater than the input distance (an individual can only ever influence as many partitions as they contribute rows), but could be tighter when supplemented by the `max_influenced_partitions` metric descriptor.
- L^∞: the greatest number of records that can be added or removed by any one individual in each partition. This is no greater than the input distance, but could be tighter when supplemented by the `max_partition_contributions` metric descriptor.

- L^1: the greatest total number of records that can be added or removed across all partitions. This is no greater than the input distance, but could be tighter when accounting for the L^0 and L^∞ distances.

By the postcondition of the map on `m_exprs`, the privacy loss, when grouped data sets may differ by this distance triple, is `d_out`.

Adapted from [Rog23] (Theorem 7). Consider S to be the set of labels that are common between x and x'. Define event E to be any potential outcome of the mechanism for which all labels are in S (where only stable partitions are released). We then lower bound the probability of the mechanism returning an event E. In the following, c_j denotes the exact count for partition j, and Z_j is a random variable distributed according to the distribution used to release a noisy count.

$$\begin{aligned}
\Pr[E] &= \prod_{j \in x \setminus x'} \Pr[c_j + Z_j \le T] \\
&\ge \prod_{j \in x \setminus x'} \Pr[\Delta_\infty + Z_j \le T] \\
&\ge \Pr[\Delta_\infty + Z_j \le T]^{\Delta_0}
\end{aligned}$$

The probability of returning a set of stable partitions ($\Pr[E]$) is the probability of not returning any of the unstable partitions. We now solve for the choice of threshold T such that $\Pr[E] \ge 1 - \delta$.

$$\begin{aligned}
\Pr[\Delta_\infty + Z_j \le T]^{\Delta_0} &= \Pr[Z_j \le T - \Delta_\infty]^{\Delta_0} \\
&= (1 - \Pr[Z_j > T - \Delta_\infty])^{\Delta_0}
\end{aligned}$$

Let `d_instability` denote the distance to instability of $T - \Delta_\infty$. By the postcondition of `integrate_discrete_noise_tail`, the probability that a random noise sample exceeds `d_instability` is at most `delta_single`. Therefore $\delta = 1 - (1 - \texttt{delta\_single})^{\Delta_0}$. This privacy loss is then added to `d_out`.

Together with the potential increase in delta for the release of the key set, then it is shown that `function`(u), `function`(v) are `d_out`-close under `output_measure`.

□

References

[Rog23] Ryan Rogers. A unifying privacy analysis framework for unknown domain algorithms in differential privacy, 2023.