Supplementary Material:

Interaction of Crime Risk across Crime Types in Hotspot Areas

Hong Zhang [1,2,3], Yongping Gao [1,2,3,*], Dizhao Yao [1,2,3] and Jie Zhang [1,2,3]

[1] Key Laboratory of Virtual Geographic Environment, Ministry of Education, Nanjing Normal University, Nanjing 210023, China
[2] State Key Laboratory Cultivation Base of Geographical Environment Evolution (Jiangsu Province), Nanjing 210023, China
[3] Jiangsu Center for Collaborative Innovation in Geographical Information Resource Development and Application, Nanjing 210023, China
* Correspondence: 211302077@njnu.edu.cn; Tel.: +86-136-9796-0344

Supplemental Material

Algorithm S1: DBSCAN algorithm

The steps of the DBSCAN algorithm are as follows:
(1) Generate a list of Eps parameter values: The Eps parameter list is generated using the K-means nearest neighbor algorithm. The main principle of this algorithm is to first calculate the K-nearest neighbor distance matrix of the input dataset and then find the median of the K-nearest neighbor distances of all element points, thus forming the K-means nearest neighbor set. To take the K-means nearest neighbor set as the Eps parameter list, the specific steps are as follows:
(1.1) Calculate distance distribution matrix $Dist_{n\times n}$， as shown in formula 1， where n is the number of element points in the dataset and *dist(i, j)* is the Euclidean distance between element point *i* and element point *j*.

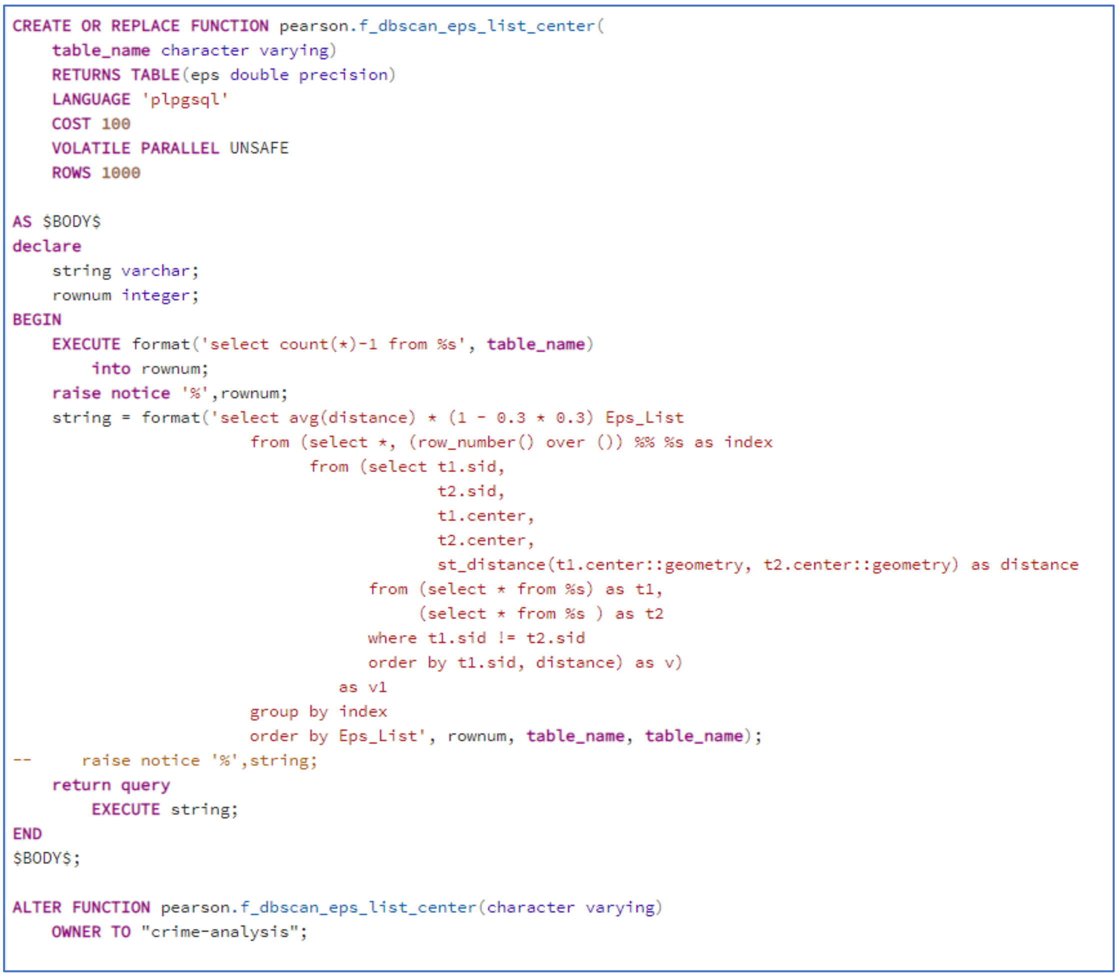
$$Dist_{n\times n} = \{dist(i,j) | 1 \leq i \leq n, 1 \leq j \leq n\} \qquad [1]$$

(1.2) Arrange each row of the distance distribution matrix in ascending order $Dist_{n\times n}$. Then, average the K（1 ≤ K ≤ n）column of the matrix to obtain $\overline{D_K}$. After subtracting the self-attenuation value (λ = 0.3), obtain the Eps parameter candidates, as shown in formula 2.

$$Eps_K = (1-\lambda^2)\overline{D_K} \qquad [2]$$

(1.3) Calculate all K values, and obtain the Eps parameter list, as shown in formula (3).

$$Eps\_list = \{Eps_K | 1 \leq K \leq n\} \qquad [3]$$



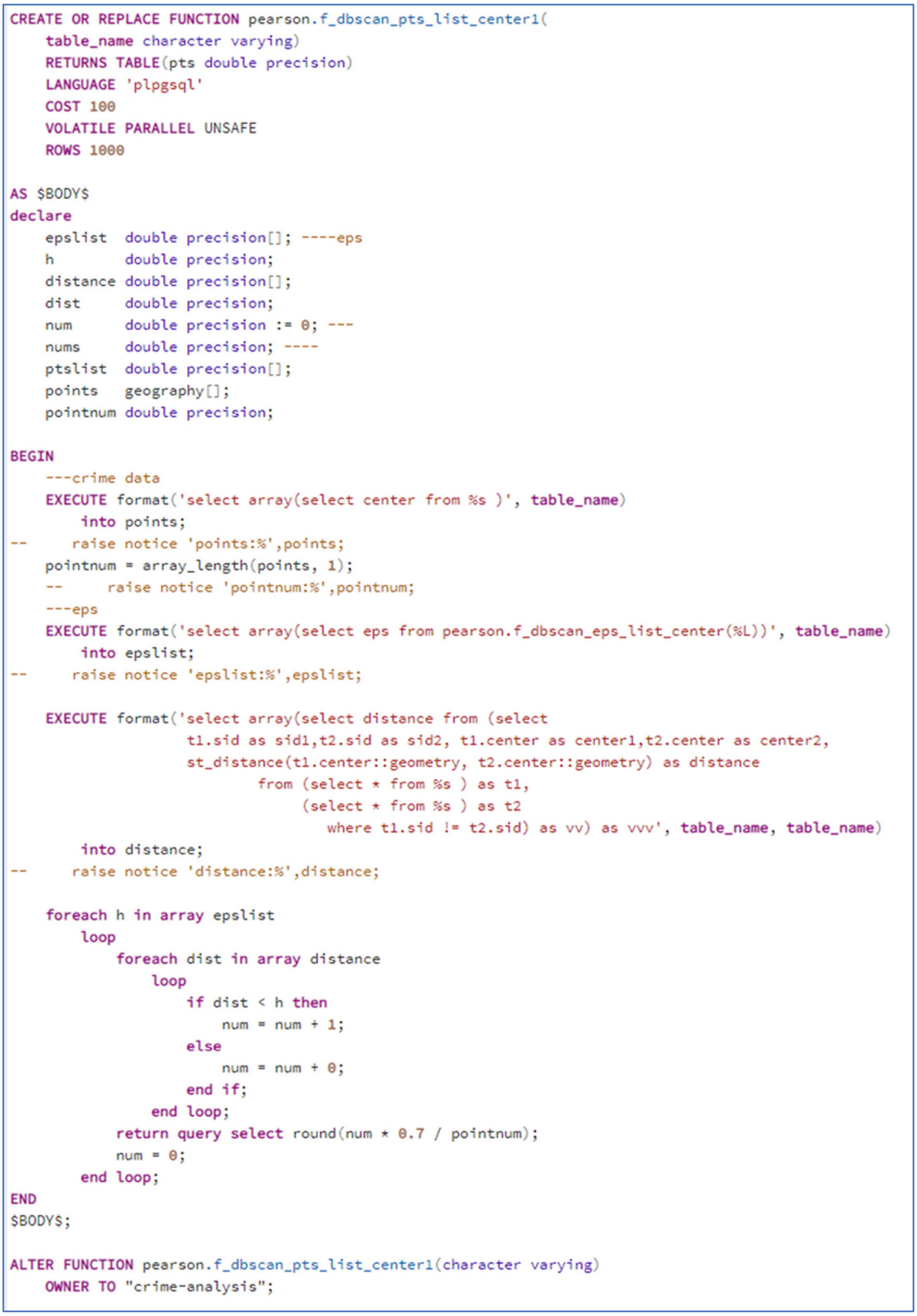
```
CREATE OR REPLACE FUNCTION pearson.f_dbscan_eps_list_center(
    table_name character varying)
    RETURNS TABLE(eps double precision)
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
    ROWS 1000

AS $BODY$
declare
    string varchar;
    rownum integer;
BEGIN
    EXECUTE format('select count(*)-1 from %s', table_name)
        into rownum;
    raise notice '%',rownum;
    string = format('select avg(distance) * (1 - 0.3 * 0.3) Eps_List
                        from (select *, (row_number() over ()) %% %s as index
                              from (select t1.sid,
                                           t2.sid,
                                           t1.center,
                                           t2.center,
                                           st_distance(t1.center::geometry, t2.center::geometry) as distance
                                    from (select * from %s) as t1,
                                         (select * from %s ) as t2
                                    where t1.sid != t2.sid
                                    order by t1.sid, distance) as v)
                                 as v1
                        group by index
                        order by Eps_List', rownum, table_name, table_name);
--     raise notice '%',string;
    return query
        EXECUTE string;
END
$BODY$;

ALTER FUNCTION pearson.f_dbscan_eps_list_center(character varying)
    OWNER TO "crime-analysis";
```

(2) Generate a list of Minpts parameter values:

The mathematical expectation method based on the self-decay value is used to generate the Minpts list, as shown in Formula 4. P_i is the number of objects in the Eps domain. After all K values are calculated, the Minpts parameter list is obtained, as shown in Formula 5.

$$MinPts_K = \frac{(1-\lambda)}{n}\sum_{i=1}^{n} P_i \qquad [4]$$

$$MinPts_{list} = \{MinPts_K | 1 \leq K \leq n\} \qquad [5]$$

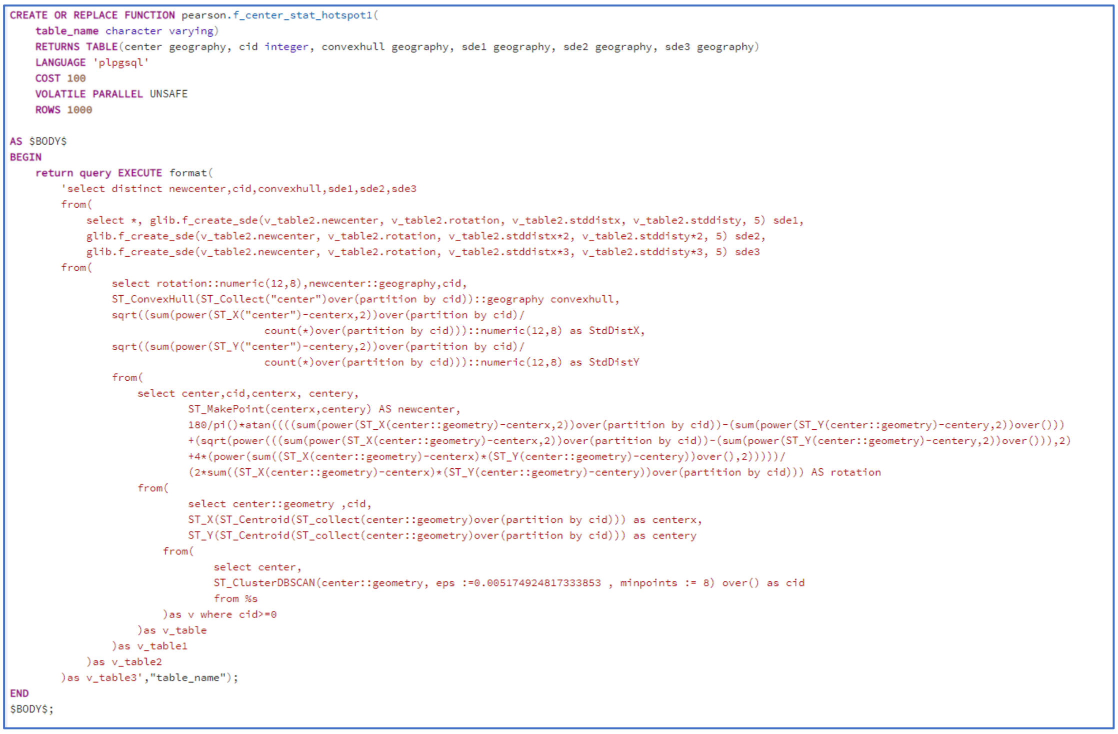


```
CREATE OR REPLACE FUNCTION pearson.f_dbscan_pts_list_center1(
    table_name character varying)
    RETURNS TABLE(pts double precision)
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
    ROWS 1000

AS $BODY$
declare
    epslist  double precision[]; ----eps
    h        double precision;
    distance double precision[];
    dist     double precision;
    num      double precision := 0; ---
    nums     double precision; ----
    ptslist  double precision[];
    points   geography[];
    pointnum double precision;

BEGIN
    ---crime data
    EXECUTE format('select array(select center from %s )', table_name)
        into points;
--     raise notice 'points:%',points;
    pointnum = array_length(points, 1);
    --      raise notice 'pointnum:%',pointnum;
    ---eps
    EXECUTE format('select array(select eps from pearson.f_dbscan_eps_list_center(%L))', table_name)
        into epslist;
--     raise notice 'epslist:%',epslist;

    EXECUTE format('select array(select distance from (select
                t1.sid as sid1,t2.sid as sid2, t1.center as center1,t2.center as center2,
                st_distance(t1.center::geometry, t2.center::geometry) as distance
                        from (select * from %s ) as t1,
                             (select * from %s ) as t2
                                where t1.sid != t2.sid) as vv) as vvv', table_name, table_name)
        into distance;
--     raise notice 'distance:%',distance;

    foreach h in array epslist
        loop
            foreach dist in array distance
                loop
                    if dist < h then
                        num = num + 1;
                    else
                        num = num + 0;
                    end if;
                end loop;
            return query select round(num * 0.7 / pointnum);
            num = 0;
        end loop;
END
$BODY$;

ALTER FUNCTION pearson.f_dbscan_pts_list_center1(character varying)
    OWNER TO "crime-analysis";
```

(3) Eps and Minpts parameter pair optimization: The Eps and Minpts parameter value pairs corresponding to different K values are selected in turn as the clustering parameters of the DBSCAN

algorithm, and the number of clustering results using Eps and Minpts parameter values corresponding to different K values is obtained, as shown in Formula 6.

$$Density_i = \frac{MinPts_i}{\pi Eps^2} \qquad [6]$$

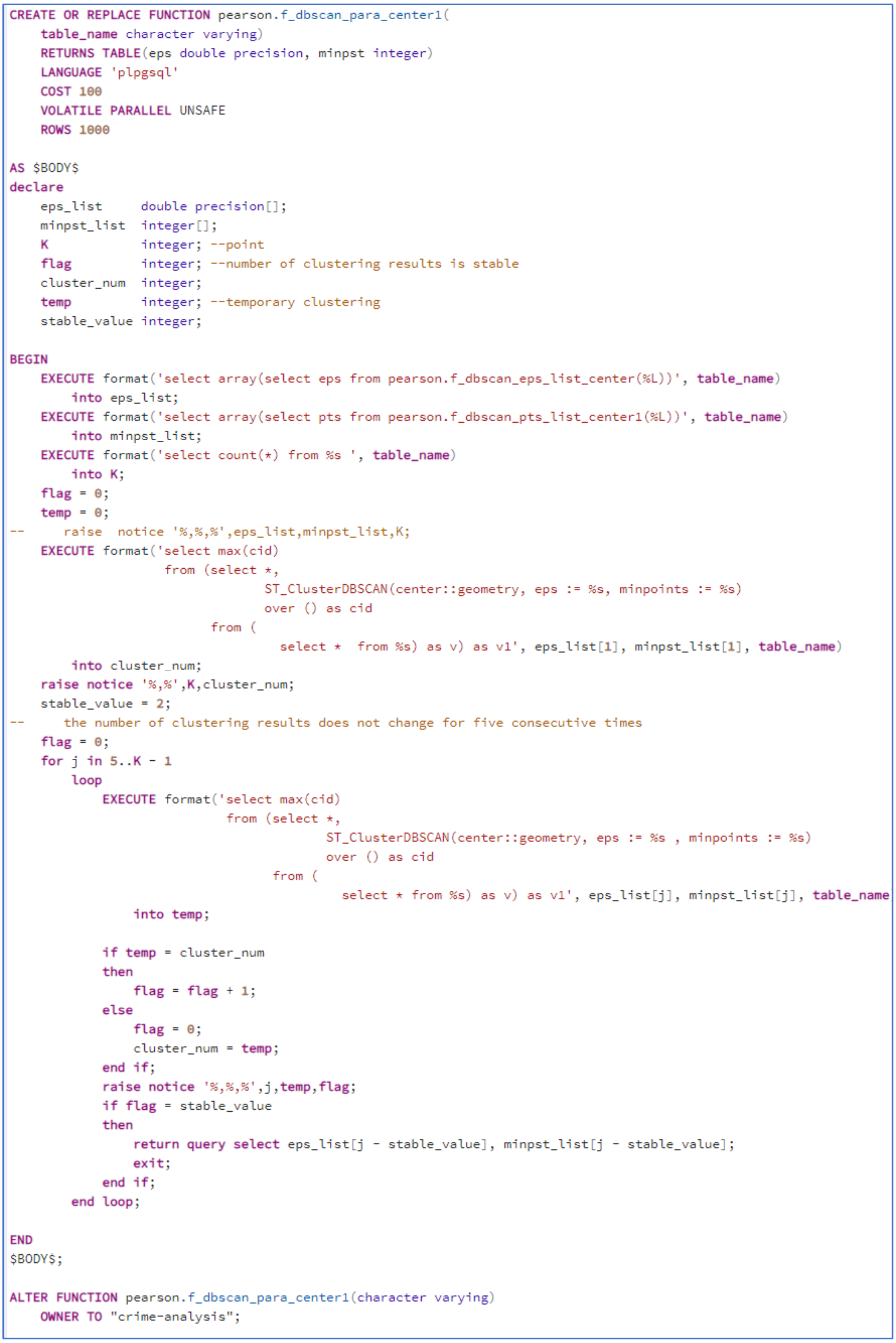


```
CREATE OR REPLACE FUNCTION pearson.f_center_stat_hotspot1(
    table_name character varying)
    RETURNS TABLE(center geography, cid integer, convexhull geography, sde1 geography, sde2 geography, sde3 geography)
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
    ROWS 1000

AS $BODY$
BEGIN
    return query EXECUTE format(
        'select distinct newcenter,cid,convexhull,sde1,sde2,sde3
        from(
            select *, glib.f_create_sde(v_table2.newcenter, v_table2.rotation, v_table2.stddistx, v_table2.stddisty, 5) sde1,
            glib.f_create_sde(v_table2.newcenter, v_table2.rotation, v_table2.stddistx*2, v_table2.stddisty*2, 5) sde2,
            glib.f_create_sde(v_table2.newcenter, v_table2.rotation, v_table2.stddistx*3, v_table2.stddisty*3, 5) sde3
        from(
                select rotation::numeric(12,8),newcenter::geography,cid,
                ST_ConvexHull(ST_Collect("center")over(partition by cid))::geography convexhull,
                sqrt((sum(power(ST_X("center")-centerx,2))over(partition by cid)/
                                        count(*)over(partition by cid)))::numeric(12,8) as StdDistX,
                sqrt((sum(power(ST_Y("center")-centery,2))over(partition by cid)/
                                        count(*)over(partition by cid)))::numeric(12,8) as StdDistY
                from(
                    select center,cid,centerx, centery,
                            ST_MakePoint(centerx,centery) AS newcenter,
                            180/pi()*atan((((sum(power(ST_X(center::geometry)-centerx,2))over(partition by cid))-(sum(power(ST_Y(center::geometry)-centery,2))over()))
                            +(sqrt(power(((sum(power(ST_X(center::geometry)-centerx,2))over(partition by cid))-(sum(power(ST_Y(center::geometry)-centery,2))over())),2)
                            +4*(power(sum((ST_X(center::geometry)-centerx)*(ST_Y(center::geometry)-centery))over(),2)))))/
                            (2*sum((ST_X(center::geometry)-centerx)*(ST_Y(center::geometry)-centery))over(partition by cid))) AS rotation
                    from(
                            select center::geometry ,cid,
                            ST_X(ST_Centroid(ST_collect(center::geometry)over(partition by cid))) as centerx,
                            ST_Y(ST_Centroid(ST_collect(center::geometry)over(partition by cid))) as centery
                        from(
                                select center,
                                ST_ClusterDBSCAN(center::geometry, eps :=0.005174924817333853 , minpoints := 8) over() as cid
                                from %s
                        )as v where cid>=0
                    )as v_table
                )as v_table1
            )as v_table2
        )as v_table3',"table_name");
END
$BODY$;
```

When the number of clusters in the clustering result tends to be stable, the optimal parameters are selected inversely through the K value. Stable interval refers to the stable interval obtained for a number of consecutive times with the same number of clusters that becomes stable with the increase in the K value, which is recorded as the optimal number of clusters. By selecting the corresponding K value, the corresponding Eps and Minpts are the optimal parameters.

This paper considers that when the number of clustering results does not change for five consecutive times, the number of clustering results is stable, and the current number of clustering results is recorded as the optimal number of clustering results. When the number of clustering results is stable, we continue the above operations until the number of clustering results changes.

The interval from the optimal number of clustering results to the change in the number of clustering results is called the stable area. For the stable area corresponding to all the Eps and Minpts parameter value pairs, the expected Eps and Minpts are obtained, respectively, and Eps and Minpts are taken as the optimal Eps and Minpts values.



```sql
CREATE OR REPLACE FUNCTION pearson.f_dbscan_para_center1(
    table_name character varying)
    RETURNS TABLE(eps double precision, minpst integer)
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
    ROWS 1000

AS $BODY$
declare
    eps_list     double precision[];
    minpst_list  integer[];
    K            integer; --point
    flag         integer; --number of clustering results is stable
    cluster_num  integer;
    temp         integer; --temporary clustering
    stable_value integer;

BEGIN
    EXECUTE format('select array(select eps from pearson.f_dbscan_eps_list_center(%L))', table_name)
        into eps_list;
    EXECUTE format('select array(select pts from pearson.f_dbscan_pts_list_center1(%L))', table_name)
        into minpst_list;
    EXECUTE format('select count(*) from %s ', table_name)
        into K;
    flag = 0;
    temp = 0;
--     raise  notice '%,%,%',eps_list,minpst_list,K;
    EXECUTE format('select max(cid)
                    from (select *,
                                 ST_ClusterDBSCAN(center::geometry, eps := %s, minpoints := %s)
                                 over () as cid
                          from (
                                   select *  from %s) as v) as v1', eps_list[1], minpst_list[1], table_name)
        into cluster_num;
    raise notice '%,%',K,cluster_num;
    stable_value = 2;
--     the number of clustering results does not change for five consecutive times
    flag = 0;
    for j in 5..K - 1
        loop
            EXECUTE format('select max(cid)
                            from (select *,
                                         ST_ClusterDBSCAN(center::geometry, eps := %s , minpoints := %s)
                                         over () as cid
                                  from (
                                           select * from %s) as v) as v1', eps_list[j], minpst_list[j], table_name
                into temp;

            if temp = cluster_num
            then
                flag = flag + 1;
            else
                flag = 0;
                cluster_num = temp;
            end if;
            raise notice '%,%,%',j,temp,flag;
            if flag = stable_value
            then
                return query select eps_list[j - stable_value], minpst_list[j - stable_value];
                exit;
            end if;
        end loop;

END
$BODY$;

ALTER FUNCTION pearson.f_dbscan_para_center1(character varying)
    OWNER TO "crime-analysis";
```