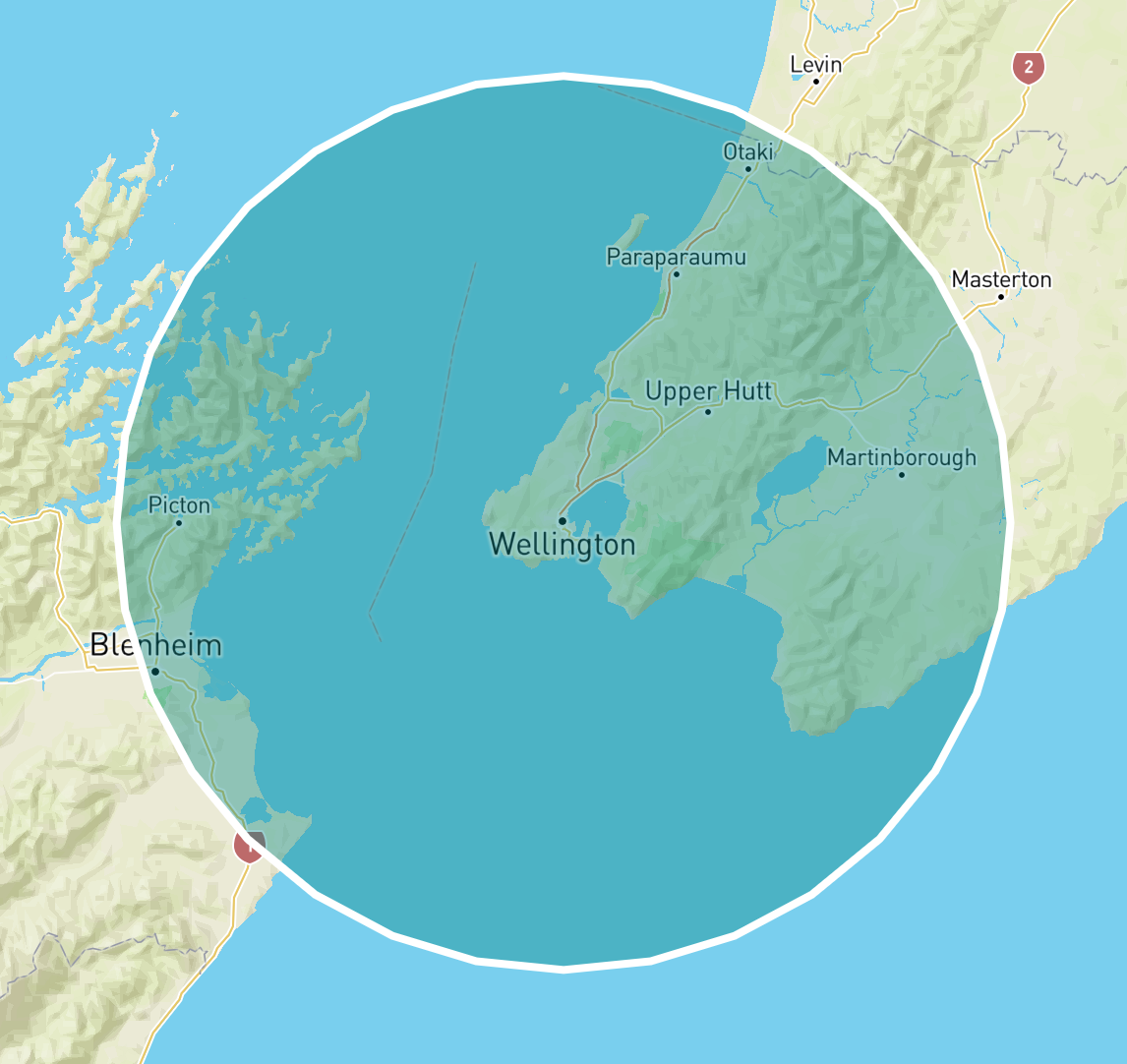
# 通过PostGIS生成受陆地限制的网格点

当我在市场上买电动车的时候，我想知道的事是，电动车在我必须给它充电前能走多远。即便知道了一个数字范围，我仍然不能知道我是否能够抵达一个特定的目的地。

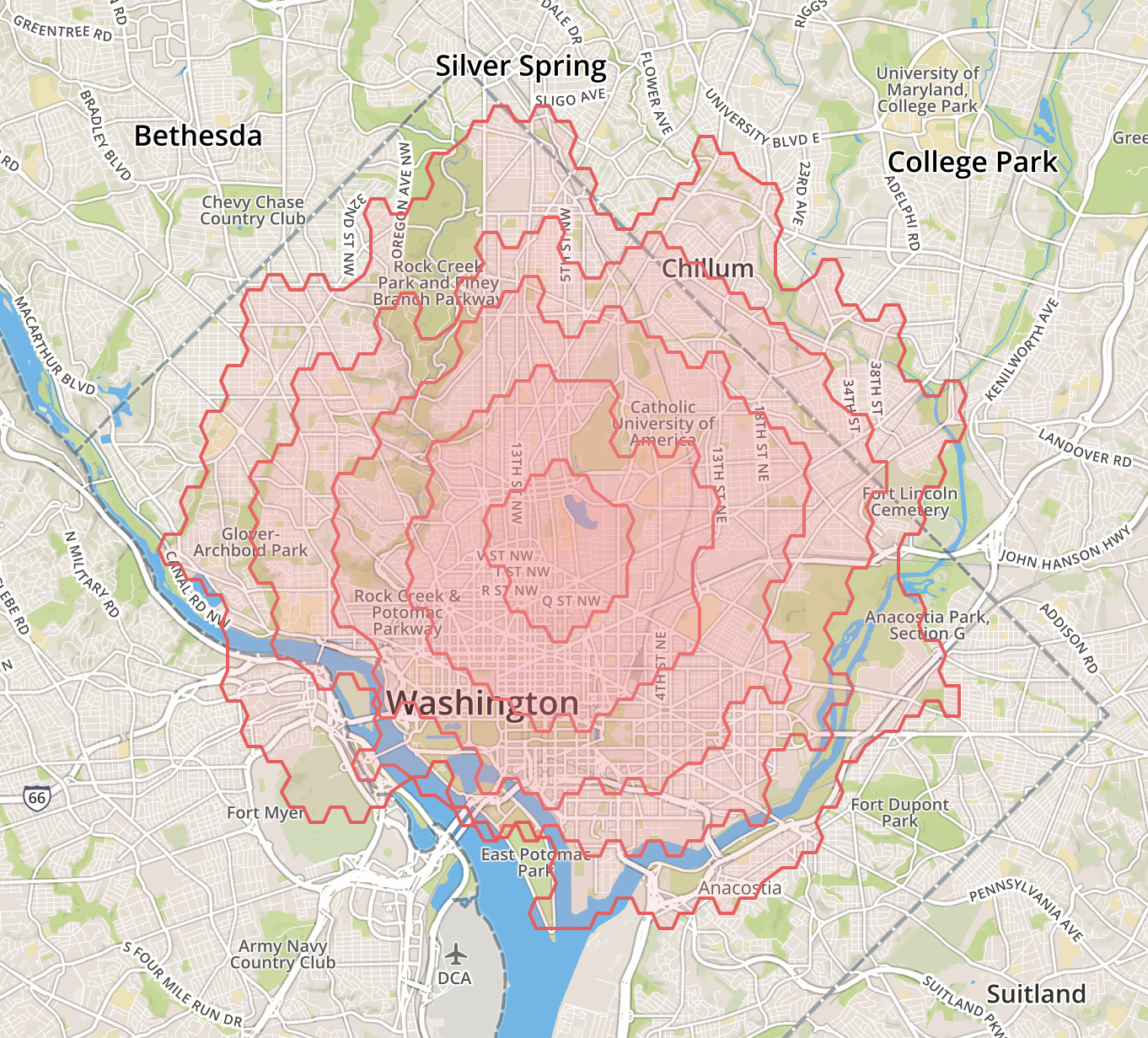
所以我想在EV guide（https://evguide.nz/）上增加一个可视化的机动车行驶距离，来使这个范围更加明显。

普通的解决方案是使用形式距离的范围作为半径来画一个圆，如下图所示：



但是，这样的方法实在是过于草率而且并不精准，因为道路的距离并不总是等于直线距离。

作为代替，在道路地图上我需要一个给定距离的轮廓，也就是**等距线**。下图展示了Mapbox博客中的几个等距线示例：

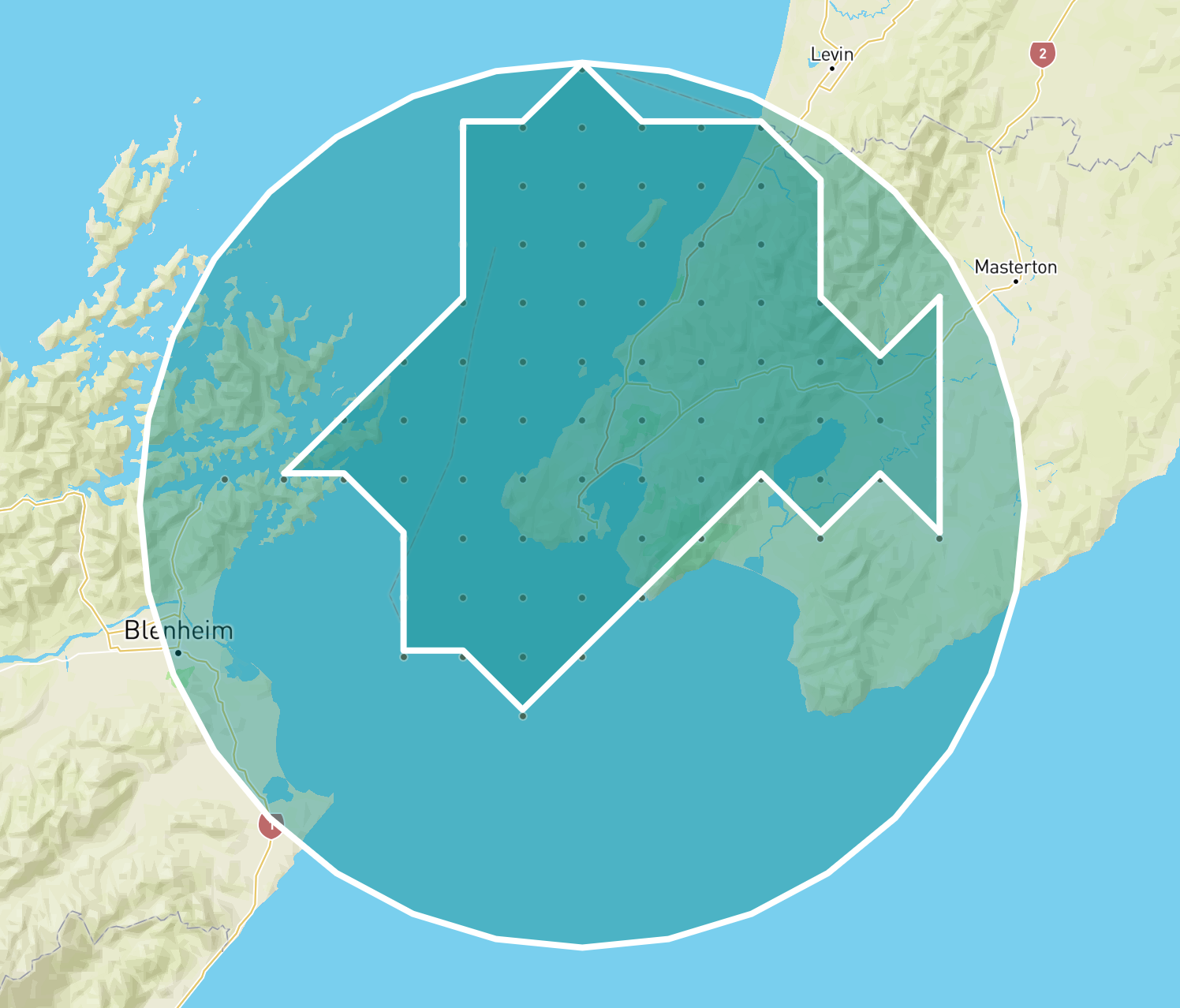


我相信我可以使用Mapbox与PostGIS结合的方式来生成等距线，并且将结果在Mapbox上进行展示。

* 基础的方法有以下几点：
* 在给定的中心点上生成网格点
* 对每个网格点，使用Mapbox Directions API（该方法提供了给定位置之间的道路行驶距离的接口）来计算路程的距离
* 丢弃所有比期望距离远的点

计算剩下的点的凸包，该凸包就是等距线。（译者注：即PostGIS中的ST\_Convexhull函数）

通过一些实践，我发现Mapbox Directions API会认为海上的一些位置是可以到达的，例如：



即便特斯拉可以漂浮，但明显不能在海上开这么远。举例而言，有些可能是由于轮渡航线，但并没有向北的轮渡航线。我认为把等距离限制在陆地上会更好。这意味着我需要排除网格中不在陆地上的点。

使用PostgreSQL和PostGIS，可以完成以下两项任务：

* 生成相关地区的网格点
* 舍弃不在陆地上的网格点

## 生成网格点

在刚创建的数据库中，需要安装PostGIS扩展：

create extension postgis;

create extension postgis\_topology;

我尝试的第一个方法包括创建框（译注：不是指外接矩形box2d，而是多边形的边界框），并使用generate\_series来生成点的坐标。

该方法需要生成边界框的坐标。这里有一个问题：当我想在EV可视化范围中展示时，我希望使用公里作为单位，而PostGIS并没有提供一个简单的方法来指定公里作为几何类型的单位来画圆。

这个问题让我转而去看地理类型（译注：PostGIS中的两种数据类型：geometry为几何类型，geography为地理类型），但我又遇到了另一个问题。在新西兰为中心的的场景下，我的多边形碰巧与国际日期变更线相交，PostGIS无法产生一个有意义的结果。

最终我发现了ST\_Project函数，该函数允许我通过距离与方位角来移动点。

我首先生成了一条间距为50公里的点线：

select st\_project(st\_setSrid(st\_point(174.8431, -41.2581), 4326)::geography, s.a \* 50000, pi() / 2) as pt

from generate\_series(-10, 10, 1) as s(a);



几句题外话：

点与点之间50km的距离并不很实用——这样的网格太稀疏了，因而不能产生有用的等距线。事实上，我用了10km，但这很难看清，因为这些点合并成一个单一的颜色点。

这篇文章的截图来自pgAdmin 4，它能够可视化几何和地理类型的结果集。它是非常有用的。（译者吐槽：用QGIS不是更好吗……）

然后，使用另一个generate\_series函数将其扩展为一个网格：

with row as (

select st\_project(st\_setSrid(st\_point(174.8431, -41.2581), 4326)::geography,

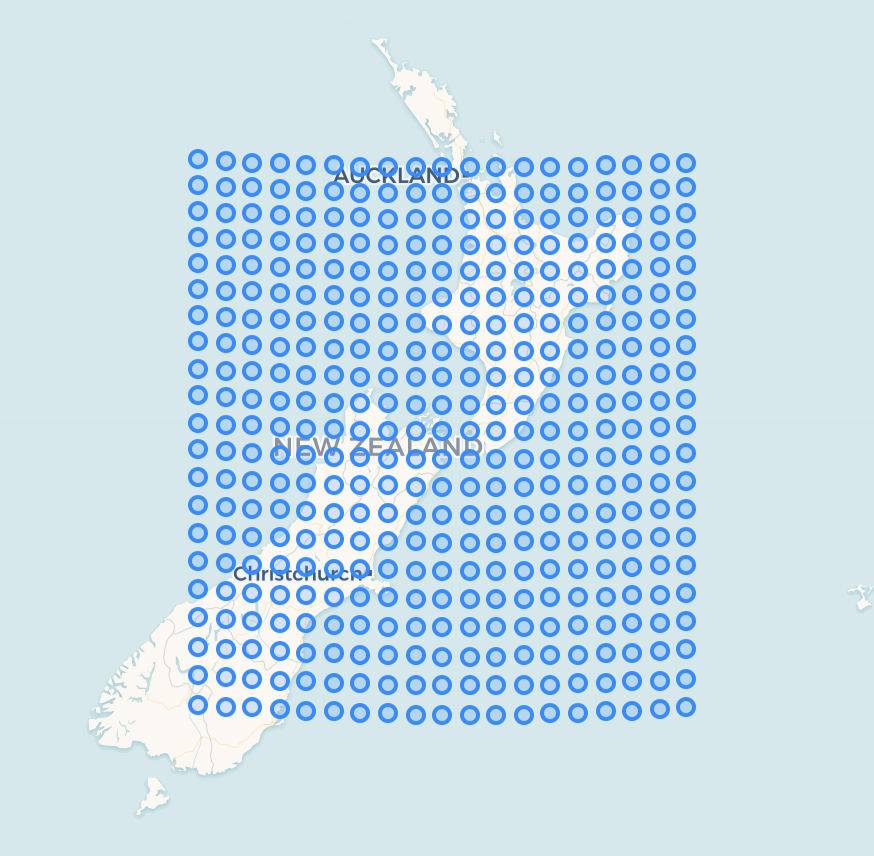
s.a \* 50000, pi() / 2) as pt

from generate\_series(-10, 10, 1) as s(a)

)

select st\_project(pt, s.a \* 50000, 0) as projected\_pt

from row, generate\_series(-10, 10, 1) as s(a);



说明

网格看起来不像正方形的原因是一种显示效果。有些点跨越日期线，所以它们不会显示，（这些点）出现在地图的另一端。

一个明显的约束条件是，点到原点的距离不能超过一定距离，所以我在里面加入了以下条件，用来去掉了半径500km以外的点：

with row as (

select st\_project(st\_setSrid(st\_point(174.8431, -41.2581), 4326)::geography,

s.a \* 50000, pi() / 2) as pt

from generate\_series(-10, 10, 1) as s(a)

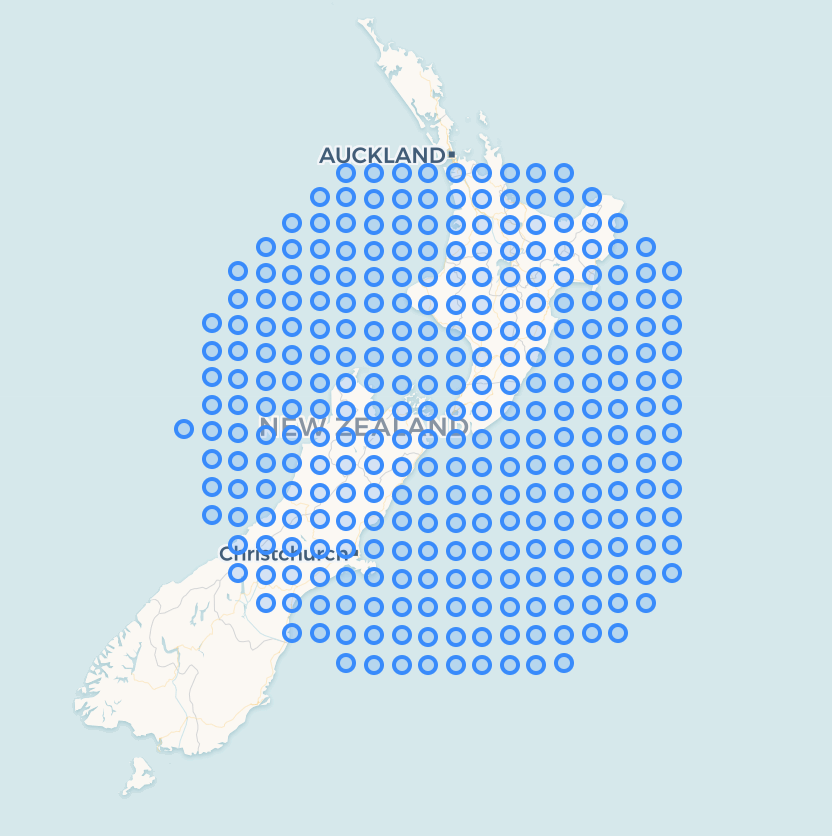
)

select st\_project(pt, s.a \* 50000, 0) as projected\_pt

from row, generate\_series(-10, 10, 1) as s(a)

where st\_distance(st\_setSrid(st\_point(174.8431, -41.2581), 4326)::geography

, st\_project(pt, s.a \* 50000, 0)) < 500000;



以上就是点网格的创建方式。接下来是第二个任务。

## 丢弃海面上的网格点

下一步是移除在海面上的点。

新西兰的土地等高线可以从Land Information New Zealand获得，它是一个21MB的KML文件，包含由几十万个点组成的多边形。为了将这些数据导入PostgreSQL，我知道我可以使用GDAL包中的ogr2ogr工具(macOS上的命令为brew install gdal):

ogr2ogr -f "PostgreSQL" PG:"host=localhost user=alex dbname=isodist password=" nz-coastlines-and-islands-polygons-topo-150k.kml

上述命令将几何对象导入到表中，并作为二进制字符串进行存储：

isodist=# \dt

List of relations

┌────────┬─────────────────────────────────────────────────┬───────┬───────┐

│ Schema │ Name │ Type │ Owner │

├────────┼─────────────────────────────────────────────────┼───────┼───────┤

│ public │ nz coastlines and islands polygons (topo 1:50k) │ table │ alex │

└────────┴─────────────────────────────────────────────────┴───────┴───────┘

isodist=# \d public."nz coastlines and islands polygons (topo 1:50k)"

┌──────────────┬───────────────────┬───────────┬──────────┬

│ Column │ Type │ Collation │ Nullable │

├──────────────┼───────────────────┼───────────┼──────────┼

│ ogc\_fid │ integer │ │ not null │

│ wkb\_geometry │ bytea │ │ │

│ name │ character varying │ │ │

│ description │ character varying │ │ │

└──────────────┴───────────────────┴───────────┴──────────┴

将表重命名为海岸线之后，运行以下查询来查看哪些几何图形包含给定的点(例如(175.9065，-38.7797)):

with geoms as (

select st\_setsrid(encode(wkb\_geometry, 'hex'), 4326) as g from coastlines

)

select st\_geometrytype(g)

, st\_npoints(g)

, st\_contains(g, st\_setsrid(st\_point(175.9065, -38.7797), 4326))

from geoms

order by 2 desc;

几何图形必须编码成文本才能传递给PostGIS的函数（译者注：其实有::geometry强制转换，作者可能不知道）。然后使用ST\_Contain来判断几何图形A是否包含B。

最后，结合点网格查询与where子句放弃那些海面上的网格点：

with geoms as (

select st\_setsrid(encode(wkb\_geometry, 'hex'), 4326) as g from coastlines

)

, row as (

select st\_project(st\_setSrid(st\_point(174.8431, -41.2581), 4326)::geography, s.a \* 50000, pi() / 2) as pt

from generate\_series(-10, 10, 1) as s(a)

)

select st\_project(pt, s.a \* 50000, 0) as projected\_pt

from row, generate\_series(-10, 10, 1) as s(a)

where st\_distance(st\_setSrid(st\_point(174.8431, -41.2581), 4326)::geography, st\_project(pt, s.a \* 50000, 0)) < 500000

and exists(select 1 from geoms where st\_nPoints(g) > 1000 and st\_contains(g, st\_project(pt, s.a \* 50000, 0)::geometry));

（上面的SQL中）只用ST\_Contains判断由超过1000个点(st\_nPoints(g) > 1000)组成的陆地等高线，因为我没有必要对所有小岛之类的地方进行全面检查。

结果如下图所示：



此时，使用PostgreSQL JSON函数的将结果集写成JSON坐标数组，并与Mapbox Directions API一起使用：

with geoms as (

select st\_setsrid(encode(wkb\_geometry, 'hex'), 4326) as g from coastlines

)

, row as (

select st\_project(st\_setSrid(st\_point(174.8431, -41.2581), 4326)::geography, s.a \* 50000, pi() / 2) as pt

from generate\_series(-10, 10, 1) as s(a)

)

, grid as (

select st\_project(pt, s.a \* 50000, 0)::geometry as projected\_pt

from row, generate\_series(-10, 10, 1) as s(a)

where st\_distance(st\_setSrid(st\_point(174.8431, -41.2581), 4326)::geography, st\_project(pt, s.a \* 50000, 0)) < 500000

and exists(select 1 from geoms where st\_nPoints(g) > 1000 and st\_contains(g, st\_project(pt, s.a \* 50000, 0)::geometry))

)

, coords as (

select st\_x(projected\_pt) as lon, st\_y(projected\_pt) as lat

from grid

)

select json\_agg(row\_to\_json(coords)) from coords;

## 结论

有几点需要在结论中说明：

PostgreSQL和PostGIS提供了强力的数据集工具，可以使用一个短的吓人的SQL，来实现生成受土地限制的网格点。

将土地等高线作为公共数据集使我们能够在像我的项目、公民科学等这样的业余项目中利用地理数据做一些有用的事情。

## 译注

主要用到的两个PostGIS函数整理如下：

### ST\_ConvexHull

描述

返回包含所有输入几何对象的凸包。

该函数支持3D对象，并且不会删除z坐标。

语法

geometry ST\_ConvexHull(geometry geomA);

参数

geomA

几何对象。

举例

SELECT ST\_AsText(ST\_ConvexHull(

ST\_Collect(

ST\_GeomFromText('MULTILINESTRING((100 190,10 8),(150 10, 20 30))'),

ST\_GeomFromText('MULTIPOINT(50 5, 150 30, 50 10, 10 10)')

)) );

st\_astext

-------------------------------------------------------

POLYGON((50 5,10 8,10 10,100 190,150 30,150 10,50 5))

(1 row)

### ST\_Contain

描述

输入两个几何对象A和B，当且仅当任一B上的点都不在A的外部，且存在至少一个B上的点在A的内部，则返回True，这种情况称为A包含B。注意，A不包含A的边界，但A包含A自身。该函数与ST\_ContainsProperly（完全包含）不同，A不完全完全包含A自身。

如果几何对象B完全在A中则返回True。为使该函数生效，A和B必须处于同一坐标投影中，即拥有相同的SRID。该函数与ST\_Within相反，除非输入的几何对象无效，ST\_Contains(A,B)等价于ST\_Within(B,A)。

GeometryCollection不能作为输入参数。

语法

boolean ST\_Contains(geometry geomA, geometry geomB);

参数

geomA

geomB

几何对象。

举例

1. LineString对象不包含在其上的Point对象：

SELECT ST\_Contains('LineString(0 0,1 1)'::geometry,'Point(1 1)'::geometry);

st\_contains

-------------

f

(1 row)

1. Polygon对象包含在其中的Point对象：

SELECT ST\_Contains('Polygon((0 0,2 0,2 2,0 2,0 0))'::geometry,'Point(1 1)'::geometry);

st\_contains

-------------

t

(1 row)