Aplicaciones Web para visualizacion de datos georeferenciados

https://github.com/ramirojc/SciPyLA

Ramiro Caro

- Ingeniero Electrónico Universidad Nacional de Córdoba.
- 8 años de experiencia en área de adquisición y procesamiento de datos en Petróleo y Gas.
- Como gerente de servicios para el área de Evaluación y Caracterización de Reservorios, comencé a utilizar herramientas de Machine Learning para optimizar la eficiencia en la utilización de los recursos.
- En 2017 deje la Empresa para especializarme en Inteligencia Artificial y Ciencia de Datos, desde entonces trabajo como desarrollador independiente, en ingeniería de proyectos de automatización y optimización de procesos.

- Desarrollo completo en Python
- Versatilidad e Interactividad
- Simple Deployment como WebApp
- Open Source

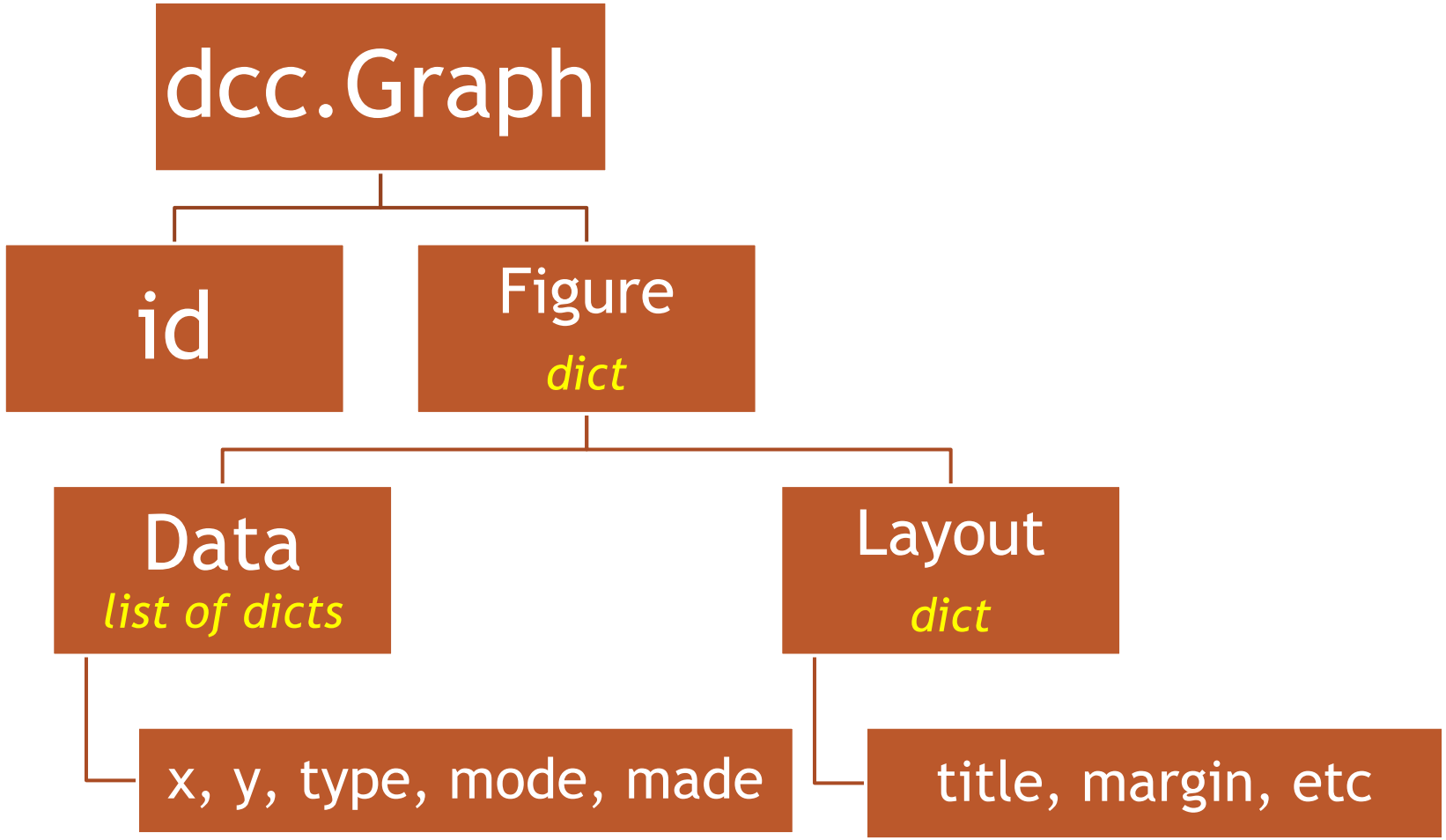
Organizacion

- Estructura de la aplicacion
- Componentes HTML, Dash Core, Graficos
- Componente Mapa: Scattermapbox
  - Mapa ScatterGeo y Choropleth
- Carga de datos
- Callbacks
  - Entradas y salidas simples
  - Entradas y salidas multiples
- Layout de la aplicacion
- Codificacion de variables con color y tamano
- Retornar componente desde callback
- Eventos generados por mapa
- Deployment en Heroku

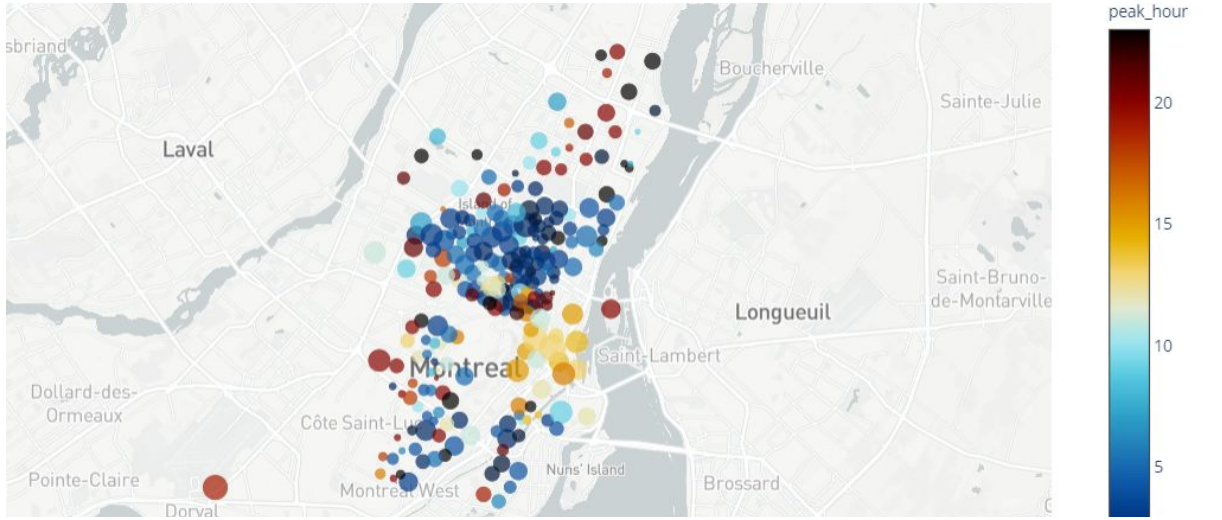
Estructura de aplicacion

- Layout (Visual)
  - Componentes HTML
  - Componentes Core
  - Otros: Industrial / Bootstrap / Personalizados
- Callbacks (Interacciones)
  - Inputs
  - Outputs
  - States

Graficos


dcc.Graph
id
Figure
dict
Data
list of dicts
Layout
dict
x, y, type, mode, made
title, margin, etc

Datos Georeferenciados



Descripcion de datos

- Puntos de distribucion, Regio Nova Friburgo, RJ, Brasil
  - lat , Latitud
  - lon , Longitud
  - FIC , Frecuencia Interrupcion Servicio
  - DIC , Duracion de las interrucpciones
  - CONJ , Conjunto de consumidores (Sub Estacion)
  - ENE_m , Consumo mensual (m = Numero de mes)
  - Otros: Subestacion, Perdidas, Barrio, Puntos de conexion

Callbacks

- Output : Componente que va a ser modificado
  - Component_id = Id del componente a modificar
  - Component_property > Propiedad / Atributo que se modifica
- Input : Componente que dispara la modificación
  - Component_id : Id del component que dispara el callback
  - Component_property : Propiedad que al modificarse dispara el callback
- State: Informacion de estado
  - Se utiliza para pasar información de un componente que no se modifica, pero que cuya información se usa en el callback

Estilizacion

- Cascading Style Sheets, CSS
  - Tipografia, Dimensiones, Colores, Margenes, etc
- Clases
  - Son encapsulamientos de propiedades que se le pueden asignar como atributo a un componente HTML
- En DASH Podemos asignar un CSS implicita o explicitamente.

Mapa de distribucion Electrica

Seleccione Regiones

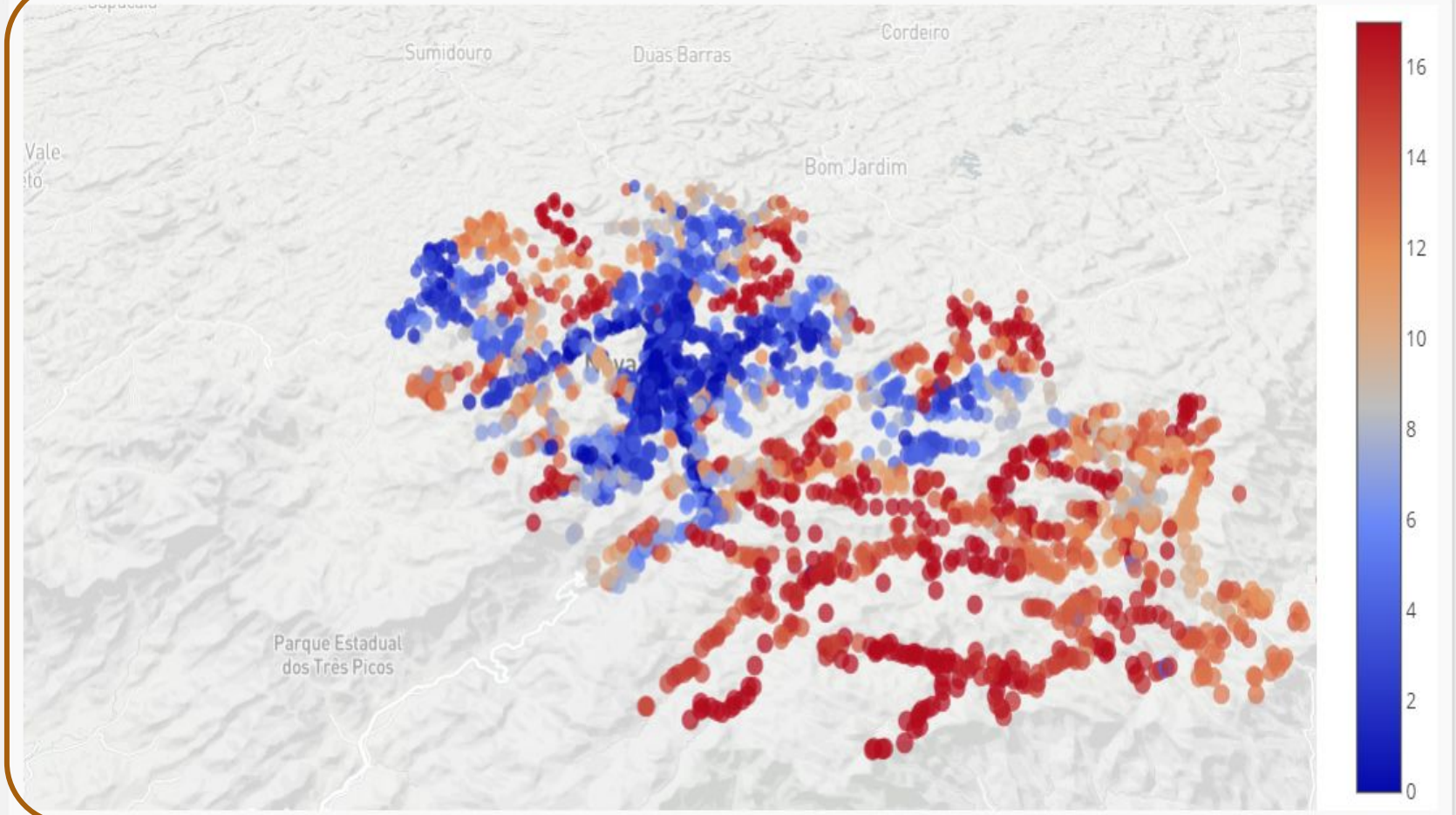
× Boa Aventura × Nova Friburgo × Campo do Coelho × Vale dos Peoes × Conselheiro Paulino

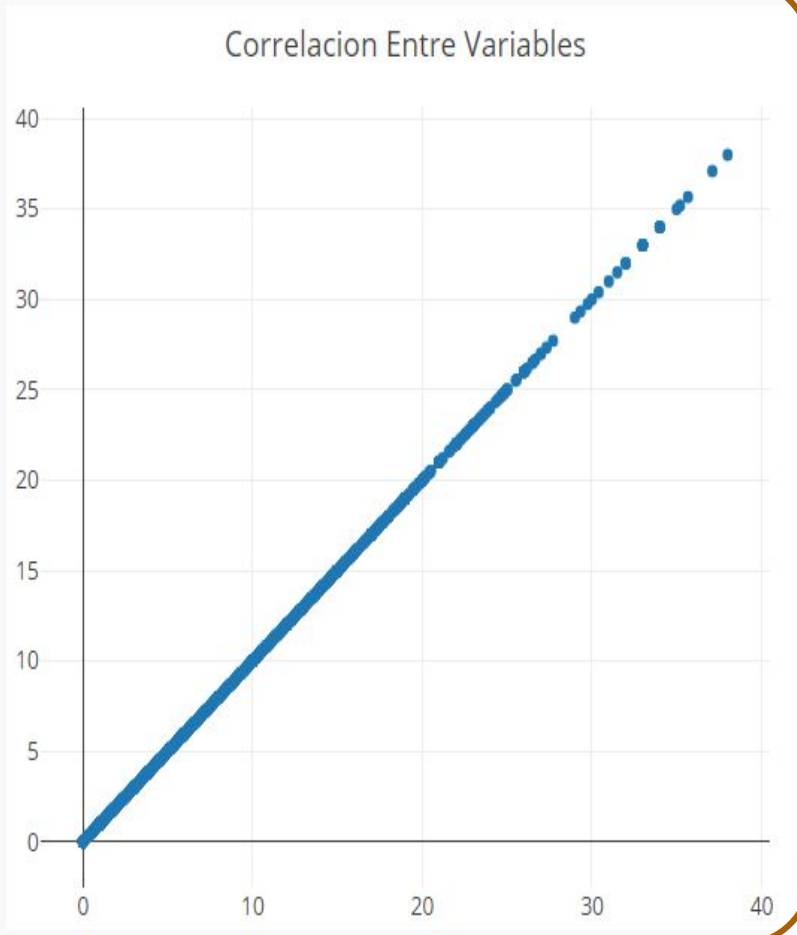
Codificacion color

- (•) Frecuencia de Cortes
- ( ) Duracion Media de Cortes
- ( ) Grupo Electrico

Codificacion tamaño

- (•) No Codificar
- ( ) Duracion Media de Cortes
- ( ) Consumo Total





Mapa de distribucion Electrica

Seleccione Regiones

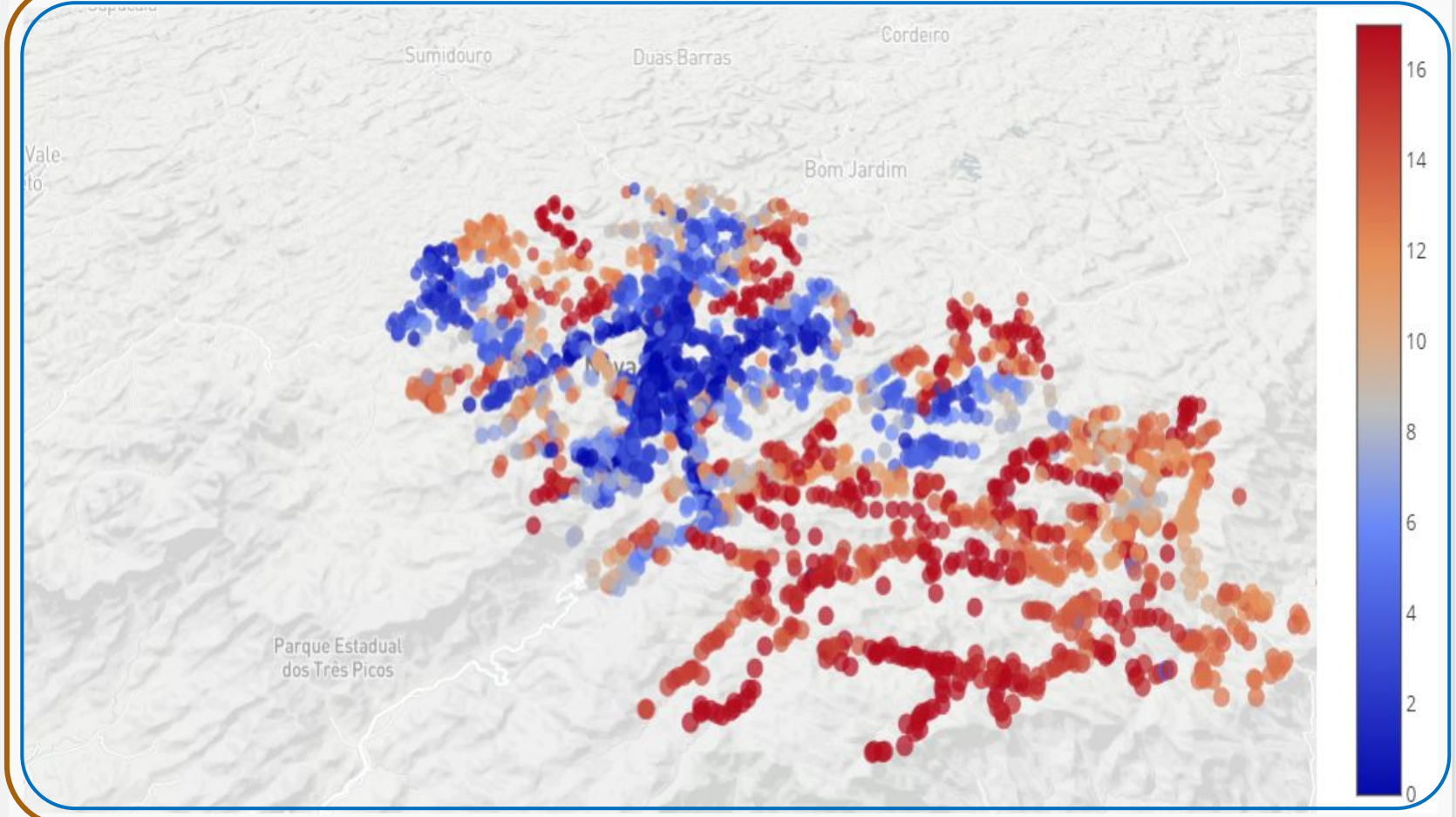
× Boa Aventura × Nova Friburgo × Campo do Coelho × Vale dos Peoes × Conselheiro Paulino

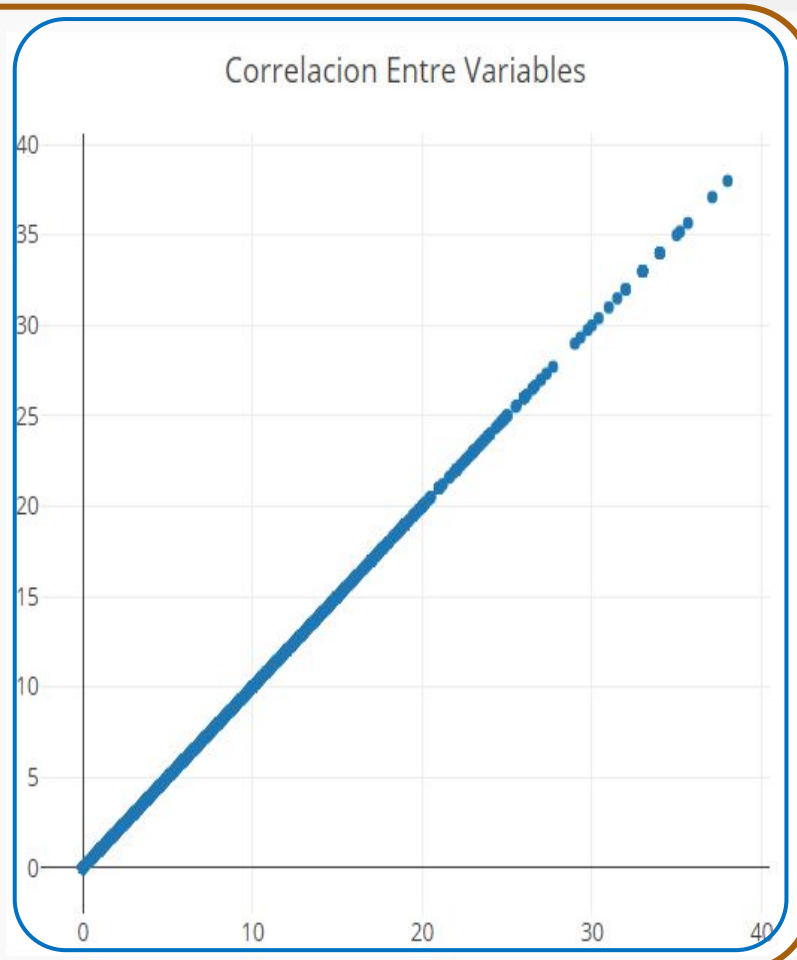
Codificacion color

- (•) Frecuencia de Cortes
- ( ) Duracion Media de Cortes
- ( ) Grupo Electrico

Codificacion tamaño

- (•) No Codificar
- ( ) Duracion Media de Cortes
- ( ) Consumo Total





Propiedades de ScatterMapBox

- clickData
- hoverData
- selectedData
- relayoutData
- className
- config

Estructura selectedData

Points es una lista de diccionarios

Cada diccionario es un punto

```
{
'points': [
    {'curveNumber': 0,
    'pointNumber': 487, 'pointIndex': 487,
    'lon': -42.58565013434617, 'lat': -22.355343793381337,
    'text': 'Hover Info', 'marker.size': 25, 'marker.color': 17}
    , (…) ],
'lassoPoints': {'mapbox': [ … ]}
}
```

Estructura Map Layout

{'mapbox.center’:
{'lon': -42.31672003043832,
'lat': -22.404623237098193},
'mapbox.zoom': 10,
'mapbox.bearing': 0,
'mapbox.pitch': 45}

Deployment en Heroku

- Instalar gunicorn
- Crear requirements.txt y Procfile (web: gunicorn app:server)
- Navegar hasta la carpeta de Proyecto
  - Git init
  - Heroku create my-dash-app
  - Git add
  - Git commit –m “Commit Inicial”
  - Git push heroku master