Hacer modelos en GEE

# Modelos en GEE

Google Earth Engine (GEE) es una herramienta muy útil, dentro de la cuál se pueden generar modelos matemáticos para las distintas problemáticas que se deseen visualizar dentro de GEE. Sin embargo, el desarrollo no se debe limitar al uso exclusivo de GEE pues esto nos restringiría al uso de otras herramientas desarrolladas por distintos individuos o grupos de trabajo. Bajo esta primicia, en este pequeño tutorial vamos a mostrar como se podemos exportar un modelo creado en otras plataformas y poder exportarlo a GEE.

En este sentido vamos a usar la plataforma GEE para exportar modelos y poder hacer la visualización de los resultados en GEE.

# Bases

Un modelo implica en términos muy simples implica poder describir como se comporta algo en lo cual estamos interesados y encontrar las causas que lo explican. Hay muchos tipos de modelos desde los muy complejos hasta los modelos muy simple, pero una forma de ver los es simplemente como funciones,

Donde normalmente los llamamos variables explicativas y a la variable descriptiva. Y es nuestro modelo, un modelo simple puede ser la combinación lineal de los elementos

o complejo

donde es una función que determina hasta que potencia se eleva a la variable .

En estos casos determinar un modelo implicaría encontrar la variables () que se consideran que mejor explican a la variable y encontrar . Haciendo uso de los datos propios de cada problemática ajustar el modelo implica poder encontrar las .

GEE tiene herramientas para poder hacer modelos sin embargo, los modelos dentro de GEE puede que no se ajusten a las problemáticas que se desean analizar, o no se tienen las herramientas disponibles para generar modelos complejos dentro de GEE. Por tal motivo, podemos obtener un modelo fuera de GEE y si se tiene una expresión como las descritas anteriormente es posible generar las predicciones dentro de GEE para su visualización.

En este ejemplo vamos a obtener el contaminante *particle matter* de menos de (**PM 2.5**) para la zona metropolitana del valle de México. Para tal motivo vamos a utilizar los datos que se encuentran dentro de GEE, en este caso vamos a utilizar la colección de imágenes **‘MODIS/006/MCD19A2\_GRANULES’** la cual contiene el resultado de algoritmo *Multi-Angle Implementation of Atmospheric Correction* (MAIAC), la banda que vamos a utilizar es la *Aerosol Optical Depth at 047 micron* esta banda es el espesor óptico de . Dicha banda ha sido utilizada para obtener los contaminantes y los cuales con lleva altos riesgos de la salud al tener una exposición prolongada.

## Colección de Imágenes

Para hacer uso de la colección de imágenes se hace usando la siguiente linea

var AOD\_MODIS\_fix = ee.ImageCollection("MODIS/006/MCD19A2\_GRANULES");

La collección de imágenes contiene las imágenes disponibles del conjunto de datos, lo que normalmente son muchas imágenes disponibles, dentro de las cuales la mayoría no vamos a utilizar, para esto los objetos ‘ee.ImageCollection’ tienen métodos que nos ayudan a poder filtrar las imágenes dentro de la colección de imágenes.

La siguiente linea de código nos filtra la colección sólo con las imágenes que se tomaron del ‘2005-01-01’ al ‘2016-01-01’.

var AOD\_MODIS\_fix = ee.ImageCollection("MODIS/006/MCD19A2\_GRANULES").filterDate('2005-01-01', '2016-01-01')

La siguiente linea filtra las imágenes que contienen al punto cercano a CentroGeo (-99.221411, 19.29183)

var AOD\_MODIS\_fix = ee.ImageCollection("MODIS/006/MCD19A2\_GRANULES").filterBounds(ee.Geometry.Point(-99.221411, 19.29183))

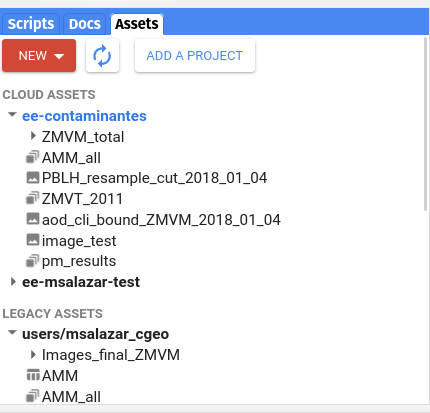
Estas las podemos combinar para aplicar las distintas acciones de forma concatenada, es decir, podemos filtrar primero por fecha y después al resultado filtrar las imágenes que intersectan al punto dentro de la misma ejecución, como se muestra a continuación

var AOD\_MODIS\_fix = ee.ImageCollection("MODIS/006/MCD19A2\_GRANULES")  
.filterDate('2005-01-01', '2016-01-01')   
.filterBounds(ee.Geometry.Point(-99.221411, 19.29183));

Este tipo de acciones las podemos implementar no sólo con filtrados sino con distintas acciones disponibles en GEE.

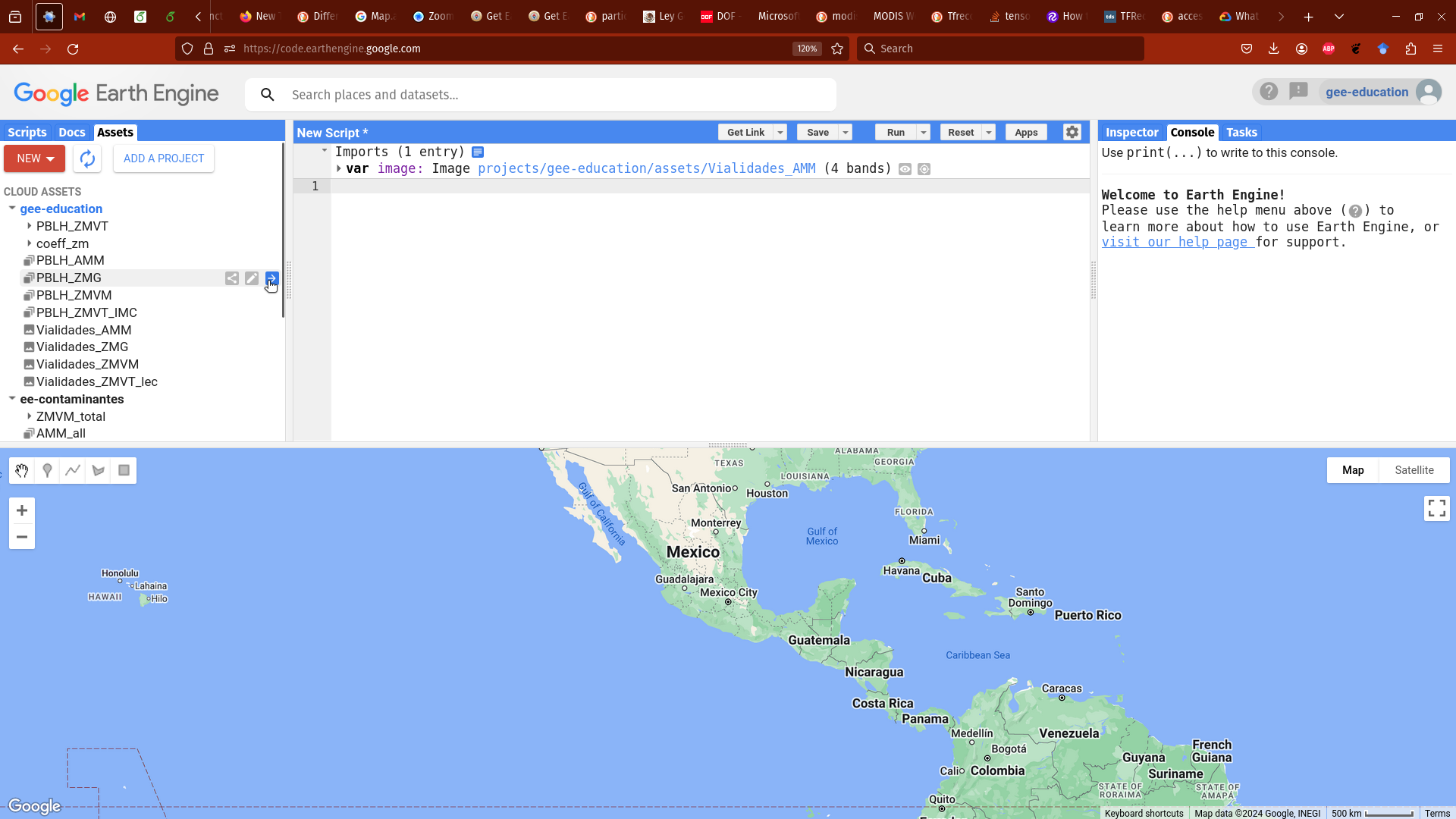
## Subir datos a GEE

En la pestaña ‘Assets’ dentro del editor de código de GEE (panel superior izquierdo) podemos añadir elementos para hacer uso de los mismos como se muestra en la imagen.

.

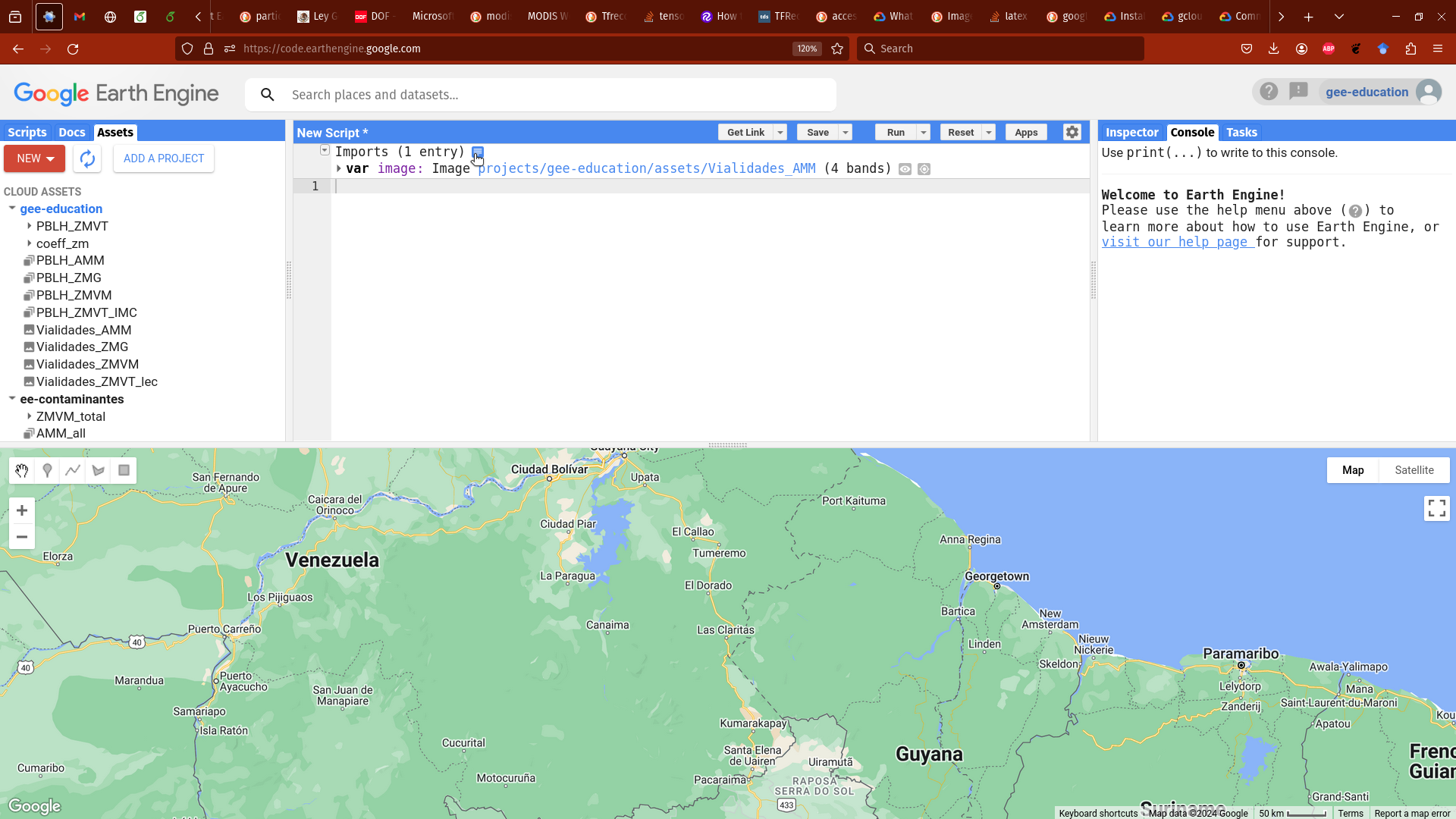
Los elementos que podemos añadir a GEE son imágenes (.tif, tiff), tablas en archivos shape (.shp, .shx, .dbf, .prj o .zip) o archivos csv (.csv). También podemos agregar colecciones de imágenes o folders que contengan este tipo de archivos. Los elementos que subimos a GEE se almacenan en proyectos en la nube de Google, esto hace que esten accesibles desde GEE cada vez que sea necesario. Al estar dentro de la nube de Google las políticas de acceso a los mismos son las de la Nube de Google y también son accesibles desde las API de Google cloud como [CLI](https://cloud.google.com/cli?hl=en).

Para hacer uso de estos elementos denro del editor de código de GEE se pueden insertar haciendo uso de la interfaz gráfica como se muestra en la imagen @fig:ins\_elem. Los elementos añadidos se insertaran al inicio de nuestro código como un preámbulo llamado Imports, se pueden editar los nombres para que los nombres de las variables donde se almacenan sean de acuerdo a las políticas del desarrollador.



insertar elementos

Si se desea compartir los elementos que se encuentran en el preámbulo Imports se hace click en el boton azul a lado de Imports esto abrira una ventana la cual contendra todos los elementos que se encuentran dentro de el preámbulo Imports. Los elementos también pueden ser añadidos al código simplemente copiando y pegando. Se puede eliminar los elementos importados usando un boton de basura que se encuentra en el margen izquierdo cuando se pasa a lado del elemento insertado.



copiar elementos

Para añadirlo de forma directa al código, se hace a través de insertar el nombre del elemento y declararlo como el objeto que es. En el caso de una imagen se utiliza ee.Image(), para una tabla se hace con ee.FeatureCollection(), para el caso de un objeto geométrico dentro de un archivo .shp se hace como si fuese un tabla con ee.FeatureCollection(), y si es un conjunto de imágenes se declara con ee.ImageCollection(). El código a continuación se muestra como importamos el raster con la agregación de las vialidades extraidas de OpenStreet map ([OSM](https://www.openstreetmap.org/)) y subida a GEE en el elemento “raster\_agregados\_vialidades” el cual se tendra que añadir dentro de los assets.

var image = ee.Image("raster\_agregados\_vialidades");

## Explicación del Modelo

Nuestro modelo, es una modificación al modelo utilizado por Just, haciendo uso de una fuente de información distinta de los datos meteorológicos para hacer las estimaciones, la fuente de datos a utilizar son los datos de [reánalisis meteorológicos](https://climate.copernicus.eu/climate-reanalysis) del [“European Centre for Medium-Range Weather Forecasts”](https://www.ecmwf.int/en/about) los cuales se encuentran disponibles dentro de GEE en el [catálogo de datos](https://developers.google.com/earth-engine/datasets/catalog/ECMWF_ERA5_LAND_HOURLY). Para importar estos datos lo hacemos a traves de el siguiente código y al igual que con la colección de MODIS la filtramos por fecha, seleccionamos las bandas que vamos a utilizar y las partes de las imágenes que se encuentren dentro del “bounding box” del polígono de la zona metropolitana el cual se encuentra en la variable ZMVM\_bb.

var land\_eras\_day\_f = ee.ImageCollection("ECMWF/ERA5\_LAND/HOURLY").filter(  
 ee.Filter.date(startdate, enddate)  
 ).select([  
 'dewpoint\_temperature\_2m',  
 'total\_evaporation',  
 'temperature\_2m',  
 'total\_precipitation\_hourly',  
 'total\_precipitation'  
 ]).filterBounds(ZMVM\_bb);

Para obtener el bounding box del polígono de la Zona Metropolitana del Valle de México (ZMVM) se hace de la siguiente forma:

var ZMVM\_tabla = ee.FeatureCollection("Poligono\_archivo\_shape\_ZMVM");  
var ZMVM\_bb = ZMVM\_tabla.geometry().bounds();

donde “Poligono\_archivo\_shape\_ZMVM” es un elemento que se deberá añadir en los assets que contenga al polígono de la ZMVM. El polígono se encuentrá dentro de un archivo .shp, para poder hacer uso de este elemento geométrico se hace usando .geometry() el cual extrae lo que se encuentra dentro del campo geometry de una tabla, en nuestro caso el polígono de la ZMVM. Después tomamos el cuadrado delimitador o “bounding box” usando .bounds() el cual es un método de los objetos geométricos en GEE.

De este conjunto de datos seleccionamos los que se encuentran en determinadas fechas y filtramos los elementos las imágenes que se intersecten con ZMVM\_bb.

## Todos igual

Por simpleza vamos a remuestrar todos los datos para que estos esten similar a los datos de MODIS.

Para tal proposito definimos una función,

var modisProj = AOD\_MODIS.first().projection();  
   
var resample\_coll = function(image){  
 return image.reproject(modisProj);  
};

# Entrenamiento fuera de GEE.

El modelo fue entrenado con los datos de forma anual y para cada día se obtuvieron los valores de los coeficientes para el modelo, de forma similar a la presentada en [Equation 2](#eq-compleja).

donde son coeficientes que se determinan en el modelo todo el año. Mientras que y se determinan de forma diaria.

## Coeficientes

Por simplicidad, los coefficientes se ponen dentro de una tabla, la cual es añadida a través de los **Assets**. La cual tiene la siguiente forma:

| Intercept | Optical\_Depth\_047\_z | temperature\_2m\_z | relative\_humidity\_z | total\_precipitation\_sqrt\_z | PBLH\_z | vialidades\_z | date |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 3.9526 | 0.4266 | 0.2033 | -0.3986 | -0.1151 | 0.0128 | 2004-01-01 | 1072915200 |
| 1.1116 | 0.2307 | 0.2033 | -0.3986 | -0.1151 | 0.0128 | 2004-01-02 | 1073001600 |

sólo mostramos los primeros elementos pero la tabla deberá contener todos los días en donde es posible obtener el modelo.

|  |
| --- |
| Note |
| Esta tabla se sube usando la pestaña **Assets**, se resalta en particular la columna ‘system:index’, esta es la fecha en [‘Unix time’](https://en.wikipedia.org/wiki/Unix_time), es importante pues dentro de GEE nos sirve para poder usar el filtro de fechas (.filterDate) si el dato no se encuentra en este formato los filtros especificos de fechas no se pueden utilizar. |

Las variables a usar dentro de nuestro modelo en [Equation 3](#eq-modelo) son es el espesor óptico de MODIS para el día en la celda , es la precipitación y la temperatura a metros para el día en la celda las cuales se obtienen de la coleccion de reánalisis meteorológicos. es la altura de la capa límite de mezcla. Y la densidad de calles en la celda .

Todas la variables anteriores las podemos extraer directamente de las colecciones de imágenes que se encuentran en los datos de GEE o bien colecciones que añadimos nosotros como es el caso de , pues esta no se encuentra directamente en GEE, es necesario descargar y añadir como colección de imágenes a GEE.

var PBLH\_ZMVT = ee.ImageCollection("PBLH\_data")

## Calcular para toda la colección

Nos falta la variable la cual es la humedad relativa en el día en la celda , pero esta puede ser obtenida usando la temperatura y el punto de rocío.

Donde es la temperatura y es el punto de rocío. Podemos obtener un raster de la humedad relativa aplicando usando las bandas que seleccionamos de la colección de imágenes de los datos de reanálisis con la siguiente función:

var relat\_humed = function(image){  
 /\*  
 actual\_water\_press = 611.21 \* math.exp((17.502 \* td) / (240.97 + td))  
 max\_water\_press = 611.21 \* math.exp((17.502 \* t) / (240.97 + t))  
   
 HR=(actual\_water\_press/ max\_water\_press)\*100  
 \*/  
   
 var lambda\_i = 243.04;  
 var beta\_i = 17.625; ///17.1; //17.625;  
   
 var dew\_point = image.select('dewpoint\_temperature\_2m');  
 var temp\_2m = image.select('temperature\_2m');  
 var prec\_total = image.select('total\_precipitation');  
 var e\_up = dew\_point  
 .multiply(beta\_i);  
 var e\_dow = dew\_point  
 .add(lambda\_i);  
 var e = ((e\_up.divide(e\_dow)).exp());  
 var e\_s\_up = temp\_2m  
 .multiply(beta\_i);  
 var e\_s\_dow = temp\_2m  
 .add(lambda\_i);  
 var e\_s = ((e\_s\_up.divide(e\_s\_dow)).exp());  
 var rel\_hum = (e.divide(e\_s)).multiply(100);  
   
 rel\_hum = rel\_hum .select(['dewpoint\_temperature\_2m']).rename(['relative\_humidity']);  
 rel\_hum = rel\_hum  
 .addBands(temp\_2m)  
 .addBands(dew\_point)  
 .addBands(prec\_total)  
 .set({'system:time\_start':image.get('system:time\_start')})  
 ;  
return rel\_hum;  
};

* Definimos una función para calcular la humedad relativa basada en el punto de rocío y la temperatura.
* La función calcula la presión de vapor y la saturación de vapor usando las formulas respectivas.
* Finalmente se calcula la humedad relativa y se añade una nueva banda (relative\_humidity) a la imagen.

Para aplicar la función a las imágenes diarias se “mapea” la función a todas los elementos de la colección.

land\_eras\_day\_f = land\_eras\_day\_f.map(relat\_humed );

## Escalas

Una problemática que nos podemos encontrar es que en algunas ocasiones al ajustar el modelo en **R** puede ser necesario escalar las variables usando **z-scores** para que el modelo converja. Si es necesario hacer esto entonces es necesario saber como se escalaron las variables y poder subir estos valores dentro de GEE usando una tabla. Si se escala con la función scale en **R** entonces nos interesan los valores “scale” y “center” donde “scale” es la desviación estandar y “center” es el promedio de los datos.

es el promedio de los valores de la variable , es la desviación estandar de los valores de la variable y es la variable escalada.

Si fue necesario escalar las variables para obtener la convergencia del modelo entonces podemos escalar los valores dentre de GEE si tenemos y . Por simpleza subimos una tabla la cual contiene dichos valores de forma anual.

| Variable | center | scale | year |
| --- | --- | --- | --- |
| PM25\_z | 29.48 | 10.95 | 2004 |
| Optical\_Depth\_047\_z | 249.99 | 149.17 | 2004 |
| temperature\_2m\_z | 292.50 | 5.26 | 2004 |
| relative\_humidity\_z | 90.55 | 3.76 | 2004 |
| total\_precipitation\_sqrt\_z | 0.02 | 0.01 | 2004 |
| PBLH\_z | 450.12 | 176.57 | 2004 |
| vialidades\_z | 1421.67 | 1855.65 | 2004 |
| PM25\_z | 30.32 | 13.84 | 2005 |
| Optical\_Depth\_047\_z | 266.90 | 151.62 | 2005 |
| temperature\_2m\_z | 293.15 | 5.34 | 2005 |
| relative\_humidity\_z | 88.98 | 5.56 | 2005 |
| total\_precipitation\_sqrt\_z | 0.02 | 0.02 | 2005 |
| PBLH\_z | 496.19 | 216.17 | 2005 |
| vialidades\_z | 1421.67 | 1855.65 | 2005 |

Para seleccionar los valores para poder escalar los datos seleccionamos los datos para un año en especifico de la siguiente forma:

var selected\_day = '2018-04-24';  
selected\_day = ee.Date(selected\_day);  
  
var scale\_values = ee.FeatureCollection("your\_scale\_values\_table");  
var all\_scale\_day = scale\_values.filter(ee.Filter.eq('year', selected\_day.get('year')));  
var optical\_scale\_dic = ee.Dictionary({  
 'center': all\_scale\_day.filter(ee.Filter.eq('1', 'Optical\_Depth\_047\_z')).first().get('center'),  
 'scale':all\_scale\_day.filter(ee.Filter.eq('1', 'Optical\_Depth\_047\_z')).first().get('scale')  
  
});  
var PBLH\_scale\_dic = ee.Dictionary({  
 'center': all\_scale\_day.filter(ee.Filter.eq('1', 'PBLH\_z')).first().get('center'),  
 'scale':all\_scale\_day.filter(ee.Filter.eq('1', 'PBLH\_z')).first().get('scale')  
});  
var PM25\_scale\_dic = ee.Dictionary({  
 'center': all\_scale\_day.filter(ee.Filter.eq('1', 'PM10\_z')).first().get('center'),  
 'scale':all\_scale\_day.filter(ee.Filter.eq('1', 'PM10\_z')).first().get('scale')  
});  
var precipitation\_scale\_dic = ee.Dictionary({  
 'center': all\_scale\_day.filter(ee.Filter.eq('1', 'total\_precipitation\_sqrt\_z')).first().get('center'),  
 'scale':all\_scale\_day.filter(ee.Filter.eq('1', 'total\_precipitation\_sqrt\_z')).first().get('scale')  
});  
var RH\_scale\_dic = ee.Dictionary({  
 'center': all\_scale\_day.filter(ee.Filter.eq('1', 'relative\_humidity\_z')).first().get('center'),  
 'scale':all\_scale\_day.filter(ee.Filter.eq('1', 'relative\_humidity\_z')).first().get('scale')  
});  
var Temp\_scale\_dic = ee.Dictionary({  
 'center': all\_scale\_day.filter(ee.Filter.eq('1', 'temperature\_2m\_z')).first().get('center'),  
 'scale':all\_scale\_day.filter(ee.Filter.eq('1', 'temperature\_2m\_z')).first().get('scale')  
});  
var vialidades\_scale\_dic = ee.Dictionary({  
 'center': all\_scale\_day.filter(ee.Filter.eq('1', 'vialidades\_z')).first().get('center'),  
 'scale':all\_scale\_day.filter(ee.Filter.eq('1', 'vialidades\_z')).first().get('scale')  
});

Para cada una de las variables tenemos un diccionario con valores center y scale para poder escalar los datos en GEE. Se hace una función especifica para las variable obtenidos de reanálisis y para los demás se hace de forma individual.

var z\_scale\_land = function(image){  
 var temp\_2m = image.select('temperature\_2m');  
 temp\_2m= temp\_2m.expression(  
 '(x - mean\_x) / sd\_x', {  
 'x':temp\_2m ,  
 'mean\_x': Temp\_scale\_dic.getNumber('center'),  
 'sd\_x': Temp\_scale\_dic.getNumber('scale')  
 }  
 ).rename('temperature\_2m\_z');  
   
 var rh = image.select('relative\_humidity');  
 rh= rh.expression(  
 '(x - mean\_x) / sd\_x', {  
 'x':rh ,  
 'mean\_x': RH\_scale\_dic.getNumber('center'),  
 'sd\_x': RH\_scale\_dic.getNumber('scale')  
 }  
 ).rename('relative\_humidity\_z') ;  
   
 var total\_prec = image.select('total\_precipitation');  
 total\_prec= total\_prec.expression(  
 '(x - mean\_x) / sd\_x', {  
 'x':total\_prec ,  
 'mean\_x': precipitation\_scale\_dic.getNumber('center'),  
 'sd\_x': precipitation\_scale\_dic.getNumber('scale')  
 }  
 ).rename('total\_precipitation\_sqrt\_z') ;  
 return temp\_2m  
 .addBands(rh)  
 .addBands(total\_prec)  
 .set({'system:time\_start':image.get('system:time\_start')});  
};  
  
  
var optical\_scale = optical\_original.expression(  
 '(x - mean\_x) / sd\_x', {  
 'x': optical\_original,  
 'mean\_x': optical\_scale\_dic.getNumber('center'),  
 'sd\_x': optical\_scale\_dic.getNumber('scale')  
 }  
 ).rename('Optical\_Depth\_047\_z');

|  |
| --- |
| Note |
| Cabe destacar el uso de [ee.Image.expression()](https://developers.google.com/earth-engine/apidocs/ee-image-expression) este método nos permite evaluar los valores en las imágenes como si fuesen fórmulas usando un string. |

Una vez escalados los valores se renombran las bandas de las imágenes, con la intención de tener un nombre que identifique que estas bandas se encuentran escaladas.

### Unir todo

Ahora vamos a hace uso de las bandas escaladas para poder obtener el resultado de tomando la combinación lineal de nuestro modelo. Aunque se puede hacer las operaciones de forma compacta en código, hacemos éstas en lineas separadas de código para que al leerlo quede de forma legible y entender los distintos pasos realizados.

* Multiplicamos cada banda por el respectivo coeficiente.
* Generamos una imagen que tenga el valor constante XIntercept
* Hacemos la suma de las distintas imágenes en una sola y la renombramos.
* Se reescalan los valores de .

///Multiplicar bandas coeficientes   
PBLH\_day\_s = PBLH\_day\_s.multiply(coeff\_day.getNumber('PBLH\_z'));   
optical\_scale = optical\_scale.multiply(coeff\_day.getNumber('Optical\_Depth\_047\_z'));  
var hum\_image = day\_land\_eras\_image\_s.select('relative\_humidity\_z').multiply(coeff\_day.getNumber('relative\_humidity\_z'));  
var tem\_image = day\_land\_eras\_image\_s.select('temperature\_2m\_z').multiply(coeff\_day.getNumber('temperature\_2m\_z'));  
var prec\_image = day\_land\_eras\_image\_s.select('total\_precipitation\_sqrt\_z').multiply(coeff\_day.getNumber('total\_precipitation\_sqrt\_z'));  
vialidades\_s = vialidades\_s.multiply(coeff\_day.getNumber('vialidades\_z'));  
var image\_interc = ee.Image(coeff\_day.getNumber('XIntercept')).clip(ZMVM\_bb);  
  
//// Suma   
var day\_pm= ee.Image().expression(  
 'opt + pblh + viali + rh + temp + prec\_sqr + Intercept',  
 {  
 'opt' :optical\_scale,   
 'pblh' :PBLH\_day\_s,   
 'viali' :vialidades\_s,   
 'Intercept': image\_interc,  
 'rh': hum\_image,  
 'temp':tem\_image,  
 'prec\_sqr':prec\_image  
   
 }  
 ).rename('PM\_scale');  
/// Rescalar a valores  
var day\_pm\_re = ee.Image().expression(  
 'y\*sd\_x + mean\_x',  
 {  
 'y': day\_pm,  
 'mean\_x': PM25\_scale\_dic.getNumber('center'),  
 'sd\_x': PM25\_scale\_dic.getNumber('scale')  
 }  
);

En la última imagen se encuentran los valores calculados de y esta es la imagen que vamos a desplegar en un mapa.

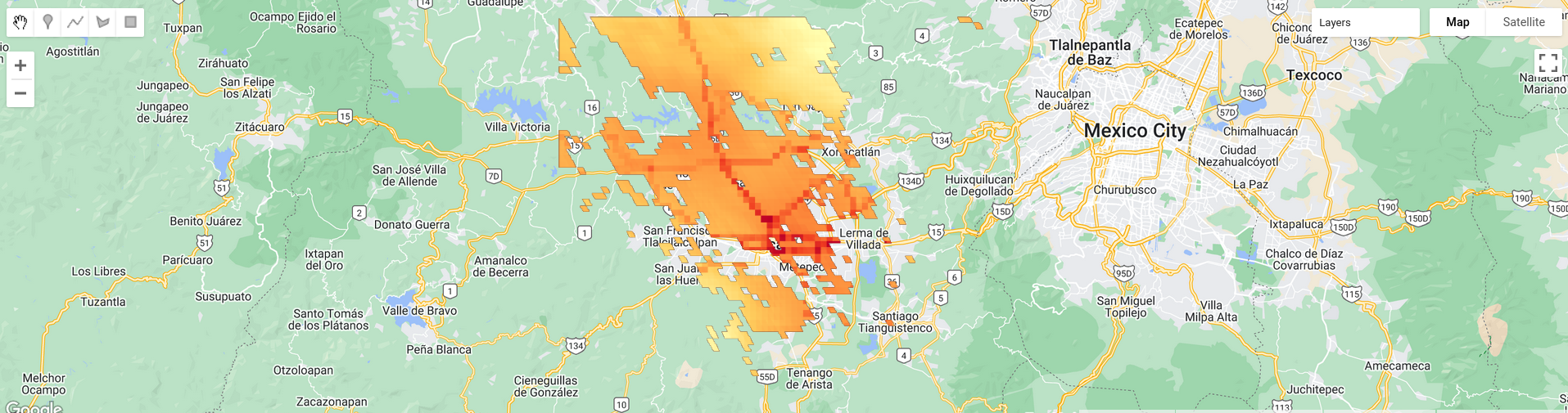
## Despliegue en mapa

Usamos la imagen y la añadimos al mapa, como los valores minimos y maximos van a ser de forma diaria para poder hacer una visualización buscamos los mínimos y máximos para ese día usando el método [ee.Image.reduceRegion()](https://developers.google.com/earth-engine/apidocs/ee-image-reduceregion) la cual nos permite hacer uso de la [operación reduce](https://en.wikipedia.org/wiki/Many-one_reduction) y [ee.Reducer.minMax()](https://developers.google.com/earth-engine/apidocs/ee-reducer-minmax) para obtener el mïnimo y el máximo.

Una vez añadida la capa al mapa centramos el mapa al centro de la zona metropolitana.

var amm\_roj = ['#ffffb2','#fecc5c','#fd8d3c','#f03b20','#bd0026'];  
var min\_max = day\_pm\_re.reduceRegion(ee.Reducer.minMax(),ZMVM\_bb)  
print(min\_max)  
var max\_pmval = min\_max.get('PM\_scale\_max').getInfo();  
var min\_pmval = min\_max.get('PM\_scale\_min').getInfo();  
Map.addLayer(  
 day\_pm\_re,  
 {   
 'bands':'PM\_scale',  
 min : min\_pmval,  
 max :max\_pmval,  
 palette: amm\_roj  
 }, 'PM 2.5'  
);  
Map.centerObject(ZMVM\_bb, 10);

Al final podemos observar como queda la estimación generada por el modelo.



Resultado Final

## Comentario final

Este tutorial donde mostramos que podemos hacer uso de distintas herramientas e integrarlas dentro de GEE.