ISSN 2618-4915

Contribuciones Técnicas
SIG e IDE N°44

FUNCIONAMIENTO DEL SCRIPT PARA CARTOGRAFÍA GEOQUÍMICA DE BAJA DENSIDAD

Autores

Buceta, Maria Guadalupe; González, Ana Carolina; Alvarez, Dolores; Candiani, José Nicolás y Ferpozzi, Federico Javier.

Diciembre 2023

INSTITUTO DE GEOLOGÍA Y RECURSOS MINERALES

SegemAR
Servicio Geológico Minero Argentino

FUNCIONAMIENTO DEL SCRIPT PARA CARTOGRAFÍA GEOQUÍMICA DE BAJA DENSIDAD

Autores

Buceta, Maria Guadalupe

González, Ana Carolina

Alvarez, Dolores

Candiani, José Nicolás

Ferpozzi, Federico Javier.

Dirección de Geomática - Instituto de Geología y Recursos Minerales - SEGEMAR

Av. General Paz 5445 (Colectora provincia) Edificio 25 | 1650 San Martín – Buenos Aires República Argentina

(11) 5670-0100

REFERENCIA BIBLIOGRÁFICA

Esta publicación debe citarse como:

Buceta, M. G., González, A. C., Alvarez, D., Candiani, J. N., Ferpozzi, F. 2023. Funcionamiento del script para cartografía geoquímica de baja densidad. Instituto de Geología y Recursos Minerales, Servicio Geológico Minero Argentino. Serie Contribuciones Técnicas SIG e IDE N° 44, 27p. Buenos Aires

PALABRAS CLAVE: script, arcpy, SIGAM, geoquímica, cartografía

ISSN 2618-4915

Contenido

Introducción

El Sistema de Información Geológica Ambiental Minera (SIGAM) ha sido diseñado y desarrollado para fortalecer la gestión de la información en los niveles técnico, gubernamental y público. El SIGAM consta de una infraestructura de datos espaciales (IDE), un SIG institucional y un sistema de producción cartográfica para la gestión de datos e información geológico-ambiental-minera, abarcando la incorporación, administración, evaluación, accesibilidad y disponibilidad a través de internet de la información existente (Servicio Geológico Minero Argentino, s.f.). En el marco del continuo desarrollo evolutivo del SIGAM, se desarrolló una nueva herramienta de salidas cartográficas para automatizar la realización de cartografía (en formato analógico y digital) en el marco del proyecto "Mapeo geoquímico regional de baja densidad de la Región Mesopotámica. Atlas Geoquímico de la Provincia de Entre Ríos, República Argentina" enmarcado en el Acuerdo de cooperación internacional trianual firmado entre el Servicio Geológico Minero Argentino (SEGEMAR) de la República Argentina y el China Geological Survey (CGS) de la República Popular China.

Objetivo del informe

El objetivo de este informe es caracterizar los aspectos relativos al diseño del módulo de automatización de salidas cartográficas para el atlas de geoquímica de baja densidad, a título de herramienta de consulta y guía para el uso y aplicación de dicha herramienta.

Funcionalidades de la herramienta

El propósito general de la herramienta es generar salidas gráficas para datos geoquímicos a partir de una base de datos GIS con datos de análisis de geoquímica, automatizando la producción cartográfica de los mapas de los diferentes elementos químicos con el objetivo de generar un producto final homogéneo en apariencia y calidad, dado que dichos mapas componen el Atlas geoquímico de baja densidad de cada provincia mesopotámica: Entre Ríos, Misiones y Corrientes. Para ello se parte de una plantilla original con normas de composición del mapa completo y criterios de edición de cada uno de los elementos que lo componen que permiten mejorar los procesos de edición y el control de calidad.

El código genera 142 archivos únicos, compuestos por un archivo de documento de mapa con extensión .mxd del software ESRI ArcMap y la visualización digital del mismo con extensión .pdf (Portable Document Format) basados en información geoquímica para cada elemento químico de un conjunto de muestras.

Requisitos

El script funciona a través de ArcPy, un paquete en lenguaje python 2.7 utilizado en el software ArcGIS de Esri para automatizar funciones de sistemas de información geográfica (GIS) y como herramienta dentro del módulo ArcToolBox del software ArcMap 10.2.2, por lo cual todos los paquetes y software mencionados son necesarios para su operación.

Además para ejecutarse requiere que los datos geoquímicos estén estructurados y estén definidos los criterios de representación

Mapa plantilla

El código se ejecuta sobre un mapa plantilla, generado como archivo de documento de mapa con extensión .mxd con un maquetado a la escala cartográfica correspondiente para un tamaño de hoja A3, que se replicará para cada elemento químico.

Diseño del mapa plantilla

El mapa plantilla es un archivo con extensión .mxd con el nombre de la provincia y sufijo *_geoquiBD* ubicado en la carpeta de mapas provinciales del disco *data_sigam.*

Dataframe principal

En el Dataframe principal se ubican los datos georeferenciales. Está proyectado en POSGAR07 según la faja que corresponda a la ubicación donde se disponen los siguentes elementos geográficos con su respectiva simbología y en el siguiente orden de visualización:

- Grupo de capas de nombre *MuestraGeoquimicaGroup*
- Topografía
    - Límites internacionales y provinciales con geometría de línea
    - Centros poblados con geometría de puntos y sus respectivas anotaciones
    - Red vial nacional y provincial con geometría de línea y sus respectivas anotaciones
    - Red fluvial con geometría de línea y sus respectivas anotaciones
    - Red fluvial de doble margen con geometría de polígono y sus respectivas anotaciones
    - Puentes con geometría de puntos y sus respectivas anotaciones
    - Provincia con geometría de polígono

- Grupo de capas de nombre *GrillaGeoquimicaGroup*
- Modelo digital de elevación como relieve sombreado

Sección carátula

En la carátula se ubican el título y descripción del proyecto, la descripción de la ubicación, los logos oficiales de los organismos que participaron y un elemento de tipo texto con nombre *S2aTxtTitulo* que el script modificará con el nombre de cada elemento químico.

Sección referencias

En las referencias se encuentran una rampa con la simbología de color proveniente del grillado y un elemento de tipo leyenda con nombre *S2bLegend* que el script modificará con los valores correspondientes a cada percentil y su simbología puntual.

Sección escala

En la sección de escala se ubicará la escala gráfica y de texto, la proyección con su meridiano central y el marco de referencia.

Datos

Los requisitos del dato geoquímico deberán cumplir con la Normativa para la Carta Geoquímica de la República Argentina (SEGEMAR, 2001) y del Diseño del Módulo de Salidas Gráficas de la Carta Geoquímica (Marquinez García et al, 2018*a*) y para ejecutar el código deberán estar almacenados dentro del entorno *sigam*, organizados de acuerdo al modelo de datos para el FeatureClass *GeoquimicaBD,* con el número de decimales definidos para cada elemento químico y con todo valor bajo el límite de detección como el valor absoluto del límite de detección y negativo. No podrá haber datos iguales a cero.

Simbología

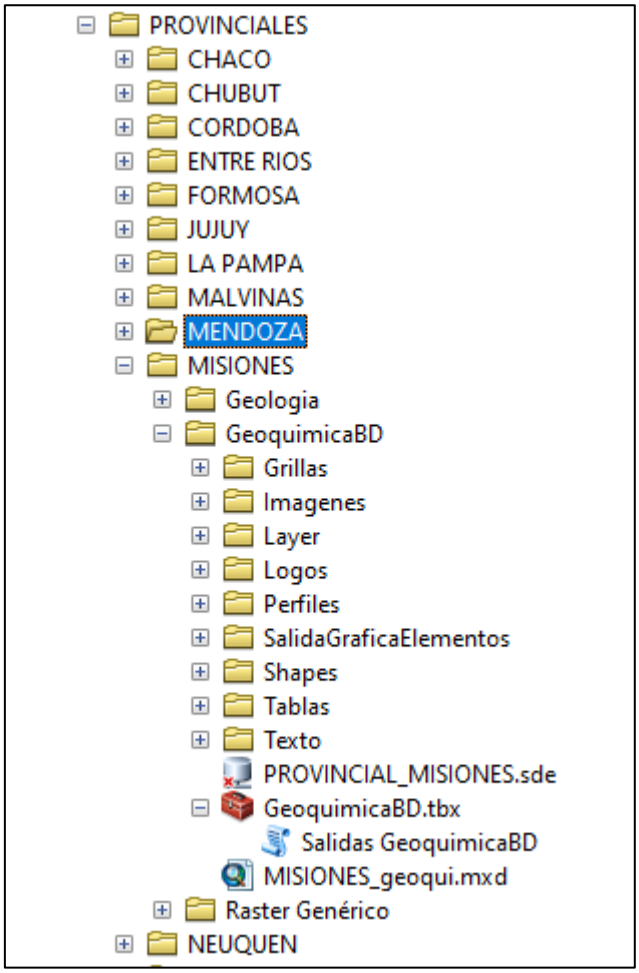
La representación del dato geoquímico agrupado como percentiles requiere capas puntuales con la simbología definida por la Normativa para la Carta Geoquímica de la República Argentina (SEGEMAR, 2001 y Marquinez García, 2018*a*) en formato .lyr ubicadas en una carpeta accesible

Grillas

El dato grillado provisto deberá estar con geometría de polígonos en formato .lyr y ubicado en una carpeta accesible cuya ruta se declara por medio de los parámetros, y se requiere además una carpeta vacía donde se guardarán los archivos generados.

Ejecución

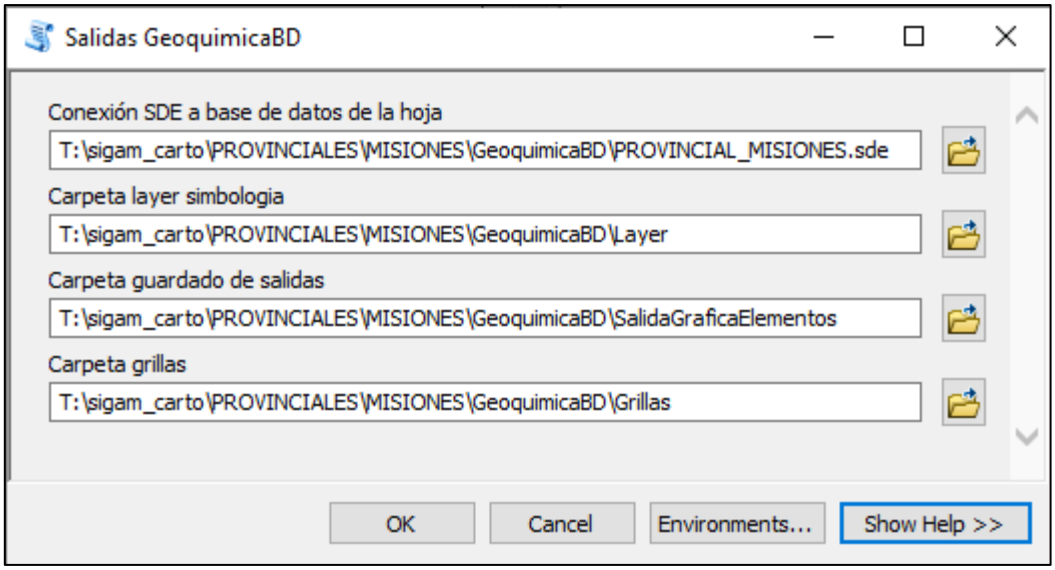
Para ejecutar el programa hay que abrir el archivo plantilla y desde el módulo de ArcCatalog, en la carpeta del proyecto se accede a una caja de herramientas ArcToolBox titulada *GeoquímicaBD* que posee una herramienta de nombre *Salidas Geoquimica BD.*



Carpeta del proyecto con la caja de herramientas ArcToolBox GeoquimicaBD.tbx *y la herramienta* Salidas GeoquimicaBD

Parámetros:

Al hacer doble click sobre la herramienta *Salidas Geoquimica BD* se visualiza un menú que solicita parámetros para la ejecución. al completar dichos parámetros se ejecuta el programa y se obtienen las salidas gráficas, que se guardarán en la carpeta ingresada.



Parámetros de la herramienta la herramienta Salidas GeoquimicaBD

Conexión SDE a la base de datos de la hoja

Parámetro con tipo de dato Workspace, donde se ingresa una conexión a una geodatabase corporativa. Esta geodatabase contiene los datos georeferenciados de los análisis de geoquímica dentro del FeatureClass *GeoquimicaBD.*

Carpeta layer simbología

Parámetro con tipo de dato Disk Connection, donde se ingresa la ruta a la carpeta donde se encuentran ocho layers con la simbología puntual correspondiente a cada uno de los percentiles.

Carpeta guardado de salidas

Parámetro con tipo de dato Disk Connection, donde se ingresa la ruta a la carpeta donde se guardarán los archivos generados.

Carpeta grillas

Parámetro con tipo de dato Disk Connection, donde se ingresa la ruta a la carpeta donde se encuentran los layers de los grillados de cada elemento, con su nombre normalizado a *elemento_unidad__contour_region.lyr*

Código

A continuación se describen los pasos de ejecución del código

```
# -*- coding: iso-8859-1 -*-
# SCRIPT SALIDA GRAFICA GEOQUIMICA BAJA DENSIDAD - SEGEMAR 2023
```

Define la codificación del texto como el estándar iso-8859-1.

Módulos y paquetes

Importa todos los módulos y paquetes necesarios para ejecutarse:

```
import arcpy
from arcpy import env
from arcpy import mapping
```

`import arcpy`: Importa el paquete ArcPy, el cual proporciona acceso a la funcionalidad de geoprocesamiento en el software ArcMap de Esri para gestionar y analizar datos SIG.

`from arcpy import env`: Importa el módulo **`env`** desde el paquete ArcPy. **`env`** proporciona funciones para gestionar los parámetros de entorno y ajustes del espacio de trabajo dentro de ArcMap.

`from arcpy import mapping`: Importa el módulo **`mapping`** desde ArcPy. **`mapping`** permite trabajar con documentos de mapas (archivos con extensión .mxd) y capas dentro de ArcMap.

```
import os
import numpy
```

`import os`: Importa el módulo **`os`** de Python, mediante el cual se manipulan los archivos y directorios del sistema operativo.

`import numpy`: Importa la librería **`numpy`** de Python, utilizada para cálculos numéricos y funciones matemáticas para manipular los datos geoquímicos.

```
arcpy.ClearWorkspaceCache_management()
```

La función **`arcpy.ClearWorkspaceCache_management()`** limpiar el cache, dado que los datos previamente almacenados pueden generar errores al ejecutarse código.

Parámetros

```
sdeHOJA = arcpy.GetParameterAsText(0)
pathLayers = arcpy.GetParameterAsText(1)
pathEspacioTrabajo = arcpy.GetParameterAsText(2)
pathGrilla = arcpy.GetParameterAsText(3)
```

Incorpora al código los parámetros ingresados a traves del ArcToolBox

Datos y espacio de trabajo

```
arcpy.env.workspace = sdeHOJA
ds = str(str(arcpy.ListDatasets(u'*Geoquimica*')).split("'")[1])
fcMU =
str(str(arcpy.ListFeatureClasses(u'*BD',feature_dataset=ds)).split("'"
)[1])
```

La propiedad **`arcpy.env.workspace`** establece el espacio de trabajo (ubicación donde se encuentran los datos) para las operaciones de geoprocesamiento dentro de ArcGIS.

Al establecer esta propiedad de entorno se simplifica el código, ya que no requiere especificar la ruta completa para cada conjunto de datos utilizado en las operaciones de geoprocesamiento y permite ejecutar las siguientes funciones:

`arcpy.ListDatasets()`: obtiene los nombres de todos los datasets dentro de una base de datos y guarda como variable `ds` al FeatureDataset *Geoquimica*.

`arcpy.ListFeatureClass()`: obtiene los nombres de todos los FeatureClass dentro de una base de datos y guarda como variable `fcMU` al FeatureClass *MuestraGeoquimicaBD*, donde se encuentran los datos georeferenciados de los análisis de geoquímica a utilizar.

Propiedades del mapa

```
prntMxd =arcpy.mapping.MapDocument("CURRENT")
RefPrincipal =
arcpy.mapping.ListLayoutElements(prntMxd,"LEGEND_ELEMENT")[0]
dfMapaPrincipal = arcpy.mapping.ListLayoutElements(prntMxd,
"DATAFRAME_ELEMENT")[0]
S2aTxtNomHoja =
arcpy.mapping.ListLayoutElements(prntMxd,"TEXT_ELEMENT","S2aTxtNomHoja
")[0]
S2aTxtTitulo =
arcpy.mapping.ListLayoutElements(prntMxd,"TEXT_ELEMENT","S2aTxtTitulo"
)[0]
targetGroupLayer = arcpy.mapping.ListLayers(prntMxd,
"MuestraGeoquimicaGroup", dfMapaPrincipal)[0]
```

La propiedad **`arcpy.mapping.MapDocument()`**: permite interactuar con el contenido del documento de mapa .mxd. Al establecerlo en "CURRENT" permite agregar, remover y modificar nombres, *definition querys* y descripciones del documento en el cual se está ejecutando el script.

Esta propiedad permite ejecutar las funciones:

`arcpy.mapping.ListLayoutElements()`: lista los elementos del maquetado dentro del proyecto. En este caso guarda como variables:

- `RefPrincipal` al elemento de tipo leyenda de nombre *S2bLegend* para modificarlo posteriormente según varíen los percentiles.
- `dfMapaPrincipal` al dataframe proyectado donde se ubican las capas georeferenciadas del proyecto.
- `S2aTxtTitulo` al elemento de texto con el nombre del elemento químico.
- `S2aTxtNomHoja` al elemento de texto que representa el título de cada mapa.

`arcpy.mapping.ListLayers()`: lista las capas existentes en el mapa y guarda como variables:

- `targetGroupLayer` al elemento de tipo grupo donde se ubicarán las capas generadas con los percentiles.
- `targetGrillaLayer` al elemento de tipo grupo donde se ubicarán las capas correspondientes al grillado.

Todas estas variables generadas serán modificadas posteriormente por el código en cada uno de los mapas para que coincidan con el elemento químico representado.

Propiedades del mapa

```
RefPrincipal.autoAdd = True
for i in ["8","7","6","5","4","3","2","1"]:
    capaTempo = arcpy.mapping.Layer(sdeHOJA + "\\" + ds +  "\\" +
fcMU)
    capaTempo.name = "MuestraGeoquimica_class"+i

    arcpy.mapping.UpdateLayer(dfMapaPrincipal,capaTempo,
arcpy.mapping.Layer(pathLayers + u"\\MuestraGeoquimica_class"
+i+".lyr"),True)
    arcpy.mapping.AddLayerToGroup(dfMapaPrincipal,
targetGroupLayer,capaTempo, "TOP")
RefPrincipal.autoAdd = False
ly_muestraGQ_class1 =
arcpy.mapping.ListLayers(prntMxd,"MuestraGeoquimica_class1",dfMapaPrin
cipal)[0]
ly_muestraGQ_class2 =
arcpy.mapping.ListLayers(prntMxd,"MuestraGeoquimica_class2",dfMapaPrin
cipal)[0]
ly_muestraGQ_class3 =
arcpy.mapping.ListLayers(prntMxd,"MuestraGeoquimica_class3",dfMapaPrin
cipal)[0]
ly_muestraGQ_class4 =
arcpy.mapping.ListLayers(prntMxd,"MuestraGeoquimica_class4",dfMapaPrin
cipal)[0]
ly_muestraGQ_class5 =
arcpy.mapping.ListLayers(prntMxd,"MuestraGeoquimica_class5",dfMapaPrin
cipal)[0]
ly_muestraGQ_class6 =
arcpy.mapping.ListLayers(prntMxd,"MuestraGeoquimica_class6",dfMapaPrin
cipal)[0]
ly_muestraGQ_class7 =
arcpy.mapping.ListLayers(prntMxd,"MuestraGeoquimica_class7",dfMapaPrin
cipal)[0]
ly_muestraGQ_class8 =
arcpy.mapping.ListLayers(prntMxd,"MuestraGeoquimica_class8",dfMapaPrin
cipal)[0]
```

La función **autoAdd** activa y desactiva la incorporación automática de capas al mapa principal `RefPrincipal`. Mediante un bucle de la función **arcpy.mapping.AddLayerToGroup**

agrego al grupo `targetGroupLayer` las ocho capas de simbología ubicadas en la carpeta de simbología previamente definida en los parámetros.

Actualizo mediante `arcpy.mapping.UpdateLayer` la fuente de los datos para que represente las datos del FeatureClass `fcMU` y defino 8 variables que representen a cada una de las capas incorporadas al mapa en el paso anterior, lo que permite hacer operaciones con las msimas.

Tuplas de elementos

```
lst_elementoGeoqui_2 = (
                        ['Ag', 'ppm', 'Plata',3],
                        ['Al', 'porciento', 'Aluminio',2],
                        ['As', 'ppm', 'Ars\xe9nico',2],
                        ['Au', 'ppb', 'Oro',2],
                        ['Ba', 'ppm', 'Bario',0],
                        ['Be', 'ppm', 'Berilio',2],
                        ['Bi', 'ppm', 'Bismuto',3],
                        ['Br', 'ppm', 'Bromo',2],
                        ['C', 'porciento', 'Carbono',2],
                        ['Ca', 'porciento', 'Calcio',2],
                        ['Cd', 'ppm', 'Cadmio',3],
                        ['Ce', 'ppm', 'Cerio',1],
                        ['Co', 'ppm', 'Cobalto',2],
                        ['Cr', 'ppm', 'Cromo',0],
                        ['Cs', 'ppm', 'Cesio',2],
                        ['Cu', 'ppm', 'Cobre',1],
                        ['Er', 'ppm', 'Erbio',2],
                        ['Eu', 'ppm', 'Europio',3],
                        ['Fe', 'porciento', 'Hierro',2],
                        ['Ga', 'ppm', 'Galio',1],
                        ['Ge', 'ppm', 'Germanio',2],
                        ['Hf', 'ppm', 'Hafnio',2],
                        ['Hg', 'ppm', 'Mercurio',4],
                        ['In', 'ppm', 'Indio',3],
                        ['Ir', 'ppb', 'Iridio'],
                        ['K', 'porciento', 'Potasio',2],
                        ['La', 'ppm', 'Lantano',2],
                        ['Li', 'ppm', 'Litio',2],
                        ['Lu', 'ppm', 'Lutecio',3],
                        ['Mg', 'porciento', 'Magnesio',2],
                        ['Mn', 'ppm', 'Manganeso',0],
                        ['Mo', 'ppm', 'Molibdeno',2],
                        ['Na', 'porciento', 'Sodio',2],
```

```
                        ['Nb', 'ppm', 'Niobio',2],
                        ['Nd', 'ppm', 'Neodimio',2],
                        ['Ni', 'ppm', 'N\xedquel',1],
                        ['P', 'ppm', 'F\xf3sforo',0],
                        ['Pb', 'ppm', 'Plomo',1],
                        ['Rb', 'ppm', 'Rubidio',2],
                        ['S', 'porciento', 'Azufre',3],
                        ['Sb', 'ppm', 'Antimonio',2],
                        ['Sc', 'ppm', 'Escandio',2],
                        ['Se', 'ppm', 'Selenio',3],
                        ['Sm', 'ppm', 'Samario',2],
                        ['Sn', 'ppm', 'Esta\xf1o',2],
                        ['TI', 'ppm', 'Talio',3],
                        ['Ta', 'ppm', 'Tantalio',3],
                        ['Tb', 'ppm', 'Terbio',2],
                        ['Te', 'ppm', 'Telurio',4],
                        ['Th', 'ppm', 'Torio',2],
                        ['Ti', 'porciento', 'Titanio',3],
                        ['U', 'ppm', 'Uranio',2],
                        ['V', 'ppm', 'Vanadio',1],
                        ['W', 'ppm', 'Wolframio',2],
                        ['Y', 'ppm', 'Itrio',1],
                        ['Yb', 'ppm', 'Iterbio',2],
                        ['Zn', 'ppm', 'Cinc',1],
                        ['Zr', 'ppm', 'Circonio',0],
                        ['Sr', 'ppm', 'Estroncio',0],
                        ['B', 'ppm', 'Boro',1],
                        ['Cl', 'ppm', 'Cloro',0],
                        ['Dy', 'ppm', 'Disprosio',2],
                        ['F', 'ppm', 'Fl\xfaor',0],
                        ['Gd', 'ppm', 'Gadolinio',2],
                        ['Ho', 'ppm', 'Holmio',3],
                        ['I', 'ppm', 'Yodo',2],
                        ['N', 'ppm', 'Nitr\xf3geno',0],
                        ['Pd', 'ppb', 'Paladio',2],
                        ['Pr', 'ppm', 'Praseodimio',2],
                        ['Pt', 'ppb', 'Platino',2],
                        ['Si', 'porciento', 'S\xedlice',1],
                        ['Tm', 'ppm', 'Tulio',3],
)
```

Dentro de la variable `lst_elementoGeoqui_2` el código guarda una tupla con la lista de los símbolos químicos, las unidades, los nombres y los decimales que corresponden a cada

elemento.

Una tupla es un espacio de memoria similar a una variable, que a diferencia de ésta no se modifica durante la ejecución y sirve para agrupar elementos de consulta, lo que mejora la performance del código.

Para que no se generen incompatibilidades entre la salida gráfica y la codificación en python 2.7, en los casos donde las palabras lleven tilde o eñe se utilizó el valor del caracter en código Unicode.

Funciones

```
def getExpressionMuestrasGQ(_campo,_rango,isLast=False):
    if isLast:
        return _campo+" >="+str(_rango[2][0])+ \
        " and "+_campo+" <="+str(_rango[2][1])
    else:
        return _campo+" >="+str(_rango[2][0])+ \
        " and "+_campo+" <"+str(_rango[2][1])
```

Define la función **getExpressionMuestrasGQ** que devuelve un texto concatenando los valores calculados para cada percentil que calcula, para ser utilizado como *Definition query*

```
def redondeoTexto(valor, decimal):
    entero = str(format((round(valor,decimal)),'4f')).split(".")[0]
    digitos =
(str(format((round(valor,decimal)),'4f')).split(".")[1])[0:decimal]
    if decimal ==0:
        return entero
    else:
        return entero +","+digitos
```

Define la función **redondeoTexto** que devuelve cada número como texto, con los lugares decimales definidos originalmente, para ser usado como nombre que define cada percentil en la leyenda.

```
def getLabelBreakMuestrasGQ(_rango,elem_decimales,isFirst=False):
    rango_min = ""
    rango_max = ""
    unidad="P"
```

```
    if _rango[0] == 0:
        rango_min = "m\xedn"
    else:
        rango_min = unidad + "<SUB>" + str(_rango[0]) + "</SUB>"

    if _rango[1] == 100:
        rango_max = "m\xe1x"
    else:
        rango_max = unidad +  "<SUB>"  + str(_rango[1]) + "</SUB>"

    if isFirst:
        return redondeoTexto(values_element_MIN,elem_decimales) + \
        " a "+ redondeoTexto(_rango[2][1],elem_decimales) + \
        " ("+str(rango_min)+ " - " + str(rango_max) + ") " + \
        "(n<SUB>=</SUB>"+str(_rango[3])+")
    else:
        return redondeoTexto(_rango[2][0],elem_decimales) + \
        " a "+ redondeoTexto(_rango[2][1],elem_decimales) + \
        " (" + str(rango_min) + " - " + str(rango_max) + ") " + \
        "(n<SUB>=</SUB>"+ str(_rango[3])+ ")"
```

Define la función **getLabelBreakMuestrasGQ** que devuelve un texto concatenado con la leyenda de cada percentil, usando los valores obtenidos de la función **redondeoTexto**.

Bucle

Posteriormente el código realiza un bucle **for**, que evalúa los datos de geoquímica para cada uno de los elementos químicos usando todas las muestras cargadas en la base de datos, los clasifica y los asigna a su respectivo percentil. Este mecanismo permite encontrar los valores que definen el límite de cada percentil, para filtrar los datos y representarlos clasificados y con sus respectivas referencias.

El bucle finaliza al guardarlo un archivo .mxd de documento de mapa y la visualización en .pdf del mismo donde además se incorporan las capas del grillado provistas.

```
for elementoGQui in lst_elementoGeoqui_2:
    simbolo = elementoGQui[0]
    unidadEle = elementoGQui[1]
    elem_decimales=elementoGQui[3]
    nombre_compuesto = simbolo + "_" + unidadEle
```

```
    if unidadEle == "porciento":
        unidadEle = "%"
    elem_descripcion = elementoGQui[2]
```

El ciclo comienza iterando todos los elementos de la tupla **`lst_elementoGeoqui_2`**, seleccionando de a un elemento químico a la vez para todo el circuito del bucle y asignando las siguientes variables:

- `simbolo` al símbolo químico
- `unidadEle` a su unidad de medida
- `elem_decimales` a los dígitos significativos
- `nombre_compuesto` a la concatenación del símbolo químico y la unidad, que replica el nombre de campo del dato en la base de datos.

```
    S2aTxtTitulo.text = u'Distribución de '.encode('iso-8859-1') +
elem_descripcion
    RefPrincipal.title = u'Distribución de '.encode('iso-8859-1') +
simbolo + \ " ("+unidadEle+")"
```

La función **`.text`** modifica el elemento de texto `S2aTxtTitulo` poniéndole el nombre del elemento químico utilizado para definir el título del mapa como nombre de dicho elemento a partir del contenido de la variable `elem_descripción` y la función **`.title`** modifica en el título del elemento leyenda el símbolo y unidades del elemento químico a través de las variables `simbolo` y `unidadEle`.

```
    ly_MuesGeoqui.definitionQuery = ""
```

Mediante la funcion **`.definitionQuery`** se modifica el valor del *definition query.* Esta *query* o consulta en lenguaje SQL filtra el dato visualizado y en este caso lo deja vacío, ya que se modificá posteriormente en cada elemento y para cada percentil.

```
    values_element_Null=[]
    values_element_NOTNull=[]
    values_element_calculos=[]
    values_element_positivos=[]
    values_element_negativ=[]
```

```
    values_element_Zero=[]
    values_element_MIN=[]

values_element_Null = [a[0] for a in
arcpy.da.FeatureClassToNumPyArray(ly_MuesGeoqui,nombre_compuesto,nombr
e_compuesto+" IS NULL", skip_nulls=False)]

    values_element_NOTNull = [a[0] for a in
arcpy.da.FeatureClassToNumPyArray(ly_MuesGeoqui,nombre_compuesto,skip_
nulls=True)]
    if len(values_element_Null)>0:
        arcpy.AddError(">>>> El elemento: "+elem_descripcion + \
        ", contiene valores nulos en algunos registros. Verifique que
         sea corecto.")
        ly_muestraGQ_class8 =
arcpy.mapping.ListLayers(prntMxd,"MuestraGeoquimica_class8",
dfMapaPrincipal)[0]
    elif len(values_element_NOTNull) == 0:
        arcpy.AddError(">>>> El elemento: "+elem_descripcion + \
        " no posee valores. Verifique que sea corecto porque no se
realizarán calculos.")
        continue

       values_element_positivos = [a[0] for a in
arcpy.da.FeatureClassToNumPyArray(ly_MuesGeoqui,nombre_compuesto,
nombre_compuesto+">0",skip_nulls=True)]
    values_element_positivos.sort()
    try:
        values_element_MIN=values_element_positivos[0]
    except:
        arcpy.AddError("El elemento " +simbolo+ " no tiene valores")
        continue

    values_element_negativ = [numpy.abs(a[0])/2 for a in
arcpy.da.FeatureClassToNumPyArray(ly_MuesGeoqui,nombre_compuesto,nombr
e_compuesto+"<0",skip_nulls=True)]

values_element_calculos =
values_element_negativ+values_element_positivos
    values_element_calculos.sort()
```

```
    values_element_Zero = [a[0] for a in
arcpy.da.FeatureClassToNumPyArray(ly_MuesGeoqui,nombre_compuesto,nombr
e_compuesto+"=0",skip_nulls=True)]
    if len(values_element_Zero)>0:
        arcpy.AddError(">>>> Un elemento: "+ elem_descripcion + \
                               ", no puede tener el valor 0 asignado.
Por favor, si posee datos completelos para poder realizar los
cálculos.")
        continue
```

Luego el código declara siete variables de tipo lista, dentro de las cuales se van adjuntando valores de la tabla de datos obtenidos de la capa declarada en **`ly_MuesGeoqui`** para el elemento químico que está siendo evaluado, según el criterio:

- `values_element_NOTNull` recopila todos los registros que tengan dato y muestra una advertencia en caso de no existir datos.
- `values_element_Null` recopila todos los registros nulos y muestra una advertencia. A estos valores los asigna a en el mapa a la capa **`MuestraGQ_class8`** donde se representan los valores nulos.
- `values_element_positivos` recopila todos los valores positivos y los ordena. Además verifica que existan valores y muestra una advertencia en caso de que no existan.
- `values_element_MIN` dentro de esta variable se almacena el valor mínimo de la lista, que al tener un dato permite confirmar si existen o no valores en la tabla y muestra una advertencia.
- `values_element_negativ` recopila todos los valores positivos de los valores negativos para cada elemento químico como la mitad de su valor absoluto. Los valores negativos representan datos por debajo del límite de detección.
- `values_element_calculos` junta los valores positivos y negativos, lo que permite calcular los valores de percentiles en el universo completo de las muestra.
- `values_element_Zero` revisa si existen valores cero en los datos. Si existen muestra una advertencia y cancela el análisis.

❗ En el caso de existir valores cero se cancela el análisis porque no deberían existir dentro de la tabla de datos y significa que hubo un error durante la instancia de carga de valores a la base de datos. Se muestra una advertencia para revisar el dato original.

Calculo de percentiles

```
   #   Los datos calculados de los percentiles se almacenan en la
lista: rangos_percent.
    rangos_percent = [[0,50],[50,75],[75,90],[90,95],[95,98],[98,100]]
    if len(values_element_calculos)>0:
       for rang in rangos_percent:
           rang.append(numpy.percentile(values_element_calculos,rang))

        # Calculando numero de entidades por rango
        for rang in rangos_percent:
            nss = 0
            if rang==rangos_percent[-1]:
                for v in values_element_positivos:
                    if v>=rang[2][0] and v<=rang[2][1]:
                        nss+=1
            else:
                for v in values_element_positivos:
                    if v>=rang[2][0] and v<rang[2][1]:
                        nss+=1
            rang.append(nss)
    else:
        arcpy.AddError(">>>> El elemento: "+elem_descripcion + \
       " no posee valores. Verifique que sea corecto porque no se
realizarán calculos.")
        continue
```

Se encuentran los límites para cada percentil y se guardan dentro de la variable de tipo lista **rangos_percent**.

Generación de capas por percentil

```
    ly_muestraGQ_class1.definitionQuery =
getExpressionMuestrasGQ(nombre_compuesto,rangos_percent[5],True)
    ly_muestraGQ_class1.name =
getLabelBreakMuestrasGQ(rangos_percent[5], elem_decimales)
    arcpy.AddMessage(ly_muestraGQ_class1.name)
    ly_muestraGQ_class2.definitionQuery =
getExpressionMuestrasGQ(nombre_compuesto,rangos_percent[4])
    ly_muestraGQ_class2.name =
getLabelBreakMuestrasGQ(rangos_percent[4], elem_decimales)
```

```
    arcpy.AddMessage(ly_muestraGQ_class2.name)
    ly_muestraGQ_class3.definitionQuery
=getExpressionMuestrasGQ(nombre_compuesto,rangos_percent[3])
    ly_muestraGQ_class3.name =
getLabelBreakMuestrasGQ(rangos_percent[3], elem_decimales)
    arcpy.AddMessage(ly_muestraGQ_class3.name)
    ly_muestraGQ_class4.definitionQuery =
getExpressionMuestrasGQ(nombre_compuesto,rangos_percent[2])
    ly_muestraGQ_class4.name =
getLabelBreakMuestrasGQ(rangos_percent[2], elem_decimales)
    arcpy.AddMessage(ly_muestraGQ_class4.name)
    ly_muestraGQ_class5.definitionQuery =
getExpressionMuestrasGQ(nombre_compuesto,rangos_percent[1])
    ly_muestraGQ_class5.name =
getLabelBreakMuestrasGQ(rangos_percent[1], elem_decimales)
    arcpy.AddMessage(ly_muestraGQ_class5.name)

    ly_muestraGQ_class6.definitionQuery =
getExpressionMuestrasGQ(nombre_compuesto,rangos_percent[0])
    ly_muestraGQ_class6.name =
getLabelBreakMuestrasGQ(rangos_percent[0], elem_decimales,  True)
    arcpy.AddMessage(ly_muestraGQ_class6.name)
```

Se modifican los definition query para cada una de las capas generadas previamente con la simbología de percentil definida según la normativa. lo que permite filtrar las muestras según el percentil en el cual se encuentran. Además se modifica el nombre de la capa que aparecerá en la leyenda de las referencias.

```
        if len(values_element_negativ)>0:
            ly_muestraGQ_class7.visible = True
            ly_muestraGQ_class7.definitionQuery = nombre_compuesto+"<0"
            ly_muestraGQ_class7.name = u'Bajo el límite de detección
(n<SUB>=</SUB>'.encode('iso-8859-1')
+str(len(values_element_negativ))+")"
            arcpy.AddMessage(ly_muestraGQ_class7.name)
    else:
            ly_muestraGQ_class7.visible = False

    if len(values_element_Null) > 0:
            ly_muestraGQ_class8.visible = True
```

```
        ly_muestraGQ_class8.definitionQuery = nombre_compuesto+" is
Null"
        ly_muestraGQ_class8.name = u'Muestras sin medición
(n<SUB>=</SUB>'.encode('iso-8859-1') + str(len(values_element_Null)) +
")"
        arcpy.AddMessage(ly_muestraGQ_class8.name)
    else:
        ly_muestraGQ_class8.visible = False
```

Sólo en el caso de que haya valores por debajo del límite de detección los represento en la capa **`MuestraGQ_class7`** y muestro la capa en el mapa. Lo mismo con valores nulos en la capa **`MuestraGQ_class8`**.

Incorporación de grilla

```
        targetGrillaLayer = arcpy.mapping.ListLayers(prntMxd,
"GrillaGeoquimicaGroup", dfMapaPrincipal)[0]
    RefPrincipal.autoAdd = False
    capaGrilla = arcpy.mapping.Layer(pathGrilla + u"\\" + simbolo +
u"_contour_region.lyr")
    arcpy.AddMessage("Cargando grilla")
    arcpy.mapping.AddLayerToGroup(dfMapaPrincipal,
targetGrillaLayer,capaGrilla, "TOP")
    RefPrincipal.autoAdd = False
```

Agrego la capa con los polígonos del grillado del elemento químico correspondiente al grupo de capas **`GrillaGeoquimicaGroup`** a partir de la carpeta declara en los parámetros. Este grupo permite visualizar la grilla debajo de las muestras y de la elementos de topografía.

Los nombres de las capas de grilla deberán estar normalizados según *simbolo_contour_region.lyr*

Salida gráfica

```
        pathMxdSave =
u'{0}\\GeoquiBD_{1}_printable.mxd'.format(pathEspacioTrabajo,
simbolo).encode('iso-8859-1')
    if os.path.exists(pathMxdSave):
        os.remove(pathMxdSave)

    prntMxd.saveACopy(pathMxdSave)
```

```
    arcpy.AddMessage("")
    arcpy.AddMessage(u'Salida gráfica generada para Elemento:
'.encode('iso-8859-1') +simbolo)
```

Guarda los archivos de proyecto de mapa en la carpeta de trabajo declarada en los parámetros. Si existe un archivo del mismo nombre, lo sobreescribe.

```
        pathPDFSave =
u'{0}\\GeoquiBD_{1}_printable.pdf'.format(pathEspacioTrabajo,
simbolo).encode('iso-8859-1')
    if os.path.exists(pathPDFSave):
        os.remove(pathPDFSave)
    arcpy.mapping.ExportToPDF(prntMxd, pathPDFSave, "PAGE_LAYOUT")
    arcpy.AddMessage(u'PDF de geoquimica y grilla generado para
Elemento: '.encode('iso-8859-1') +simbolo)
    arcpy.AddMessage("Ok")
```

Guarda las visualizaciones en pdf en la carpeta de trabajo declarada en los parámetros. Si existe un archivo del mismo nombre, lo sobreescribe.

```
        capaRemover = arcpy.mapping.ListLayers(prntMxd,
'*contour_region*', dfMapaPrincipal)[0]
    arcpy.mapping.RemoveLayer(dfMapaPrincipal, capaRemover) #remueve
grilla del elemento anterior
```

Remuevo la grilla del elemento anterior del mapa, para volver a empezar un nuevo ciclo con el siguiente elemento

Resultados

A partir del código se obtienen como máximo 144 archivos, dos tipos por cada elemento químico analizado:

Archivo de salida .mxd

Archivo de proyecto de mapa, para software ArcMap, con extension ,mxd donde se pueden modificar las capas y simbología

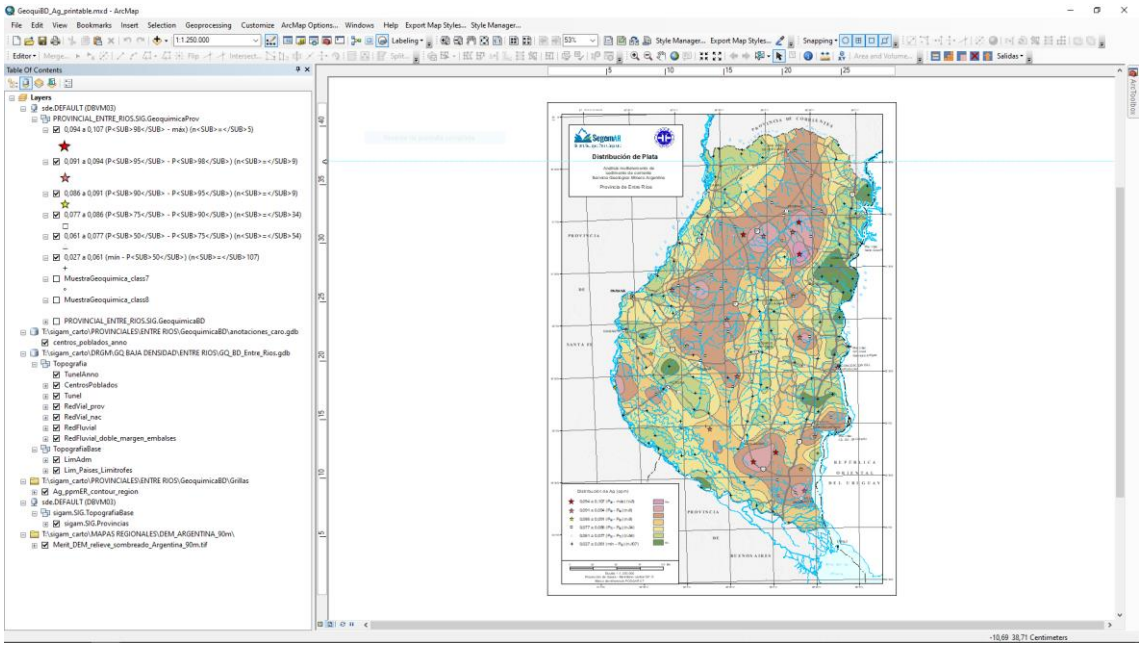
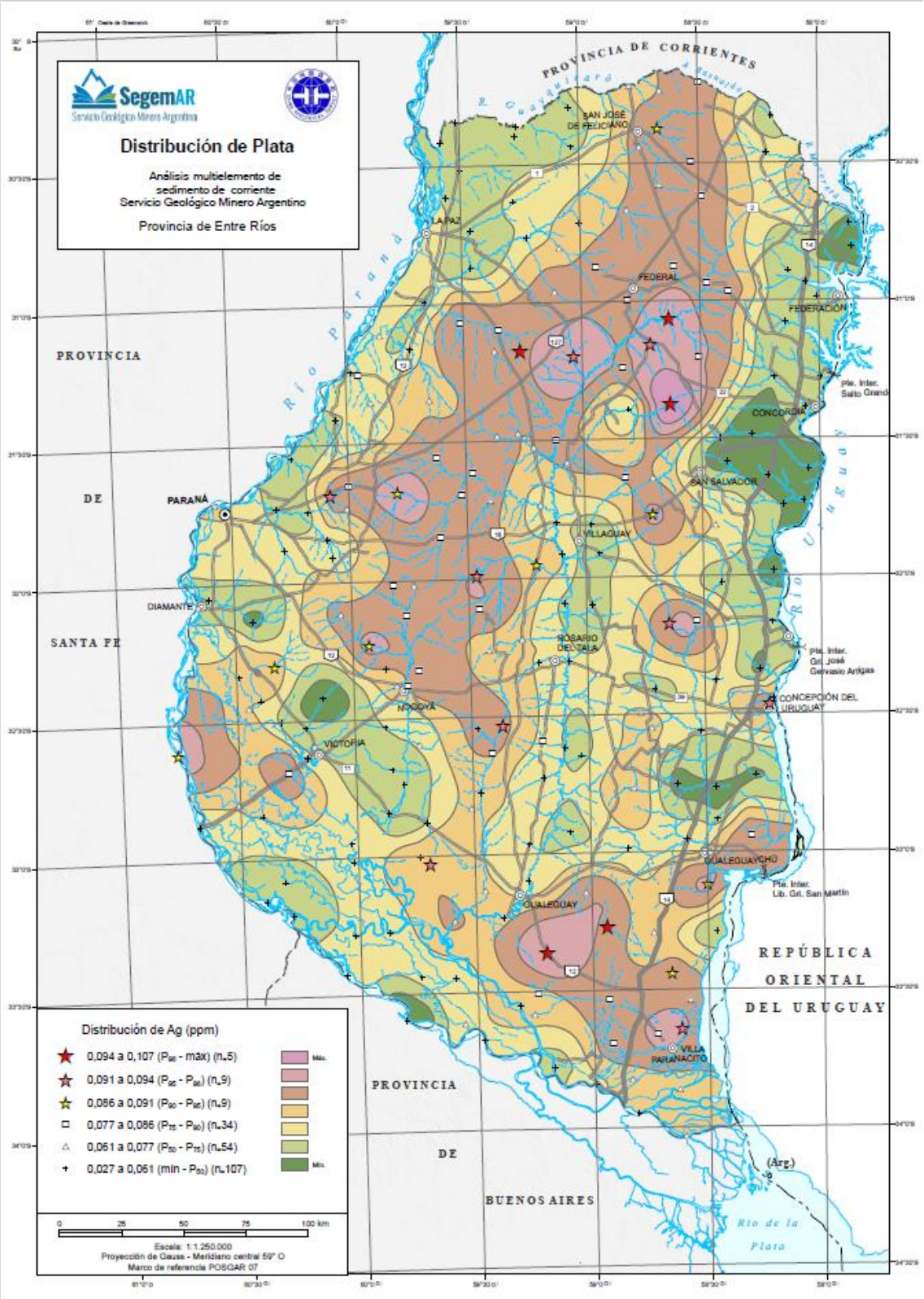


Imagen del archivo de mapa en ArcMap donde se encuentran todas las capas georeferenciadas del proyecto con su respectiva simbología

Archivo de salida .pdf



Archivo de visualización digital del mapa en formato .pdf

Bibliografía

Environmental Systems Research Institute (ESRI), 2014. ArcGIS Desktop Help 10.2 Arcpy https://resources.arcgis.com/en/help/main/10.2/index.html

Fernández, Juan Carlos, 2017. Script Salida Gráfica Geoquímica para entorno sigam. INDUROT

Marquinez García, Jorge; García Manteca, Pilar; Colina, A.; Turel, Andrea Vilma; Moser, Leda Cecilia; Candaosa, Norberto Gabriel; Ferpozzi, Federico Javier; Chavez, Silvia Beatríz. 2018*a*. Diseño del Módulo de Salidas Gráficas de la Carta Geoquímica. Contribuciones Técnicas - SIG e IDE N.º 20 37p. Buenos Aires. SEGEMAR. Instituto de Geología y Recursos Minerales. Unidad Sensores Remotos y SIG.

Marquínez García, Jorge; García Manteca, Pilar; Sánchez, D.; Colina, A.; Candaosa, Norberto Gabriel; Chavez, Silvia Beatríz; Ferpozzi, Federico Javier; Olmos, María Isabel; Benítez, Javier; Rodríguez, Valentín; Gambande, Liliana; Tavitian Serrano, Ana Felisa; Oyola, Matías; 2018*b*. Diseño del Módulo de Salidas Gráficas: Especificaciones Generales. Contribuciones Técnicas - SIG e IDE N.º 18 37p. Buenos Aires. SEGEMAR. Instituto de Geología y Recursos Minerales. Unidad Sensores Remotos y SIG.

Marquínez García, Jorge; García Manteca, Pilar; Sánchez, D.; Colina, A.; Fernández Iglesias, Juan Carlos; Candaosa, Norberto Gabriel; Chavez, Silvia Beatríz; Ferpozzi, Federico Javier. 2018*c*. Manual de Usuario de las Herramientas del Módulo de Salidas Gráficas del SIGAM. Contribuciones Técnicas - SIG e IDE N.º 16 15p. Buenos Aires. SEGEMAR. Instituto de Geología y Recursos Minerales. Unidad Sensores Remotos y SIG.

Marquínez García, Jorge; García Manteca, Pilar; Sánchez, D.; Colina, A.; Fernández Iglesias, Juan Carlos; Candaosa, Norberto Gabriel; Ferpozzi, Federico Javier; Chavez, Silvia Beatríz. 2018*d*. Manual del Administrador de las Herramientas del SIGAM para la Escala 1:250.000. Contribuciones Técnicas - SIG e IDE Nº29 59p. Buenos Aires. SEGEMAR. Instituto de Geología y Recursos Minerales. Unidad Sensores Remotos y SIG.

Servicio Geológico Minero Argentino, 2001. Normativa para la Carta Geoquímica de la República Argentina. Programa Nacional de Cartas Geológicas y Temáticas de la República Argentina. 26 p. Buenos Aires, Servicio Geológico Minero Argentino. Sector Geoquímica

Servicio Geológico Minero Argentino s.f. Sistema de Información Geológica Ambiental Minera (SIGAM). Fecha de consulta: 13 de diciembre de 2023.