datosaire

September 4, 2024

[12]:
```
import pandas as pd
import numpy as np
from matplotlib.pylab import plt
import seaborn as sns
import matplotlib
import matplotlib.gridspec as gridspec
import matplotlib as mpl
from pathlib import Path
from libmolina import *
from datetime import datetime
from iminuit import Minuit
from probfit import Chi2Regression, Polynomial
from notebook.services.config import ConfigManager
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[12], line 13
     11 from iminuit import Minuit
     12 from probfit import Chi2Regression, Polynomial
---> 13 from notebook.services.config import ConfigManager

ModuleNotFoundError: No module named 'notebook.services'
```

[10]:
```
matplotlib.rcParams["text.usetex"] = True
cmap = plt.colormaps["viridis"]  # plasma
viridis = mpl.colormaps["plasma"]
```

[11]:
```
variables_contaminacion = {
    "co_concentracion": {"variable": "CO", "unidad": "$\mu$g$/$m$^3$"},
    "direccion_viento": {"variable": "Dirección del Viento", "unidad":␣
 ↪"$^{\circ}$"},
    "humedad_relativa": {"variable": "Humedad Relativa", "unidad": "\%"},
    "humedad_relativa_10m": {"variable": "Humedad Relativa 10 m", "unidad":␣
 ↪"\%"},
    "humedad_relativa_2m": {"variable": "Humedad Relativa 2 m", "unidad": "\%"},
    "no_concentracion": {"variable": "NO", "unidad": "$\mu$g$/$m$^3$"},
```

```
    "no2_concentracion": {"variable": "NO2", "unidad": "$\mu$g$/$m$^3$"},
    "o3_concentracion": {"variable": "O3", "unidad": "$\mu$g$/$m$^3$"},
    "pm10_concentracion": {"variable": "PM10", "unidad": "$\mu$g$/$m$^3$"},
    "pm25_concentracion": {"variable": "PM2.5", "unidad": "$\mu$g$/$m$^3$"},
    "pst_concentracion": {"variable": "PST", "unidad": "$\mu$g$/$m$^3$"},
    "precipitacion_liquida": {"variable": "Precipitación Líquida", "unidad":␣
 ↪"mm"},
    "presion_atmosferica": {"variable": "Presión Atmosférica", "unidad":␣
 ↪"mmHg"},
    "radiacion_solar_global": {
        "variable": "Radiación Solar Global",
        "unidad": "W$/$m$^2$",
    },
    "radiacion_uvb": {"variable": "Radiación UVB", "unidad": "MED$/$h"},
    "so2_concentracion": {"variable": "SO2", "unidad": "$\mu$g$/$m$^3$"},
    "temperatura": {"variable": "Temperatura", "unidad": "$^{\circ}$C"},
    "temperatura_10m": {"variable": "Temperatura a 10 m", "unidad":␣
 ↪"$^{\circ}$C"},
    "temperatura_2m": {"variable": "Temperatura a 2 m", "unidad":␣
 ↪"$^{\circ}$C"},
    "velocidad_viento": {"variable": "Velocidad del Viento", "unidad": "m$/$s"},
}
```

[3]:
```
data_frame_subconjunto = pd.read_csv(
    "datos_segmentados/co_concentracion.csv",
    dtype=str,
)
```

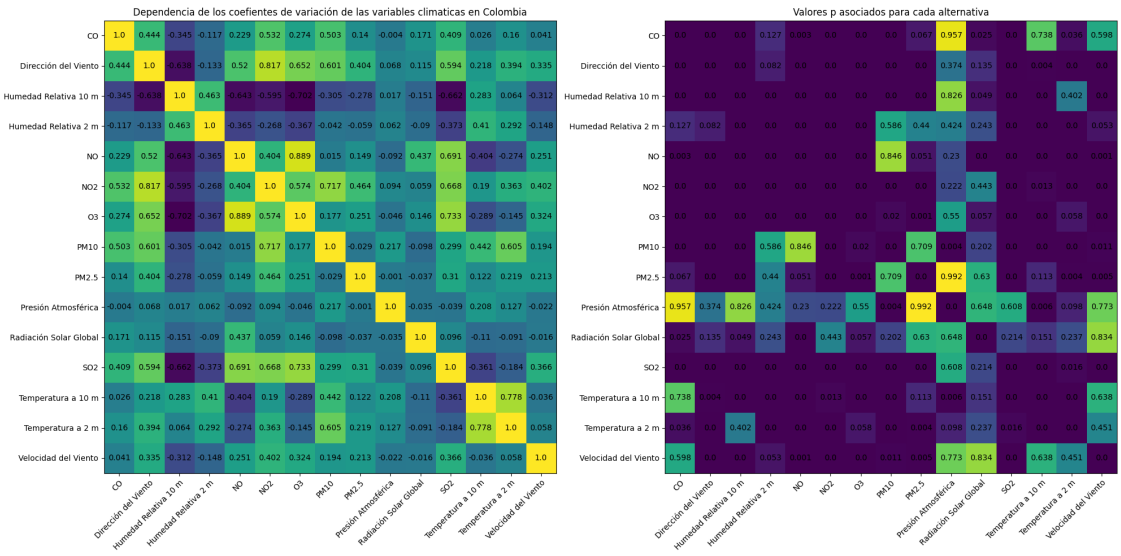
[ ]:
```
# datf.info()
# datf.isnull().sum()
data_frame_subconjunto = data_frame_subconjunto.dropna()
```

[6]:
```
mxsizeclas=[]
variable_climaticas=[]
for us, ot in enumerate(list(variables_contaminacion.keys())):
    data_frame_subconjunto = pd.read_csv("datos_segmentados/" + ot + ".csv",␣
 ↪dtype=str)
    data_frame_subconjunto = data_frame_subconjunto.dropna()
    quitarcoma = [float(dato.replace(",", "")) for dato in␣
 ↪data_frame_subconjunto["Concentración"]]
    variable_clima_boxplot = datos_caja_biogotes(quitarcoma)
    k = len(np.histogram_bin_edges(variable_clima_boxplot, bins="auto"))
    if k > 75:
        variable_climaticas.append(list(variables_contaminacion.values())[us])
        frc,_= np.histogram(variable_clima_boxplot, bins=171)
        frcnormal = frc / sum(frc)
```

```
        frcnormal=list(frcnormal)
        mxsizeclas.append(frcnormal)
    sizeclas = np.array(mxsizeclas)
```

[7]:
```
mx_corr_pearsonr, mx_valores_p = correlacion_pearsonr_difrentes_variables(
    np.array(sizeclas)
)
```

[8]:
```
fig = plt.figure(figsize=(20, 17), tight_layout=True)
gs = gridspec.GridSpec(1, 2)
ax0 = fig.add_subplot(gs[0, 0])
ax1 = fig.add_subplot(gs[0, 1])
ax = [ax0, ax1]

titulos = [
    "Dependencia de los coeficientes de variación de las variables climaticas␣
 ↪en Colombia",
    "Valores p asociados para cada alternativa",
]

ffarmers = [variable_climaticas, variable_climaticas]
harvest = [mx_corr_pearsonr, mx_valores_p]
graficar_correlacines_pearsonr(harvest, ax, plt, titulos, ffarmers)

fig.tight_layout()
fig.savefig("cvclima.pdf")
plt.show()
```



[ ]:
```
fig = plt.figure(figsize=(20, 20), tight_layout=True)
nN=len(variable_climaticas)
gs = gridspec.GridSpec(nN, nN)

for i, vlimai in enumerate(variable_climaticas):
    for j, vclimaj in enumerate(variable_climaticas):

        fig.add_subplot(gs[i, j]).plot(
            list(sizeclas[np.array(variable_climaticas) == vlimai][0]),
            list(sizeclas[np.array(variable_climaticas) == vclimaj][0]),
            "o",
        )
plt.show(),
```

[33]:
```
cv_variable_climaticas=[]
cv_datos_clima=[]
dist_prob_climaticas=[]
for us, ot in enumerate(list(variables_contaminacion.keys())):
    data_frame_subconjunto = pd.read_csv("datos_segmentados/" + ot + ".csv",␣
 ↪dtype=str)
    data_frame_subconjunto = data_frame_subconjunto.dropna()
    lista_grade = [float(dato.replace(",", "")) for dato in␣
 ↪data_frame_subconjunto["Concentración"]]
    datos_concent_bigotes = datos_caja_biogotes(lista_grade)
    k = len(np.histogram_bin_edges(datos_concent_bigotes, bins="auto"))
    if k > 75:
        xfrc,intervalo_de_clasess = np.histogram(datos_concent_bigotes,␣
 ↪bins=171)
        categorias_de_clase = pd.cut(datos_concent_bigotes,␣
 ↪bins=intervalo_de_clasess)
        datos_agrupados_por_clase = (
            pd.Series(datos_concent_bigotes).groupby(categorias_de_clase)
        )
        coeficiete_de_variacion = list(datos_agrupados_por_clase.apply(lambda x:
 ↪ x.std() / x.mean()))
        nan_indic = np.where(np.isnan(coeficiete_de_variacion))[0]
        if len(nan_indic) == 0:
            xfrc = xfrc / sum(xfrc)
            dist_prob_climaticas.append(xfrc)
            cv_variable_climaticas.append(list(variables_contaminacion.
 ↪values())[us])
            cv_datos_clima.append(coeficiete_de_variacion)
```

[34]:
```
# mx_corr_pearsonr,mx_valores_p = correlacion_pearsonr_difrentes_variables(np.
 ↪array(cv_datos_clima))
matriz = np.array(cv_datos_clima)
```

```
cv_flatten=matriz.flatten()
_, interclasescv = np.histogram(cv_flatten, bins=171)
```

[ ]:
```
fig = plt.figure(figsize=(15, 15), tight_layout=True)
gs = gridspec.GridSpec(1,1)
ax = fig.add_subplot(gs[0, 0])

for ur in dist_prob_climaticas:
    ax.plot(interclasescv[:-1][ur<0.1], ur[ur<0.1], "o")

fig.tight_layout()
fig.savefig("cvclima.pdf")
plt.show()
```

[ ]:
```
ui = np.array([[i, j] for i in range(1) for j in range(2)])
for ur in range(10,11,1):#len(list(variables_contaminacion.keys())),2):
    fig = plt.figure(figsize=(20, 5), tight_layout=True)
    gs = gridspec.GridSpec(1, 2)
    for us in range(ur,ur+2):
        uz = us % 2
        ot=list(variables_contaminacion.keys())[us]
        ax0 = fig.add_subplot(gs[ui[:, 0][uz], ui[:, 1][uz]])
        data_frame_subconjunto = pd.read_csv("datos_segmentados/" + ot + ".
 ↪csv",dtype=str)
        data_frame_subconjunto["Concentración"] =␣
 ↪data_frame_subconjunto["Concentración"].str.replace(",", "")
        data_frame_subconjunto["Concentración"] =␣
 ↪data_frame_subconjunto["Concentración"].astype(float)
        data_frame_subconjunto.dropna(axis=0, subset=["Concentración"])
        lista_grade = data_frame_subconjunto["Concentración"]
        # [float(dato.replace(",", "")) for dato in datf["Concentración"]]
        lista_pequena = datos_caja_biogotes(lista_grade)
        io = np.where(np.isin(lista_grade, lista_pequena))[0]
        fecstr = [dato for dato in data_frame_subconjunto["Fecha"]]
        fstr = list(np.array(fecstr)[io])
        fhstr = [dato.split(" ") for dato in fstr]
        try:
            fechas_str = [
                dato[0]
                + " "
                + dato[1]
                + " "
                + dato[2].split(".")[0].upper()
                + dato[3].split(".")[0].upper()
                for dato in fhstr
            ]
            fechas_str = covertir_formato_24_horas(fechas_str)
```

```
        except:
            fechas_str = [datos[0] + " " + datos[1]+":00" for datos in fhstr]

        fechas_datetime = pd.to_datetime(fechas_str, format="%d/%m/%Y %H:%M:%S")

        conjunto_datos = {"Fecha": fechas_datetime, "Concentracion":␣
↪lista_pequena}

        datdataframe=pd.DataFrame(conjunto_datos)

        datdataframe["Semana"] = datdataframe["Fecha"].dt.to_period("W")

        promedio_semanal = datdataframe.groupby("Semana")["Concentracion"].
↪mean()
        errs = datdataframe.groupby("Semana")["Concentracion"].std()
        # contandodatos = datdataframe.groupby("Semana")["Concentracion"].
↪count()

        # grupo_de_datos = datdataframe.groupby("Semana")
        # cvfT = grupo_de_datos["Concentracion"].apply(lambda x: x.std() / x.
↪mean())

        # k = len(np.histogram_bin_edges(lista_pequena, bins="auto"))

        # if k > 75:
        datetimes = promedio_semanal.to_timestamp()
        fecha_semanal=np.array(datetimes.keys())
        values_semanal = np.array(datetimes.values)
        errs_semanal = np.array(errs.values)
        ax0.plot(fecha_semanal,values_semanal,"-o", markersize=3, color='blue')
        # ax0.fill_between(
        #     fecha_semanal,
        #     values_semanal - errs_semanal,
        #     values_semanal + errs_semanal,
        #     color="black",
        #     alpha=0.1,
        # )

        # frc, cbin = np.histogram(lista_pequena, bins=171)
        # print(len(lista_pequena), " ", ot, " ", k)
        # x = cbin[:-1]
        # y = frc / sum(frc)
        # ax1.plot(x, y, "o")
        # for u in range(len(x)):
        #     ax1.bar(x, y, width=140, ec="w", color=viridis.colors[u])

        unidad_variable=variables_contaminacion[ot]["unidad"]
```

```
        nombre_variable=variables_contaminacion[ot]["variable"]
        ax0.xaxis.set_tick_params(labelsize=17)
        ax0.yaxis.set_tick_params(labelsize=17)
        ax0.set_xlabel(f"Tiempo(wk)", fontsize=20)
        ax0.set_ylabel(f"{nombre_variable} ({unidad_variable})", fontsize=18)
        ax0.set_title(nombre_variable, fontsize=20)



    fig.tight_layout()
    plt.subplots_adjust(wspace=0.15)
    ncont=int(ur/2)
    # fig.savefig(f"imgseriestmp/img{ncont}.pdf")
    plt.show()
```

[342]:
```
data_frame_subconjunto = data_frame_subconjunto.dropna()
sc = np.array(data_frame_subconjunto["Concentracion"])
datxf = [float(dato.replace(",", "")) for dato in sc]
```

[342]:

	Tiempo	Concentracion	Variable	Id_municipal	Unidades
0	16116.0	1946.297182	CO	5001	µg/m3
1	16117.0	1831.809112	CO	5001	µg/m3
2	16118.0	1831.809112	CO	5001	µg/m3
3	16119.0	2289.761390	CO	5001	µg/m3
4	16120.0	2289.761390	CO	5001	µg/m3
…	…	…	…	…	…
960359	70065.0	171.700000	CO	11001	ugm3
960360	70066.0	157.400000	CO	11001	ugm3
960361	70067.0	157.400000	CO	11001	ugm3
960362	70068.0	163.600000	CO	11001	ugm3
960363	70069.0	171.700000	CO	11001	ugm3

[877028 rows x 5 columns]

1 Busqueda de avalanchas de series temporales climáticas en datos outliers

1.0.1 Diviendo datos por variable y municipio

Dada la gran cantidad de datos, es necesario segmentarlos antes de cargarlos. Además, como las muestras contienen un formato de fecha incorrecto, es esencial corregirlo para poder analizar los datos como una serie temporal.

[ ]:
```
conjunto_de_datos = pd.read_csv(
    "datos_aire.csv",
    dtype=str,
    chunksize=5000000,
```

```
)
grupos_acumulados = {}
for chunk in conjunto_de_datos:
    grouped = chunk.groupby("Variable")
    for nombre_grupo, grupo in grouped:
        if nombre_grupo in grupos_acumulados:
            grupos_acumulados[nombre_grupo] = pd.concat(
                [grupos_acumulados[nombre_grupo], grupo]
            )
        else:
            grupos_acumulados[nombre_grupo] = grupo

grupos_acumulados_condicional = {}
for us, ot in enumerate(list(variables_contaminacion.keys())):
    vrclima = variables_contaminacion[ot]["variable"]
    varPM = grupos_acumulados[vrclima]["Fecha"].str.replace("p. m.", "PM")
    grupos_acumulados[vrclima]["Fecha"] = varPM
    varAM = grupos_acumulados[vrclima]["Fecha"].str.replace("a. m.", "AM")
    grupos_acumulados[vrclima]["Fecha"] = varAM
    vrcondicion = grupos_acumulados[vrclima]["Fecha"]
    grupos_acumulados[vrclima]["Condicional"] = vrcondicion.str.contains("M")
    remplazofv = grupos_acumulados[vrclima]["Condicional"].map(
        {True: "Fhora_AMPM", False: "Fhora_24HH"}
    )
    grupos_acumulados[vrclima]["Condicional"] = remplazofv
    grupo_condicional = grupos_acumulados[vrclima].groupby("Condicional")
    for nombre_grupo, grupo in grupo_condicional:
        if nombre_grupo in grupos_acumulados_condicional:
            grupos_acumulados_condicional[nombre_grupo] = pd.concat(
                [grupos_acumulados_condicional[nombre_grupo], grupo]
            )
        else:
            grupos_acumulados_condicional[nombre_grupo] = grupo
grupo_condicional_acumulados = pd.concat(grupos_acumulados_condicional.values())
grup_condicional = grupo_condicional_acumulados.groupby("Condicional")
grupos_formatoHora_12h_24H = {}
for nombre_grupo, grupo in grup_condicional:
    if nombre_grupo in grupos_formatoHora_12h_24H:
        grupos_formatoHora_12h_24H[nombre_grupo] = pd.concat(
            [grupos_formatoHora_12h_24H[nombre_grupo], grupo]
        )
    else:
        grupos_formatoHora_12h_24H[nombre_grupo] = grupo

grupos_formatoHora_12h_24H["Fhora_AMPM"]["Fecha_24h"] = pd.to_datetime(
    grupos_formatoHora_12h_24H["Fhora_AMPM"]["Fecha"],
    format="%d/%m/%Y %I:%M:%S %p",
```

```
    errors="coerce",
)

fecha_minima = grupos_formatoHora_12h_24H["Fhora_AMPM"]["Fecha_24h"].min()
fecha_referencia = pd.to_datetime(fecha_minima, format="%d/%m/%Y %H:%M:%S")
# # Calcular la diferencia en horas entre cada fecha y la hora de referencia
grupos_formatoHora_12h_24H["Fhora_AMPM"]["Fecha_horas"] = (
    grupos_formatoHora_12h_24H["Fhora_AMPM"]["Fecha_24h"] - fecha_referencia
).dt.total_seconds() / 3600

grupos_formatoHora_12h_24H["Fhora_AMPM"].to_csv("Fhora_AMPM.csv", index=False)

grupos_formatoHora_12h_24H["Fhora_24HH"]["Fecha_24h"] = pd.to_datetime(
    grupos_formatoHora_12h_24H["Fhora_24HH"]["Fecha"],
    format="%d/%m/%Y %H:%M",
    errors="coerce",
)

grupos_formatoHora_12h_24H["Fhora_24HH"]["Fecha_horas"] = (
    grupos_formatoHora_12h_24H["Fhora_24HH"]["Fecha_24h"] - fecha_referencia
).dt.total_seconds() / 3600

grupos_formatoHora_12h_24H["Fhora_24HH"].to_csv("Fhora_24HH.csv", index=False)
```

Debido a la limitación de memoria disponible (8 GB de RAM y 20 GB de swap), fue necesario almacenar los conjuntos de datos 'Fhora_AMPM.csv' y 'Fhora_24HH.csv' antes de cargarlos. Posteriormente, se dividieron y guardaron por municipio, y adicionalmente se guardaron los datos correspondientes a cada variable de forma independiente.

[ ]:
```
grupos_acumulados = {}
for datsclima in ["Fhora_AMPM", "Fhora_24HH"]:
    conjunto_de_datos = pd.read_csv(
        datsclima + ".csv",
        dtype=str,
        chunksize=5000000,
    )

    for chunk in conjunto_de_datos:
        grouped = chunk.groupby("Variable")
        for nombre_grupo, grupo in grouped:
            if nombre_grupo in grupos_acumulados:
                grupos_acumulados[nombre_grupo] = pd.concat(
                    [grupos_acumulados[nombre_grupo], grupo]
                )
            else:
                grupos_acumulados[nombre_grupo] = grupo
```

```
for us, ot in enumerate(list(variables_contaminacion.keys())):
    # dirpath = Path(f"datos_outliers/{ot}").mkdir(parents=True, exist_ok=True)
    vrclima = variables_contaminacion[ot]["variable"]
    df_colconcet_noNulos = (
        grupos_acumulados[vrclima].dropna(axis=0, subset=["Concentración"]).
 ↪copy()
    )
    df_noNulos_ordenados = df_colconcet_noNulos.sort_values(by="Fecha_horas").
 ↪copy()
    df_noNulos_ordenados["Concentración"] = df_noNulos_ordenados[
        "Concentración"
    ].str.replace(",", "")

    df_noNulos_ordenados["Concentración"] = df_noNulos_ordenados[
        "Concentración"
    ].astype(float)

    Q1 = df_noNulos_ordenados["Concentración"].quantile(0.25)
    Q3 = df_noNulos_ordenados["Concentración"].quantile(0.75)
    iqr = Q3 - Q1

    lower_limit = Q1 - 1.5 * iqr
    upper_limit = Q3 + 1.5 * iqr

    outliers_municipal = df_noNulos_ordenados[
        (df_noNulos_ordenados["Concentración"] > lower_limit)
        & (df_noNulos_ordenados["Concentración"] < upper_limit)
    ]
    nombre_archivo = f"datos_sin_outliers/{ot}.csv"
    outliers_municipal.to_csv(nombre_archivo, index=False)

    # grouped_id_municipio = outliers_municipal.groupby("Código del municipio")
    # grupos_acumulados_municipio = {}
    # for nombre_grupo, grupo in grouped_id_municipio:
    #     if nombre_grupo in grupos_acumulados_municipio:
    #         grupos_acumulados_municipio[nombre_grupo] = pd.concat(
    #             [grupos_acumulados_municipio[nombre_grupo], grupo]
    #         )
    #     else:
    #         grupos_acumulados_municipio[nombre_grupo] = grupo

    # for nombre_grupo, grupo in grupos_acumulados_municipio.items():
    #     nombre_archivo = f"datos_outliers/{ot}/{nombre_grupo}.csv"
    #     grupo.to_csv(nombre_archivo, index=False)
    #     print(f"Guardado: {nombre_archivo}")
```

[3]:
```
# df_grupos_acumulados = pd.concat(grupos_acumulados.values())
```

[39]:
```
df_subconjunto=df_grupos_acumulados[
    [
        "Fecha_horas",
        "Concentración",
        "Código del municipio",
        "Nombre del municipio",
        "Variable"
    ]
]
```

[ ]:
```
for nombre_grupo, grupo in df_subconjunto_grupos_acumulados.items():
    lista_var=[]
    for namgvar in list(variables_contaminacion.keys()):
        lista_var.append(variables_contaminacion[namgvar]["variable"])

    i = np.where(np.array(lista_var) == nombre_grupo)[0]
    name_archivo = np.array(list(variables_contaminacion.keys()))[i]

    nombre_archivo = f"{name_archivo[0]}.csv"

    grupo.to_csv("datos_outliers_varible/" + nombre_archivo, index=False)
    print(f"Guardado: {nombre_archivo}")
```

1.0.2 Buscando avanchas en datos para variable climatica

[86]:
```
dataframe_outliers=[]
for us, ot in enumerate(list(variables_contaminacion.keys())):
    try:
        data_frame_csv = pd.read_csv(
            "datos_outliers_varible/" + ot + ".csv", dtype=str
        )
    except:
        print("El archivo de datos "+ot+" no existe")
        continue

    data_frame_csv["Fecha_horas"] = data_frame_csv["Fecha_horas"].astype(float)
    dataframe_outliers.append(data_frame_csv)
```

1.0.3 binarizando datos climaticos

[ ]:
```
conjunto_datos_outliers = pd.read_csv("datos_outliers.csv")
grupos_temporales = conjunto_datos_outliers.groupby("Código del municipio")
events_temp = grupos_temporales.describe()
df_describ = events_temp.reset_index()
lista_outliers = []
for id_mun in list(df_describ["Código del municipio"]):
    grupo_municipio_outlirs = grupos_temporales.get_group(id_mun)
```

```
    lista_outliers.append(list(grupo_municipio_outlirs["Fecha_horas"]))
```

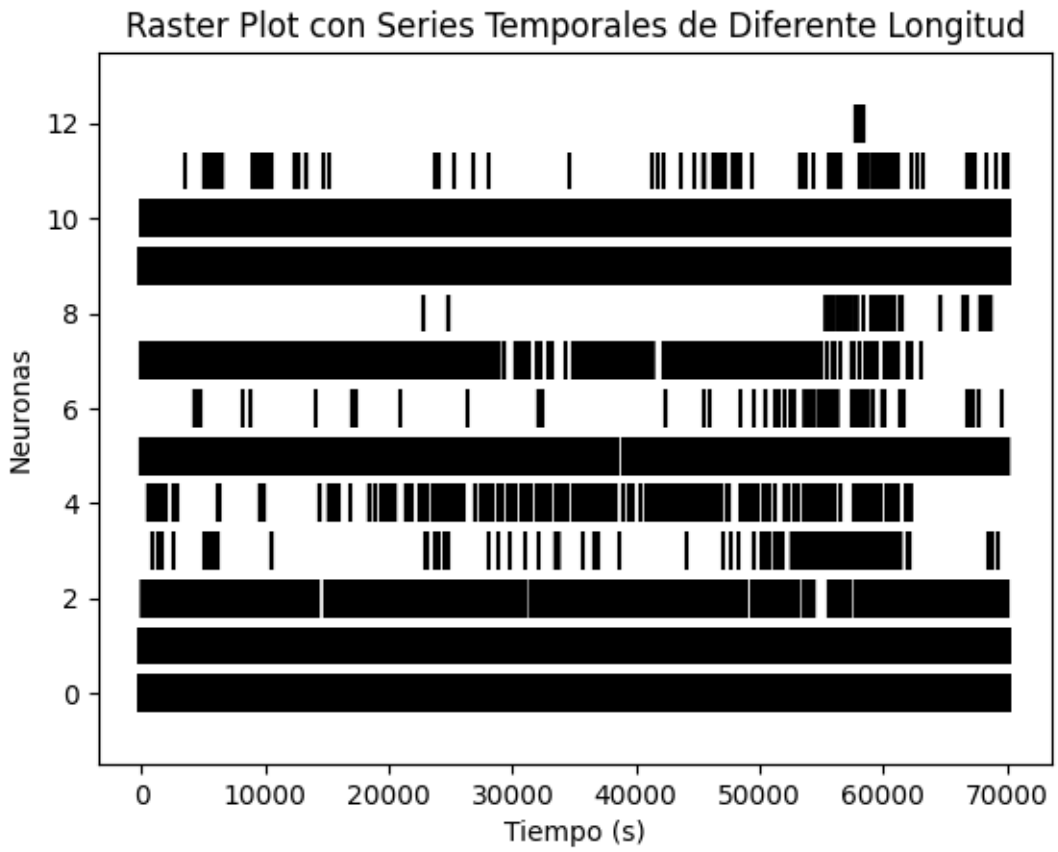
[ ]:
```
bins_size = 3  # horas
outliers_concatenate = []
for idmun in range(len(lista_outliers)):
    outliers_concatenate.extend(lista_outliers[idmun])
outliers_concatenate_sort = np.sort(outliers_concatenate)
intervalo = int(
    (outliers_concatenate_sort[-1] - outliers_concatenate_sort[0]) / bins_size
)
outliers_array_binary = np.zeros((len(lista_outliers), (intervalo + 1)))
```

[ ]:
```
for i in range(len(lista_outliers)):
    print("Binarization %s" % i)
    for j in range(intervalo + 1):
        outlist=np.array(lista_outliers[i])
        bins_size_min=j*bins_size+outliers_concatenate_sort[0]
        bins_size_max = (j + 1) * bins_size + outliers_concatenate_sort[0]
        count_outliers = outlist[(outlist >= bins_size_min) & (outlist <=␣
 ↪bins_size_max)]
        if len(count_outliers) >= 1:
            outliers_array_binary[i, j] = 1
```

[98]:
```
numcols = ["col" + str(int(u)) for u in range(len(outliers_array_binary[0,:]))]
mpanda=pd.DataFrame(outliers_array_binary,columns=numcols)
mpanda.to_csv("outliers_array_binary.csv",index=False)
```

1.0.4 Analisando raster plot en datos climaticos

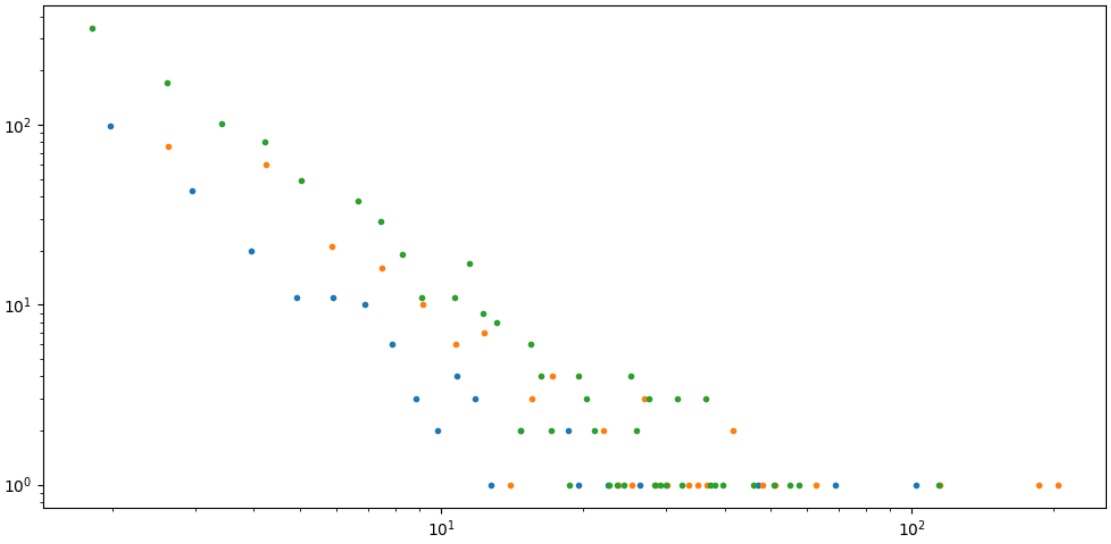
[18]:
```
plt.eventplot(dataframe_outliers, linelengths=0.8, color="black")

# Etiquetas y formato
plt.xlabel("Tiempo (s)")
plt.ylabel("Neuronas")
plt.title("Raster Plot con Series Temporales de Diferente Longitud")

plt.show()
```



1.0.5 Calcular tamaños de avalancha

[101]:
```
def size_avalancha_outliers(samples):
    bin_size = 6
    step = np.arange(samples[0], samples[-1], bin_size)
    spk_count, edges = np.histogram(samples, bins=step)
    I = np.argwhere(spk_count == 0).T[0]

    somas = []
    for i in range(0, len(I) - 1):
        somaX = np.sum(spk_count[I[i] : I[i + 1]])
        somas.append(somaX)
        # somas=np.concatenate((somas,somaX))
    somas = np.array(somas, dtype=float)
    size = somas[somas != 0]

    return size
```

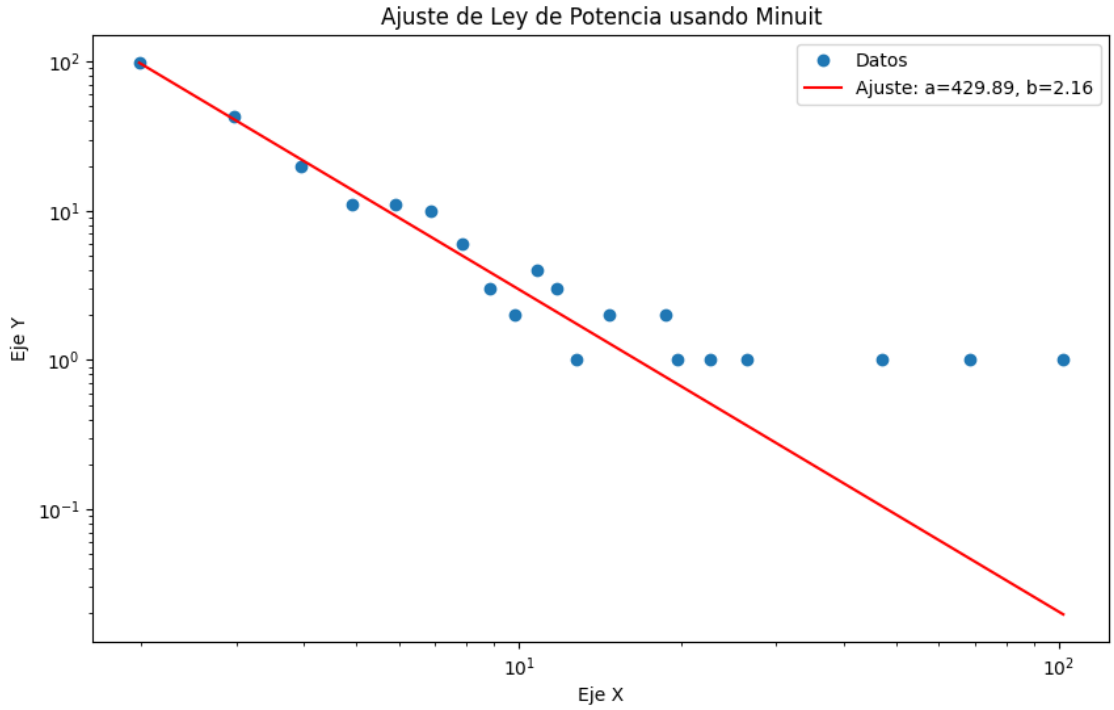
[102]:
```
fig = plt.figure(figsize=(10, 5), tight_layout=True)
gs = gridspec.GridSpec(1, 1)
ax = fig.add_subplot(gs[0, 0])
array_avalanchas_size = []
for idmun_arr in dataframe_outliers:
    if len(idmun_arr)>2:

        k = len(
            np.histogram_bin_edges(
                size_avalancha_outliers(list(idmun_arr["Fecha_horas"])),␣
 ↪bins="auto"
            )
        )
        spk_count, edges = np.histogram(
            size_avalancha_outliers(list(idmun_arr["Fecha_horas"])), bins=k
        )
        ax.plot(edges[1:], spk_count, ".")
# plt.title(list(idmun_arr["Variable"])[0])
plt.yscale("log")
plt.xscale("log")
plt.show()
```



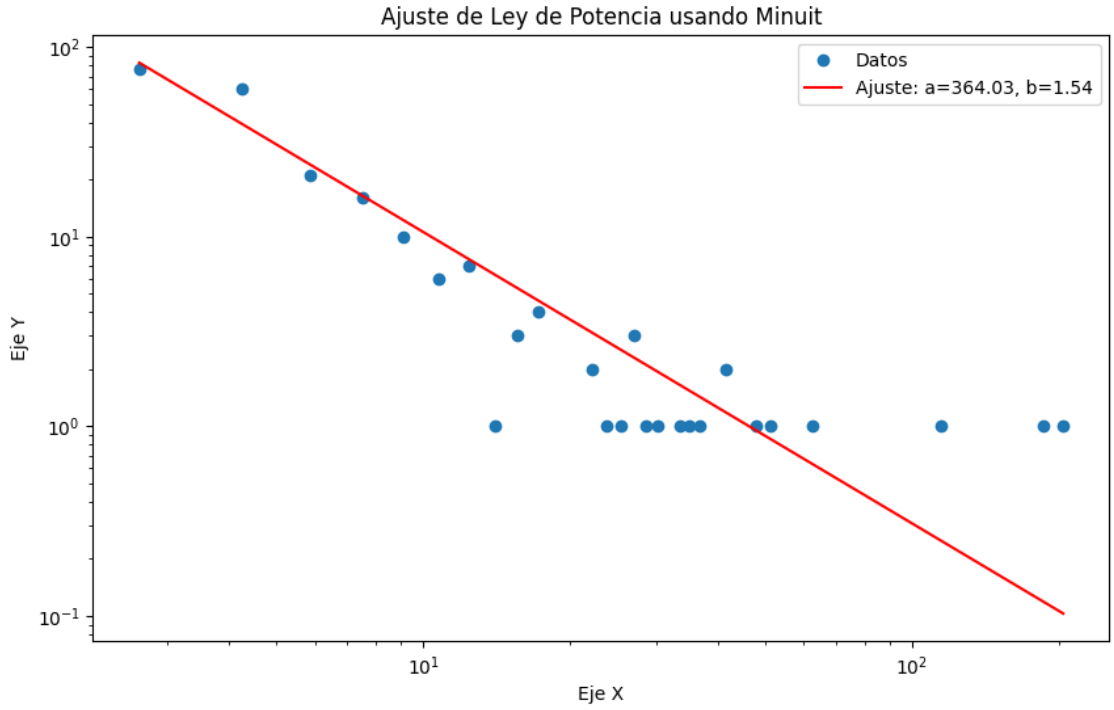
[103]:
```
def model(x, a, b):
    f = a * x**-b
    return f

for idmun_arr in dataframe_outliers:
```

```
    k = len(
    np.histogram_bin_edges(
            size_avalancha_outliers(list(idmun_arr["Fecha_horas"])), bins="auto"
        )
    )
    spk_count, edges = np.histogram(
        size_avalancha_outliers(list(idmun_arr["Fecha_horas"])), bins=k
    )
    ax.plot(edges[1:], spk_count, ".")
    x = np.array(list(edges[1:]), dtype=float)
    y = np.array(list(spk_count), dtype=float)
    yerr = np.array(list(len(spk_count) * [1]), dtype=float)

    # yerr = yerr[x < 10]
    # y=y[x<10]
    # x=x[x<10]

    chi2 = Chi2Regression(model, x, y, yerr)
    m = Minuit(chi2, a=0.4, b=1)
    m.migrad()

    a_fit = m.values["a"]
    b_fit = m.values["b"]
    print(f"Parámetros ajustados: a={a_fit:.2f}, b={b_fit:.2f}")

    # Graficar datos y ajuste
    plt.figure(figsize=(10, 6))
    plt.loglog(x, y, "o", label="Datos")
    plt.loglog(
        x, model(x, a_fit, b_fit), "r-", label=f"Ajuste: a={a_fit:.2f},␣
↪b={b_fit:.2f}"
    )
    plt.xlabel("Eje X")
    plt.ylabel("Eje Y")
    plt.title("Ajuste de Ley de Potencia usando Minuit")
    plt.legend()
    plt.show()
```

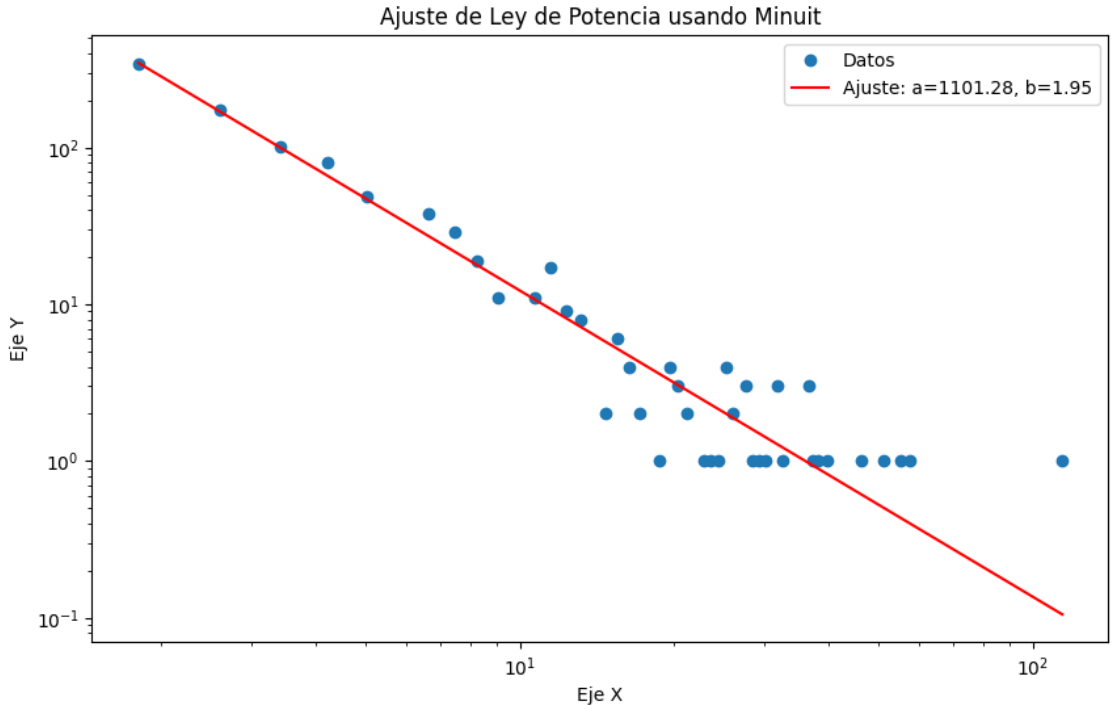
```
Parámetros ajustados: a=429.89, b=2.16
```



Parámetros ajustados: a=364.03, b=1.54



```
Parámetros ajustados: a=1101.28, b=1.95
```



2 Análisis de datos por id municipal

[4]:
```
df_unique = pd.read_csv("datos_segmentados/minicipios_unicos.csv")
```

[ ]:
```
for us, ot in enumerate(list(variables_contaminacion.keys())):
    if ot!="pm10_concentracion" and ot!="velocidad_viento":
        fig = plt.figure(figsize=(35, 10), tight_layout=True)
    else:
        fig = plt.figure(figsize=(50, 10), tight_layout=True)

    gs = gridspec.GridSpec(1, 1)
    ax = fig.add_subplot(gs[0, 0])
    data_frame_subconjunto = pd.read_csv("datos_segmentados/" + ot + ".csv",␣
 ↪dtype=str)

    data_frame_subconjunto = data_frame_subconjunto.dropna(axis=0,␣
 ↪subset=["Concentración"])
    data_frame_subconjunto["Concentración"] =␣
 ↪data_frame_subconjunto["Concentración"].str.replace(",","")
    data_frame_subconjunto["Concentración"] =␣
 ↪data_frame_subconjunto["Concentración"].astype(float)
```

```
    Q1 = data_frame_subconjunto["Concentración"].quantile(0.25)
    Q3 = data_frame_subconjunto["Concentración"].quantile(0.75)
    iqr= Q3 - Q1

    lower_limit = Q1 - 1.5 * iqr
    upper_limit = Q3 + 1.5 * iqr

    outliers = data_frame_subconjunto[
        (data_frame_subconjunto["Concentración"] < lower_limit) |␣
↪(data_frame_subconjunto["Concentración"] > upper_limit)
    ]

    dfsort = outliers.groupby("Código del municipio")
    grupos_acumulados = {}
    for nombre_grupo, grupo in dfsort:
        if nombre_grupo in grupos_acumulados:
            grupos_acumulados[nombre_grupo] = pd.concat(
                [grupos_acumulados[nombre_grupo], grupo]
            )
        else:
            grupos_acumulados[nombre_grupo] = grupo

    dfgacum = pd.concat(grupos_acumulados.values())
    dfgracum = dfgacum.drop_duplicates(subset="Código del municipio")
    asxc = dfsort["Concentración"].describe()
    df_describ = asxc.reset_index()
    df_describ.insert(
        1, "Nombre del municipio", list(dfgracum["Nombre del municipio"])
    )

    sqrcont = np.sqrt(np.array(list(df_describ["count"].to_numpy())))
    x = df_describ["Nombre del municipio"].to_numpy()
    y = df_describ["count"].to_numpy()
    error = df_describ["std"].to_numpy() / sqrcont

    ax.bar(x=x, height=y, capsize=10, color="#619cff", ecolor="r")
    unidad_variable = variables_contaminacion[ot]["unidad"]
    nombre_variable = variables_contaminacion[ot]["variable"]
    ax.xaxis.set_tick_params(labelsize=17)
    ax.yaxis.set_tick_params(labelsize=25)
    ax.set_xlabel(f"Municpios de Colombia", fontsize=35)
    ax.set_ylabel(f"{nombre_variable} ({unidad_variable})", fontsize=35)
    ax.set_title(
        f"Promedio muestral de datos {nombre_variable} ({unidad_variable}) por ␣
↪municipio",
        fontsize=45,
```

```
    )
    plt.setp(ax.get_xticklabels(), rotation=45, ha="right",␣
 ↪rotation_mode="anchor")
    fig.savefig(f"img_media_municipio/{ot}.pdf")
    plt.show()
```

```
El kernel se bloqueó al ejecutar código en la celda actual o en una celda␣
 ↪anterior.

Revise el código de las celdas para identificar una posible causa del error.

Haga clic <a href='https://aka.ms/vscodeJupyterKernelCrash'>aquí</a> para␣
 ↪obtener más información.

Vea Jupyter <a href='command:jupyter.viewOutput'>log</a> para obtener más␣
 ↪detalles.
```

[45]:
```
data_frame_subconjunto.info()
# datf.isnull().sum()
# np.array(df_unique)[io, jo]
# np.array(list(np.array(df_unique)[:, 0]))

# np.array(list(np.array(df_unique)[:, 1]))[io, jo]
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 42333 entries, 5 to 950529
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Fecha                           42333 non-null  object
 1   Autoridad Ambiental             42333 non-null  object
 2   Nombre de la estación           42333 non-null  object
 3   Tecnología                      39184 non-null  object
 4   Latitud                         42333 non-null  object
 5   Longitud                        42333 non-null  object
 6   Código del departamento         42333 non-null  object
 7   Departamento                    42333 non-null  object
 8   Código del municipio            42333 non-null  object
 9   Nombre del municipio            42333 non-null  object
 10  Tipo de estación                42333 non-null  object
 11  Tiempo de exposición            42333 non-null  object
 12  Variable                        42333 non-null  object
 13  Unidades                        42333 non-null  object
 14  Concentración                   42333 non-null  float64
 15  Nueva columna georreferenciada  42333 non-null  object
dtypes: float64(1), object(15)
```

```
memory usage: 5.5+ MB
```

[ ]:

[10]:
```
!jupyter nbconvert --to pdf datosaire.ipynb
# k!pip install jupyter
```

```
Traceback (most recent call last):
  File "/home/alejandro/.pyenv/versions/djworkspace/bin/jupyter-nbconvert", line
5, in <module>
    from nbconvert.nbconvertapp import main
  File "/home/alejandro/.pyenv/versions/djworkspace/lib/python3.8/site-
packages/nbconvert/nbconvertapp.py", line 193, in <module>
    class NbConvertApp(JupyterApp):
  File "/home/alejandro/.pyenv/versions/djworkspace/lib/python3.8/site-
packages/nbconvert/nbconvertapp.py", line 252, in NbConvertApp
    Options include {get_export_names()}.
  File "/home/alejandro/.pyenv/versions/djworkspace/lib/python3.8/site-
packages/nbconvert/exporters/base.py", line 145, in get_export_names
    e = get_exporter(exporter_name)(config=config)
  File "/home/alejandro/.pyenv/versions/djworkspace/lib/python3.8/site-
packages/nbconvert/exporters/base.py", line 106, in get_exporter
    exporter = items[0].load()
  File "/home/alejandro/.pyenv/versions/djworkspace/lib/python3.8/site-
packages/importlib_metadata/__init__.py", line 184, in load
    module = import_module(match.group('module'))
  File "/usr/lib/python3.8/importlib/__init__.py", line 127, in import_module
    return _bootstrap._gcd_import(name[level:], package, level)
  File "/home/alejandro/.pyenv/versions/djworkspace/lib/python3.8/site-
packages/jupyter_contrib_nbextensions/nbconvert_support/__init__.py", line 5, in
<module>
    from .collapsible_headings import ExporterCollapsibleHeadings
  File "/home/alejandro/.pyenv/versions/djworkspace/lib/python3.8/site-packages/
jupyter_contrib_nbextensions/nbconvert_support/collapsible_headings.py", line 6,
in <module>
    from notebook.services.config import ConfigManager
ModuleNotFoundError: No module named 'notebook.services'
```