

PETE219

Project Report

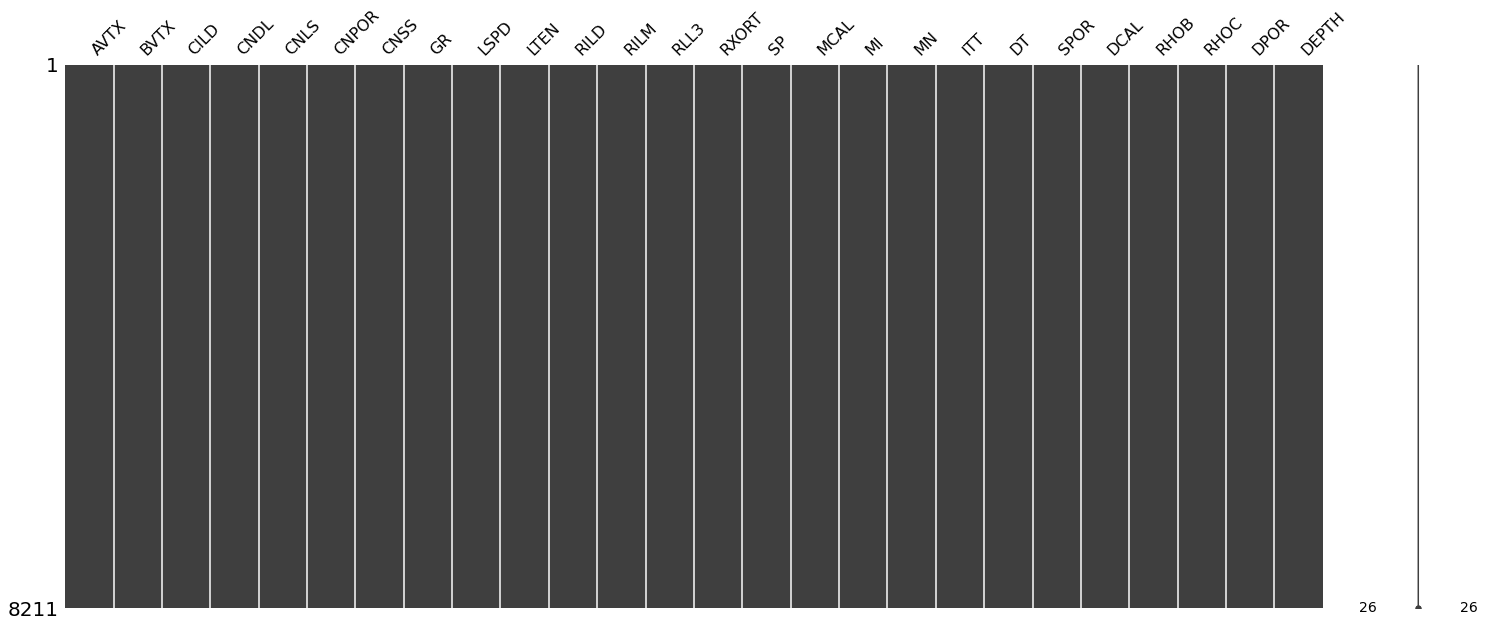
**Group 4**

**Data Cleaning, Visualization, and Analysis**

**Date Submitted**: Sunday November 6, 2022

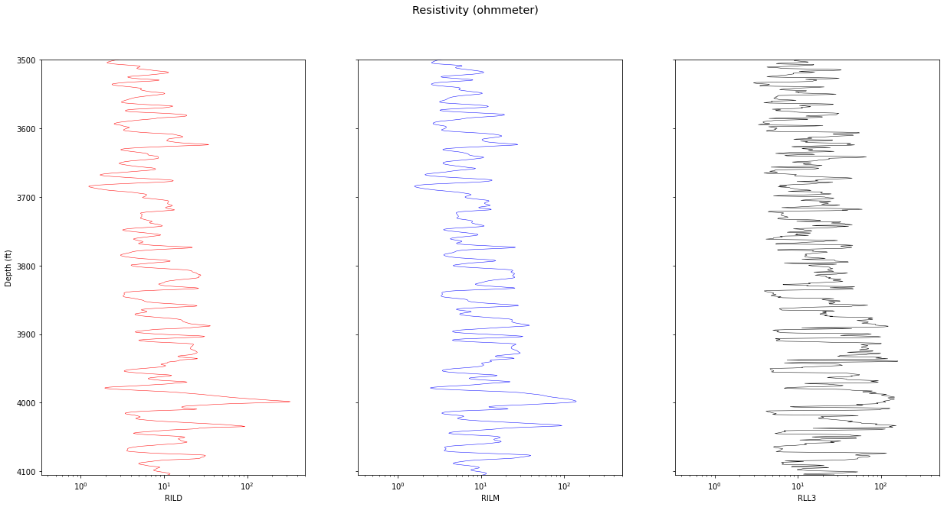
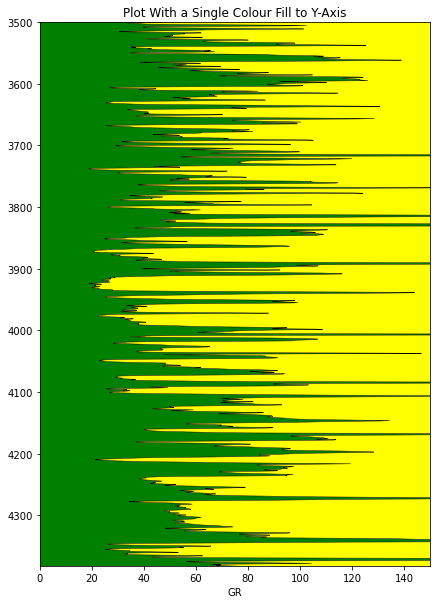
**Student Name**: Ahmad Al-Areeki, Omar Belal Al-Khateeb

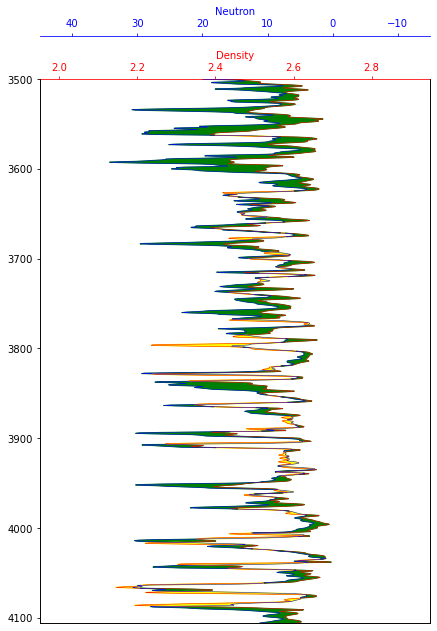
**Group 4:**

The data has been imported as a lasio file, it was visualized as a matrix using missingno. The data has then been cleaned using missingno and numpy library by converting values out of the range as nan values. For CNDL, values larger than 70 or less than zero have been assigned as nan. For GR, values less than zero or greater than 500 have been converted to nan. For CNLS, values less than zero have been converted to nan. For RILD, larger than 2500. For RILM larger than 8500. For DPOR, RHOB, RHOC, SPOR, DT, ML, MI, MCAL negative values have been assigned as nan. However, for DPOR, RHOB and RHOC values larger than 90, 5 as well as 0 have been considered as nan the types of data respectively. The result of this cleaning gave a clean matrix as displayed below:

*Figure 1: Shows the mssingno matrix (Appendix I)*

Initially, the data has been plotted using matplotlib.pyplot, the data has then been replotted using a column semi-log plot using the pandas library as well. The plots are shown below:

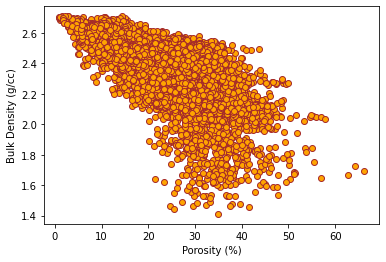




*Figure 2,3,4: Illustrates the Gamma Ray, Resistivity, and Neutron Porosity logs individually (Appendix II)*

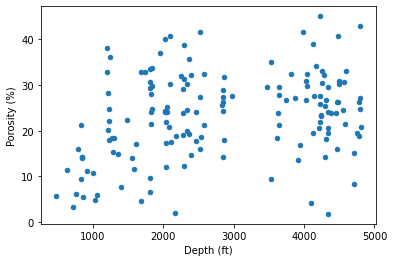
These logs estimated a reservoir thickness of 403ft for the GR log, and 118ft for the Resistivity logs. Both of which are estimates, but the resistivity log is more reliable as it has three depths and shows gas and oil more clearly.

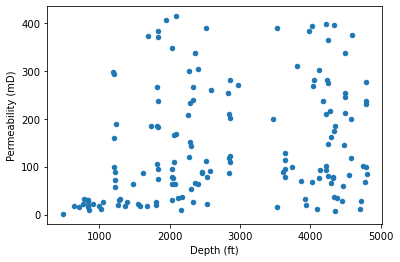
Porosity has also been plotted against density using CNPOR and RHOB respectively. This showed a very wide range of values but it was quite expected as the logs measured in 0.5ft increments.



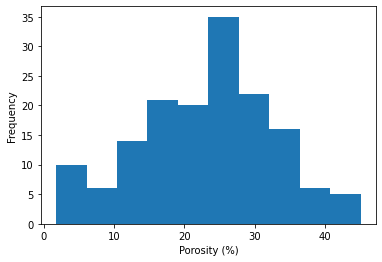
*Figure 5: Displays the Porosity against Bulk Density (Appendix III)*

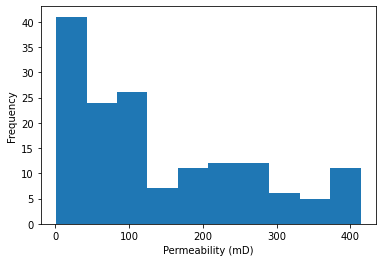
Following, regression graphs have then been plotted for the porosity-permeability data. Firstly, the data was cleaned by removing the negative values. Next, scatter graphs have been plotted for porosity and permeability including histograms to discover the correlation between the data.





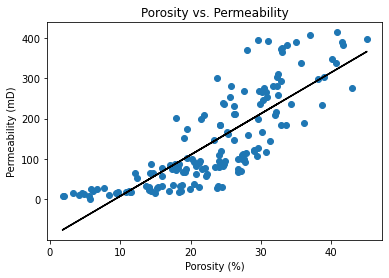
*Figure 6, 7: Displays the Porosity and Permeability against Depth (Appendix III)*





*Figure 8,9: Displays the Porosity and Permeability against their frequencies (Appendix III)*

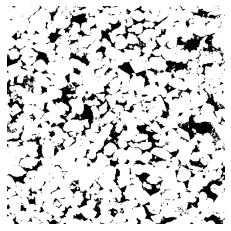
Porosity was normally distributed by the histogram, while permeability was skewed to the right. These can be somewhat seen in the scatter plots as well. A PP plot was made showing the relation between porosity and permeability, this was also split into the four facies:



*Figure 10: Shows the linear relationship between porosity and permeability(Appendix III)*

Linear regression has been used to discover the correlation between porosity and permeability in addition to calculating R^2 as a scale. Linear regression was also used for the facies channel, crevasse splay and overbanks of the porosity and permeability data.

Also, skimage was used to show the image of the thin section from the reservoir, the image was corrected and the borders were excluded in the calculation of the section, and the porosity was 20.22%. The image is shown below:



*Figure 11: Displays the picture of the core sample (Appendix IIII)*

We can see from the data that we have visualized that we might have a potential reservoir. From the logs, the potential reservoir could be from around the 3000ft to 4100ft. This range is quite large because the reservoir can be an unclean sandy type, which would give this effect. Also, the neutron density logs show that this could be an oil reservoir. Moreover, the poro-perm values look good for a reservoir as well, from the image it is seen as 20% which is acceptable for an oil reservoir. The correlations made all make sense and this seems like a good quality of a reservoir.

**Appendix I:**

############ IMP0RT LASI0 FILE ############

import lasio as las #pip install lasio

data = las.read('1051661435.las')

data = data.df()

############ CLEANING DATA ############

import numpy as np

import missingno as msno #pip install missingno

msno.matrix(data)

data['CNDL'][data['CNDL'] < 0] = np.nan

data['CNDL'][data['CNDL'] > 70] = np.nan

data['CNLS'][data['CNLS'] < 0] = np.nan

data['CNPOR'][data['CNPOR'] < 0] = np.nan

data['GR'][data['GR'] > 500] = np.nan

data['GR'][data['GR'] < 0] = np.nan

data['RILD'][data['RILD'] > 2500] = np.nan

data['RILM'][data['RILM'] > 8500] = np.nan

data['MCAL'][data['MCAL'] < 0] = np.nan

data['MI'][data['MI'] < 0] = np.nan

data['MN'][data['MN'] < 0] = np.nan

data['DT'][data['DT'] < 0] = np.nan

data['SPOR'][data['SPOR'] < 0] = np.nan

data['RHOB'][data['RHOB'] < 0] = np.nan

data['RHOB'][data['RHOB'] > 5] = np.nan

data['RHOC'][data['RHOC'] <= 0] = np.nan

data['DPOR'][data['DPOR'] < 0] = np.nan

data['DPOR'][data['DPOR'] > 90] = np.nan

data.index.names = ['DEPTH']

data['DEPTH'] = data.index

msno.matrix(data)

**Appendix II:**

############ VISUALIZAING DATA ############

import matplotlib.pyplot as plt

data.plot(x='GR', y='DEPTH', c='black', lw=0.5, legend=False, figsize=(7,10))

plt.ylim(8764/2, 7000/2)

plt.xlim(0,150)

plt.title('Plot With a Single Colour Fill to Y-Axis')

plt.fill\_betweenx(data['DEPTH'], data['GR'], 0, facecolor='green')

plt.fill\_betweenx(data['DEPTH'], data['GR'], 150, facecolor='yellow')

plt.show()

print('Thickness of reservoir is', (data['GR'] < 40).sum()\*0.5, 'ft')

## [OUT] : 403.0ft

fig\_res, axes = plt.subplots(1,3, sharex=True, sharey=True, figsize=(21,10))

data.plot(x='RILD', y='DEPTH', ax=axes[0], legend=False, logx=True, lw=0.5, c='red')

data.plot(x='RILM', y='DEPTH',ax=axes[1], legend=False, logx=True, lw=0.5, c ='blue')

data.plot(x='RLL3', y='DEPTH',ax=axes[2], legend=False, logx=True, lw=0.5, c='black')

axes[0].set\_xlim(0,500)

axes[0].set\_ylim(8211/2, 7000/2)

fig\_res.suptitle('Resistivity (ohmmeter)', fontsize='x-large')

axes[0].set\_ylabel('Depth (ft)')

print('Thickness of reservoir is', (data.loc[(data['GR'] < 40),'RILM'] > 15).sum()\*0.5, 'ft')

## [OUT] : 118.0ft

fig = plt.subplots(figsize=(7,10))

ax1 = plt.subplot2grid((1,1), (0,0), rowspan=1, colspan=1)

ax2 = ax1.twiny()

ax1.plot('RHOB', 'DEPTH', data=data, color='red', lw=0.5)

ax1.set\_xlim(1.95, 2.95)

ax1.set\_ylim(8211/2, 7000/2)

ax1.set\_xlabel('Density')

ax1.xaxis.label.set\_color("red")

ax1.tick\_params(axis='x', colors="red")

ax1.spines["top"].set\_edgecolor("red")

ax2.plot('CNPOR', 'DEPTH', data=data, color='blue', lw=0.5)

ax2.set\_xlim(45, -15)

ax2.set\_ylim(8211/2, 7000/2)

ax2.set\_xlabel('Neutron')

ax2.xaxis.label.set\_color("blue")

ax2.spines["top"].set\_position(("axes", 1.08))

ax2.tick\_params(axis='x', colors="blue")

ax2.spines["top"].set\_edgecolor("blue")

x1=data['RHOB']

x2=data['CNPOR']

x = np.array(ax1.get\_xlim())

z = np.array(ax2.get\_xlim())

nz=((x2-np.max(z))/(np.min(z)-np.max(z)))\*(np.max(x)-np.min(x))+np.min(x)

ax1.fill\_betweenx(data['DEPTH'], x1, nz, where=x1>=nz, interpolate=True, color='green')

ax1.fill\_betweenx(data['DEPTH'], x1, nz, where=x1<=nz, interpolate=True, color='yellow')

for ax in [ax1, ax2]:

ax.set\_ylim(8211/2, 7000/2)

ax.xaxis.set\_ticks\_position("top")

ax.xaxis.set\_label\_position("top")

########### ANOTHER WAY TO VISUALIZE DATA ################

def well\_log\_display(df, column\_depth, column\_list,

column\_semilog=None, min\_depth=None, max\_depth=None,

column\_min=None, column\_max=None, colors=None,

fm\_tops=None, fm\_depths=None,

tight\_layout=1, title\_size=10):

"""

Display log side-by-side style

Input:

df is your dataframe

specify min\_depth and max\_depth as the upper and lower depth limit

column\_depth is the column name of your depth

column\_list is the LIST of column names that you will display

column\_semilog is specific for resistivity column; if your resistivities are

in column 3, specify as: column\_semilog=2. Default is None, so if you don't

specify, the resistivity will be plotted in normal axis instead

column\_min is list of minimum values for the x-axes.

column\_max is list of maximum values for the x-axes.

colors is the list of colors specified for each log names. Default is None,

so if don't specify, the colors will be Matplotlib default (blue)

fm\_tops and fm\_depths are the list of formation top names and depths.

Default is None, so no tops are shown. Specify both lists, if you want

to show the tops

"""

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import random

if column\_semilog==None:

# column semilog not defined, RT will be plotted in normal axis

logs = column\_list

# create the subplots; ncols equals the number of logs

fig, ax = plt.subplots(nrows=1, ncols=len(logs), figsize=(20,10))

# looping each log to display in the subplots

if colors==None:

# color is None (default)

for i in range(len(logs)):

# normal axis plot

ax[i].plot(df[logs[i]], df[column\_depth])

ax[i].set\_title(logs[i], size=title\_size)

ax[i].minorticks\_on()

ax[i].grid(which='major', linestyle='-', linewidth='0.5', color='lime')

ax[i].grid(which='minor', linestyle=':', linewidth='0.5', color='black')

if column\_min!=None and column\_max!=None:

# x-axis limits defined

ax[i].set\_xlim(column\_min[i], column\_max[i])

if min\_depth!=None and max\_depth!=None:

# y-axis limit defined

ax[i].set\_ylim(min\_depth, max\_depth)

ax[i].invert\_yaxis()

else:

# colors are defined (as list)

for i in range(len(logs)):

# normal axis plot

ax[i].plot(df[logs[i]], df[column\_depth], color=colors[i])

ax[i].set\_title(logs[i], size=title\_size)

ax[i].minorticks\_on()

ax[i].grid(which='major', linestyle='-', linewidth='0.5', color='lime')

ax[i].grid(which='minor', linestyle=':', linewidth='0.5', color='black')

if column\_min!=None and column\_max!=None:

# x-axis limits defined

ax[i].set\_xlim(column\_min[i], column\_max[i])

if min\_depth!=None and max\_depth!=None:

# y-axis limit defined

ax[i].set\_ylim(min\_depth, max\_depth)

ax[i].invert\_yaxis()

else:

# column semilog is defined, RT will be plotted in semilog axis

logs = column\_list

# create the subplots; ncols equals the number of logs

fig, ax = plt.subplots(nrows=1, ncols=len(logs), figsize=(20,10))

# looping each log to display in the subplots

if colors==None:

# color is None (default)

for i in range(len(logs)):

if i == column\_semilog:

# for resistivity, semilog plot

ax[i].semilogx(df[logs[i]], df[column\_depth])

else:

# for non-resistivity, normal plot

ax[i].plot(df[logs[i]], df[column\_depth])

ax[i].set\_title(logs[i], size=title\_size)

ax[i].minorticks\_on()

ax[i].grid(which='major', linestyle='-', linewidth='0.5', color='lime')

ax[i].grid(which='minor', linestyle=':', linewidth='0.5', color='black')

if column\_min!=None and column\_max!=None:

# x-axis limits defined

ax[i].set\_xlim(column\_min[i], column\_max[i])

if min\_depth!=None and max\_depth!=None:

# y-axis limit defined

ax[i].set\_ylim(min\_depth, max\_depth)

ax[i].invert\_yaxis()

else:

# colors are defined (as list)

for i in range(len(logs)):

if i == column\_semilog:

# for resistivity, semilog plot

ax[i].semilogx(df[logs[i]], df[column\_depth], color=colors[i])

else:

# for non-resistivity, normal plot

ax[i].plot(df[logs[i]], df[column\_depth], color=colors[i])

ax[i].set\_title(logs[i], size=title\_size)

ax[i].minorticks\_on()

ax[i].grid(which='major', linestyle='-', linewidth='0.5', color='lime')

ax[i].grid(which='minor', linestyle=':', linewidth='0.5', color='black')

if column\_min!=None and column\_max!=None:

# x-axis limits defined

ax[i].set\_xlim(column\_min[i], column\_max[i])

if min\_depth!=None and max\_depth!=None:

# y-axis limit defined

ax[i].set\_ylim(min\_depth, max\_depth)

ax[i].invert\_yaxis()

if fm\_tops!=None and fm\_depths!=None:

# Formation tops and depths are specified, they will be shown

# produce colors

rgb = []

for j in range(len(fm\_tops)):

\_ = (random.random(), random.random(), random.random())

rgb.append(\_)

for i in range(len(logs)):

for j in range(len(fm\_tops)):

# rgb = (random.random(), random.random(), random.random())

ax[i].axhline(y=fm\_depths[j], linestyle=":", c=rgb[j], label=fm\_tops[j])

# y = fm\_depths[j] / (max\_depth - min\_depth)

# ax[i].text(0.5, y, fm\_tops[j], fontsize=5, va='center', ha='center', backgroundcolor='w')

# plt.legend()

# plt.legend(loc='upper center', bbox\_to\_anchor=(-3, -0.05),

# fancybox=True, shadow=True, ncol=5)

plt.show()

df = data

column\_depth = 'DEPTH'

column\_list = ['RHOB', 'DT', 'GR', 'CNPOR', 'RILM']

column\_semilog = 6

column\_min=None

column\_max=None

min\_depth= 3500

max\_depth= 4150

colors=["black","blue", "orange", "red","grey","green","yellow"]

well\_log\_display(df, column\_depth, column\_list, column\_semilog, min\_depth, max\_depth, column\_min, column\_max, colors)

def triple\_combo(df, column\_depth, column\_GR, column\_resistivity,

column\_NPHI, column\_RHOB, min\_depth, max\_depth,

min\_GR=0, max\_GR=150, sand\_GR\_line=60,

min\_resistivity=0.01, max\_resistivity=1000,

color\_GR='black', color\_resistivity='green',

color\_RHOB='red', color\_NPHI='blue',

figsize=(6,10), tight\_layout=1,

title\_size=15, title\_height=1.05):

"""

Producing Triple Combo log

Input:

df is your dataframe

column\_depth, column\_GR, column\_resistivity, column\_NPHI, column\_RHOB

are column names that appear in your dataframe (originally from the LAS file)

specify your depth limits; min\_depth and max\_depth

input variables other than above are default. You can specify

the values yourselves.

Output:

Fill colors; gold (sand), lime green (non-sand), blue (water-zone), orange (HC-zone)

"""

import matplotlib.pyplot as plt

from matplotlib.ticker import AutoMinorLocator

import numpy as np

fig, ax=plt.subplots(1,3,figsize=(8,10))

fig.suptitle('Triple Combo Log', size=title\_size, y=title\_height)

ax[0].minorticks\_on()

ax[0].grid(which='major', linestyle='-', linewidth='0.5', color='lime')

ax[0].grid(which='minor', linestyle=':', linewidth='1', color='black')

ax[1].minorticks\_on()

ax[1].grid(which='major', linestyle='-', linewidth='0.5', color='lime')

ax[1].grid(which='minor', linestyle=':', linewidth='1', color='black')

ax[2].minorticks\_on()

ax[2].grid(which='major', linestyle='-', linewidth='0.5', color='lime')

ax[2].grid(which='minor', linestyle=':', linewidth='1', color='black')

# First track: GR

ax[0].get\_xaxis().set\_visible(False)

ax[0].invert\_yaxis()

gr=ax[0].twiny()

gr.set\_xlim(min\_GR,max\_GR)

gr.set\_xlabel('GR',color=color\_GR)

gr.set\_ylim(max\_depth, min\_depth)

gr.spines['top'].set\_position(('outward',10))

gr.tick\_params(axis='x',colors=color\_GR)

gr.plot(df[column\_GR], df[column\_depth], color=color\_GR)

gr.minorticks\_on()

gr.xaxis.grid(which='major', linestyle='-', linewidth='0.5', color='lime')

gr.xaxis.grid(which='minor', linestyle=':', linewidth='1', color='black')

gr.fill\_betweenx(df[column\_depth], sand\_GR\_line, df[column\_GR], where=(sand\_GR\_line>=df[column\_GR]), color = 'gold', linewidth=0) # sand

gr.fill\_betweenx(df[column\_depth], sand\_GR\_line, df[column\_GR], where=(sand\_GR\_line<df[column\_GR]), color = 'lime', linewidth=0) # shale

# Second track: Resistivity

ax[1].get\_xaxis().set\_visible(False)

ax[1].invert\_yaxis()

res=ax[1].twiny()

res.set\_xlim(min\_resistivity,max\_resistivity)

res.set\_xlabel('Resistivity',color=color\_resistivity)

res.set\_ylim(max\_depth, min\_depth)

res.spines['top'].set\_position(('outward',10))

res.tick\_params(axis='x',colors=color\_resistivity)

res.semilogx(df[column\_resistivity], df[column\_depth], color=color\_resistivity)

res.minorticks\_on()

res.xaxis.grid(which='major', linestyle='-', linewidth='0.5', color='lime')

res.xaxis.grid(which='minor', linestyle=':', linewidth='1', color='black')

# Third track: NPHI and RHOB

ax[2].get\_xaxis().set\_visible(False)

ax[2].invert\_yaxis()

## NPHI curve

nphi=ax[2].twiny()

nphi.set\_xlim(-0.15,0.45)

nphi.invert\_xaxis()

nphi.set\_xlabel('NPHI',color='blue')

nphi.set\_ylim(max\_depth, min\_depth)

nphi.spines['top'].set\_position(('outward',10))

nphi.tick\_params(axis='x',colors='blue')

nphi.plot(df[column\_NPHI], df[column\_depth], color=color\_NPHI)

nphi.minorticks\_on()

nphi.xaxis.grid(which='major', linestyle='-', linewidth='0.5', color='lime')

nphi.xaxis.grid(which='minor', linestyle=':', linewidth='1', color='black')

## RHOB curve

rhob=ax[2].twiny()

rhob.set\_xlim(1.95,2.95)

rhob.set\_xlabel('RHOB',color='red')

rhob.set\_ylim(max\_depth, min\_depth)

rhob.spines['top'].set\_position(('outward',50))

rhob.tick\_params(axis='x',colors='red')

rhob.plot(df[column\_RHOB], df[column\_depth], color=color\_RHOB)

# solution to produce fill between can be found here:

# https://stackoverflow.com/questions/57766457/how-to-plot-fill-betweenx-to-fill-the-area-between-y1-and-y2-with-different-scal

x2p, \_ = (rhob.transData + nphi.transData.inverted()).transform(np.c\_[df[column\_RHOB], df[column\_depth]]).T

nphi.autoscale(False)

nphi.fill\_betweenx(df[column\_depth], df[column\_NPHI], x2p, color="orange", alpha=0.4, where=(x2p > df[column\_NPHI])) # hydrocarbon

nphi.fill\_betweenx(df[column\_depth], df[column\_NPHI], x2p, color="blue", alpha=0.4, where=(x2p < df[column\_NPHI])) # water

res.minorticks\_on()

res.grid(which='major', linestyle='-', linewidth='0.5', color='lime')

res.grid(which='minor', linestyle=':', linewidth='1', color='black')

plt.show()

df = data

column\_depth = 'DEPTH'

column\_GR = 'GR'

column\_resistivity = 'RILM'

column\_NPHI = 'CNPOR'

column\_RHOB = 'RHOB'

min\_depth= 3500

max\_depth= 4200

triple\_combo(df, column\_depth, column\_GR, column\_resistivity, column\_NPHI, column\_RHOB, min\_depth, max\_depth)

**Appendix III:**

Density = data['RHOB']

Density = Density.to\_numpy()

Porosity = data['CNPOR']

Porosity = Porosity.to\_numpy()

fig = plt.figure()

ax1 = fig.add\_subplot(111)

ax1.scatter(Porosity, Density, color = 'orange', edgecolors = 'brown')

plt.xlabel("Porosity (%)")

plt.ylabel("Bulk Density (g/cc)")

plt.show()

############ REGRESSION ############

import matplotlib.pyplot as plt

import pandas as pd

poro = pd.read\_csv('poro\_perm\_data.csv')

poro = poro.dropna()

dataclean = (poro["Permeability (mD)"] > 0) & (poro['Porosity (%)'] > 0)

poro = poro[dataclean]

poroplot = poro.plot('Depth (ft)','Porosity (%)', kind = 'scatter')

permplot = poro.plot('Depth (ft)','Permeability (mD)',kind = 'scatter')

poro['Porosity (%)'].plot(kind='hist')

plt.ylabel('Frequency')

plt.xlabel('Porosity (%)')

plt.show()

poro['Permeability (mD)'].plot(kind='hist')

plt.ylabel('Frequency')

plt.xlabel('Permeability (mD)')

plt.show()

from sklearn.linear\_model import LinearRegression

porox=poro.loc[:,'Porosity (%)'].to\_numpy().reshape(-1,1)

permy=poro.loc[:,'Permeability (mD)'].to\_numpy().reshape(-1,1)

fig\_reg, ax = plt.subplots()

ax.scatter(porox,permy)

ax.set\_xlabel('Porosity (%)')

ax.set\_ylabel('Permeability (mD)')

ax.set\_title('Porosity vs. Permeability')

LR = LinearRegression()

LR.fit(porox, permy)

r\_sq = LR.score(porox, permy)

y\_pred = LR.predict(porox)

ax.plot(porox,y\_pred, color="black")

chfacies=poro.loc[:,'Facies']=="'channel'"

chfacies=poro[chfacies]

poroch=chfacies.loc[:,'Porosity (%)'].to\_numpy().reshape(-1,1)

permch=chfacies.loc[:,'Permeability (mD)'].to\_numpy().reshape(-1,1)

plotch, ax = plt.subplots()

ax.scatter(poroch,permch)

ax.set\_xlabel('Porosity (%)')

ax.set\_ylabel('Permeability (mD)')

ax.set\_title('Channels')

model = LinearRegression()

model.fit(poroch, permch)

r2ch = model.score(poroch, permch)

ypredch = model.predict(poroch)

ax.plot(poroch,ypredch, color="black")

csfacies=poro.loc[:,'Facies']=="'crevasse splay'"

csfacies=poro[csfacies]

porocs=csfacies.loc[:,'Porosity (%)'].to\_numpy().reshape(-1,1)

permcs=csfacies.loc[:,'Permeability (mD)'].to\_numpy().reshape(-1,1)

plotcs, ax = plt.subplots()

ax.scatter(porocs,permcs)

ax.set\_xlabel('Porosity (%)')

ax.set\_ylabel('Permeability (mD)')

ax.set\_title('Crevasse Splay')

model = LinearRegression()

model.fit(porocs, permcs)

r2cs = model.score(porocs, permcs)

ypredcs = model.predict(porocs)

ax.plot(porocs,ypredcs, color="black")

obfacies=poro.loc[:,'Facies']=="'overbanks'"

obfacies=poro[obfacies]

obfacies=poro.loc[:,'Facies']=="'overbanks'"

obfacies=poro[obfacies]

poroob=obfacies.loc[:,'Porosity (%)'].to\_numpy().reshape(-1,1)

permob=obfacies.loc[:,'Permeability (mD)'].to\_numpy().reshape(-1,1)

plotob, ax = plt.subplots()

ax.scatter(poroob,permob)

ax.set\_xlabel('Porosity (%)')

ax.set\_ylabel('Permeability (mD)')

ax.set\_title('Overbanks')

model = LinearRegression()

model.fit(poroob, permob)

r2ob = model.score(poroob, permob)

ypredob = model.predict(poroob)

ax.plot(poroob,ypredob, color="black")

**Appendix IIII:**

#################### PICTURE ####################

from skimage import io

import matplotlib.pyplot as plt

import numpy as np

img = io.imread('berea8bit.tif')

plt.imshow(img,cmap='gray')

plt.axis('off')

plt.show()

imarray = np.array(img)

imV = imarray.reshape((500\*500, 1))

plt.hist(imV, density=True, bins=30, range=[0,255])

plt.title('Porosity Histogram')

plt.show()

BW = imarray

BW[BW<100] = 0

BW[BW>=100] = 255

BW2 = BW.astype(np.bool)

BW2 = np.array(BW2)

plt.imshow(imarray, cmap='Greys\_r')

plt.axis('off')

area = np.size(BW2)

fw = np.sum(BW2)

print('Porosity of the thin section:', (1 - (fw/area))\*100)