Inter-Bedding Frequency (IBF) and Inter-Bedding Brittleness Index (IBBI)

Dicman Alfred

Unconventional petrophysics is a tale of two cities – Reservoir and Completions. Reservoir quality is only one half of the story (maybe the better -half), the other half is the completion quality. "You produce from what you create" – This line sums up the production philosophy of unconventionals. The current completion quality computations in the industry are limited to brittleness index and minimum horizontal stress.

The problem is unlike reservoir quality, completion quality may not be an "at-depth" computation, which is the norm for petrophysics so far. Completion quality may depend on the rocks above and below the landing target. The stress changes can occur at every bed or interface change. When hydraulic fractures interact with these beds/interfaces there are 4 possible outcomes : crossing, arresting, opening and offsetting. So characterizing these interfaces becomes important.

How do we express this attribute? Not only the degree of interbedding but also the overall effect due to the inherent quality of the interbeds, because one could have brittle/brittle , brittle/ductile or ductile/ductile interfaces. And the big underlying question, at what scale should completion quality be described?

The major issue impeding this computation is the ability to detect beds or interface change from the current tools available in commercial petrophysical packages. From my experience, even if you get very creative, you are still not there. (Trust me, I have tried every trick in the book).

Thanks to the advent of Python, now we have access to tools from signal processing that would make this computation possible and accurate to the extent we require.

In this example, I am going to show a detailed procedure of how to compute this interbedding completion quality and I am going to compute this at the scale of a target window (usually 15/20 ft) and for a desired purpose (increased frac height), with an Eagle ford example. I am going to introduce this computation as ***Inter Bedding Brittleness Index (IBBI)***, Why IBBI? Because I thought it sounded cool and I called it first! 😊

The method I am proposing involves the computation of facies, which helps us group similar rocks and also eliminate data noise. The facies computation in this workflow is done with unsupervised methods. This is an integral part of this workflow.

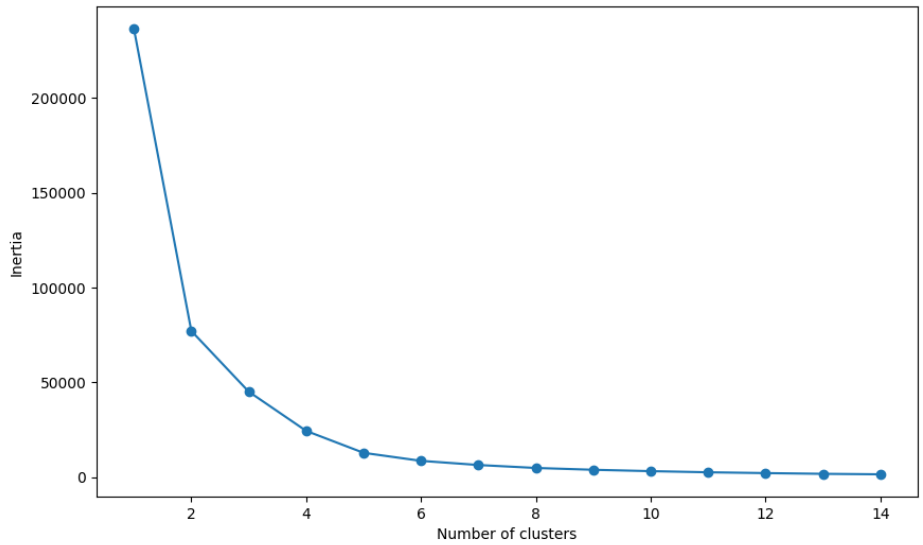
Let's start by importing the necessary python libraries and loading the well las file and limiting the depths to the zone of interest.

```
from welly import Well
from scipy import stats
import matplotlib.colors as colors
from sklearn.mixture import GaussianMixture
from sklearn.mixture import BayesianGaussianMixture
from sklearn.cluster import Birch
from sklearn.cluster import KMeans
import math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
pd.options.mode.chained_assignment = None


#Read Las file

W = Well.from_las('Well.las', index='f')
alias = {
      "GR" : ["GR","GRN","GRR"],
      "RT" : ["RDEEP","AT90","RT","ILD","RT90","RES90","RESD","M2R9"],
      "RHOB":["RHOB","RHO8","ZDEN"],
      "PHIN" : ["PHIN","NPHI","NLPS","NPOR","CNC"],
       "DT" : ["DTC","DTCO", "DT","DT24"],
       "DTS" : ["DTS","DTSM"],
       "VP": ["VP","VPM"],
}
# Logs to import
keys = ['GR','RT','RHOB','PHIN','DT','DTS','VPM']

df = W.df(keys=keys, alias=alias)
df.reset_index(inplace=True)

#Filtering data to the Desired Interval

top = 12180
bottom = 12350
logs = df.loc[(df['Depth'] >= top) & (df['Depth'] <= bottom)]
```

Now let's make facies. The whole exercise of making facies is to distinguish them geo-mechanically. Several people have different criteria to what that means, mine is very simple, their ***velocities*** have to be different. The next section of the code is to make unsupervised facies. First, we generate the "elbow" plot (aka, the inertia plot, google if you want to learn more about it) and then based on its findings we generate the desired number of facies through a model of our choice. In this exercise I am going to make facies from 3 logs, GR, PHIN and RHOB from Kmeans Clustering. There are several ways of making facies, so you are not bound by what I am showing here. I have also shown the other clustering techniques commonly used.

```
#Lets Make Facies
data = logs[['GR','RHOB','PHIN']]
# Making outliers  Null and Interpolating data
data = data.mask(data.sub(data.mean()).div(data.std()).abs().gt(2.5))
facies_data = data.interpolate()

# fitting multiple k-means algorithms and storing the values in an empty list
SSE = []
for cluster in range(1,15):
    kmeans = KMeans(n_jobs = -1, n_clusters = cluster, init='k-means++')
    kmeans.fit(facies_data)
    SSE.append(kmeans.inertia_)

frame = pd.DataFrame({'Cluster':range(1,15), 'SSE':SSE})
plt.figure(figsize=(10,6))
plt.plot(frame['Cluster'], frame['SSE'], marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.show()
```

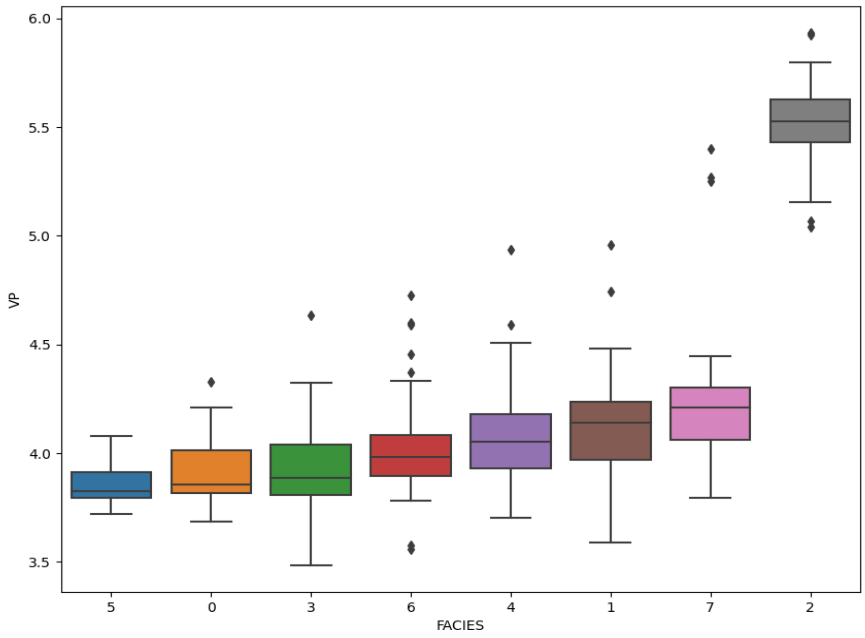
Let's see how an inertia plot looks.



The inertia plot for this well shows that 6 facies are enough to describe this well, but for our exercise we need the best possible description of beds, so we are going to choose 8. So, let's create these facies and visually insect their properties.

```
# Facies Models

log_input=logs[['GR','RHOB','PHIN']]

####Models###

model_gmm = GaussianMixture(n_components=8, covariance_type='diag', init_params='kmeans',
n_init=8, random_state=42)

model_gmb =BayesianGaussianMixture(n_components=8,covariance_type='tied',init_params='random',
n_init=3,  weight_concentration_prior_type='dirichlet_distribution', warm_start='True',
random_state=42)

model_kmeans = KMeans(init="k-means++", n_clusters=8, random_state=0)

##############################################################################
model = model_kmeans
labels = model.fit_predict(log_input)
logs['FACIES'] = labels
logs=logs.reset_index()

fig, ax = plt.subplots(figsize=(10,8))
order= logs.groupby('FACIES').median()['VP'].sort_values().index
sns.boxplot(x=logs.FACIES,y=logs.VP, order=order)
```

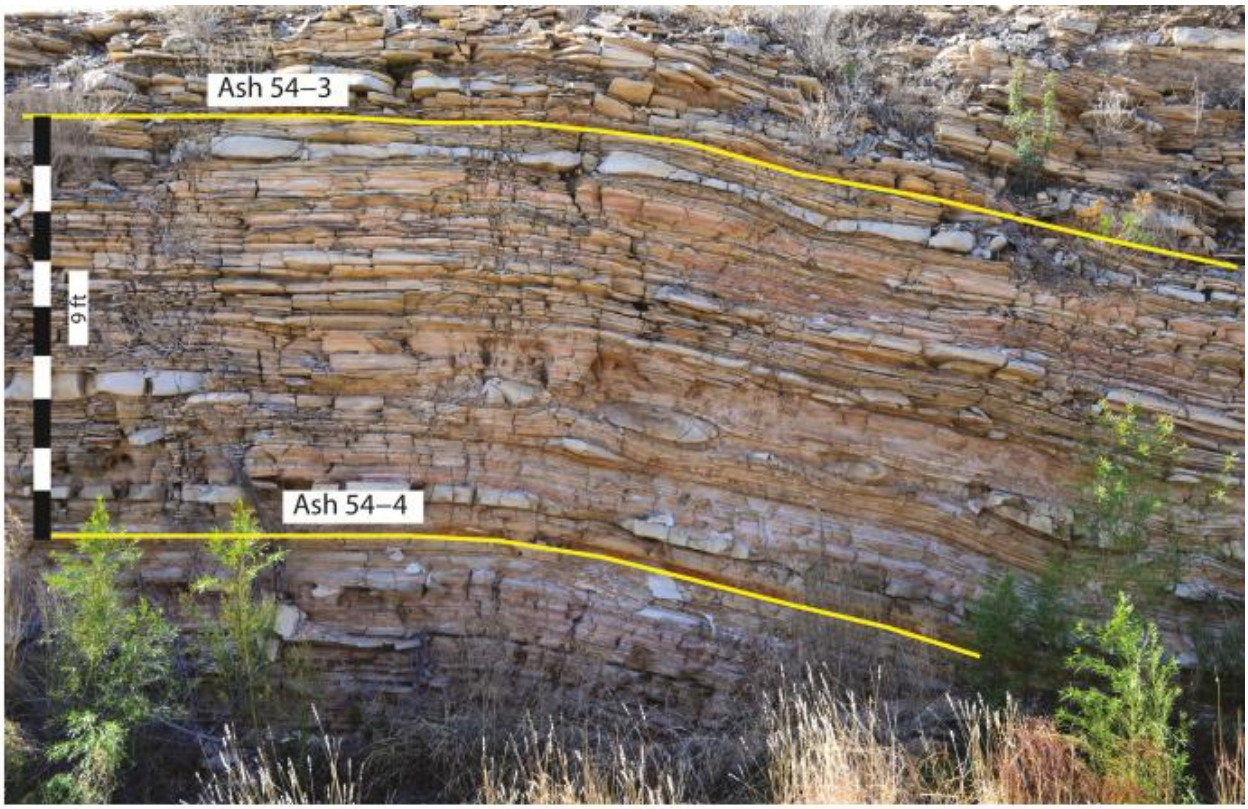
Making facies is that simple!! (Yes, that simple in terms of coding but figuring out if they mean anything is why you get paid 😊). Here I have shown other models I use, like Gaussian Mixture, Bayesian Gaussian Mixture (refer documentation on them) and you can try several models and see which one describes the data better. Let's look at the 8 facies we created and compare the velocities Vp.



As I mentioned earlier, my base assumption is higher the velocities, more brittle they get. You can start a whole argument about deriving failure criteria from elastic domain, but given the workflows that's floating around for brittleness, this one is a safe bet.

So, lets come to the part which is most critical to this analysis -detecting bed or interfaces and its thickness. In general, commercial petrophysical packages do not address this issue, just because it was never a part of the workflow solution. Usually in packages that allow user to code, a general solution to detecting beds or interfaces is to compute a "smoothed" curve and then addressing the changes positive and negative with respect to it. While this solution is somewhere in the ballpark, we need more than this. The other solution is just using facies as beds. But in most cases, we do not get the granularity we require. We need a finer scale than that. This is where python modules come into play.

So, lets first define what is a bed or interface in this example. Several people can have different definitions but I am going to make it simple, wherever the log reading changes direction (magnitude), it is a change in rock properties, hence a bed or interface change. Given below is an actual example from an Eagle ford out crop near Comstock , Texas which illustrates the point of interfaces or beds that make up the Eagle ford stratigraphic framework. Interpreting the frequency of this interbedding is paramount to completion success.



(Source : John D. Pierce et al., BEG University of Texas)

Data noise can be filtered with a gaussian filter prior to applying this. So, what logs would one use for this exercise? The one which has the smallest vertical resolution. Density log and Gamma ray are the best indicators of change in lithology in my opinion, so either one will do. In this exercise I will use Gamma ray.

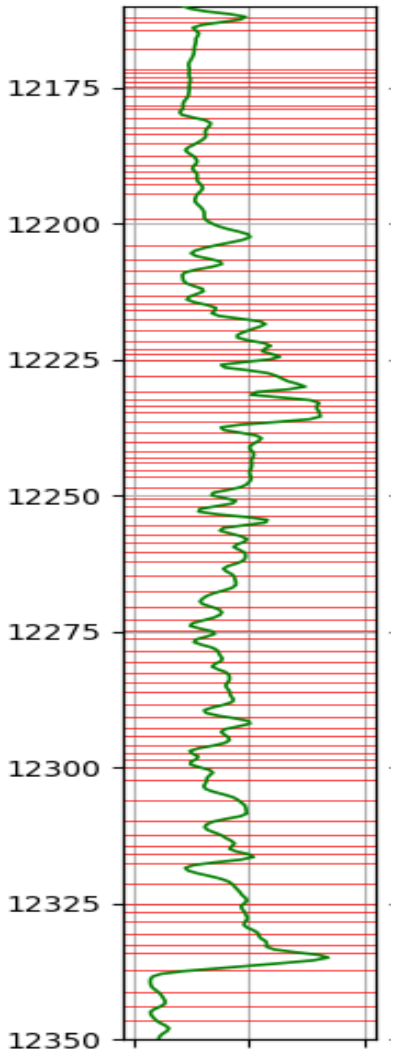
Let's look at the module in python ***scipy.signal***. The ***argrelextrema*** module lets you accurately detect the peaks and troughs in a signal. By detecting this, we can estimate the bed thickness and changes in beds/interface. In the next part of the code, we are going to detect the peaks and troughs of a signal and combine them.

```
data_gr=logs['GR'].values
from scipy.signal import argrelextrema
# Detecting Peaks
y = argrelextrema(data_gr, np.greater)
peak_list=list(y)[0].tolist()
# Detecting Troughs
x = argrelextrema(data_gr, np.less)
trough_list=list(x)[0].tolist()
#Combining then
all_list=sorted(peak_list+trough_list)


#Making  a dataframe
peak_trough_df=pd.DataFrame({'Index_val':all_list})
peak_trough_df['Beds']=1
peak_trough_df=peak_trough_df.set_index('Index_val')
```

Now let's merge this with the original data frame and compute bed thickness. In order to compute bed thickness, the mid-point between peak and trough would be a close approximation.

```
merged_df=logs[['Depth']].merge(peak_trough_df, how='inner', left_index=True, right_index=True)
merged_df['Bed_Top']=(merged_df['Depth']+ merged_df['Depth'].shift(1))/2
merged_df['Bed_Top'].iloc[0]=merged_df['Depth'].iloc[0]
logs=pd.merge(logs,merged_df, on='Depth', how='outer')
```

This would give us a very close approximation for the bed/interfaces shown in figure below. The module seems to have captured every fine detail in the gamma ray.



There are other methods to detect signal slope changes like applying the savgol filter and looking for where the second derivative of the signal is large. Feel free to try them out. The detection of beds enables us to compute bed thickness and frequency.

The next order of business is to compute the Interbedding brittleness index (IBBI). Before we do that lets have a little commentary on what it really means. What we are doing here is somehow quantify the impact of interbeds or laminae in hydraulic fracture propagation.

The IBBI computation will be a fit-for-purpose computation. Do we need large or small frac heights? Sometimes we do not desire large height growth that , if there is a water bearing zone above the landing target. We are not attempting to calculate height growth here rather assign a relative probability/value to achieve the desired outcome.

There are 4 parameters of interest in this exercise.

1. Bed thickness
2. Interbedding frequency (higher the frequency smaller the bed thickness)
3. Velocity of the bed (Vp) – a measure of relative brittleness.
4. Velocity contrast – (pseudo for stress contrast)

Rather than using actual values, I am going to assign a "relative value" to each of them for the desired outcome. From literature review on the effects of interbeds/laminae on fracturing this is what I came up with in terms relative value assignment.

	Desired Outcome		Weights
	Large Frac Height	Small Frac Height	
High	RELATIVE VALUE ASSIGNMENT		
Inter bedding frequency	High	Low	W1
Bed thickness	Low	High	W2
Vp	High	Low	W3
Vp Contrast	Low	High	W4

For example, higher interbedding frequencies (IBF), can help you achieve larger frac heights (thinner beds are easy to break) but higher velocity contrast between them can lower the height growth (preferential paths, complexity etc.). So, for this outcome, higher IBF is rewarded while the Vp contrast is penalized. Also, each of these parameters do not affect the outcome equally, hence we assign weights to the parameters.

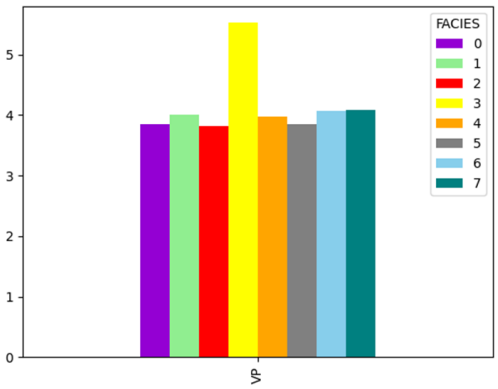
This value assignment is akin to a point system rewarding the attributes for the required outcome. Here we assign a scale from 0.05 to 1. This relative assignment of value could be non-linear or even equal increments. In this example, I am assigning values based on their relative values. So, for a desired outcome for large frac height growth my IBBI function will look something like this.

$$IBBI = \frac{IBF^{W1} * VP^{W3}}{VP\ contrast^{W4} * Bed\ Thickness^{W2}}$$

(Some of you can come up with other creative assignments of value. Also, the weight assignment currently is arbitrary; more thoughts need to go into this)

```
#Computing Bed thickness, Avg Vp over facies and its contrast to the previous facies
data = logs[logs['Beds'] > 0]
data['Bed Thickness']=-(data['Bed_Top']-data['Bed_Top'].shift(-1))
data=data[['Depth','Bed Thickness']]
logs=pd.merge(logs,data, on='Depth', how='outer')
logs['Bed Thickness']=logs['Bed Thickness'].fillna(method='ffill')
logs=logs.fillna(0)

logs['Vp_Avg'] = logs.groupby('FACIES')['VP'].transform('median')
logs['Vp_contrast']= abs((logs['Vp_Avg']-logs['Vp_Avg'].shift(1))/logs['Vp_Avg'])
logs['Vp_contrast']= logs['Vp_contrast'].replace(to_replace=0, method='ffill').fillna(method='bfill')

vp_max=logs['Vp_Avg'].max()
vp_min=logs['Vp_Avg'].min()
contrast_vp_max=logs['Vp_contrast'].max()
contrast_vp_min=logs['Vp_contrast'].min()
bed_thick_max=logs['Bed Thickness'].max()
bed_thick_min=logs['Bed Thickness'].min()

#Assign Reward Points
logs['Value_Vp'] =0.05+0.95*((logs['Vp_Avg']-vp_min)/(vp_max-vp_min))
logs['Value_Vp_contrast'] = 0.05 + 0.95 * ((logs['Vp_contrast']-contrast_vp_min) / (contrast_vp_max -
contrast_vp_min))
logs['Value_Bed_Thickness']= 0.05 + 0.95 * ((logs['Bed Thickness']-bed_thick_min) / (bed_thick_max -
bed_thick_min))
```

FACIES	VP	Value_Vp
2	3.824	0.050
0	3.842	0.060
5	3.848	0.064
4	3.973	0.133
1	4.012	0.155
6	4.070	0.187
7	4.091	0.199
3	5.526	1.000



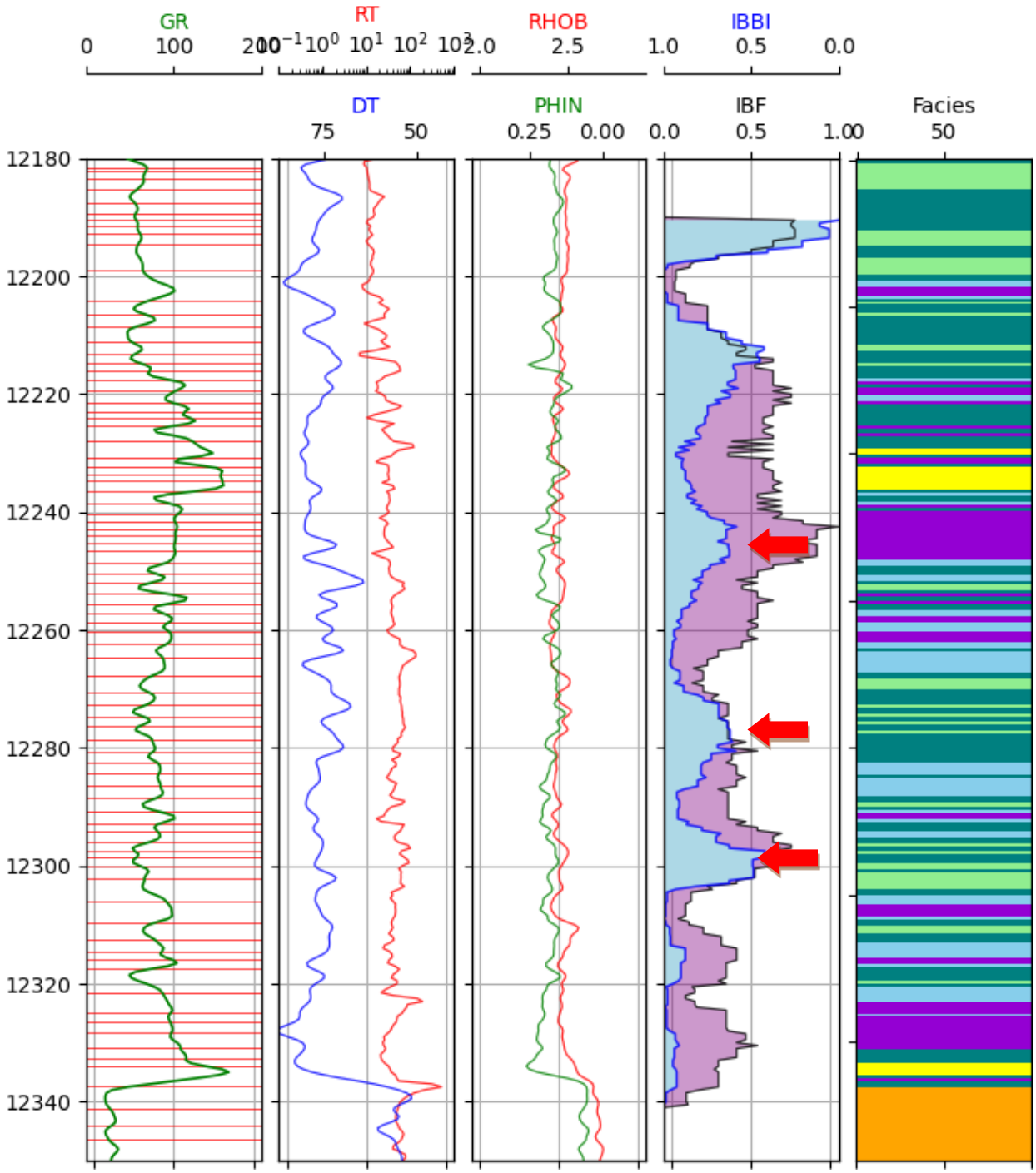
Having assigned a relative value for the facies, we then proceed to compute the desired completion attributes for the desired scale (target tolerance window). We would then slide the desired target window along the well to find,

1. The Interbedding frequency (IBF) of the target window. This computation includes 3 parameters, the number of beds, median thickness of the beds and thickness of the target interval.
2. The corresponding Interbedding Brittleness Index (IBBI), which is the harmonically averaged relative values for the 4 parameters described earlier.

```
for i in range(0, len(logs)):
    if i <= target_window:
        logs.at[i, 'Interbedding'] =0
    elif i > (len(logs)-target_window):
        logs.at[i, 'Interbedding'] = 0
    else:
        data = data_beds.iloc[(i - target_window //2):(i + target_window // 2), ]
        data = data[data['Beds'] > 0]
        data['Thickness'] = -(data['Bed_Top']-data['Bed_Top'].shift(-1))
        Median_Thickness = data['Thickness'].median()
        Mean_value_vp=harmonic_mean(data['Value_Vp'].tolist())
        Mean_value_contrast=harmonic_mean(data['Value_Vp_contrast'].tolist())
        Mean_value_thickness=harmonic_mean(data['Value_Bed_Thickness'].tolist())
        sum_cross =(data['Beds'].sum())

        logs.at[i, 'IBF'] =sum_cross/(Median_Thickness*target_window)
        logs.at[i, 'IBBI'] =((Mean_value_vp**2*logs.at[i,
'Interbedding']**1)/(Mean_value_contrast**1.5*Mean_value_thickness**1))


#Fill Missing and Normalize between 0-1

logs['IBBI']=logs['IBBI'].ffill()
logs['IBF']=logs['IBF'].ffill()
logs['IBF']=(logs['IBF']-logs['IBF'].min())/(logs['IBF'].max()-logs['IBF'].min())
logs['IBBI']=(logs['IBBI']-logs['IBBI'].min())/(logs['IBBI'].max()-logs['IBBI'].min())
```

The harmonic averaging method for relative values is very similar to the vertical permeability upscaling (in series) in geo-cellular modeling. Basically, to sum it up in layman's terms, in the given target window, let's say with respect to Vp, we penalize the presence of ductile rocks a lot more than we appreciate the presence of brittle rocks. This comes from real life examples that we have seen how ductile interfaces can inhibit frac height growths, imbeds proppant , act as pinch points etc. This is very similar to the presence of low perm beds among those with high perms.

In this Eagle ford example below, we see the application of this method. Between 12215-12260, we see high interbedding or interface changes (IBF). But the IBBI value is less because of the presence of ductile beds/thicker beds or higher contrast. The best landing targets within Eagle ford for a 20ft target tolerance window from a completions quality perspective (desired outcome- increased frac height) is shown by the red arrows, where the IBBI values are high. Of course, the final target will depend on integrating the reservoir quality as well.



Another way to evaluate this without the need for facies, using bed averages than facies, since we already have resolved the beds.

I hope you found this methodology useful and applicable in your reservoirs to determine the optimal landing target with respect to completions. This is a work in progress, I welcome other suggestions to compute the IBBI factor and physical backups to the choice of weights. Bring it on.