

AutoPlot

Just press a Button

Content

- Background
- Problems
- Features
- Demo: DE, Diffu., WE
- Scripting/Coding
- Coding Essentials
- Modules: CP, ER, Uniformity
- Utilities

Background

- **AutoPlot** is an Indigenous Software created for generating plots automatically from the data (especially QC) maintained in all the sections of VFD.
- applicable for any type of data –
 - CSV,
 - TSV,
 - Excel (with single/multiple sheets)

Problems

- In order to generate Plots for QC data (maintained in Excel), currently we drag each parameter (like Date, Control limits, Spec limits, Etch Rate/Thickness/CP, etc.) manually to create/update the charts. And this consumes a good amount of valuable time in Engineer's data analysis job.
- In some sections, they are dependent on plots created in DMIS, which is not that satisfactory as it doesn't contain '**Remarks**' column feature and more.

Features

- It has 3 operation modes => **Button**, **Shell** & **Auto**.
- It shows the '**Remarks**' column while hovering on the data points in the Plot.
- It is very *Interactive* (zoom-in/out) and is quite *Modern*.
- A button is built-in within, for saving the plot's snapshot and can be attached in PPTs, Mail, etc.
- FYI, the entire codebase is written in Python, Bash (for automation) programming languages.
- There is no need of learning a programming language. All the modules/functions will be provided in form of packages and hence can be applied (with little parameter tweaks) for any form of data (in CSV, Excel).
- **ViEW** (another Indigenous Software) is also integrable with **AutoPlot** for adding "**Version Control**" feature to the Codebase, Excel, Word, Image (or any other format) files.
Here, **ViEW** acts as the foundation layer for **AutoPlot**.
- It can also be used in applications where JMP software is normally preferred, here in SCL. In this, there are separate modules/functions which can replace the scripts written in **JMP Scripting Language (JSL)**.

Testing

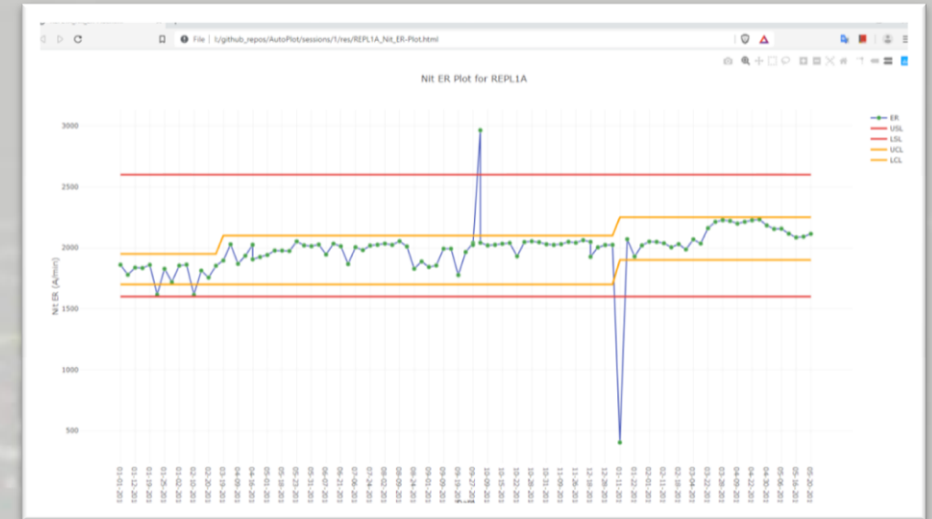
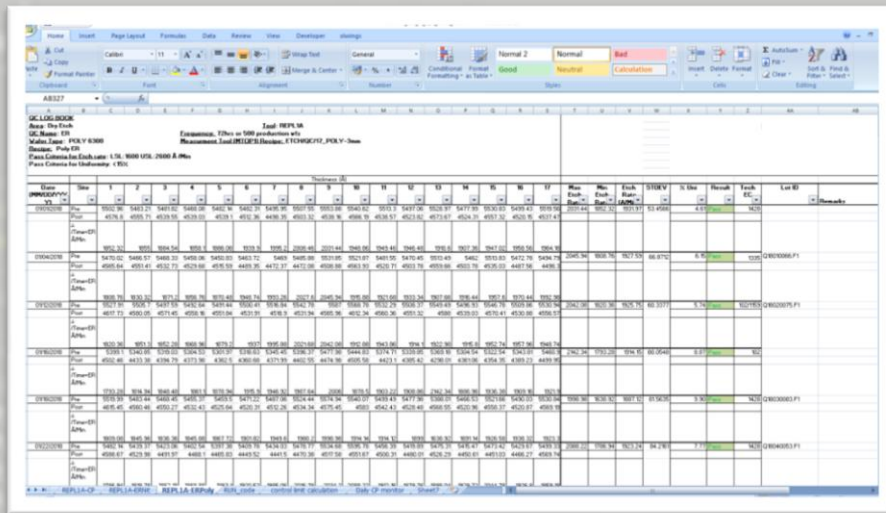
- Fully implemented in **DRY ETCH** and successfully tested for more than 2 months now.
- A sample QC data of PRS01 Equipment, **WET ETCH** has also been successfully tested as well.
- A sample QC data of FRST1 Equipment, **DIFFUSION** has also been successfully tested as well.

Demo: Dry Etch

Demo: Diffusion

Demo: Wet Etch

Design Model



Cleaning,
Filtering,
Re-structuring

index labels		column names				
		Mountain	Height (m)	Range	Coordinates	Parent mountain
0		Mount Everest / Sagarmatha / Chomolungma	8848	Mahalangur Himalaya	27°59'17"N 86°55'31"E	N/A
1		K2 / Qogir / Godwin Austen	8611	Baltoro Karakoram	35°52'53"N 76°30'48"E	Mount Everest
2		Kangchenjunga	8586	Kangchenjunga Himalaya	27°42'12"N 88°08'51"E	Mount Everest
3		Lhotse	8516	Mahalangur Himalaya	27°57'42"N 86°55'59"E	Mount Everest
4		Makalu	8485	Mahalangur Himalaya	27°53'23"N 87°05'20"E	Mount Everest
5		Cho Oyu	8188	Mahalangur Himalaya	28°10'30"N 86°39'39"E	Mount Everest
6		Dhaulagiri I	8167	Dhaulagiri Himalaya	28°41'48"N 83°29'35"E	K2
7		Manaslu	8163	Manaslu Himalaya	28°33'00"N 84°33'35"E	Cho Oyu
8		Nanga Parbat	8126	Nanga Parbat Himalaya	35°14'14"N 74°35'21"E	Dhaulagiri
9		Annapurna I	8091	Annapurna Himalaya	28°25'44"N 83°49'13"E	Cho Oyu

Generate Plots

- Button
- Shell
- Auto

Coding Essentials

- Editor, Compiler

Modules: Global variables

```
# Global variables
line_color = '#3f51b5'      # line (trace1) color for any plot
marker_color = '#43a047'    # marker color for any plot
marker_border_color = '#ffffff' # marker border color for any plot
cl_color = '#ffa000'        # control limit line color for any plot
sl_color = '#e53935'        # spec limit line color for any plot
cp_plot_title = 'CP Plot for RESP1B' # title for CP plot
cp_plot_xlabel = 'Date'      # xaxis name for CP plot
cp_plot_ylabel = 'delta CP (no.s)' # yaxis name for CP plot
cp_plot_html_file = 'RESP1B_CP-Plot.html' # HTML filename for CP plot
cp_plot_trace_count = 2      # no. of traces in CP plot
er_barcode_plot_title = 'BARC ER Plot for RESP1B' # title for ER plot
er_barcode_plot_xlabel = 'Date' # xaxis name for ER plot
er_barcode_plot_ylabel = 'BARC ER (A/min)' # yaxis name for ER plot
er_barcode_plot_html_file = 'RESP1B_BARC_ER-Plot.html' # HTML filename for ER plot
er_barcode_plot_trace_count = 5 # no. of traces in ER plot
unif_barcode_plot_title = 'BARC Uniformity Plot for RESP1B' # title for Unif plot
unif_barcode_plot_xlabel = 'Date' # xaxis name for Unif plot
unif_barcode_plot_ylabel = 'BARC Unif (%)' # yaxis name for Unif plot
unif_barcode_plot_html_file = 'RESP1B_BARC_Unif-Plot.html' # HTML filename for Unif plot
unif_barcode_plot_trace_count = 2 # no. of traces in Unif plot

sht_cp_columns = ["Date (MM/DD/YYYY)", "delta CP", "USL", "Remarks"] # Columns for CP
# Columns for BARC Layer Dataframe
sht_er_barcode_columns = ["Date (MM/DD/YYYY)", "Layer", "Etch Rate (A/Min)", "% Uni",
"Remarks", "LSL", "USL", "LCL", "UCL", "% Uni USL", "% Uni UCL"]

# Excel file directory
excel_file_directory = "\\vmfg\VFD FILE SERVER\SECTIONS\DRY ETCH" +
"\\QC Log Book\Final QC Log Book\UNT_02_LOG_BOOK\UNT02_Ch_A_QC_LOG_BOOK.xlsm"
```

Modules: Date Formatter

```
"""
    """
    "Description": Date formatter to format the excel date (issue: one date less in plotly chart) as "%m-%d-%Y %H:%M:%S"
    "x": datetime list
    "return": formatted datetime list
    """
def date_formatter(x):
    x_fmt = []
    for a in x:
        a = a.strftime("%m-%d-%Y %H:%M:%S")
        x_fmt.append(a)
    return x_fmt
```

Modules: CP Plot

```
"""
>Description: This function plots CP Chart with traces v/s
Date
draw_plotly_resplb_cp_plot": Draw Plotly's Plot for RESP1B CP
"x": Date (x-axis) for CP Chart
"y1": Delta-CP (y-axis) for CP Chart
"y2": USL (y-axis) for CP Chart
# "y3": UCL (y-axis) for CP Chart
"""
def draw_plotly_resplb_cp_plot(x, y1, y2, remarks):
    trace1 = go.Scatter(
        x = x,
        y = y1,
        name = 'delta-CP',
        mode = 'lines+markers',
        line = dict(
            color = line_color,
            width = 2),
        marker = dict(
            color = marker_color,
            size = 8,
            line = dict(
                color = marker_border_color,
                width = 0.5),
            ),
        text = remarks
    )

    trace2 = go.Scatter(
        x = x,
        y = y2,
        name = 'USL',
        mode = 'lines',
        line = dict(
            color = sl_color,
            width = 3)
    )

    trace3 = go.Scatter(
        x = x,
        y = y3,
        name = 'UCL',
        mode = 'lines',
        line = dict(
            color = cl_color,
            width = 3)
    )

    data = [trace1, trace2, trace3]
    layout = dict(
        title = cp_plot_title,
        xaxis = dict(title= cp_plot_xlabel),
        yaxis = dict(title= cp_plot_ylabel)
    )

    fig = dict(data= data, layout= layout)
    py.offline.plot(fig, filename= cp_plot_html_file)
```

Modules: ER Plot

```
"""
"Description": This function plots ER Chart with traces v/s Date.
"draw_plotly_resplb_er_barb_plot": Draw Plotly's Plot for RESP1B BARC
ER": Date (x-axis) for ER Chart
"y1": ER (y-axis) for ER Chart
"y2": USL (y-axis) for ER Chart
"y3": LSL (y-axis) for ER Chart
"y4": UCL (y-axis) for ER Chart
"y5": LCL (y-axis) for ER Chart
"""
def draw_plotly_resplb_er_barb_plot(x, y1, y2, y3, y4, y5, remarks):
    trace1 = go.Scatter(
        x = x,
        y = y1,
        name = 'ER',
        mode = 'lines+markers',
        line = dict(
            color = line_color,
            width = 2),
        marker = dict(
            color = marker_color,
            size = 8,
            line = dict(
                color = marker_border_color,
                width = 0.5),
            ),
        text = remarks
    )

    trace2 = go.Scatter(
        x = x,
        y = y2,
        name = 'USL',
        mode = 'lines',
        line = dict(
            color = sl_color,
            width = 3)
    )

    trace3 = go.Scatter(
        x = x,
        y = y3,
        name = 'LSL',
        mode = 'lines',
        line = dict(
            color = sl_color,
            width = 3)
    )

    trace4 = go.Scatter(
        x = x,
        y = y4,
        name = 'UCL',
        mode = 'lines',
        line = dict(
            color = cl_color,
            width = 3)
    )

    trace5 = go.Scatter(
        x = x,
        y = y5,
        name = 'LCL',
        mode = 'lines',
        line = dict(
            color = cl_color,
            width = 3)
    )

    data = [trace1, trace2, trace3, trace4, trace5]
    layout = dict(
        title = er_barb_plot_title,
        xaxis = dict(title= er_barb_plot_xlabel),
        yaxis = dict(title= er_barb_plot_ylabel)
    )
    fig = dict(data= data, layout= layout)
    py.offline.plot(fig, filename= er_barb_plot_html_file)
```

Modules: Unif Plot

```
"""
"Description": This function plots Unif Chart with traces v/s Date.
"draw_plotly_respb1b_unif_barb_plot": Draw Plotly's Plot for RESP1B BARC
Unif Date (x-axis) for Unif Chart
"y1": Unif (y-axis) for Unif Chart
"y2": USL (y-axis) for Unif Chart
"y3": UCL (y-axis) for Unif Chart
"""
def draw_plotly_respb1b_unif_barb_plot(x, y1, y2, y3, remarks):
    trace1 = go.Scatter(
        x = x,
        y = y1,
        name = 'Unif',
        mode = 'lines+markers',
        line = dict(
            color = line_color,
            width = 2),
        marker = dict(
            color = marker_color,
            size = 8,
            line = dict(
                color = marker_border_color,
                width = 0.5),
            ),
        text = remarks
    )

    trace2 = go.Scatter(
        x = x,
        y = y2,
        name = 'USL',
        mode = 'lines',
        line = dict(
            color = sl_color,
            width = 3)
    )

    trace3 = go.Scatter(
        x = x,
        y = y3,
        name = 'UCL',
        mode = 'lines',
        line = dict(
            color = cl_color,
            width = 3)
    )

    data = [trace1, trace2, trace3]
    layout = dict(
        title = unif_barb_plot_title,
        xaxis = dict(title= unif_barb_plot_xlabel),
        yaxis = dict(title= unif_barb_plot_ylabel)
    )
    fig = dict(data= data, layout= layout)
    py.offline.plot(fig, filename= unif_barb_plot_html_file)
```


Utilities

- **Stage-2:** Applying Nelson rules
(Under development)
- Like in JMP, one can write custom script to generate plots out of data (in any format).
- Various types of plots can be created – Line, Scatter, Box, Qiver, Contour (2-D, 3-D), Heatmap,

