Appendix 4: Code for plotting normalized MCC data

This code was run in a python notebook on a windows PC. Linux PCs seem to have an issue with the code.

```
import csv
import matplotlib.pyplot as plt

# Surfaces, image scales and case numbers for the plot titles
surfaces = ["Q0","Q1","Q2","Q3","Q4"]
zooms    = ["200","100","050","025","010","005","001"]
cases    = ["01","02","03","04","05","06","07","08"]

# Headers for the "total averages" CSV file
total_average = [["Complete average"] + zooms]

# Importing the file created in the downscalign script
with open('Hits.csv', 'r', encoding='utf-8-sig') as readFile:

    # Creating a list of the data
    reader = csv.DictReader(readFile)
    reader_list = list(reader)

    # Headers for the "case averages" CSV files
    case_averages = [["Averages"] + zooms]

    # Creating a tally and total for average values in total
    values_total  = {"200":0,"100":0,"050":0,"025":0,"010":0,"005":0,"001":0}
    numbers_total = {"200":0,"100":0,"050":0,"025":0,"010":0,"005":0,"001":0}

    # Itterating over each case
    for case in cases:

        # Start of a new line for "case averages" CSV
        average_newline = [case]

        # Headers for the "case-by-case" CSV files
        table   = [["case " + case] + zooms]

        # Creating a tally and total for average values per case
        values  = {"200":0,"100":0,"050":0,"025":0,"010":0,"005":0,"001":0}
        numbers = {"200":0,"100":0,"050":0,"025":0,"010":0,"005":0,"001":0}

        # Itterating over possible surfaces
        for surface in surfaces:
```

```
    # Start of a new line for "case-by-case" CSV
    new_line = [surface]

    # Itterating over possible zoom levels
    for zoom in zooms:

        # Itterating over all data points
        for lines in reader_list:

            # Finding correct data point throuhgh case, zoom and surface
            if zoom == lines["Zoom"] and surface == lines["surface"] and case == lines["Case"]:

                # Checking for error values
                if lines["MCC_observe"] != "Error" and lines["X95_MCC"] != "Error":

                    # Calculating normalized MCC
                    calibrated_MCC = float(lines["MCC_observe"])-float(lines["X95_MCC"])

                    # Updating tallies and totals
                    values[zoom] += calibrated_MCC
                    values_total[zoom] += calibrated_MCC
                    numbers[zoom] += 1
                    numbers_total[zoom] += 1

                    # Adding data point to the new "case-by-case" CSV line
                    new_line.append(calibrated_MCC)
                else:
                    # An error is added if there was no MCC
                    new_line.append("Error")

    # Line is added to the "case-by-case" CSV
    table.append(new_line)

# "case averages" average is calculated per scale
# (the average over all horizons is calculated)
for zoom2 in zooms:
    average_newline.append(values[zoom2]/numbers[zoom2])

# The "case-by-case" CSVs are created
with open(table[0][0]+' table.csv', 'w+', newline='') as newfile:
    writer = csv.writer(newfile, quoting=csv.QUOTE_ALL)
    writer.writerows(table)

# Line is added to the "case averages" CSV
```

```
        case_averages.append(average_newline)


    # The "case averages" CSVs are created
    with open(case_averages[0][0]+' table.csv', 'w+', newline='') as newfile:
            writer = csv.writer(newfile, quoting=csv.QUOTE_ALL)
            writer.writerows(case_averages)

# Start of a new line for "total averages" CSV
total_average_newline = ["Average"]

# "total averages" average is calculated per scale
# (the average over all horizons and cases is calculated)
for zoom3 in zooms:
    total_average_newline.append(values_total[zoom3]/numbers_total[zoom3])

# Line is added to the "total averages" CSV
total_average.append(total_average_newline)

# The "total averages" CSV is created
with open(total_average[0][0]+' table.csv', 'w+', newline='') as newfile:
        writer = csv.writer(newfile, quoting=csv.QUOTE_ALL)
        writer.writerows(total_average)



# Making sure the figure is empty
plt.clf()

# Defining the X-axis values
X = ["1:200","1:100","1:50","1:25","1:10","1:5","1:1"]

# Opening the "case averages" data and the "total averages" data
with open('Averages table.csv', 'r', encoding='utf-8-sig') as file_a, open('Complete average table.csv',
'r', encoding='utf-8-sig') as file_b:

    # making a list of the "case averages"
    reader = csv.DictReader(file_a)
    reader_list = list(reader)

    # making a list of the "total averages"
    reader_b = csv.DictReader(file_b)
    reader_list_b = list(reader_b)

    # Defining plot types
    plots = ["Frequency","Elastic properties","Illumination","All cases"]

    # Hard coding the Y-axis ranges per plot type
    plot_yscale = [[-0.7, 0.3],[-1.3,0.4],[-0.8,0.3],[-1.3,0.4]]

    # Defining wether to add the "total average" per plot type
    plot_avg = ["No","No","No","Yes"]

    # Defining what cases to plot per plot type
    plots_cases =
[["01","02","03"],["01","06","07","08"],["01","04","05"],["01","02","03","04","05","06","07","08"]]

    # Defining the unit of the feature of interest per plot type
    case_features =
[["30Hz","60Hz","90Hz"],["Standard","Low","High","Contrast"],["45°","25°","15°"],["Case 1","Case 2","Case
3","Case 4","Case 5","Case 6","Case 7","Case 8"]]

    # Itterating over plot types, plot count to determine the features of the plots
    for plot_count, plot_name in enumerate(plots):

        # Fetching Y-axis range
        plt.ylim(plot_yscale[plot_count])

        # Defining plot titles, labels and style
        plt.style.use('ggplot')
        plt.title(plot_name +"\n A) Dolomite threshold: 8%")
        plt.xlabel("Scale factor")
        plt.ylabel("MCC normalized on the 99.5th percentile")

        # Generating file name
        filename = case_averages[0][0] + " " + plot_name

        # Fetching and itterating over cases to plot
        for case_count, case_name in enumerate(plots_cases[plot_count]):

            # Itterating over all data points
            for lines in reader_list:

                # Finding case number for data point
                case = list(lines.values())[0]

                # Checking wether to plot the case
                if case == case_name:

                    # Fetching data points
                    Y = list(map(float,list(lines.values())[1:]))

                    # Plotting data points
                    plt.plot(X,Y,  label = case_features[plot_count][case_count])

        # Checking whether to plot the "total average"
        if plot_avg[plot_count] == "Yes":

            # Plotting the "total average"
            plt.plot(X,list(map(float,list(reader_list_b[0].values())[1:])),"k--", label = "Average")

        # Plotting the X-axis
        plt.plot(["1:200","1:1"],[0,0], "k")

        # Adding a legend
        plt.legend(loc = 'lower right')

        # Saving the plot to a png and cleaning the plot
        plt.savefig("Results_chart/" + filename + "_plot.png", bbox_inches="tight", dpi=300)
        plt.clf()
```