# Changes to vis code:

1. Sheet order

A screenshot of a calendar

Description automatically generated

workbook\_creation\_functions.order\_sheets(workbook, plotting\_specifications)

1. Y axis values:

Made a few fixes for this. First goal was to make it so the positive values worked as expected but then realized that we should include minimum values (to consider for negative values). So it ended up a bit more complicated than expected.

def extract\_max\_and\_min\_values(data, max\_and\_min\_values\_dict, total\_plotting\_names):

    unique\_sheets = data['sheet\_name'].unique()

    unique\_chart\_types = data['chart\_type'].unique()

    unique\_table\_ids = data['table\_id'].unique()

    for sheet in unique\_sheets:

        for chart\_type in unique\_chart\_types:

            for table\_id in unique\_table\_ids:

                subset = data[

                    (data['sheet\_name'] == sheet) &

                    (data['chart\_type'] == chart\_type) &

                    (data['table\_id'] == table\_id)

                ]

                if subset.empty:

                    continue

                if subset.aggregate\_column.iloc[0] == 'fuels\_plotting':

                    subset.loc[subset['sectors\_plotting'].isin(total\_plotting\_names), 'value'] = 0

                elif subset.aggregate\_column.iloc[0] == 'sectors\_plotting':

                    subset.loc[subset['fuels\_plotting'].isin(total\_plotting\_names), 'value'] = 0

                postive\_values = subset[subset['value'] >= 0].copy()

                negative\_values = subset[subset['value'] <= 0].copy()

                if len(postive\_values) > 0:

                    max\_value = postive\_values.groupby(['year', 'scenario'])['value'].sum().max()

                    if chart\_type == 'line':#we dont want to sum the values for line charts

                        max\_value = postive\_values['value'].max()

                else:

                    max\_value = 0

                if len(negative\_values) > 0:

                    min\_value = negative\_values.groupby(['year', 'scenario'])['value'].sum().min()

                    if chart\_type == 'line':#we dont want to sum the values for line charts

                        min\_value = negative\_values['value'].min()

                else:

                    min\_value = 0

                if chart\_type == 'area' and min\_value < 0 and max\_value > 0:

                    #area plots dont really work when we have negative and positive values. so let user know but dont raise an error

                    print('WARNING: Area chart for ' + sheet + ' with table\_id ' + str(table\_id) + ' has both negative and positive values. This will not work well. Please consider changing the chart type to line or bar')

                # Calculate max y-axis value for the chart

                if max\_value is not None and not np.isnan(max\_value):

                    if max\_value == 0:

                        y\_axis\_max = 0

                    elif max\_value > 0:

                        y\_axis\_max = calculate\_y\_axis\_value(max\_value)

                    else:

                        y\_axis\_max = None

                    key\_max = (sheet, chart\_type, table\_id, "max")

                    max\_and\_min\_values\_dict[key\_max] = y\_axis\_max

                # Calculate min y-axis value for the chart

                if min\_value is not None and not np.isnan(min\_value):

                    if min\_value == 0:

                        y\_axis\_min = 0

                    elif min\_value < 0:

                        y\_axis\_min = calculate\_y\_axis\_value(min\_value)

                    else:

                        y\_axis\_min = None

                    key\_min = (sheet, chart\_type, table\_id, "min")

                    max\_and\_min\_values\_dict[key\_min] = y\_axis\_min

    # Remove items with None values

    max\_and\_min\_values\_dict = {k: v for k, v in max\_and\_min\_values\_dict.items() if v is not None}

    return max\_and\_min\_values\_dict

def calculate\_y\_axis\_value(value):

    # Adjust the value by 5% in the appropriate direction

    if value > 0:

        y\_axis\_value = value + (0.05 \* value)

    else:

        y\_axis\_value = value - (0.05 \* value)

    try:

        # Use absolute value to handle the logarithm for negatives

        order\_of\_magnitude = 10 \*\* math.floor(math.log10(abs(y\_axis\_value)))

        rounding\_step = order\_of\_magnitude / 2

        # If the value is positive, round up. If negative, round down.

        if y\_axis\_value > 0:

            y\_axis\_value = math.ceil(y\_axis\_value / rounding\_step) \* rounding\_step

        else:

            y\_axis\_value = math.floor(y\_axis\_value / rounding\_step) \* rounding\_step

    return y\_axis\_value

3.

Ordered chart types:

A screen shot of a computer code

Description automatically generated

1. Nas in tables:
   1. Ichanged this to be >= and <= because otherwise 0’s were being set to nan: A computer screen shot of a code

      Description automatically generated
   2. Then for ‘Other renewables’ in power input, I changed the plotting name to ‘Other Renewables’ heh.
2. Error when a plotting name isn’t in the colors sheet fixed below

A screen shot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

1. Added a space under ?or? above charts number. this probably will need to be implemented better lately as I haven’t tested it (it resulted perfectly when I first implemented it)

A screen shot of a computer

Description automatically generated

1. Added line thickness parameter in master config.xlsx
2. Turned 1\_map\_9th\_data… into a function which is now called in code\1\_prepare\_9th\_workbook.py