```
# -*- coding: utf-8 -*-
"""
Created on Mon May  6 14:10:47 2024

@author: enis_kostak

This script is designed to generate visualizations for the catch share rate and
retention probability
for different mesh sizes in krill for trawl nets. It processes excel data files to
create plots using
the altair visualization library. The generated plots include catch share points,
modeled rates, and
size selection curves with confidence bands.

For paths and other documents please visit:

https://github.com/eniskostak/enk/tree/5430375cd7d59082ef7955eb9ce14132a9334312/fin
al_assignment

"""

import pandas as pd   # pandas imported for data manipulation
import altair as alt  # altair for plotting
import webbrowser     # webbrowser to open and display the html file with the plots
import re             # re for regular expression operations (to name titles from
file name)


### catch share rate function
def generate_chart1(catch_share_paths):

    """
   Generate the left column chart for catch share rate plot.
   This function extracts the mesh size information from the filename, loads the
excel data,
   applies numeric conversion, and constructs the altair chart with lines and
points.

   Codes for naming: MB14; Mesh Belly 14mm, reperesents 14mm mesh size for belly
section of the trawl net.

   X1 and Y1: Column for catch share curve data
   X0 and Y0: Column for catch points data
   X2 and Y2: Column for population structure (control)
   X3 and Y3: Column for population structure (test)
   all X columns are for length data, statistical software output gives separate
columns.
   """
   # function extracts mesh size from file path for title suffix (from re module)
    match = re.search(r'c_share_krill_(MB\d+)_', catch_share_paths) #regex search
```

```
for: MB followed by one or more digits, like MB14.
    title_suffix1 = match.group(1) if match else 'Unknown' # if the search finds a 
match it returns match object as title, if not error as Unknown


    data = pd.read_excel(catch_share_paths,    # funct. reads excel data into 
pandas data frame, and paths excel file
                         header=1)             # skipping first row (index 0) and 
starts from second row (index 1)due to unused headers
    data = data.apply(pd.to_numeric,    # applies fun data pd frame, convert 
numeric
                      errors='coerce')  # converts non convertible values into 
NaN(not a number) to prevent conversation fails
    """
    # combine line (catch share rate) and circle (catch share points) for left axis
    """
    #Catch share rate curve
    left_axis_chart = alt.Chart(data).mark_line(color='black',  # creates altair 
chart object as 'data', and mark it as line chart, in black
                                                clip=True,      # to prevent 
extended data clip chart
                                                size=5          # thickness of the 
line
                                                ).encode(       # encode() method 
used to bind data columns like x, y axis configuration
        x=alt.X('X1:Q', # config the x-axis: X1 is the column name from dataframe 
in numerical(Q:quantative)
                scale=alt.Scale(domain=[15, 55]), # sets range scale
                title='Length (mm)',  # sets title name
            axis=alt.Axis(values=[15, 25, 35, 45, 55], # sets labels
                          grid=False, # no grids
                          titleFontSize=30, # sets title font size
                          labelFontSize=25, # sets label font size
                          titlePadding=20)),# sets spacing between labels and title
        y=alt.Y('Y1:Q',  # configs the y-axis: Y1 is the column name.
                scale=alt.Scale(domain=[0, 1.01]),
                title='Catch share rate',
            axis=alt.Axis(values=[0, 0.25, 0.50, 0.75, 1.00],
                          labelFontSize=25,
                          titleFontSize=30,
                          format='.2f', # for two deciamls in y-axis labels
                          grid=False))
        #Catch share points
    ) + alt.Chart(data).mark_point(shape='circle',  # make point chart with circles
                                   clip=True,
                                   filled=True,     # fill circle
                                   fill='white',    # fill color
                                   stroke= 'black', # black outer line for circle
                                   fillOpacity=1,   # opacity for filling
                                   size=80          # circle size
                                   ).encode(
        x=alt.X('X0:Q', # catch point data
                scale=alt.Scale(domain=[15, 55])), # set range scale
        y='Y0:Q'        # y axis
    )
    """
    This part here for to combine population structure for control and test nets
    """
    # dotted population structure (control+test) with right y axis
    right_axis_chart = alt.Chart(data).mark_line(size = 1,          # creates
altair chart object as 'data', and mark it as line chart
                                                 color='darkgrey',  # line color
darkgrey
                                                 clip=True).encode(
        x='X3:Q',       # test data column
        y=alt.Y('Y3:Q', # y axis
                scale=alt.Scale(domain=[0, 2000]), # range scale
                title='Number captured',           # title for right axis
            axis=alt.Axis(orient='right',          # oriantation of the y axis
                          labelFontSize=25,
                          titleFontSize=30,
                          titlePadding=20,
                          grid=False,
                          format="d",              # removes comma on thousand
sperator
                          values=[0, 500, 1000, 1500, 2000])) # right y axis labels
    ) + alt.Chart(data).mark_line(size = 1,
                                  color='black',
                                  clip=True).encode(
        x='X2:Q',       # control data column
        y=alt.Y('Y2:Q',
                scale=alt.Scale(domain=[0, 2000]))
    )

    # specify right y axis for combined pop
    right_axis_chart = right_axis_chart.encode(
        y=alt.Y('Y2:Q',
                axis=alt.Axis(orient='right', # orient y axis
                              grid=False))
    )

    # combine all charts
    """
    Below I combined all left and right charts in to one like:
    catch share rate + catch share points + pop str (test) + pop str (control)
    """
    final_chart = alt.layer(right_axis_chart, #combine left and right charts
                            left_axis_chart).resolve_scale(  # this method here is
to make shared or independed of the y axis
        y='independent'
    ).properties(    # here properities for the size of the chart
        width=400,
        height=250,
        title=title_suffix1 # adds title to plot
    )

    return final_chart # ends function for catch share rate
"""
Next function generates chart for "selection curve".
It uses similar structure with the previous function.
Basically I copied and just changed some parts of it.

In this section x and y axes data is different then before:
Y0-X0: Probability curve (main) data.
Y1-X1: Lower 95% confidence limit data
Y2-X2: Upper 95% confidence limit data
"""

### selection curve function
def generate_chart2(sel_cur_paths):  # only changed names in this part, logic is
the same (line 141-145)
    match = re.search(r'sel_cur_krill_(MB\d+)_', sel_cur_paths)
    title_suffix = match.group(1) if match else 'Unknown'
    data = pd.read_excel(sel_cur_paths, header=1)
    data = data.apply(pd.to_numeric, errors='coerce')

    # line chart for selection curve

    chart = alt.Chart(data).mark_line(color='black',                # main curve
                                      clip=True,
                                      size=5).encode(
        x=alt.X('X0:Q',
                scale=alt.Scale(domain=[15, 55]),                   # main x-axis
                title='Length (mm)',
            axis=alt.Axis(values=[15, 25, 35, 45, 55],
                          grid=False,
                          titleFontSize=30,
                          labelFontSize=25,
                          titlePadding=20)),
        y=alt.Y('Y0:Q',                                             # main y-axis
                scale=alt.Scale(domain=[0, 1.01]),
                title='Retention probability',
            axis=alt.Axis(values=[0, 0.25, 0.50, 0.75, 1.00],
                          labelFontSize=25,
                          titleFontSize=30,
                          titlePadding=20,
                          format='.2f',
                          grid=False))

    ) + alt.Chart(data).mark_line(color='black',                    # lower limit
```

```
                                  clip=True,
                                  strokeDash=[5, 5],  #configure dash length and
space
                                  size=2
                                  ).encode(
        x=alt.X('X1:Q',
                scale=alt.Scale(domain=[15, 55])),
        y='Y1:Q'
    ) + alt.Chart(data).mark_line(color='black',                    # upper limit
                                  clip=True,
                                  strokeDash=[5, 5],
                                  size=2
                                  ).encode(
        x=alt.X('X2:Q',
                scale=alt.Scale(domain=[15, 55])),
        y='Y2:Q'
    )
    final_chart = alt.layer(chart).resolve_scale(
        y='independent'
    ).properties(
        width=400,
        height=250,
        title=title_suffix
        )
    return final_chart # this ends the function.

"""
In next function creates and returns a combined chart
by generating individual catch share rate charts and selection curve charts
from specified file paths. It horizontally concatenates each pair of these
charts with independent y-axes but shared x-axes, then vertically stacks these
combined charts to produce a final visualization. This part ensures each
pair of charts is displayed together in a coherent and structured layout.
"""

# combined chart function
def generate_combined_chart(catch_share_paths, sel_cur_paths):
    # generate individual catch share charts for each path
    catch_share_charts = [generate_chart1(path) for path in catch_share_paths]
    # generate individual selection curve charts for each path
    sel_cur_charts = [generate_chart2(path) for path in sel_cur_paths]

    # create a list of combined charts for each pair of catch share and selection
curve charts
    combined_row_charts = [
        alt.hconcat(catch_share,
                    sel_cur,
                    spacing=90).resolve_scale(  # spcaing between charts
            x='shared', y='independent' # in here x axis is shared because we have
one type of y axis data
```

```
        ) for catch_share, sel_cur in zip(catch_share_charts, # zip() takes two
lists and return a tuple
                                          sel_cur_charts)
    ]

    # vertically concatenated all combined row charts
    combined_chart = alt.vconcat(*combined_row_charts, # alt.vconcat() method
vertically concatenate  charts
                                 spacing=50).configure_view(
        stroke=None  # remove the border around the charts
    )

    return combined_chart  # return final combined chart

# file paths - all functions are calling the file paths here ands runs each file in
order
# for file paths please go:
    #
https://github.com/eniskostak/enk/tree/5430375cd7d59082ef7955eb9ce14132a9334312/fin
al_assignment
catch_share_paths = [
    r'C:\Users\eko067\Desktop\My Paper - Presentation
etc\Paper\meso_sel_study\DATA\sup
files\plot_data\c_share\c_share_krill_MB14_22.xlsx',
    r'C:\Users\eko067\Desktop\My Paper - Presentation
etc\Paper\meso_sel_study\DATA\sup
files\plot_data\c_share\c_share_krill_MB20_22.xlsx',
    r'C:\Users\eko067\Desktop\My Paper - Presentation
etc\Paper\meso_sel_study\DATA\sup
files\plot_data\c_share\c_share_krill_MB30_22.xlsx',
]

sel_cur_paths = [
    r'C:\Users\eko067\Desktop\My Paper - Presentation
etc\Paper\meso_sel_study\DATA\sup
files\plot_data\sel_cur\sel_cur_krill_MB14_22.xlsx',
    r'C:\Users\eko067\Desktop\My Paper - Presentation
etc\Paper\meso_sel_study\DATA\sup
files\plot_data\sel_cur\sel_cur_krill_MB20_22.xlsx',
    r'C:\Users\eko067\Desktop\My Paper - Presentation
etc\Paper\meso_sel_study\DATA\sup
files\plot_data\sel_cur\sel_cur_krill_MB30_22.xlsx',
]

# generate the combined chart
combined_chart = generate_combined_chart(catch_share_paths, sel_cur_paths) # calls
functions
combined_chart = combined_chart.properties(
    title='Krill' # sets title as header to the plot
)
```

```
combined_chart = combined_chart.configure_title(fontSize=30) # title font size
# generate charts to html
combined_chart_file = "figure8_exam.html" # assigns the file name in to the
variable
combined_chart.save(combined_chart_file)  # saves the plots as HTML file
webbrowser.open(combined_chart_file)      # opens the saved HTML file in the web
browser to display the chart

"""
I chose it to be an html file because it is easy to comare for me the changes
that I make afterwards by skipping between tabs. Also there is a possibility to
edit it with
Vega Editor.
"""
```