Příloha 1: Skript GP nástroje provádějící výpočet Cropperových hodnot a následnou regresní analýzu a interpolaci, určený pro připojení do widgetu webové aplikace jako GP služba

```
################################### DENDROCHRONOLOGY - TREE RING ANALYSIS ###################################

### IMPORT MODULES ############################################################################################
import arcpy
import os
import tempfile
import shutil
from math import sqrt, floor, ceil
from statistics import median
import collections
import sys
import time
###############################################################################################################

# timer
all_start = time.time()

### ENVIRONMENT ###############################################################################################
# set environment
arcpy.env.overwriteOutput = True
arcpy.env.qualifiedFieldNames = False
###############################################################################################################

### FUNCTIONS #################################################################################################
def C_values_medians_function(dictionary):
    """Calculates Cropper values & medians for one site.

    Input: dictionary of years and corresponding annual ring growths (dict).
    Output: ordered dictionary of years and corresponding medians of Cropper values (dict).
    """
    w = 13
    k = floor(w/2)
    C_vals = {}
    catalog = {}

    for key,value in dictionary.items():    # item == sample
        C = []
        cy = [] # current year index (year that we base the window in)

        for l in range(len(dictionary[key]['years'])):  # create list of usable 'cy's
            cy.append(l)

        for h in range(k):
            cy.remove(h)
        for j in range(k):
            delete = len(value['values'])-1-j
            cy.remove(delete)

        # loops through usable 'cy's
        for i in range(cy[0],cy[-1]+1):    # i is current mid-window year
            mid_year = value['values'][i]   # value of current mid-window year

            sum_val = 0     # sum of all years' values in window
            for ii in range(-k,k+1):
                sum_val += value['values'][i+ii]
            window_mean = sum_val/(2*k+1) # mean of all years' values in window

            sum_q_dev = 0  # sum of deviations
            for iii in range(-k,k+1):
                sum_q_dev += ((value['values'][i+iii]) - window_mean)**2
            stdev_cy = sqrt(sum_q_dev/(2*k)) # two-direction stdev - both sides (w-1)

            c = round((mid_year - window_mean)/stdev_cy,2)   # 2 decimal places
            C.append({value['years'][i]: c})

            year_num = value['years'][i]
            if year_num not in catalog.keys():
                catalog[year_num] = [c]
            else:
                catalog[year_num].append(c)

        C_vals[key] = C

    new_cat = {}
    sorted_catalog = collections.OrderedDict(sorted(catalog.items()))
    for item in sorted_catalog.items():
        if len(item[1]) >= 5:   # new_cat does not include years represented by 4 or less samples
            new_cat[item[0]] = round(median(item[1]),2)
    snew_cat = collections.OrderedDict(sorted(new_cat.items()))
    return snew_cat

def newfield(input_table,f_name,f_type,f_scale):
    """Adds new field with given name, type and scale to the input table.
```

```
    Input: input table (str), field name (str), field type (str), field scale (number of decimal places) (int).
    """
    arcpy.management.AddField(
        in_table=os.path.join(input_table),
        field_name=f_name,
        field_type=f_type,
        field_scale=f_scale)

def range_list(min,max):
    """Creates a list of lists specifying class ranges for reclassification.
    Outer boundaries are according to interpolated raster's min or max value (whichever is absolutely larger).

    Input: minimal raster value (float), maximal raster value (float).
    Output: remap list of lists with class boundaries and values (list), step (class interval) (float).
    """
    if abs(min) > abs(max):
        outer = abs(min)
    else:
        outer = abs(max)

    outer = ceil(outer*100)/100 # round UP to 2 decimal places
    step = 0.15 # fixed step
    border = step*5

    rangeList = []
    # classes from -6 to 6; 0 == 0, -6 & 6 is used for greater than 'border' or -'border', respectively
    if outer > border:
        classList = [-outer,-border,-6]  # 1st class if needed
        rangeList.append(classList)

    a=-border
    c=-5

    for _ in range(11):
        b=a+step
        if c == 0:
            classList=[-0.0001,0.0001,c]    # zero class
        elif c == -1:
            classList=[round(a,2),-0.0001,c]    # next-to-zero class (due to zero's boundaries)
        elif c == 1:
            a=0
            b=a+step
            classList=[0.0001,round(b,2),c]     # next-to-zero class (due to zero's boundaries)
        else:
            classList=[round(a,2),round(b,2),c]

        rangeList.append(classList)
        a+=step
        c+=1

    if outer > border:
        classList = [border,outer,6]     # last class if needed
        rangeList.append(classList)

    return rangeList,step
########################################################################################################################


### USER SELECTION PARAMETERS from JS API frontend ######################################################################
det = arcpy.GetParameterAsText(0)                       # USER INPUT         (bool) - selection of dataset
if det == 'True' or det == 'true':
    data = 'detrend_data'
    dataset_name = "detrendovaná data"
elif det == 'False' or det == 'false':
    data = 'raw_data'
    dataset_name = "nedetrendovaná data"
min_length = arcpy.GetParameterAsText(1)                # USER INPUT      (int) - minimal chronology length
tree_height_min = arcpy.GetParameterAsText(2)           # USER INPUT      (float) - tree height range (m)
tree_height_max = arcpy.GetParameterAsText(3)           # USER INPUT
cr_height_min = arcpy.GetParameterAsText(4)             # USER INPUT      (float) - crown height range (m)
cr_height_max = arcpy.GetParameterAsText(5)             # USER INPUT
dbh_min = arcpy.GetParameterAsText(6)                   # USER INPUT      (float) - diameter at breast height range
(cm) (average...)
dbh_max = arcpy.GetParameterAsText(7)                   # USER INPUT

# export parameter
delim = arcpy.GetParameterAsText(8)                     # USER INPUT      (str) - delimiter for the export ";" or
","

log_selection = f"\n---------- Zvolené parametry výběru: ----------\n-> dataset: {dataset_name}\n-> minimální délka
chronologie: {min_length}\n-> rozsah výšek stromů: {tree_height_min}–{tree_height_max}\n-> rozsah výšek korun:
{cr_height_min}–{cr_height_max}\n-> rozsah průměrů ve výšce hrudi: {dbh_min}–{dbh_max}\n"
arcpy.AddMessage(log_selection)
arcpy.SetParameterAsText(12,log_selection)
########################################################################################################################


### USER ANALYSIS PARAMETERS from JS API frontend ########################################################################
explanatory = arcpy.GetParameterAsText(9)               # USER INPUT      (list) - explanatory variables elevation/
latitude/longitude/slope/aspect... (=> list of strings)
```

```
# if type(explanatory) == str:  # for testing in python terminal only
#     explanatory = explanatory.strip('[]').replace(" ", "").split(',')
polynomial = int(arcpy.GetParameterAsText(10))          # USER INPUT      (int) - order of interpolation polynomial
0/1/2/3          note: although the interpolation allows it to be 'float', only previously mentioned options will
be available
selected_year = arcpy.GetParameterAsText(11)            # USER INPUT      (int) - year to do the regression and
interpolation in

log_analysis = f"\n---------- Zvolené parametry analýzy: ----------\n-> zvolený rok: {selected_year}\n->
vysvětlující proměnné (regrese): {str(explanatory).strip('[]')}\n-> stupeň polynomu (interpolace): {polynomial}.\n"
arcpy.AddMessage(log_analysis)
arcpy.SetParameterAsText(13,log_analysis)
########################################################################################################################


### GET INFO ABOUT DATASET ###############################################################################
connection_file = os.path.join("
","Users","Anička","Škola","3.LS","Bakalářka","DendroApp","DendroConnect","DendroConnect.sde")
select_all = f"""
SELECT DISTINCT site_code,site_name,species,lat_site,lon_site,elev_site,slope_site,asp_site,rooting_depth,soil,
tree_id,dbh,height,crown_height,crown_projection,social_status,sapwood,elev_tree,slope_tree,asp_tree,
sample_id,year,value FROM
(
SELECT
v.site_code,v.site_name,v.species,v.lat_site,v.lon_site,v.elev_site,v.slope_site,v.asp_site,v.rooting_depth,v.soil,
v.tree_id,
v.dbh,v.height,v.crown_height,v.crown_projection,v.social_status,v.sapwood,v.lat_tree,v.lon_tree,fgt.elevation AS
elev_tree,
fgt.slope AS slope_tree,fgt.aspect AS asp_tree,v.sample_id,v.year,v.value FROM fg_tree AS fgt RIGHT JOIN
(
SELECT
x.site_code,x.site_name,x.species,x.lat_site,x.lon_site,x.elev_site,x.slope_site,x.asp_site,x.rooting_depth,x.soil,
x.tree_id,
dbh,height,crown_height,crown_projection,social_status,sapwood,ST_Y(t.geom) AS lat_tree,ST_X(t.geom) AS
lon_tree,x.sample_id,x.year,x.value FROM tree AS t INNER JOIN
(
SELECT y.site_code,y.site_name,y.species,y.lat_site,y.lon_site,elevation AS elev_site,slope AS slope_site,aspect AS
asp_site,rooting_depth,soil,y.tree_id,y.sample_id,y.year,y.value
FROM fg_site AS fgs RIGHT JOIN
(
SELECT s.site_code,site_name,species,ST_Y(s.geom) AS lat_site,ST_X(s.geom) AS lon_site,tree_id,sample_id,year,value
FROM site AS s INNER JOIN
(
SELECT * FROM {data} AS d WHERE (site_code,tree_id,sample_id) IN
(SELECT DISTINCT site_code,tree_id,sample_id FROM {data} GROUP BY site_code,sample_id,tree_id HAVING COUNT(*)>=0)
) AS z
ON z.site_code=s.site_code
) AS y
ON y.site_code=fgs.site_code
) AS x
ON x.site_code=t.site_code AND x.tree_id=t.tree_id
) AS v
ON fgt.site_code=v.site_code AND fgt.tree_id=v.tree_id
) AS vse
"""
arcpy.management.MakeQueryLayer(connection_file,"database_all_tableview",query=select_all)  # makes TableView
arcpy.management.CopyRows("database_all_tableview", os.path.join(arcpy.env.scratchFolder,"db_all.txt")) # adds
OBJECTID
arcpy.management.CopyRows(os.path.join(arcpy.env.scratchFolder,"db_all.txt"), "database_all")   # back to TableView

cursor = arcpy.da.SearchCursor("database_all", ["site_code","sample_id"]) # site_code == row[0], sample_id ==
row[1]
all_sitecodes = []
all_samples = []
for row in cursor:
    if row[0] not in all_sitecodes:
        all_sitecodes.append(row[0])
    if row[1] not in all_samples:
        all_samples.append(row[1])
del cursor
########################################################################################################################


### DENDROCHRONOLOGICAL DATA SELECTION ###################################################################
select = f"""
SELECT DISTINCT site_code,site_name,species,lat_site,lon_site,elev_site,slope_site,asp_site,rooting_depth,soil,
tree_id,dbh,height,crown_height,crown_projection,social_status,sapwood,elev_tree,slope_tree,asp_tree,
sample_id,year,value FROM
(
SELECT
v.site_code,v.site_name,v.species,v.lat_site,v.lon_site,v.elev_site,v.slope_site,v.asp_site,v.rooting_depth,v.soil,
v.tree_id,
v.dbh,v.height,v.crown_height,v.crown_projection,v.social_status,v.sapwood,v.lat_tree,v.lon_tree,fgt.elevation AS
elev_tree,
fgt.slope AS slope_tree,fgt.aspect AS asp_tree,v.sample_id,v.year,v.value FROM fg_tree AS fgt RIGHT JOIN
(
SELECT
x.site_code,x.site_name,x.species,x.lat_site,x.lon_site,x.elev_site,x.slope_site,x.asp_site,x.rooting_depth,x.soil,
x.tree_id,
dbh,height,crown_height,crown_projection,social_status,sapwood,ST_Y(t.geom) AS lat_tree,ST_X(t.geom) AS
```

```
lon_tree,x.sample_id,x.year,x.value FROM tree AS t INNER JOIN
(
SELECT y.site_code,y.site_name,y.species,y.lat_site,y.lon_site,elevation AS elev_site,slope AS slope_site,aspect AS
asp_site,rooting_depth,soil,y.tree_id,y.sample_id,y.year,y.value
FROM fg_site AS fgs RIGHT JOIN
(
SELECT s.site_code,site_name,species,ST_Y(s.geom) AS lat_site,ST_X(s.geom) AS lon_site,tree_id,sample_id,year,value
FROM site AS s INNER JOIN
(
SELECT * FROM {data} AS d WHERE (site_code,tree_id,sample_id) IN
(SELECT DISTINCT site_code,tree_id,sample_id FROM {data} GROUP BY site_code,sample_id,tree_id HAVING
COUNT(*)>={min_length})
) AS z
ON z.site_code=s.site_code
) AS y
ON y.site_code=fgs.site_code
) AS x
ON x.site_code=t.site_code AND x.tree_id=t.tree_id
) AS v
ON fgt.site_code=v.site_code AND fgt.tree_id=v.tree_id
) AS vse

WHERE
(dbh>={dbh_min} AND dbh<={dbh_max})

AND (height>={tree_height_min} AND height<={tree_height_max})
AND (crown_height>={cr_height_min} AND crown_height<={cr_height_max})
"""
arcpy.management.MakeQueryLayer(connection_file,"database_selection_tableview",query=select)    # makes TableView
arcpy.management.CopyRows("database_selection_tableview", os.path.join(arcpy.env.scratchFolder,"db_selection.txt"))
# adds OBJECTID
arcpy.management.CopyRows(os.path.join(arcpy.env.scratchFolder,"db_selection.txt"), "database_selection")   # back
to TableView
arcpy.AddMessage('Data úspěšně načtena z databáze.')
##########################################################################################################################


### GET THE DATA ###########################################################################################
# get sitecodes
cursor = arcpy.da.SearchCursor("database_selection", ["site_code","site_name","species","lat_site","lon_site"]) #
site_code == row[0], site_name == row[1], species == [2], lat_site == [3], lon_site == [4]
sitecodes = []
site_packets = []
for row in cursor:
    if row[0] not in sitecodes:
        sitecodes.append(row[0])
        site_packet = (row[0],(row[4],row[3]),row[1],row[2],row[3],row[4]) # site_packet = (site_code,
(longitude,latitude),site_name,species,latitude,longitude)
        site_packets.append(site_packet)
del cursor

n_sites = len(sitecodes)
log_sites_in = f"Počet zahrnutých lokalit z databáze: {n_sites} z {len(all_sitecodes)}\n(Nemusí být ale zahrnuty
všechny jejich vzorky.)"
arcpy.AddMessage(log_sites_in)
arcpy.SetParameterAsText(14,log_sites_in)

missing_sites = [a for a in all_sitecodes if a not in set(sitecodes)]
if len(missing_sites) != 0:
    log_miss = f"Stanoviště zcela eliminovaná filtry: {str(missing_sites).strip('[]')}"
    arcpy.AddMessage(log_miss)
    arcpy.SetParameterAsText(15,log_miss)

# separate sites in dictionary; keys == site codes
cursor = arcpy.da.SearchCursor("database_selection", ["site_code", "sample_id", "year", "value"]) # site_code ==
row[0], sample_id == row[1], year == row[2], value == row[3]
sites_dict = {}
for row in cursor:  # each row of the table
    for sitecode in sitecodes:  # tries all site codes
        if row[0] == sitecode:  # if current site code fits a site code in list 'sitecodes'
            if sitecode in sites_dict.keys():   # if key-value exists
                if row[1] in sites_dict[sitecode].keys(): # if current sample_id is a key in the dictionary
                    sites_dict[sitecode][row[1]]['years'].append(int(row[2]))   # append year to list 'years'
                    sites_dict[sitecode][row[1]]['values'].append(row[3])       # append value to list 'values'
                else:
                    sites_dict[sitecode][row[1]] = {'years':[int(row[2])], 'values': [row[3]]}
            else:   # create key-value
                record = {row[1]: {'years':[int(row[2])], 'values':[row[3]]}}   # {sample_id: {'years':[year],
'values':[value]}}
                sites_dict[sitecode] = record # add under sitecode key
del cursor
##########################################################################################################################


### GET CALCULATED SITE VALUES INTO DICTIONARY WITH SITECODES ##############################################
# count of sample_id keys of each location
sample_counts = {}
sumnsam = 0
lengths_chrono = []
for sitec in sitecodes:
```

```
    nsam = len(sites_dict[sitec].keys())
    for g in (sites_dict[sitec].values()):
        lengths_chrono.append(len(g['years']))
    sample_counts[sitec] = nsam
    sumnsam += nsam
log_samples_n = f"\nPočet zahrnutých vzorků: {sumnsam} z {len(all_samples)}"
arcpy.AddMessage(log_samples_n)
arcpy.SetParameterAsText(16,log_samples_n)
log_chrono = f"Rozsah délek chronologií vzorků: {min(lengths_chrono)}–{max(lengths_chrono)}"
arcpy.AddMessage(log_chrono)
arcpy.SetParameterAsText(17,log_chrono)

# calculate Cropper values
sites_c_dict = {}
for s_key,s_value in sites_dict.items():
    sites_c_dict[s_key] = C_values_medians_function(s_value)
#########################################################################################################################


### GET AVAILABLE YEARS RANGE ###########################################################################################
yearmin = []
yearmax = []
for site,chronology in sites_c_dict.items():
    if len(list(chronology.items())) != 0:
        yearmin.append(list(chronology.items())[0][0])     # first year of Cropper value medians chronology
(because it is sorted) // (dict.items would return an iterable dict view object rather than a list. We need to wrap
the call onto a list in order to make the indexing possible (via StackOverflow))
        yearmax.append(list(chronology.items())[-1][0])    # last year of Cropper value medians chronology
    else:
        print(f"Stanoviště {site} neobsahuje žádné mediány Cropperových hodnot.")   # print in terminal only
yearMIN = max(yearmin)  # latest first year -> results in only taking into account years that are present in all
sites' chronologies (of sites that went through the filter and was possible to calculate C. values
yearMAX = min(yearmax)  # earliest last year -> dtto

log_all_cropp = f"Rozpětí let, ve kterých byly vypočítány mediány Cropperových hodnot pro všechna neodfiltrovaná
stanoviště: {yearMIN}–{yearMAX}"
arcpy.AddMessage(log_all_cropp)
arcpy.SetParameterAsText(18,log_all_cropp)
#########################################################################################################################


### PREPARE THE TABLE OF SITES ##########################################################################################
# create empty feature class with sites present in selected records
sr = arcpy.SpatialReference(4326)
arcpy.management.CreateFeatureclass(
    out_path=arcpy.env.scratchGDB,
    out_name="sites_empty",
    geometry_type="POINT",
    spatial_reference = sr)
new_sites = os.path.join(arcpy.env.scratchGDB,"sites_empty")

# add new fields: site_code,site_name,species,latitude, longitude, samples_n, Cm{selected_year}
Cm_fieldname = str('Cm' + selected_year)
fieldnames_new = [('site_code','TEXT',0),('site_name','TEXT',0),('species','TEXT',0),('latitude','FLOAT',7),
('longitude','FLOAT',7),('samples_n','SHORT',0),(Cm_fieldname,'FLOAT',2)] # field_scale used in fact only for float
and double
for fn in fieldnames_new:
    newfield(new_sites,fn[0],fn[1],fn[2])

# add sitecode etc. rows to the feature class + coordinates via 'SHAPE@XY'
with arcpy.da.InsertCursor(new_sites, ["site_code","SHAPE@XY","site_name","species","latitude","longitude"]) as
cursor:
    for row in site_packets:    # site_packet = (site_code,
(longitude,latitude),site_name,species,latitude,longitude)
        cursor.insertRow(row)

# add fields with data from the rasters
arcpy.sa.ExtractMultiValuesToPoints(
    in_point_features=new_sites,
    in_rasters=[["https://unce-18.natur.cuni.cz/arcgis/rest/services/elevation/ImageServer","elevation"],
#   >>> URL REFERENCE TO RASTER IMAGERY LAYER (ELEVATION) <<<
    ["https://unce-18.natur.cuni.cz/arcgis/rest/services/slope/ImageServer","slope"],
#   >>> URL REFERENCE TO RASTER IMAGERY LAYER (SLOPE) <<<
    ["https://unce-18.natur.cuni.cz/arcgis/rest/services/aspect/ImageServer","aspect"]])
#   >>> URL REFERENCE TO RASTER IMAGERY LAYER (ASPECT) <<<
#########################################################################################################################


### FILL IN CROPPER VALUES & SAMPLE COUNTS FROM DICTIONARY ##############################################################
sites_with_C = []
sites_without_C = []
with arcpy.da.UpdateCursor(new_sites, ["site_code", Cm_fieldname, "samples_n"]) as cursor: # site_code == row[0],
C{selected_year} == row[1], samples_n == row[2]
    for row in cursor:
        for ss_key,ss_value in sites_c_dict.items():    # for each record (sitecode)
            if row[0] == ss_key:    # if site code of the row == current site code
                if int(selected_year) in ss_value:   # if current year has C. value
                    C_selected_year = ss_value[int(selected_year)]
                    if C_selected_year is not None: sites_with_C.append(ss_key)
                    row[1] = C_selected_year    # fill Cropper value median of the selected year to the
```

```
"C{selected_year}" field of the table
                    row[2] = sample_counts[row[0]]
                else:
                    sites_without_C.append(ss_key)
        cursor.updateRow(row)
if len(sites_without_C) != 0: arcpy.AddWarning(f"Stanoviště {str(sites_without_C).strip('[]')} neobsahují
Cropperovu hodnotu pro rok {selected_year}.\n")
log_sites_to_analysis = f"Počet stanovišť s vypočítaným mediánem Cropperových hodnot v roce {selected_year}:
{len(sites_with_C)}\nStanoviště vstupující do regrese: {str(sites_with_C).strip('[]')}"
arcpy.AddMessage(log_sites_to_analysis)
arcpy.SetParameterAsText(19,log_sites_to_analysis)

if len(sites_with_C) < 10:  # terminate analysis if not enough values for relevance     >>> CHANGE HERE THE MINIMAL
NUMBER OF VALUES GOING INTO REGRESSION <<<
    arcpy.AddError(f"Příliš málo stanovišť s mediány Cropperových hodnot za rok {selected_year} pro relevantní
analýzu. Zvolte prosím větší vzorek dat nebo jiný rok.")
    exit()
##########################################################################################################################


### PREPARATION FOR ANALYSIS ###############################################################################################
arcpy.management.MakeFeatureLayer("https://unce-18.natur.cuni.cz/arcgis/rest/services/Hosted/prediction_points/
FeatureServer/0","point_net_lyr")    #   >>> URL REFERENCE TO VECTOR FEATURE LAYER (PREDICTION POINTS) <<<
##########################################################################################################################


### ANALYSIS ###############################################################################################################
in_features = new_sites

prediction_locations = "point_net_lyr"

# Generalized Linear Regression
arcpy.stats.GeneralizedLinearRegression(
    in_features=in_features,
    prediction_locations=prediction_locations,
    dependent_variable=Cm_fieldname,
    model_type="CONTINUOUS",
    output_features=os.path.join(arcpy.env.scratchGDB, "regression_output"),
    output_predicted_features=os.path.join(arcpy.env.scratchGDB, "regression_output_predicted"),
    explanatory_variables=explanatory)

log_regression = arcpy.GetMessages()
arcpy.AddMessage(log_regression)
arcpy.SetParameterAsText(20,log_regression)
arcpy.AddMessage("Obecná lineární regrese úspěšně proběhla.")

# Local Polynoial Interpolation
search = arcpy.ga.SearchNeighborhoodStandardCircular(
    radius=0.0045,   # approximately equal to 500 meters
    angle=0,
    sectorType="ONE_SECTOR")

arcpy.ga.LocalPolynomialInterpolation(
    in_features=os.path.join(arcpy.env.scratchGDB, "regression_output_predicted"),
    z_field="PREDICTED",
    out_raster=os.path.join(arcpy.env.scratchFolder,"interpol_r"),   # >> raster output <<
    cell_size=0.0009, # approximately equal to 100 meters
    power=polynomial,
    search_neighborhood=search,
    kernel_function="GAUSSIAN",
    output_type="PREDICTION")

output_raster_interpolated = os.path.join(arcpy.env.scratchFolder,"interpol_r")
arcpy.SetParameterAsText(22,output_raster_interpolated)

arcpy.AddMessage("Lokální polynomická interpolace úspěšně proběhla.")
##########################################################################################################################


### DEVIATIONS FROM DATAPOINTS #############################################################################################
# fill in predicted values
pred_name = str('pred' + selected_year)
arcpy.sa.ExtractMultiValuesToPoints(
    in_point_features=new_sites,
    in_rasters=[[os.path.join(arcpy.env.scratchFolder,"interpol_r"),pred_name]])
arcpy.management.CalculateField(
    in_table=new_sites,
    field=pred_name,
    expression=f"round(!{pred_name}!,3)",
    expression_type="PYTHON3")

# calculate & fill in deviations
dev_name = str('dev' + selected_year)
newfield(new_sites,dev_name,'FLOAT',2)
difference=f"round(!{Cm_fieldname}!,3)-!{pred_name}!"     # PYTHON3 !Cm1997!-!pred1997!
arcpy.management.CalculateField(
    in_table=new_sites,
    field=dev_name,
    expression=difference,
    expression_type="PYTHON3")
```

```
arcpy.AddMessage("Proběhl výpočet odchylek odhadovaných dat od původních dat.")
#########################################################################################################################


### RASTER STATISTICS ###################################################################################################
rasterMINobject = arcpy.management.GetRasterProperties(os.path.join(arcpy.env.scratchFolder,"interpol_r"),
"MINIMUM")
rasterMAXobject = arcpy.management.GetRasterProperties(os.path.join(arcpy.env.scratchFolder,"interpol_r"),
"MAXIMUM")
rasterMIN = str(rasterMINobject.getOutput(0)).replace(',','.')
rasterMAX = str(rasterMAXobject.getOutput(0)).replace(',','.')
arcpy.AddMessage(f"Minimální hodnota interpolovaného rasteru: {rasterMIN}")
arcpy.AddMessage(f"Maximální hodnota interpolovaného rasteru: {rasterMAX}")
#########################################################################################################################


### RASTER RECLASSIFICATION #############################################################################################
inRaster = os.path.join(arcpy.env.scratchFolder,"interpol_r")
reclassField="Value"
ranges,interval_step = range_list(float(rasterMIN),float(rasterMAX))
remap = arcpy.sa.RemapRange(ranges)

outReclassify = arcpy.sa.Reclassify(
    in_raster=inRaster,
    reclass_field=reclassField,
    remap=remap,
    missing_values="NODATA")
outReclassify.save(os.path.join(arcpy.env.scratchGDB, "reclassified"))

arcpy.conversion.RasterToPolygon(
    in_raster=os.path.join(arcpy.env.scratchGDB, "reclassified"),
    out_polygon_features=os.path.join(arcpy.env.scratchGDB,"interpolated_polygons"),   # >> polygons for rendering
<<
    simplify="NO_SIMPLIFY",
    create_multipart_features="MULTIPLE_OUTER_PART")

output_polygons_reclassified = os.path.join(arcpy.env.scratchGDB,"interpolated_polygons")
arcpy.SetParameterAsText(23,output_polygons_reclassified)

log_interpretation = f"\nInterpretace vykreslených polygonů:\nKaždá třída vyjadřuje interval hodnot (predikovaných
mediánů Cropperových hodnot) o šířce {round(interval_step,3)}.\nTřída 0 vyjadřuje nulové hodnoty, případná třída 6,
resp. -6 vyjadřuje kladné nebo záporné odlehlé hodnoty vyšší než {round(5*interval_step,2)}, resp. nižší než {-
round(5*interval_step,2)}.\n"
arcpy.AddMessage(log_interpretation)
arcpy.SetParameterAsText(21,log_interpretation)
#########################################################################################################################


### OUTPUTS #############################################################################################################
# output point features
arcpy.management.CopyFeatures(new_sites,os.path.join(arcpy.env.scratchFolder,"sites.shp"))
output_sites_shp = os.path.join(arcpy.env.scratchFolder,"sites.shp")
arcpy.SetParameterAsText(24,output_sites_shp)
#########################################################################################################################


### CLEAN UP ############################################################################################################
arcpy.management.Delete(os.path.join(arcpy.env.scratchFolder,"db_all.txt"))
arcpy.management.Delete(os.path.join(arcpy.env.scratchFolder,"db_selection.txt"))
#########################################################################################################################

# timer
all_end = time.time()
print(f"Uplynulý čas: {all_end - all_start}")
```

Příloha 2: Možnost definování symbologie výstupní polygonové vrstvy s využitím JSON rendereru

```
symbology_polygons = """{
    "type" : "uniqueValue",
    "field1" : "gridcode",
    "field2" : null,
    "field3" : null,
    "fieldDelimiter" : ", ",
    "defaultSymbol" : {
        "type" : "esriSFS",
        "style" : "esriSFSSolid",
        "color" : [130,130,130,255],
        "width" : 1
        "outline": {
            "type": "esriSLS",
            "style": "esriSLSSolid",
            "color": [0,0,0,255],
            "width": 1
            }
        },
    "defaultLabel" : "Ostatní hodnoty",
    "uniqueValueInfos" : [
      { "value" : "-6",
        "label" : "Třída -6",
        "description" : "Třída -6",
        "symbol" :
        { "type" : "esriSFS",
          "style" : "esriSFSSolid",
          "color" : [255,0,0,255],
          "outline": {
            "type": "esriSLS",
            "style": "esriSLSSolid",
            "color": [0,0,0,255],
            "width": 1
            }
        }
      },
      { "value" : "-5",
        "label" : "Třída -5",
        "description" : "Třída -5",
        "symbol" :
        { "type" : "esriSFS",
          "style" : "esriSFSSolid",
          "color" : [255,95,0,255],
          "outline": {
            "type": "esriSLS",
            "style": "esriSLSSolid",
            "color": [0,0,0,255],
            "width": 1
            }
        }
      },
      { "value" : "-4",
        "label" : "Třída -4",
        "description" : "Třída -4",
        "symbol" :
        { "type" : "esriSFS",
          "style" : "esriSFSSolid",
          "color" : [255,140,0,255],
          "outline": {
            "type": "esriSLS",
            "style": "esriSLSSolid",
            "color": [0,0,0,255],
            "width": 1
            }
        }
      },
      { "value" : "-3",
        "label" : "Třída -3",
        "description" : "Třída -3",
        "symbol" :
        { "type" : "esriSFS",
          "style" : "esriSFSSolid",
          "color" : [252,177,26,255],
          "outline": {
            "type": "esriSLS",
            "style": "esriSLSSolid",
            "color": [0,0,0,255],
            "width": 1
            }
        }
      },
      { "value" : "-2",
        "label" : "Třída -2",
        "description" : "Třída -2",
        "symbol" :
        { "type" : "esriSFS",
          "style" : "esriSFSSolid",
          "color" : [249,210,72,255],
          "outline": {
            "type": "esriSLS",
            "style": "esriSLSSolid",
            "color": [0,0,0,255],
            "width": 1
            }
        }
      },
      { "value" : "-1",
        "label" : "Třída -1",
        "description" : "Třída -1",
        "symbol" :
        { "type" : "esriSFS",
          "style" : "esriSFSSolid",
          "color" : [241,224,79,255],
          "outline": {
            "type": "esriSLS",
            "style": "esriSLSSolid",
            "color": [0,0,0,255],
```

```
      "width": 1
      }
  }
},
{ "value" : "0",
  "label" : "Třída 0",
  "description" : "Třída 0",
  "symbol" :
  { "type" : "esriSFS",
    "style" : "esriSFSSolid",
    "color" : [243,248,102,255],
    "outline": {
      "type": "esriSLS",
      "style": "esriSLSSolid",
      "color": [0,0,0,255],
      "width": 1
      }
  }
},
{ "value" : "1",
  "label" : "Třída 1",
  "description" : "Třída 1",
  "symbol" :
  { "type" : "esriSFS",
    "style" : "esriSFSSolid",
    "color" : [231,238,89,255],
    "outline": {
      "type": "esriSLS",
      "style": "esriSLSSolid",
      "color": [0,0,0,255],
      "width": 1
      }
  }
},
{ "value" : "2",
  "label" : "Třída 2",
  "description" : "Třída 2",
  "symbol" :
  { "type" : "esriSFS",
    "style" : "esriSFSSolid",
    "color" : [220,251,103,255],
    "outline": {
      "type": "esriSLS",
      "style": "esriSLSSolid",
      "color": [0,0,0,255],
      "width": 1
      }
  }
},
{ "value" : "3",
  "label" : "Třída 3",
  "description" : "Třída 3",
  "symbol" :
  { "type" : "esriSFS",
    "style" : "esriSFSSolid",
    "color" : [194,252,82,255],
    "outline": {
      "type": "esriSLS",
      "style": "esriSLSSolid",
      "color": [0,0,0,255],
      "width": 1
      }
  }
},
{ "value" : "4",
  "label" : "Třída 4",
  "description" : "Třída 4",
  "symbol" :
  { "type" : "esriSFS",
    "style" : "esriSFSSolid",
    "color" : [160,253,62,255],
    "outline": {
      "type": "esriSLS",
      "style": "esriSLSSolid",
      "color": [0,0,0,255],
      "width": 1
      }
  }
},
{ "value" : "5",
  "label" : "Třída 5",
  "description" : "Třída 5",
  "symbol" :
  { "type" : "esriSFS",
    "style" : "esriSFSSolid",
    "color" : [115,254,46,255],
    "outline": {
      "type": "esriSLS",
      "style": "esriSLSSolid",
      "color": [0,0,0,255],
      "width": 1
      }
  }
},
{ "value" : "6",
  "label" : "Třída 6",
  "description" : "Třída 6",
  "symbol" :
  { "type" : "esriSFS",
    "style" : "esriSFSSolid",
    "color" : [0,255,35,255],
    "outline": {
      "type": "esriSLS",
      "style": "esriSLSSolid",
      "color": [0,0,0,255],
      "width": 1
      }
  }
},
  ],
}"""
symbology_renderer =
f"JSONRENDERER={symbology_polygons}
```