PaleolithicGreening

Supplement for the publication: **Where the grass is greener: Estimated vegetation periods and their explanatory potential for site distribution in the East European Plain**

Authors: Florian Linsel, Louise Tharandt

Date: 03/31/2022

Description:

> Our study relies on the following work routine. This supplement is written in R. One of our primary goals is to perform all functions provided. The documentation and the complete workflow are published. All consecutive analyses can follow the exemplary procedure. This script contains the main functionality of our research, in which all parameters needed for conducting the functions used in scripts, libraries and packages are introduced. Our project is reproducible, which is easily done by running the following script (main.R).

Workflow

To start, open the **main.R** script in the VegPer_R_Scripts folder, select all and run the script. From this main.R script all other scripts are invoked and run. To see the progress of the funcions and script, check the folder **VegPer_Results** (see section folder structure) and view the created csv files. Depending on the computer this code runs on and the number of data that is used, the results can appear after a while.

To see the vegetation period pull factors in a geographical context, the next step is to open the **PaleolithicGreeningQGIS.qgz** file with QGIS (3.10). Select the Python Console once your QGIS file has opened. Next click on the notepad with pen icon to show the Python editor. If the Python files are not visible, go to the folder symbol to open the Python scripts nodes.py, links.py and print.py (see folder structure below). Select the nodes layer in the layer browser to the left, then run the nodes.py script. Repeat with the links.py script (select the layer and run the script). The last step in QGIS is to run the print.py script. Nothing has to be selected, just run the Python script and the resulting vegetation period pull factor images will be exported to the folder extdata/QGIS.

To get a better overview and be able to compare the pull factors of spring and fall in each period, use the **plot_2x2.R** script. This will combine the single plot images, that were exported from QGIS. Open the R script, select all and press run. The results can be fund in the folder extdata/2x2.

Notes

Please cite the use of this script as:

Code

The following scripts were used to automatically calculate the vegetation periods and pull factors over multiple stadials and interstadials for BlackSea and NGRIP data sets.

---

main.R

Variables

parameter : LGM, Black Sea, NGRIP

LGM : base data file

period : GS-3, GI-3, GS-4, GI-4, GS-5.1, GI-5.1, GS-5.2, GI-5.2, GS-6, GI-6, GS-7, GI-7, GS-8, GI-8, GS-9, GI-9

month : 1 through 12 (1:12)

step_day_size : 2, 7, 14, 21, 28

step_list : 20, 60, 47.5, 65, 2.5

df_result_result : create new data frame

Details

As one of the most important variables to distinguish, the three different data files (LGM, Black Sea, NGRIP) are listed in the classification **parameter**.

The base data file is assigned to **LGM** from which all subsequent results stem.

There are 16 different time periods, eight stadial and eight interstadial periods (GS-3, GI-3, GS-4, GI-4, GS-5.1, GI-5.1, GS-5.2, GI-5.2, GS-6, GI-6, GS-7, GI-7, GS-8, GI-8, GS-9, GI-9), which are assigned to **period**.

The parameter **month** is assigned with the number of months of a year, to show the change in temperature over the period of a year.

To show the transition of the vegetation period and temperature over an area in a time frame, the parameter **step_day_size** (2,7,14,21,28) is used.

To define and edit the coordinates as well as the distance between the coordinate points, the parameter **step_list** (20, 60, 47.5, 65, 2.5) is used. Latidute south to north: 47.5 to 65, longitude west to east: 20 to 60, distance: 2.5

df_result_result creates a data frame to build a new and better table, filtering the vegetation period table results.

Folder structure and contents

> this folder structure has been built to clearly structure the data input and the results that are generated through this script.

- Base Data:
    - VegPer_Base_Data/LGM_Base_Data.csv
    - VegPer_Base_Data/Period_BlackSea.csv
    - VegPer_Base_Data/Period_NGRIP.csv
    - VegPer_Base_Data/QGIS_ID_File.csv
    - VegPer_Base_Data/QGIS_ID_File.xlsx
- QGIS
    - pycache
    - exdata (Result Images from plotting the Vegetation Period in QGIS)
        - 2x2
        - QGIS
    - geographical_data (QGIS geographical layers)
    - layout_data (QGIS symbology and layout files)
    - link_node_data (QGIS node files)
    - links (QGIS link files for all parameters - LGM, BlackSea, NGRIP)
    - python_scripts
        - nodes.py
        - links.py
        - print.py
    - rasterfiles (Elevation map)
    - sites (archaeological sites)
    - PaleolithicGreeningQGIS.qgz
- Scripts
    - main.R
    - Calc21Day.R
    - CalcStepsStandardised.R
    - CalcFilter.R
    - plot_2x2.R
- Result Folder Structure
    - VegPer_Results/21Day_BlackSea_Results
    - VegPer_Results/21Day_LGM_Results
    - VegPer_Results/21Day_NGRIP_Results
    - VegPer_Results/Calc_Step_Results
        - VegPer_Results/Calc_Step_Results/Calc_Step_BlackSea
        - VegPer_Results/Calc_Step_Results/Calc_Step_LGM
        - VegPer_Results/Calc_Step_Results/Calc_Step_NGRIP
    - VegPer_Results/Filter_Results
        - VegPer_Results/Filter_Results/Filter_Results_BlackSea
            - VegPer_Results/Filter_Results/Filter_Results_BlackSea/nodes
        - VegPer_Results/Filter_Results/Filter_Results_LGM
            - VegPer_Results/Filter_Results/Filter_Results_LGM/nodes

- VegPer_Results/Filter_Results/Filter_Results_NGRIP
  - VegPer_Results/Filter_Results/Filter_Results_NGRIP/nodes

```
#libraries

library(tidyverse)
library(dplyr)
library(purrr)

# links to the required R scripts
setwd(dirname(rstudioapi::getSourceEditorContext()$path))
source("../VegPer_R_Scripts/Calc21Day.R")
source("../VegPer_R_Scripts/CalcStepsStandardised.R")
source("../VegPer_R_Scripts/CalcFilter.R")
source("../VegPer_R_Scripts/plot_2x2.R")

# set workspace
setwd("../") #reference inside of the github / working directory folder

# global parameter:
parameter <- c("BlackSea" ,"NGRIP", "LGM") #name of the period data sets

# create new directories and folders for the results
for (p in parameter){
  dir.create(file.path(paste0('VegPer_Results/')),
  showWarnings = FALSE)
  dir.create(file.path(paste0('VegPer_Results/Calc_Step_Results/Calc_Step_',p)),
  showWarnings = FALSE)
  dir.create(file.path(paste0('VegPer_Results/21Day_',p,'_Results')),
  showWarnings = FALSE)
  dir.create(file.path(paste0('VegPer_Results/Filter_Results')),
  showWarnings = FALSE)
  dir.create(file.path(paste0('VegPer_Results/Filter_Results/Filter_Results_',p)),
  showWarnings = FALSE)

dir.create(file.path(paste0('VegPer_Results/Filter_Results/Filter_Results_',p,'/no
des')),
  showWarnings = FALSE)
}

# input files
# LGM base data that will be used to add factors to get the temperature data of
stadial and interstadial periods
LGM <- read.csv("VegPer_Base_Data/LGM_base_data.csv", header = T,
fileEncoding="UTF-8-BOM")

# list of stadial and interstadial climatic periods of interest:
period <- c("GS-3", "GI-3", "GS-4", "GI-4", "GS-5.1", "GI-5.1", "GS-5.2", "GI-
5.2", "GS-6", "GI-6", "GS-7", "GI-7", "GS-8", "GI-8", "GS-9", "GI-9")
```

```
### Calc21Day.R

# list of months
month <- c(1:12)

# function to calculate the 21 day average of all periods and data sets
lapply(parameter,DAY_21,period,month)


### CalcStepsStandardised.R

# list of parameters needed for the step-function:
step_day_size <- c(2,7,14,21,28)

# list of coordinates:
step_list <- c(20, 60, 47.5, 65, 2.5)

# set workspace and folders for results
setwd("VegPer_Results/")

# function to calculate the steps between each coordinate node of all periods and
data sets:
lapply (parameter,stepfunction,period,step_day_size)


### CalcFilter.R

setwd("../")
setwd("VegPer_Results/")

df_result_result <- data.frame()

# function to filter the calculated results of all periods and data sets:
lapply (parameter,VegPer_Results,period,df_result_result)


#### create the plots with Python and QGIS ####
```

Calc21Day.R

> The base temperature LGM (last glacial maximum) data is used to calculate the vegetation period during the LGM and between stadial periods (GS-3 until GS-9) as well as interstadial periods (GI-3 until GI-9). Two different period correction factor tables are used, which calculate the vegetation period with NGRIP data and with Black Sea data in the LGM and stadial/interstadial periods. The temperature data is then filtered to display the vegetation period of each time period. First the mean of 3 temperatures has to be equal or above 5 degrees Celsius. The temperature data is then filtered and shows each day,

> that has a mean temperature >= 5 °C. The vegetation period start and end is defined as 21 consecutive days with a mean temperature of equal or above 5 degrees Celsius. The new temperature data is filtered and shows the start and end of the vegetation periods for every time period.

Calc21Day Functions

`calc_21`

`period_correction`

`Day_21`

Details Calc21Day Functions

The function `Day_21` sets up the basic calculation and code for the vegetation period data. The base LGM (last glacial maximum) data shows the temperature each day of the year (*Temp*) as well as the mean temperature in another column (*Temp_mean*). This file is imported and renamed **LGM_temp** to work with and and change the file, without corrupting the main data file. The function `Day_21` sets up the calculation of the base LGM data first, to have **LGM_temp** run through the `calc_21` function and afterwards use the calculated Vegetation data for LGM to then add the season factors (`period_correction` function) to calculate the vegetation periods for the defined interstadial and stadial time periods.

With `calc_21` the start and end of the vegetation period is calculated. We have set the start and end of the vegetation period as 21 days of consecutive temperatures above 5 degrees Celsius. `calc_21` uses two subsets which define if the mean temperature was above or below 5 degrees Celsius for 21 consecutive days before or after the currently checked Day of Year. The function runs trough the column *Veg* of the sorted data (LGM_temp) and shows the vegetation period of a temperature mean of 5 degrees Celsius for at least 21 consecutive days by depicting either a 1 (if at least 21 consecutive days) or 0 (if less than 21 consecutive days) in the column *Veg.21Day*.

The function `period_correction` uses the base data (LGM), which has the calculated mean temperature, and sorts the data to show if the mean temperature is above or below 5 degrees Celsius. It is sorted by depicting either a 1 (if temperature is over 5 degrees) or 0 (if temperature is below 5 degrees) into the column *Veg*. The function also adds the period correction factor data (period_correction) from NGRIP and Black Sea to the base data (LGM). Afterwards the `calc_21` function is used to sort and show the vegetation period of the new data with the added season factors.

```
# generate the vegetation period for LGM

calc_21 <- function (LGM_temp) {
 i = 1
 while (i < nrow(LGM_temp)){ # all rows will be calculated (1-49640)

  if(LGM_temp[i,]$Temp_mean >= 5){
   LGM_temp[i,]$Veg = 1
  }else {
   LGM_temp[i,]$Veg = 0
```

```
  }

  test1 <- subset(LGM_temp, ID >= i & ID < i+21)
  test2 <- subset(LGM_temp, ID <= i & ID > i-21)

  if(sum(test1$Veg == 1) == 21 || sum(test2$Veg == 1) == 21){
   LGM_temp[i,]$Veg.21Day = 1
  }else { #
   LGM_temp[i,]$Veg.21Day = 0
  }

  i=i + 1
 }
 return (LGM_temp)
}

# generate new temperature data and vegetation periods for BlackSea and NGRIP

period_correction <- function (x,p,month) {

  LGM_temp <- LGM # the existing LGM data is assigned to a new object, so the
original data won't be manipulated

  LGM_temp$Period = x # the period list is assigned

  period_corr <- read.csv(paste("VegPer_Base_Data/Period_", p, ".csv", sep=""),
header =T,  check.names = F)
  a <- period_corr[period_corr[,"Time"]==x,]

  for(y in month) {
   b <- a[,paste(y)]
   LGM_temp[LGM_temp[,"Month"]==y,]$Temp = LGM_temp[LGM_temp[,"Month"]==y,]$Temp
+b
   LGM_temp[LGM_temp[,"Month"]==y,]$Temp_mean =
LGM_temp[LGM_temp[,"Month"]==y,]$Temp_mean +b
  }

  i = 1

  while (i < nrow(LGM_temp)){
   if(LGM_temp[i,]$Temp_mean >= 5){
    LGM_temp[i,]$Veg = 1
   }else {
    LGM_temp[i,]$Veg = 0
   }

   i=i + 1
  }

  # calc_21 function for the data sets for BlackSea and NGRIP

  i = 1
  while (i < nrow(LGM_temp)){
```

```
   test1 <- subset(LGM_temp, ID >= i & ID < i+21)
   test2 <- subset(LGM_temp, ID <= i & ID > i-21)


   if(sum(test1$Veg == 1) == 21 || sum(test2$Veg == 1) == 21){
    LGM_temp[i,]$Veg.21Day = 1
   }else {
    LGM_temp[i,]$Veg.21Day = 0
   }

   i=i + 1
  }

  # export path
  path <- paste("VegPer_Results/21Day_", p, "_Results/",sep="")

  # export the data in the for loop as new .csv file with the same period name as
the period factor that was used
  write.csv(LGM_temp, paste(path, x,".csv",sep=""))
}

# start the function to generate new temperature data and vegetation periods for
LGM, BlackSea and NGRIP

DAY_21 <- function(p,period,month){
 LGM <- read.csv("VegPer_Base_Data/LGM_base_data.csv", header = T,
fileEncoding="UTF-8-BOM")

 if (p == "LGM") {

  path <- paste("VegPer_Results/21Day_", p, "_Results/",sep="")
  LGM_temp <- LGM
  LGM_temp <- calc_21(LGM_temp)
  write.csv(LGM_temp, paste(path,'21Day_',p, ".csv",sep=""))

 }else{

  # calculate new temperature data for BlackSea and NGRIP with the
period_correction function
  lapply(period,period_correction,p,month)
 }
}
```

CalcStepStandardised.R

> After finalising the vegetation period results, the data is divided into two main datasets – the spring/summer and the autumn/winter sets. While the spring/summer data is used to estimate pull factors towards a later greening, the autumn/winter data aims at estimating a later end of the fall. The

> basic rule to determine the movement is a step function, which is represented each by drawing a directed link between two nodes. The estimated begin respectively end of the vegetation period is regarded in this calculation and determines whether a link is defined. Each link is directed and can show the principle movement directory. The minimum threshold between two points is defined as two days.

CalcStepStandardised Functions

`stepping`

`QGIS_ID`

`stepfunction`

Details CalcStepStandardised Functions

The `stepfunction` seperate the LGM and BlackSea/NGRIP datasets and sets up the function `stepping` for LGM, NGRIP and BlackSea, with the *step_day_size* (2,7,14,21,28).

The function `stepping` calculates the vegetation change and the route of transmission of the fauna migration in spring and autumn, by seperating the data into to sets: spring/summer and fall/winter. It will calculate the amount of days between coordinate nodes and vegetation period starts end ends.

To define the coordinate nodes and step directions for the images in QGIS the function `QGIS_ID` sets up the calculation for ID points.

```
# calculate the vegetation change and the route of transmission in spring and
autumn

stepping <- function  (csv_file, long_west, long_east, lati_bottom, lati_top,
step, step_day_size, p){

  tbl <-
    list.files(pattern = paste (csv_file,".csv", sep= '')) %>%
    map_df(~read_csv(.))


  # the for loop starts with the set up of the stadial or interstadial that is
calculated

  temp_data <- tbl

  #long_west = 20, long_east = 60, lati_bottom = 47.5, lati_top = 65, step = 2.5

  VegPeriod_complete <- tbl[tbl$Veg.21Day == 1,]
  df_result <- data.frame(c("a","b","c","d","e","f","g","h"))
  df_result_max <- data.frame(c("a","b","c","d","e","f","g","h"))

  text =''
```

```
  # calculation of the min, max, sum and mean of DOY and sum of temperatures

  for (long in seq(from = long_west, to = long_east, step)){ # runs through every
longitude

    #calculation of the vegetation change and the route of transmission of the
fauna

    # migration in spring
    for (lat in seq(from = lati_bottom, to = lati_top, step)){ # runs through
every latitude
      result <- VegPeriod_complete[VegPeriod_complete$Long == long &
VegPeriod_complete$Lat == lat,] #filtering point
      data <- min(result$DOY, na.rm=T)
      for (xlong in seq(from = -step, to = step, step)){
        for (ylati in seq(from = 0, to = step, step)){
          for (a in step_day_size){

            test <- VegPeriod_complete[VegPeriod_complete$Long == long + xlong &
VegPeriod_complete$Lat == lat + ylati,]
            data_test <- min(test$DOY, na.rm=T)
            if (data_test == Inf){break
            } else if (data == -Inf){break
            } else if (data == 0){break
            } else if (data == data_test+1){break
            } else if (data == data_test){break
            } else if (data <= data_test-a)
              {stats <- c(long,lat,long+xlong,lat+ylati,data_test-data,a,QGIS_ID
(lat,long),QGIS_ID (lat+ ylati,long + xlong))

            } else {
              break
            }
            df_result <- cbind (df_result, stats)

          }
        }
      }

      # migration in autumn
      data_max <- max(result$DOY, na.rm=T)
      for (xlong in seq(from = -step, to = step, step)){
        for (ylati in seq(from = -step, to = 0, step)){
          for (a in step_day_size){

            test <- VegPeriod_complete[VegPeriod_complete$Long == long + xlong &
VegPeriod_complete$Lat == lat + ylati,]
            data_test_max <- max(test$DOY, na.rm=T)
            if (data_max == -Inf){ break
            } else if (data_max == Inf){break
            } else if (data_max == 0){break
            } else if (data_max == data_test_max){break
```

```
            } else if (data_max == data_test_max-1){break
            } else if (data_max <= data_test_max-a)
              {stats_max <- c(long,lat,long+xlong,lat+ylati,data_test_max-
data_max,a,QGIS_ID (lat,long),QGIS_ID (lat+ ylati,long + xlong))

            } else {
              break
            }

            df_result_max <- cbind (df_result_max, stats_max)

          }
        }
      }
    }
    if (p == "NGRIP" && csv_file == "GS-6" && lat == 65 && long == 20) {stats_max
<- ''}
    df_result <- cbind (df_result, stats)
    df_result_max <- cbind (df_result_max, stats_max)
  }

  df_result_min <- df_result %>% t(.) %>% data.frame(.) %>% .[-c(0,1),]
  df_result_max <- df_result_max %>% t(.) %>% data.frame(.) %>% .[-c(0,1),]
  rownames(df_result_min) <- NULL
  rownames(df_result_max) <- NULL


  df_result_min [0,1] <- paste('"', df_result_min [0,1], sep ="")
  df_result_max [0,1] <- paste('"', df_result_min [0,1], sep ="")


  path <- paste("../Calc_Step_Results/Calc_Step_",p,"/",sep="")
  x = "export"

  write.csv(df_result_min, paste(path,p,'_',csv_file,"_min.csv",sep=""), row.names
= F, col.names=F)
  write.csv(df_result_max, paste(path,p,'_',csv_file,"_max.csv",sep=""), row.names
= F, col.names=F)
}

# define the coordinate nodes and step directions

QGIS_ID <- function (lat,long) {
  QGIS_ID <- read_csv('../../VegPer_Base_Data/QGIS_ID_File.csv')
  val <- paste(long,'/',lat,sep='')
  QGIS_ID$Target == val
  out <- QGIS_ID[QGIS_ID$Target == val,]$ID
  return (out)
}

# seperate the LGM and BlackSea/NGRIP datasets and sets up the function stepping

stepfunction <- function(p,period,step_day_size){
```

```
  if (p == "LGM") {
    period = "21Day_LGM"

    adress <- paste0("21Day_",p,"_Results",sep="")
    setwd(adress)
    stepping (period, step_list [1], step_list [2], step_list [3], step_list [4],
step_list [5],step_day_size,p)

  }else {

    adress <- paste0("21Day_",p,"_Results",sep="")
    setwd(adress)

    lapply(period, step_list [1], step_list [2], step_list [3], step_list [4],
step_list [5],step_day_size,p)
    setwd("../")
  }
}
```

CalcFilter.R

> The data created with the R script 21Day_ containing new temperature data with the use of the different period correction factor data tables as well as the creation of Vegetation periods, is now being sorted and compacted to show the result in a clearer overview. It will also be used to create nodes, which are necessary to portray the vegetation movement in QGIS.

CalcFilter Functions

`publication`

`interstadial`

`VegPer_Results`

Details CalcFilter Functions

`VegPer_Results` separates the LGM file and the NGRIP/BlackSea files and sets up the calculation of the `interstadial` and `publication` script.

`publication` loops over all newly created data and filters the results of the first day of the vegetation period *Start veg. day*, the last day of the Vegetation Period *End veg. day*, the sum of days within the Vegetation Period *Veg. sum day* and the sum of the temperatures within the Vegetation Period *Sum of temperature*.
The function `publication` also creates the nodes, which are used for the stepping function (**CalcStep standardised**) and are necessary for the plotting of images in QGIS.

The function `interstadial` sets up the `publication` script and uses the filtered and compacted data to export the filtered results of each period.

```
#

publication <- function (path,tbl,inter) {

 temp_data <- tbl

 VegPeriod_complete <- tbl[tbl$Veg.21Day == 1,]

 df_result <- data.frame()
 text = p

 # creation of rows for the filtered csv file
 for (i in seq(from = 47.5, to = 65, 2.5)){ text = paste (c(text,'', i, 'Start
veg. day', 'End veg. day', 'Veg. sum days','Sum of temperature'))}
 df_result <- cbind(text)

 #creation of nodes:
 text = paste (c('','ID','DOY','LAT','LONG','MIN','QGIS_ID'))
 df_node_min <- cbind(text)
 text = paste (c('','ID','DOY','LAT','LONG','MAX','QGIS_ID'))
 df_node_max <- cbind(text)

 # calculation of the min, max, sum and mean of DOY and sum of temperature
 for (long in seq(from = 20, to = 60, 2.5)){ # runs through every longitude

  node_min = ''
  node_max = ''
  stats=''

  for (lat in seq(from = 47.5, to = 65, 2.5)){ # runs through every latitude
   result <- VegPeriod_complete[VegPeriod_complete$Long == long &
VegPeriod_complete$Lat == lat,] # filtering according to coordinates
   stats <- c(stats,'',long,min(result$DOY,
na.rm=T),max(result$DOY,na.rm=T),max(result$DOY,na.rm=T)-min(result$DOY,
na.rm=T),sum(result$Temp_mean,na.rm=T))

   #creating and merging the data with the node files
   node_min = c('',paste(long,'/',lat, sep=''),min(result$DOY,
na.rm=T),lat,long,'Min',QGIS_ID (lat,long))
   node_max = c('',paste(long,'/',lat, sep=''),max(result$DOY,
na.rm=T),lat,long,'Max',QGIS_ID (lat,long))

   df_node_min = cbind(df_node_min, node_min)
   df_node_max = cbind(df_node_max, node_max)
  }

  # preparation of the csv export (publication)
  assign(paste(long), stats)
  name <- paste(long)
```

```
  df_result <- cbind (df_result, stats)

 }

 if (inter == "GI-3"){df_result_result <- data.frame(df_result)} else
{df_result_result <- rbind(df_result_result[,],df_result[,])}


 print(paste(path,inter,".csv",sep=""))
 print(path)

 write.table(df_result, paste(path,inter,".csv",sep=""), sep=',', row.names =
FALSE, col.names = FALSE)
 write.table(df_node_min %>% t(.) %>% data.frame(.) %>% .[,-1] %>% .[.[,2] !=
'Inf',] %>% .[.[,2] != '-Inf',], paste(path,'nodes/nodes_min_',inter,".csv",
sep=""), sep=',', row.names = FALSE, col.names = FALSE)
 write.table(df_node_max %>% t(.) %>% data.frame(.) %>% .[,-1] %>% .[.[,2] !=
'Inf',] %>% .[.[,2] != '-Inf',], paste(path,'nodes/nodes_max_',inter,".csv",
sep=""), sep=',', row.names = FALSE, col.names = FALSE)
}

# set up the publication function and export the filtered results

interstadial <- function(inter,p) {

 # export path
 path <- paste("../Filter_Results/Filter_Results_",p,"/",sep="")

 tbl <-
  list.files(pattern = paste0(inter,".csv",sep="")) %>%
  map_df(~read_csv(.))
 print(paste0(inter,".csv"))

 publication(path,tbl,inter)
}

# separate the LGM file and the NGRIP/BlackSea files and set up the functions
interstadial and publication

VegPer_Results <- function (p,period,df_result_result){

  adress <- paste0("21Day_",p,"_Results",sep="")

  setwd(adress)

  if (p == "LGM"){
    period <- list("21Day_LGM")
  }
  print(p)
  lapply(period,interstadial,p)

  setwd("../")
```

```
}
```

---

additional procedure to create plots using QGIS and Python:

- Open the PaleolithicGreeningQGIS.qgz file with QGIS (3.10)
- Select the Python Console once your QGIS file has opened
- Click on the notepad with pen icon to show the Python editor
- Open the Python scripts nodes.py, links.py and print.py
- Select the nodes layer in the layer browser to the left, then run the nodes.py script
- Repeat with the links.py script (select the layer and run the script)
- The last step in QGIS is to run the print.py script
- The resulting vegetation period pull factor images will be exported to the folder extdata/QGIS

---

nodes.py

> This Python script goes over all start and end dates of each vegetation period of all time periods and parameters (LGM, BlackSea and NGRIP) and proceeds to place defined symbols on each node/coordinate that shows a vegetation period. For the data used in this case, the start of the vegetation period goes from 100 - 210 DOY (Day of the Year), the end of the vegetation period goes from 200 - 320 DOY.

nodes.py code

```
#import geopandas as gpd
from qgis.core import QgsVectorFileWriter
import gdal , ogr
import inspect

def create_point_shape (csvFile, name, min_max):
    #source_ds = ogr.Open(csvFile)
    #layer = source_ds.GetLayer()

    layer = QgsVectorLayer(csvFile, name, "ogr")#, 'memory')

    #prov = layer.dataProvider()


#prov.addAttributes([QgsField("ID",QVariant.Double),QgsField("DOY",QVariant.Double
),QgsField("LAT",QVariant.Double),QgsField("LONG",QVariant.Double),QgsField("MAX",
QVariant.Double),QgsField("QGIS_ID",QVariant.Double)])
    #layer.updateFields()
    #print (layer.fields().names())
    #print(csvFile)
    # add layer to the map
    QgsProject.instance().addMapLayer(layer)
```

```
    '''
    # open the csv-file for reading and skip the header row
    lineStrings = open(csvFile, encoding="utf-8")
    print(q.decode())
    next(lineStrings)

    # start editing
    layer.startEditing()

    # loop over the lines, split them into 4 coordinates, build points from pairs
of
    # them, and connect the pair of points
    feats = []



    for line in lineStrings:
        line = re.sub("['\"]",'',line)
        lineStringAsList = line.split(",")

        from_node =
QgsPoint(float(lineStringAsList[0]),float(lineStringAsList[1]))
        to_node = QgsPoint(float(lineStringAsList[2]),float(lineStringAsList[3]))
        feat = QgsFeature()
        fields = QgsFields()
        feat.setFields(fields)

        feat.setAttributes([float(i) for i in lineStringAsList[4:8]])
        feat.setGeometry(QgsGeometry.fromPolyline([from_node, to_node]))
        #feat['point_id']="Start Point"
        #feat.setAttribute (float(lineStringAsList[4]))
        feats.append(feat)

    # finally add all created features and save edits
    prov.addFeatures(feats)
    layer.updateExtents()
    '''
    styling_nodes (layer, min_max)

    #export_layout()
    layer.commitChanges()

QgsProject.instance().layerTreeRoot().findLayer(layer.id()).setItemVisibilityCheck
ed(False)


def styling_nodes (layer, min_max):
    dir = 'layout_data/'
    #renderer = layer.renderer().QgsGraduatedSymbolRenderer


    if min_max == 'min':
        print(1)
```

```
        layer.loadNamedStyle(dir + 'standardised_style_nodes_min.qml')
        layer.triggerRepaint()

    else:
        print(1)
        layer.loadNamedStyle(dir + 'standardised_style_nodes_max.qml')
        layer.triggerRepaint()

    layer.reload()
    renderer = layer.renderer()

    #renderer.setMode(QgsGraduatedSymbolRendererV2.Quantile)
   #renderer.updateClasses(layer,renderer.Quantile,5)
    #renderer.updateRangeLabels()
    #iface.layerTreeView().refreshLayerSymbology(layer.id()) # Refresh legend on
the interface
    layer.reload()
    #renderer.updateClasses(layer,renderer.Quantile,9)
    iface.layerTreeView().refreshLayerSymbology(layer.id()) # Refresh legend on
the interface

filename = inspect.getframeinfo(inspect.currentframe()).filename
rel_path = os.path.dirname(os.path.dirname(os.path.abspath(filename)))


for i in ['BlackSea','NGRIP','LGM']:
    for j in ['min','max']:
        if i == 'LGM':
            k = '21Day_LGM'



            path = "file:///" + rel_path +
"/VegPer_Results/Filter_Results/Filter_Results_" + i + "/nodes/nodes_" + j + "_" +
k + ".csv"
            uri = path + "?
encoding=%s&delimiter=%s&xField=%s&yField=%s&crs=%s&useHeader=%s" % ("UTF-8",",",
"LONG", "LAT","epsg:4326","yes")

            name = "nodes_" + i + "_" + k + '_' + j
            vlayer = QgsVectorLayer(uri, name, 'delimitedtext')
            #print(vlayer.isValid())
            print (name)
            #iface.addVectorLayer(uri, name,'delimitedtext')
            path = "../VegPer_Results/Filter_Results/Filter_Results_" + i +
"/nodes/nodes_" + j + "_" + k + ".shp"
            QgsVectorFileWriter.writeAsVectorFormat(vlayer, path, "UTF-8",
vlayer.crs(), "ESRI Shapefile")

            create_point_shape (path, name, j)

        else:
            for k in ["GS-3","GI-3", "GS-4", "GI-4", "GS-5.1", "GI-5.1", "GS-5.2",
```

```
"GI-5.2",
                "GS-6", "GI-6", "GS-7", "GI-7", "GS-8", "GI-8", "GS-9", "GI-9"]:

                path = "file:///" + rel_path +
"/VegPer_Results/Filter_Results/Filter_Results_" + i + "/nodes/nodes_" + j + "_" +
k + ".csv"
                uri = path + "?
encoding=%s&delimiter=%s&xField=%s&yField=%s&crs=%s&useHeader=%s" % ("UTF-8",",",
"LONG", "LAT","epsg:4326","yes")

                name = "nodes_" + i + "_" + k + '_' + j
                vlayer = QgsVectorLayer(uri, name, 'delimitedtext')
                #print(vlayer.isValid())

                #iface.addVectorLayer(uri, name,'delimitedtext')
                path = "../VegPer_Results/Filter_Results/Filter_Results_" + i +
"/nodes/nodes_" + j + "_" + k + ".shp"
                QgsVectorFileWriter.writeAsVectorFormat(vlayer, path, "UTF-8",
vlayer.crs(), "ESRI Shapefile")

                create_point_shape (path, name, j)
```

---

links.py

> The links.py script calculates and shows the strength of the green wave and the pull factors for each period and parameter (LGM, BlackSea and NGRIP). The layout and symbology are defined to show the strength of the green wave by displaying different shades of green to show the difference between the starting or endpoints of the vegetation period between each node. The difference in days bewteen the vegetation periods from the nearest coordinates are in this case from 2 days to 50 days.

links.py script

```
import re

import gdal , ogr
import pandas as pd
import numpy as np
import os, sys
import qgis.utils

#from Froot_Loops_QGIS_print import *

def create_line_model (csvFile,name,export_path):

    # create an empty memory layer for polylines
    layer = QgsVectorLayer('LineString?crs=EPSG:4326', name, 'memory')
```

```
    #print(vlayer.isValid())

    prov = layer.dataProvider()

prov.addAttributes([QgsField("total_number_days",QVariant.Double),QgsField("limit_
of_days",QVariant.Double),QgsField("ID_source",QVariant.Double),QgsField("ID_targe
t",QVariant.Double)])
    layer.updateFields()
    print (layer.fields().names())
    # add layer to the map
    QgsProject.instance().addMapLayer(layer)

    # open the csv-file for reading and skip the header row
    lineStrings = open(csvFile, "rU")

    next(lineStrings)

    # start editing
    layer.startEditing()

    # loop over the lines, split them into 4 coordinates, build points from pairs
of
    # them, and connect the pair of points
    feats = []



    for line in lineStrings:
        line = re.sub("['\"]",'',line)
        lineStringAsList = line.split(",")

        from_node =
QgsPoint(float(lineStringAsList[0]),float(lineStringAsList[1]))
        to_node = QgsPoint(float(lineStringAsList[2]),float(lineStringAsList[3]))
        feat = QgsFeature()
        fields = QgsFields()
        feat.setFields(fields)

        feat.setAttributes([float(i) for i in lineStringAsList[4:8]])
        feat.setGeometry(QgsGeometry.fromPolyline([from_node, to_node]))
        #feat['point_id']="Start Point"
        #feat.setAttribute (float(lineStringAsList[4]))
        feats.append(feat)

    # finally add all created features and save edits
    prov.addFeatures(feats)
    layer.updateExtents()

    styling (layer)
    #export_layout()
    layer.commitChanges()

QgsProject.instance().layerTreeRoot().findLayer(layer.id()).setItemVisibilityCheck
```

```
ed(False)

    path = "C:/Users/SFB
806/Documents/GitHub/Vegetation_Period_Data/VegPer_QGIS/links/" + export_path +
name + ".shp"
    QgsVectorFileWriter.writeAsVectorFormat(layer, path, "UTF-8", layer.crs(),
"ESRI Shapefile")


    #QgsProject.instance().removeMapLayers( [vl.id()] )

def styling (layer):
    #layer = iface.activeLayer()
    #layer.geometryType() == QGis.Point:
    dir = 'layout_data/'
    #renderer = layer.renderer().QgsGraduatedSymbolRenderer

    layer.loadNamedStyle(dir + 'standardised_style_4.qml')
    layer.triggerRepaint()
    #renderer = layer.renderer()
    #print("Type:", renderer.type())


    #renderer.updateClasses(layer,renderer.Quantile,9)
    #renderer.updateRangeLabels()
    iface.layerTreeView().refreshLayerSymbology(layer.id()) # Refresh legend on
the interface
    layer.reload()


def plotting_export_png ():
    export_layout()


def reduce_resolution (name, x):
    #try:
    os.chdir('rasterfiles/')
    print(name)
    in_ds = gdal.Open(name)
    out_ds = 'reduced/' + x + name

    in_band = in_ds.GetRasterBand(1)
    myarray = np.array(in_band.ReadAsArray())

    limit = 8

    x = in_band.XSize - (in_band.XSize % limit)
    y = in_band.YSize - (in_band.YSize % limit)

    '''if x < 100 or y < 100:
        resolution = x,y
    else:'''
    if in_band.XSize % limit > 0 and in_band.YSize % limit > 0:
```

```
            resolution = (x /limit +1, y/limit+1)
    elif in_band.YSize % limit > 0:
        resolution =  in_band.XSize/limit, y/limit +1
    elif in_band.XSize % limit > 0:
        resolution = x/limit +1, in_band.YSize / limit
    else:
        resolution = (in_band.XSize / limit), (in_band.YSize / limit)


    gtiff_driver = gdal.GetDriverByName('GTIFF')
    dst_ds = gtiff_driver.Create(out_ds,
                           int(resolution[0]),
                           int(resolution[1]),
                           1,
                           gdal.GDT_Float64)


    dst_ds.GetRasterBand(1).SetNoDataValue(0)

    dst_ds.SetProjection(in_ds.GetProjection())
    geotransform = list(in_ds.GetGeoTransform())
    geotransform [1] *= limit
    geotransform [5] *= limit
    dst_ds.SetGeoTransform(geotransform)
    myarray = in_band.ReadAsArray(buf_xsize=int(resolution[0]), buf_ysize=
int(resolution[1]))

    myarray [myarray < 0] = 0
    out_band = dst_ds.GetRasterBand(1)
    out_band.WriteArray(myarray)

    #except:
    print('ohoh')
    in_ds = None
    dst_ds = None



for i in ['LGM','BlackSea','NGRIP']:#,
    for j in ['min','max']:
        if i == 'LGM':
            k = '21Day_LGM'
            name = i + '_' + k + '_' + j #"GS-3_min_QGIS_second"
            csvFile = "../VegPer_Results/Calc_Step_Results/Calc_Step_" + i + "/" +
name + ".csv"
            export_path = i + "/" + name
            create_line_model (csvFile,name,export_path)
        else:
            for k in ["GS-3","GI-3", "GS-4", "GI-4", "GS-5.1", "GI-5.1", "GS-5.2",
"GI-5.2",
                "GS-6", "GI-6", "GS-7", "GI-7", "GS-8", "GI-8", "GS-9", "GI-9"]:
```

```
                # specify your csv-file
                name = i + '_' + k + '_' + j #"GS-3_min_QGIS_second"
                csvFile = "../VegPer_Results/Calc_Step_Results/Calc_Step_" + i +
"/" + name + ".csv"
                export_path = i + "/" + name
                create_line_model (csvFile,name,export_path)

#printpdfmulti("Froot_Loops")

'''
import os

directory_in_str = 'F:/Aster' # + ASTGTMV003_N58E018_dem

directory = os.fsencode(directory_in_str)

for file in os.listdir(directory):
     filename = os.fsdecode(file)
     print(filename)

     if filename.endswith(".tif"):
         # print(os.path.join(directory, filename))
         reduce_resolution(filename, 'reduced_')
         #break
         continue
     else:
         continue

'''
```

---

print.py

> Running the print.py script, allows QGIS to combine the layers, that have been collected through the nodes.py script and the links.py script, and give these images a legend and a title. Through the print.py script the images now show the pull factors and the strength of the green wave for each period and parameter (LGM, BlackSea and NGRIP). The images are exported and can be found in the folder VegPer_QGIS/extdata/QGIS.

print.py script

```
#!/usr/bin/env python3
import os
from qgis.core import (QgsProject, QgsLayoutExporter, QgsApplication)
import qgis.utils

def export_layout():
```

```
    QgsApplication.setPrefixPath("extdata/QGIS", True)

    gui_flag = False
    app = QgsApplication([], gui_flag)

    app.initQgis()

    project_path = os.getcwd() + '/Froot_Loops_QGIS_3.qgz'

    project_instance = QgsProject.instance()
    project_instance.setFileName(project_path)
    project_instance.read()

    manager = QgsProject.instance().layoutManager()
    layout = manager.layoutByName("Froot_Loops") # name of the layout
    # or layout = manager.layouts()[0] # first layout

    exporter = QgsLayoutExporter(layout)
    exporter.exportToPdf(project_instance.absolutePath() +
"/extdata/QGIS/layout.png",
                         QgsLayoutExporter.PdfExportSettings())

    app.exitQgis()




def printpdfmulti(layoutname,link,node):
    for i in range(1):
        #selected_layers = qgis.utils.iface.layerTreeView().selectedLayers()
        projectInstance = QgsProject.instance()
        #projectInstance_2 = QgsProject.instance()
        layoutmanager = projectInstance.layoutManager()
        layout = layoutmanager.layoutByName(layoutname)
         #Layout nameprojectInstance = QgsProject.instance()



#QgsProject.instance().layerTreeRoot().findLayer(layer.name('cut_n30e030_reduced')
).setItemVisibilityChecked(True)

#QgsProject.instance().layerTreeRoot().findLayer(layer.name('cut_n30e060_reduced')
).setItemVisibilityChecked(True)

#QgsProject.instance().layerTreeRoot().findLayer(layer.name('cut_n30e000_reduced')
).setItemVisibilityChecked(True)

#QgsProject.instance().layerTreeRoot().findLayer(layer.name('aster_reduced')).setI
temVisibilityChecked(True)

        link_node (layout,link,node,projectInstance)
```

```
        print(1)
        del projectInstance, layout

def link_node (layout,link,node,projectInstance):
    print (link)
    print (node)
    node_layer = projectInstance.mapLayersByName(str(node))[0]
    link_layer = projectInstance.mapLayersByName(str(link))[0]


QgsProject.instance().layerTreeRoot().findLayer(node_layer.id()).setItemVisibility
Checked(True)

QgsProject.instance().layerTreeRoot().findLayer(link_layer.id()).setItemVisibility
Checked(True)
    #iface.layerTreeView().refreshLayerSymbology(node_layer.id())
    #iface.layerTreeView().refreshLayerSymbology(link_layer.id())

#iface.layerTreeView().layerTreeRoot().findLayer(link_layer.id()).refreshItems()
    #iface.layerTreeView().layerTreeModel().refreshLayerLegend(link_layer.id())
    '''
    layout_temp = layout
    #legend = layout.selectedLayoutItems()
    legend = QgsLayoutItemLegend(layout_temp)
    layerTree = QgsLayerTree()
    layerTree.addLayer(node_layer)
    layerTree.addLayer(link_layer)
    legend.model().setRootGroup(layerTree)
    legend.refresh()
    layout.addLayoutItem(legend)
    layout.refresh()

    newFont = QFont("MS Shell Dlg 2", 8)
    legend.setStyleFont(QgsLegendStyle.SymbolLabel, newFont)
    '''


    exporter = QgsLayoutExporter(layout)
    exporter.exportToImage(path + '/' + link + ".png",
QgsLayoutExporter.ImageExportSettings() )

QgsProject.instance().layerTreeRoot().findLayer(link_layer.id()).setItemVisibility
Checked(False)

QgsProject.instance().layerTreeRoot().findLayer(node_layer.id()).setItemVisibility
Checked(False)




#for i in
list(['cut_n30e030_reduced','cut_n30e060_reduced','cut_n30e000_reduced','aster_red
uced','ice_sheet_interstadial','ice_sheet_LGM','world_rivers_dSe']):
```

```
for i in list(['Elevation (m)','Ice sheet (interstadial)','Ice sheet
(LGM)','Rivers']):
        projectInstance = QgsProject.instance()
        layer = projectInstance.mapLayersByName(i)[0]

QgsProject.instance().layerTreeRoot().findLayer(layer.id()).setItemVisibilityCheck
ed(True)

path = "extdata/QGIS"



for i in ['LGM','BlackSea','NGRIP']:#,:
    for j in ['min','max']:
        if j == 'min':
            layoutname = 'Froot_Loops min'
        elif j == 'max':
            layoutname = 'Froot_Loops max'

        if i == 'LGM':
            k = '21Day_LGM'

            node = "nodes_" + i + "_" + k + '_' + j
            link =  i + "_" + k + '_' + j
            printpdfmulti(layoutname,link,node)
        else:
            #for k in ["GS-4"]:#
            for k in ["GS-3","GI-3", "GS-4", "GI-4", "GS-5.1", "GI-5.1", "GS-5.2",
"GI-5.2",
                "GS-6", "GI-6", "GS-7", "GI-7", "GS-8", "GI-8", "GS-9", "GI-9"]:
            #for k in ["GI-3", "GS-4"]:
                node = "nodes_" + i + "_" + k + '_' + j
                link =  i + "_" + k + '_' + j
                printpdfmulti(layoutname,link,node)

'''
for i in list(['Elevation (m)','Ice sheet (interstadial)','Ice sheet
(LGM)','Rivers']):
    projectInstance = QgsProject.instance()
    layer = projectInstance.mapLayersByName(i)[0]

QgsProject.instance().layerTreeRoot().findLayer(layer.id()).setItemVisibilityCheck
ed(False)
    '''
```

Additional script

---

plot_2x2.R

> The resulting calculations of the vegetation period in the different time periods (GS-3 to GI-9), are being made visible using QGIS and are now compacted in R into a better overview with 2x2 images shown next to each other after export.

Variables:

period_concat : list of climatic periods of interest

plot_2x2 Functions

`print2x2`

`plot2x2`

Details plot_2x2 Functions

The function `plot2x2` takes the parameters *period* and *season* and splits to either use the base data (LGM) and then use the results and then plot the images in the function `print2x2` or use the data from Black Sea / NGRIP and then use `print2x2` to build and export the images.

`print2x2` image aspects are defined and images are placed in a two by two grid to give a better overview and easier comparison. The images are then exported with the correct names being linked from the base data.

```
# libraries for plot2x2.R

library(tidyverse)
library(dplyr)
library(purrr)
library(png)
library(gridExtra)

library(RStoolbox)

library(raster)
library(rgdal)

# plotting the spatial data including the calculated arrows to show the green wave

print2x2 <- function (period,parameter) {
 print(paste(parameter,'_',period[[1]][1],'_min.png',sep=''))
 print(paste(parameter,'_',period[[2]][1],'_min.png',sep=''))

 im1 <- stack(paste('QGIS/',parameter,'_',period[[1]][1],'_min.png',sep=''))

 im2 <- stack(paste('QGIS/',parameter,'_',period[[1]][1],'_max.png',sep=''))

 im3 <- stack(paste('QGIS/',parameter,'_',period[[2]][1],'_min.png',sep=''))

 im4 <- stack(paste('QGIS/',parameter,'_',period[[2]][1],'_max.png',sep=''))
```

```
 plot_output_png <- grid.arrange(
  ggRGB(img = im1,r=1,g=2,b=3,maxpixels = 2e+06) + ggtitle(paste(parameter,'
',period[[1]][1],' (spring)',sep='')) + xlab ("") + ylab ("") + guides(fill=FALSE)
+
   theme(axis.title=element_blank(),
      axis.text=element_blank(),
      axis.ticks=element_blank(),
      panel.grid.major = element_blank(),
      panel.grid.minor = element_blank(),
      panel.background = element_blank()),
  ggRGB(img = im2,r=1,g=2,b=3,maxpixels = 2e+06) + ggtitle(paste(parameter,'
',period[[1]][1],' (autumn)',sep='')) + xlab ("") + ylab ("") + guides(fill=FALSE)
+
   theme(axis.title=element_blank(),
      axis.text=element_blank(),
      axis.ticks=element_blank(),
      panel.grid.major = element_blank(),
      panel.grid.minor = element_blank(),
      panel.background = element_blank()),
  ggRGB(img = im3,r=1,g=2,b=3,maxpixels = 2e+06) + ggtitle(paste(parameter,'
',period[[2]][1],' (spring)',sep='')) + xlab ("") + ylab ("") + guides(fill=FALSE)
+
   theme(axis.title=element_blank(),
      axis.text=element_blank(),
      axis.ticks=element_blank(),
      panel.grid.major = element_blank(),
      panel.grid.minor = element_blank(),
      panel.background = element_blank()),
  ggRGB(img = im4,r=1,g=2,b=3,maxpixels = 2e+06) + ggtitle(paste(parameter,'
',period[[2]][1],' (autumn)',sep='')) + xlab ("") + ylab ("") + guides(fill=FALSE)
+
   theme(axis.title=element_blank(),
      axis.text=element_blank(),
      axis.ticks=element_blank(),
      panel.grid.major = element_blank(),
      panel.grid.minor = element_blank(),
      panel.background = element_blank()),
  ncol=2)

 # output as png

 plot_output <- paste('2x2/',parameter,'/',period[[1]][1],'_',period[[2]]
[1],'.png', sep='')
 ggsave(plot_output, plot_output_png, device = png(), units = c("cm"), dpi = 1200,
width=25, height=16)
 while (!is.null(dev.list()))  dev.off()
}

# start the print2x2 function to plot the QGIS images

plot_2x2 <- function(parameter,period){

 if (parameter == "LGM"){
```

```
  print2x2 (list('21Day','21Day'),parameter)

 }else{
  lapply(period,parameter,print2x2)

 }
}
```