BON-in-a Box 2.0 - Block 2 eBird data workflow

Instituto de Investigación de Recursos Biológicos Alexander von Humboldt

2023-01

Organize workflow environment

1. Load packages to use in the script.

```
# name libraries
packagesList<-list("rstudioapi", "magrittr", "dplyr", "plyr", "purrr", "raster", "terra", "auk", "sf","
# load libraries
lapply(packagesList, library, character.only = TRUE)
```

2. Set working directory.

```
# name working directory
dirfolder<- "~/Bloque2/draft_ebird_bloque2_06_12_22"

# set working directory
setwd(dirfolder)
```

Specify the basic parameters for eBird data cleaning

1. Create a reference to an eBird Basic Dataset.

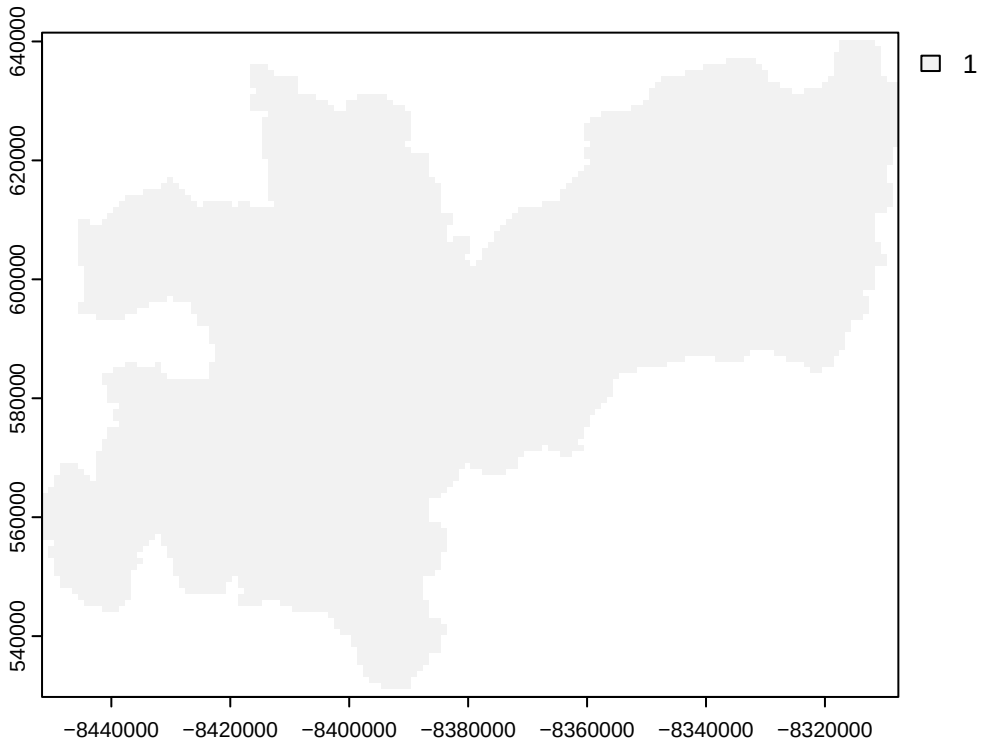
```
# Name ebird records database file. This file is available for download as relist at https://ebird.org/
file_data<- "ebd_CO_relFeb-2022.txt"

# Name ebird sampling lists database file. This file is available for download as sampling relist at ht
file_lists<- "ebd_sampling_relFeb-2022.txt"

# Create a reference file for the ebird database
setwd(dirfolder)
ebd <- auk_ebd(file = file_data, file_sampling = file_lists)
```

2. Filter the ebird database

```
#### Define filters

## Define filter by Date
date_filter<- c("2010-01-01", "2020-12-31")
```

```r
##  Define filter by duration - sampling time
duration_filter<- c(0, 300)

## Define filter by protocol sampling
protocol_filter<- c("Stationary", "Traveling")

## Define filter by sampling effort - distance in km
distance_filter<- c(0,10)

##  Define filter by study area

# set spatial parameters
dir_basemap = "~/Bloque2/draft_ebird_bloque2_06_12_22/Caldas.shp" # set the directory path of shape fil
resolution_basemap<- 1000 # set the resolution grid in meters
crs_basemap<- CRS("+init=epsg:3395") # set the projection system. We use EPSG 3395 as global coordinate

setwd(dirfolder)
info_layer<- st_read(dir_basemap)
extentBase<- info_layer%>% st_bbox() %>% st_as_sfc() %>% st_transform(crs = crs_basemap) %>% st_bbox() %
rasterbase<- raster(extentBase,crs = crs_basemap, res= resolution_basemap )


# create base grid
tname2 = "base_grid.tif"; t_file = writeStart(rasterbase, filename = tname2,  overwrite = T); writeStop
gdalUtilities::gdal_rasterize(dir_basemap, tname2, burn =1, at=T)
raster_area = rast(t_file) %>% {terra::mask(setValues(., seq(ncell(.))), .)}

# create copy extent in WGS84 EPSG 4326 coordinate system This is necessary because the ebrid data is i
area_4326<-  raster(raster_area) %>% projectRaster(crs = st_crs(4326)$proj4string, method = "ngb")
ebd_bbox<- st_bbox(area_4326)


#### Specify the function with filters
filter_bbox<- ebd %>% auk_bbox(ebd_bbox) %>% auk_protocol(protocol = c("Stationary", "Traveling")) %>%
  auk_distance(distance = c(0,10), distance_units= "km") %>% auk::auk_duration(duration = c(0, 300)) %>%
```

```r
# plot covars_raster
plot(raster_area)
```



Load and organize eBird data file

1. Set output folder and name files for save results.

```
## Set folder to save the results
# Define name of output_auk_ebird folder
outputs_auk_ebird<- "outputs_auk_ebird"

# Check and create output_auk_ebird folder
setwd(dirfolder)
if(!dir.exists(outputs_auk_ebird)){dir.create(outputs_auk_ebird)}

# Set output_auk_ebird directory
setwd(dirfolder); setwd(outputs_auk_ebird)

## Set specific subfolder to save results
# Define name of output folder
output_folder<- basename(file_path_sans_ext(dir_basemap)) # In this case we set the name according the

# set output folder
setwd(dirfolder); setwd(outputs_auk_ebird)
if(!dir.exists(output_folder)){dir.create(output_folder)}; setwd(output_folder)

# Define name to save filter ebird records database file
output_studyarea_ebd<- paste(output_folder, file_data, sep= "_")
```

```
# Define name to save filter ebird sampling lists database file
output_studyarea_lists<- paste(output_folder, file_lists, sep= "_")
```

2. Load filtered ebird data files according the parameter filters.

```
## Load filtered ebird data files
auk_filter(filter_bbox, file = output_studyarea_ebd, file_sampling = output_studyarea_lists, overwrite =
```

Loading study area spatial covariates

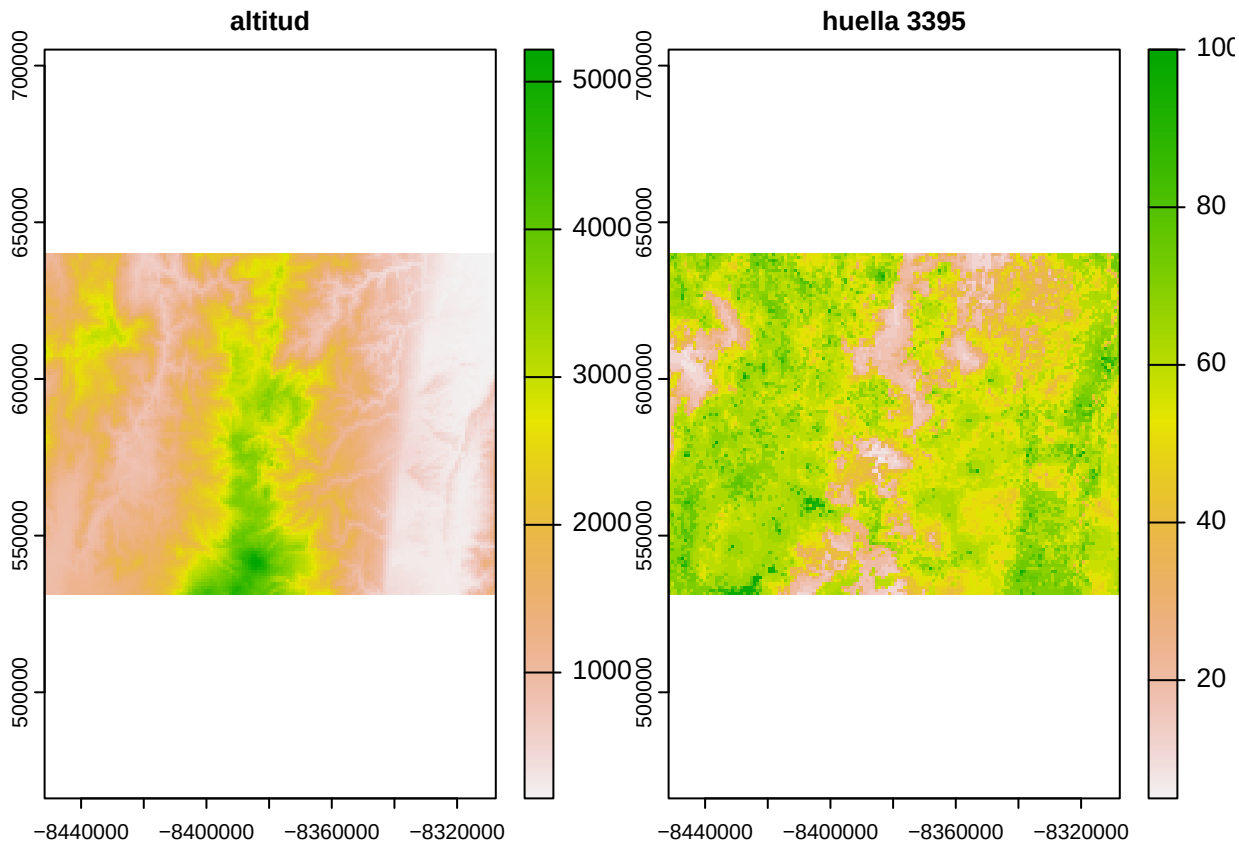
1. Define folder in which the spatial files of the covariates of interest are saved. The script supports files in format "tif" or "shp".

```
# Specify de covariates folder
setwd(dirfolder)
folder_covars<- "defaultcovars"

# Get files names of folder covars
dir_covars<- list.files(folder_covars, pattern =  c("\\.tif$", "\\.shp$"))
```

2. Load and adjust covariates according study area.

```
covars_raster<- lapply(dir_covars, function(x) {setwd(dirfolder); setwd(folder_covars); print(x)
  tname2<- tempfile(fileext = '.tif')
  t_file<- writeStart(rasterbase, filename = tname2,  overwrite=T); writeStop(t_file);

  gdalUtilities::gdalwarp(srcfile = x, dstfile= tname2,
                          tr= res(rasterbase), te = st_bbox(extent(rasterbase)),
                          overwrite=TRUE, r= "near")
  rast(tname2) %>% setNames(gsub(".tif", "",x))
}) %>% setNames(sapply(., function(x) gsub(" ", "_", names(x)) ))
```

```
# plot covars_raster
plot(rast(covars_raster))
```



3. Organize covariates table.

```
# Create table with cells of study area
covars_data<-  data.frame(Pixel= cells(raster_area))

# Add covariates values to table
for(i in names(covars_raster)){
  covars_data[,i]<-covars_raster[[i]][covars_data$Pixel]
}
```

Estimation of species occupancy models

1. Specify the interest species to estimate the model

```
sp<- "Zonotrichia capensis"
```

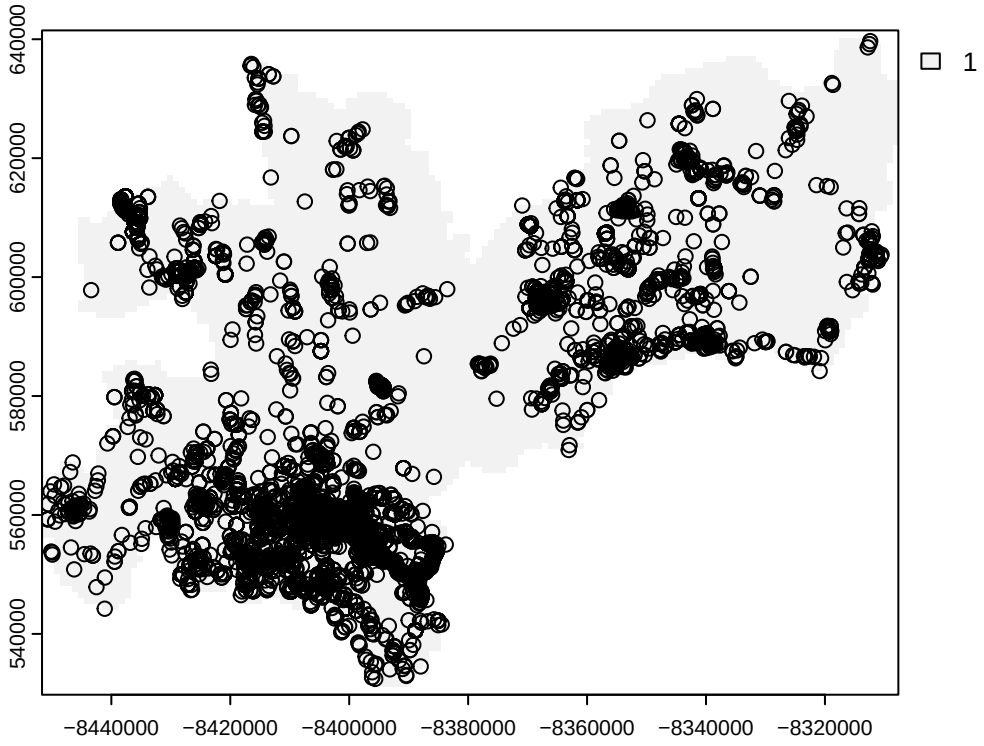
2. Filter species database by interest species

```
ebird_data <- auk_zerofill(output_studyarea_ebd, output_studyarea_lists, species = sp , collapse = T, c
  mutate(observation_count= ifelse(observation_count == "X", 1, observation_count)) %>%
  mutate(observation_count= as.numeric(observation_count))
```

3. Filter species database by study area

```r
# Spatialize the data
ebird_spatial<- ebird_data %>%  st_as_sf(coords = c("longitude", "latitude"), crs = 4326)

# Filter by mask of study area
ebird_spatial_mask<- as.data.frame(ebird_spatial) %>% mutate(Pixel= raster::extract(area_4326, ebird_spa
  dplyr::filter(!is.na(Pixel))  %>%  st_as_sf() %>% st_transform(crs_basemap)

ebird_data_mask<- st_drop_geometry(ebird_spatial_mask)
```

```r
plot(raster_area)
plot(ebird_spatial_mask[, "geometry"], add= T)
```



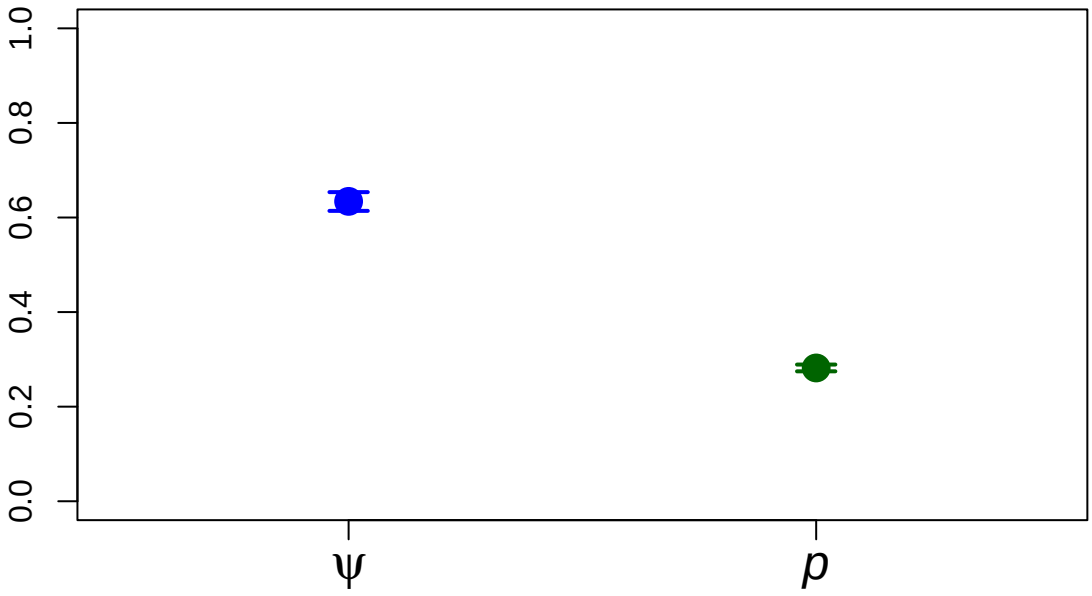
4. Organize matrix by pixels and covariates

```r
# Join species database with covariates data
summ_sp<- ebird_data_mask %>% mutate(month= format(observation_date,'%Y') ) %>%
  group_by(Pixel, month) %>% dplyr::summarise(occurrence= ifelse(sum(observation_count)>0,1,0)  ) %>%
  data.frame(xyFromCell(raster_area, .$Pixel)) %>% list(covars_data) %>% join_all()

# Organize data as matrix as pixels as rows and covariates as columns
matriz_input<- reshape2::acast(summ_sp, Pixel~month, value.var = "occurrence", fill=0)

matriz_input_covars<- list(rownames_to_column(as.data.frame(matriz_input), "Pixel"), dplyr::select(summ
  join_all() %>% dplyr::select(-"Pixel")
```

5. Explore detection probabilities

```
# Organize data for the single season occupancy models
Data.1 <- unmarkedFrameOccu(y = dplyr::select(matriz_input_covars, -names(covars_raster)), siteCovs = dp

DataMod <- occu(~1 ~ 1, Data.1)

# Original scale of data
ests <- plogis(coef(DataMod))

# Get results at normal scale with standard error
psiSE <- backTransform(DataMod, type="state")
pSE <- backTransform(DataMod, type="det")

# Get confidence intervals
ciPsi <- confint(psiSE)
ciP <- confint(pSE)

# Organize results
resultsTable <- rbind(psi = c(ests[1], ciPsi), p = c(ests[2], ciP))
colnames(resultsTable) <- c("Estimate", "lowerCI", "upperCI")
```

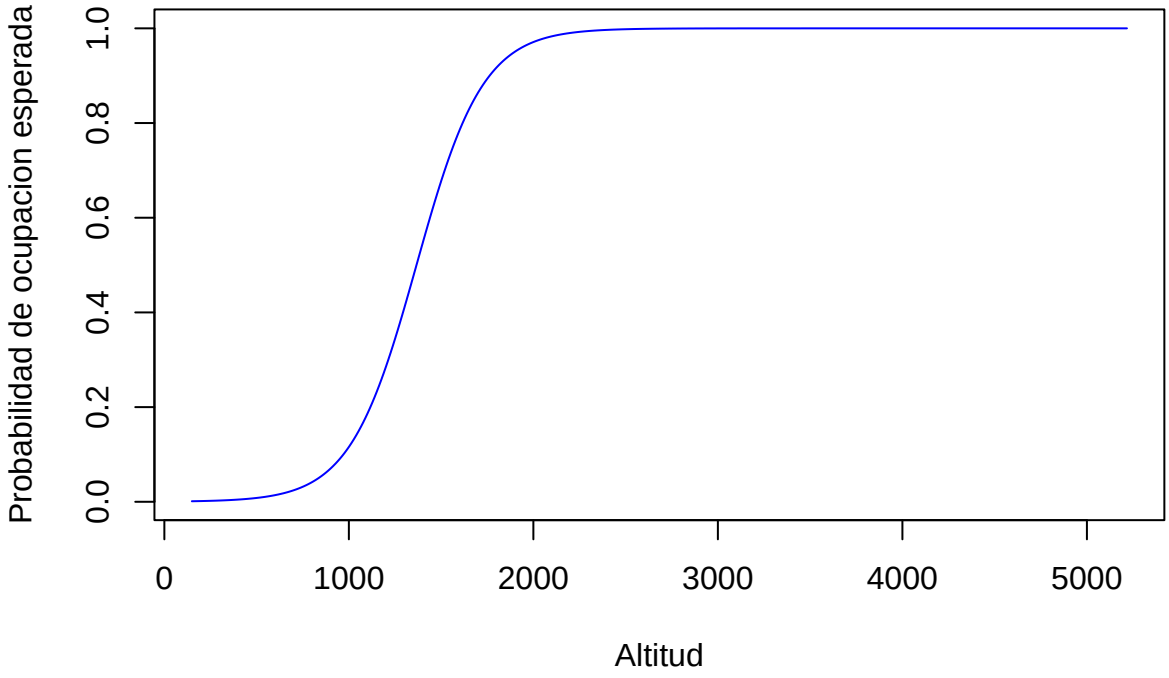
```
# plot resutls
plot(1:2, resultsTable[,"Estimate"], xlim=c(0.5, 2.5), ylim=0:1,
     col=c("blue", "darkgreen"), pch=16, cex=2, cex.lab=1.5,
     xaxt="n", ann=F)
axis(1, 1:2, labels=c(expression(psi), expression(italic(p))), cex.axis=1.5)
arrows(1:2, resultsTable[,"lowerCI"], 1:2, resultsTable[,"upperCI"],
       angle=90, length=0.1, code=3, col=c("blue", "darkgreen"), lwd=2)
```



6. Estimate occupancy model according (Mackenzy et al 2002)

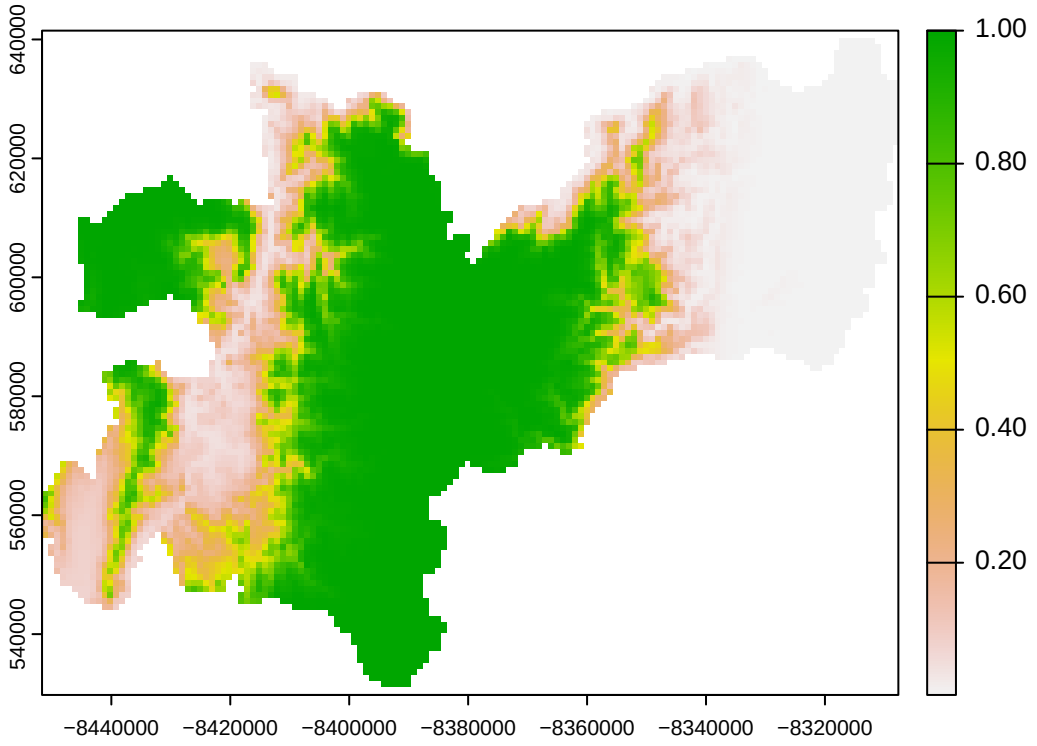
```
# Estimate occupancy
m1<-occu(~1 ~altitud,Data.1)

# create table of variables attribute to test
newData2 <- data.frame(altitud=  sort(unique(covars_data$altitud)))
E.psi <- predict(m1, type="state", newdata=newData2, appendData=TRUE)
```

```
# plot resutls
plot(Predicted ~ altitud, E.psi, type="l", col="blue",
     xlab="Altitud",
     ylab="Probabilidad de ocupacion esperada")
```




```
# create data for map predict
match_pixels<- list(covars_data, E.psi) %>% join_all()
occupancy_model_raster<- raster_area
occupancy_model_raster[match_pixels$Pixel]<- match_pixels$Predicted
```

```
# plot resutls
plot(occupancy_model_raster)
```


640000
620000
600000
580000
560000
540000
−8440000
−8420000
−8400000
−8380000
−8360000
−8340000
−8320000
1.00
0.80
0.60
0.40
0.20