Suppoting Information 1

Nathan Fox

25 March 2020

Installation

photosearcher is available as part of the R OpenSci package.

```
# pacman package allows for better loading of uninstalled packages
if(!"pacman" %in% installed.packages()) install.packages("pacman")
library(pacman)

# load required libraries and install if needed
p_load_gh("ropensci/photosearcher")
p_load(ggplot2,
       ggthemes,
       dplyr,
       ggmap,
       tmaptools,
       USAboundaries)
```

Getting data for cultural ecosystem service studies

To speed up the running of the code, all examples presented here are a smaller subset of the ones presented in the main articles (i.e US hiking for 6 months instead of 5 years)

The following lines of code are reproducible and were used to carry out obtaining hiking dataset such as presented in the article.

```
#Get photograph metadata for images of hiking in the USA

#Load shapefile
contiguous_us <- USAboundaries::us_states()
contiguous_us <- contiguous_us[!contiguous_us$name == "Alaska", ]
contiguous_us <- contiguous_us[!contiguous_us$name == "Hawaii", ]
contiguous_us <- contiguous_us[!contiguous_us$name == "Puerto Rico", ]

#add col for mapping points by state
contiguous_us$mapid <- 1:nrow(contiguous_us)

#search flickr for photographs
```

```
USA_hiking <- photosearcher::photo_search(mindate_taken = "2018-06-01",
                                          maxdate_taken = "2019-01-01",
                                          maxdate_uploaded = "2020-01-01",
                                          text = "hiking",
                                          sf_layer = contiguous_us)



)

#add state name to the the hiking photographs to plot as colours
USA_hiking <- USA_hiking %>%
  rename(mapid = within)

USA_hiking <- merge(USA_hiking, contiguous_us, by = "mapid")

#plot map
ggplot(data = USA_hiking,
       aes(x = longitude, y = latitude, colour = state_abbr)) +
  borders("state", colour = "white", fill = "gray87") +
  geom_point(size = 0.5) +
  scale_x_continuous(name = "Longitude") +
  scale_y_continuous(name = "Latitude") +
  scale_colour_viridis_d(option = "B") +
  coord_fixed() +
  theme_bw(base_size = 9) +
  theme(strip.background = element_blank(),
        plot.margin=grid::unit(c(0.25,0.25,0.25,0.25), "mm"),
        legend.position = "none")

#save plot
ggsave(filename = "USA_hike.png",
       height = 7.365,
       width = 14.287,
       units = "cm")
```

As with searches where `has_geo = TRUE` searches with a shapefile will only return photographs with associated latitude and longitude data. The location of these points can be plotted. To reproduce this with any other shapefile, users can use `sf::st_read()` to read in their chosen shapefile.

The following lines of code are reproducible and were used to carry out obtaining the city dataset such as presented in the article.

```
#extract user ids
user_ids <- data.frame(USA_hiking$owner)

#extract unique users
user_ids <- distinct(user_ids)
```

```r
#search for their information
user_info <- photosearcher::user_info(user_id = user_ids$USA_hiking.owner)

#only get users that have a city listed
user_city <- subset(user_info, city > 0)

#add country to end to increase geocode accuracy
user_city$addr <- paste(user_city$city, user_city$country, sep = " ")

#correct coding for geocoding
Encoding(user_city$addr) <- "UTF-8"
user_city$addr <- iconv(user_city$addr, "UTF-8", "UTF-8",sub='')
user_city$addr <- iconv(user_city$addr, 'utf-8', 'ascii', sub='')

#get geocoded location sample first 100 to speed up example
geo_city <- tmaptools::geocode_OSM(user_city$addr[1:100])

#make a spatial layer
geo_city <- sf::st_as_sf(geo_city,
                         coords = c("lon", "lat"),
                         remove = FALSE,
                         crs = 4326)

geo_city$Location <- as.character(sf::st_intersects(geo_city, contiguous_us))
geo_city$Location[geo_city$Location != "integer(0)"] <- "USA"
geo_city$Location[geo_city$Location != "USA"] <- "World"

#facet wrap has issues adding boarder to differnt scales - this fixes issue
mapdata <- map_data("world")

US_city <- subset(geo_city, Location == "USA")
world_city <- subset(geo_city, Location != "USA")

mapdata$Location <- ifelse(
  findInterval(mapdata$lon, range(US_city$lon)) == 1 &
    findInterval(mapdata$lat, range(US_city$lat)) == 1,
  "USA",
  ifelse(
    findInterval(mapdata$lon, range(world_city$lon)) == 1 &
      findInterval(mapdata$lat, range(world_city$lat)) == 1,
    "World",
    NA)
)

#re-add the us cities to the world map
US_city$Location <- "World"
geo_city <- rbind(geo_city, US_city)

US_map <- subset(mapdata, Location == "USA")
```

```r
mapdata$Location <- "World"
mapdata <- rbind(US_map, mapdata)
mapdata <- subset(mapdata, Location > 0)

#map geotagged cities
ggplot(geo_city, aes(x = lon, y = lat, colour = Location)) +
  geom_polygon(data = mapdata, aes(x=long, y=lat, group=group),
               colour = "gray87", fill = "gray87") +
  geom_point(size = 0.5) +
  scale_x_continuous(name = "Longitude") +
  scale_y_continuous(name = "Latitude") +
  facet_wrap(~ Location, nrow = 2, scales = "free") +
  theme_bw(base_size = 9) +
  theme(strip.background = element_blank(),
        plot.margin=grid::unit(c(0.25,0.25,0.25,0.25), "mm"),
        legend.position = "none")

#save plot
ggsave(filename = "city_map.png",
       height = 9.525,
       width = 9.525,
       units = "cm")
```

Species data

The following lines of code are reproducible and were used to carry out obtaining the barn owl and brown bear datasets presented in the article. To search for other species users should enter the species name in question into the `text =` argument.

Mapping spatial distributions

As `geo = TRUE` images returned will have associated longitude and latitude, which can be used to plot species distributions.

```r
#Search for images
#common name
barn_owl <- photosearcher::photo_search(mindate_taken = "2000-01-01",
                                        maxdate_taken = "2020-01-01",
                                        maxdate_uploaded = "2020-01-01",
                                        text = "barn_owl")
#Latin name
Tyto_alba <- photosearcher::photo_search(mindate_taken = "2000-01-01",
                                         maxdate_taken = "2020-01-01",
                                         maxdate_uploaded = "2020-01-01",
                                         text = "Tyto alba")



#add column for taxa
```

```
barn_owl$taxa <- "Barn owl"
Tyto_alba$taxa <- "Tyto alba"

#summarise data
dat <- rbind(barn_owl,
             Tyto_alba)

summary_dat <- dat %>%
  mutate(lat = round(latitude),
         long = round(longitude)) %>%
  group_by(lat, long, taxa) %>%
  summarise(n_photo = n()) %>%
  mutate(taxa = paste0(toupper(substr(taxa, 1, 1)), substr(taxa, 2,
nchar(taxa))))

#plot map
ggplot(summary_dat, aes(x = long, y = lat, colour = taxa)) +
  borders("world", colour = "gray87", fill = "gray87") +
  geom_point(size = 0.5) +
  scale_x_continuous(name = "Longitude") +
  scale_y_continuous(name = "Latitude") +
  facet_wrap(~ taxa, nrow = 2) +
  coord_fixed() +
  theme_bw(base_size = 9) +
  theme(strip.background = element_blank(),
        plot.margin=grid::unit(c(0.25,0.25,0.25,0.25), "mm"),
        legend.position = "none")

#save plot
ggsave(filename = "tyto_dist.png",
       height = 9.525,
       width = 9.525,
       units = "cm")
```

Plotting temporal distributions

The following lines of code were used to create the species distribution plots.

```
#Search for images
#common name
brown_bear <- photosearcher::photo_search(mindate_taken = "2000-01-01",
                                            maxdate_taken = "2020-01-01",
                                            maxdate_uploaded = "2020-01-01",
                                            text = "brown bear")
#Latin name
Ursus_arctos <- photosearcher::photo_search(mindate_taken = "2000-01-01",
                                            maxdate_taken = "2020-01-01",
                                            maxdate_uploaded = "2020-01-01",
                                            text = "Ursus arctos")
#add column for taxa
```

```r
brown_bear$taxa <- "Brown bear"
Ursus_arctos$taxa <- "Ursus arctos"

#number of photographs per month
brown_bear$month <- as.numeric(substr(brown_bear$datetaken, 6, 7))
bb_months <- data.frame(table(unlist(brown_bear$month))) %>%
  rename(Month = Var1) %>%
  rename(Frequency = Freq) %>%
  mutate(Taxa = "Brown bear")

Ursus_arctos$month <- as.numeric(substr(Ursus_arctos$datetaken, 6, 7))
ua_months <- data.frame(table(unlist(Ursus_arctos$month))) %>%
  rename(Month = Var1) %>%
  rename(Frequency = Freq) %>%
  mutate(Taxa = "Ursus arctos")

months_dat <- rbind(bb_months, ua_months)

ggplot(data = months_dat, aes(x = Month, y = Frequency, fill = Taxa)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_bw(base_size = 9) +
  theme(strip.background = element_blank()) +
  scale_y_continuous(limits = c(0,4000), expand = c(0, 0)) +
  scale_x_discrete(labels=c("1" = "Jan",
                            "2" = "Feb",
                            "3" = "Mar",
                            "4" = "Apr",
                            "5" = "May",
                            "6" = "Jun",
                            "7" = "Jul",
                            "8" = "Aug",
                            "9" = "Sep",
                            "10" = "Oct",
                            "11" = "Nov",
                            "12" = "Dec"))

#save plot
ggsave(filename = "bear_times.png",
       height = 7.365,
       width = 14.287,
       units = "cm")
```