Appendix S1 - Compiling NA BBS Observations

Nicholas J Clark, David J Harris, Ceridwen Fraser

Specify the years to gather North American Breeding Bird Survey (BBS) data. Note, crashes can sometimes occur when using multiple years. We found it is best to run the below functions separately for each year and save the resulting object as an `.Rdata` file. You can then bring all saved files in as a list and compile them at once (see below for methods to filter and compile)

```r
year <- 2003
```

Gather the full dataset of BBS observations from USA (840) for the specified years

```r
# install.packages('pullBBS')
species.df <- pullBBS::pullBBS(year = year, country = 840,
    useCache = T)
```

Add BirdTree.org species names and eliminate records of unrecognised species

```r
# install.packages('devtools')
devtools::install_github("nicholasjclark/BBS.occurrences")
library(BBS.occurrences)
species.df <- validateSpecies(species.df)
```

Save the dataset to a new folder

```r
dir.create("./BBS.Data.cleaned")
save(species.df, file = "./BBS.Data.cleaned/USA.BBS.2003cleaned.Rdata")
```

Repeat for the next year, etc. Once all of the data is downloaded, we can read in all the separate yearly BBS datasets and filter them

```r
all.files <- list.files(path = "./BBS.Data.cleaned/",
    pattern = "USA.BBS")

mylist <- lapply(all.files, function(x) {
    load(file = paste("./BBS.Data.cleaned/", x, sep = ""))
    get(ls()[ls() != "filename"])
})

names(mylist) <- all.files
species.df <- do.call("rbind", mylist)
```

BBS runs are only valid if `RunType == 1` and if `Run protocol (RPID) == 101`. In addition, only `RouteTypeDetailID == 1` and `RouteTypeDetailID == 2` are considered random. We filter to match these conditions, and add a `ydays` numeric variable observation for the day

```r
# install.packages('dplyr')
# install.packages('lubridate')
species.df = species.df %>%
dplyr::filter(RPID == 101 & RunType == 1 & RouteTypeID ==
    1 & RouteTypeDetailId %in% c(1, 2)) %>%
dplyr::mutate(ydays = lubridate::yday(lubridate::ymd(paste(year,
    Month, Day, sep = "-")))) %>%
dplyr::mutate(start.times = as.numeric(as.difftime(lubridate::hm(gsub("^(.)(.*)$",
    "\\1:\\2", StartTime))))/60/60)
```

We can now create site by species abundance and occurrence matrices

```r
route.abundance = species.df %>% dplyr::mutate_at(dplyr::vars(dplyr::starts_with("Stop")),
    dplyr::funs(as.numeric)) %>% dplyr::mutate(Sp.abun = rowSums(.[grep("Stop",
    names(.))], na.rm = TRUE)) %>% dplyr::select(RouteDataID,
    BirdTree.species, Sp.abun) %>% dplyr::group_by(RouteDataID,
    BirdTree.species) %>% dplyr::summarise(total = sum(Sp.abun)) %>%
    tidyr::spread(BirdTree.species, total, fill = 0)

row.names(route.abundance) <- route.abundance$RouteDataID
route.abundance$RouteDataID <- NULL

route.presence.absence <- route.abundance
route.presence.absence[route.presence.absence > 0] <- 1
```

Finally, we compress the `pullBBS` dataset to one row per site, using `slice` to remove duplicate sites with both faranheit & celsius records (keeping only the farenheit record)

```r
sites.df = species.df %>% dplyr::select(matches("Route|Temp|Wind|Sky|Type"),
    year, countrynum, statenum, RPID, Month, Day, ObsN,
    Stratum, Latitude, Longitude, ydays, start.times,
    BCR) %>% distinct %>% group_by(RouteDataID) %>%
    slice(which.max(StartTemp))
```

Before saving, we order site data to match row orders of site x species matrices

```r
sites.df <- sites.df[match(rownames(route.abundance),
    sites.df$RouteDataID), ]
```

Save the matrices, site-level data and a matrix of lats and longs (for downloading `MODIS` landcover data)

```r
latlon <- cbind(sites.df$Longitude, sites.df$Latitude)
save(route.presence.absence = route.presence.absence,
    route.abundance = route.abundance, latlon = latlon,
    sites.df = sites.df, file = "./BBS.Data.cleaned/BBS.2003-2009.filtered.Rdata")
```