Supporting Information

Monitoring population risk with community science data

Authors…(removed for peer review in Methods in Ecology and Evolution)

07 December 2024

Contents

1 Read me (begin here)

This file is the code used for a paper *submitted* to ***Methods in Ecology and Evolution***. We hope this code serve as a practical tutorial and applied for different users in an intuitive way. Our aim is to describe and test a framework to estimate risk-based population viability - local persistence probability (ϕ) from time series of community science data (eBird). Using a case study, we assessed the accuracy of local persistence probability estimates by comparing estimates using eBird data with those from standardized monitoring. Then, we conducted a sensitivity analysis to assess how behaves our approach to limitation in the number of eBird observations available. We used the risk-based viable population monitoring (VPM) framework (Staples *et al.*, 2005), fitting continuous state-space population models under density-independent dynamics (Humbert *et al.*, 2009, but see below).

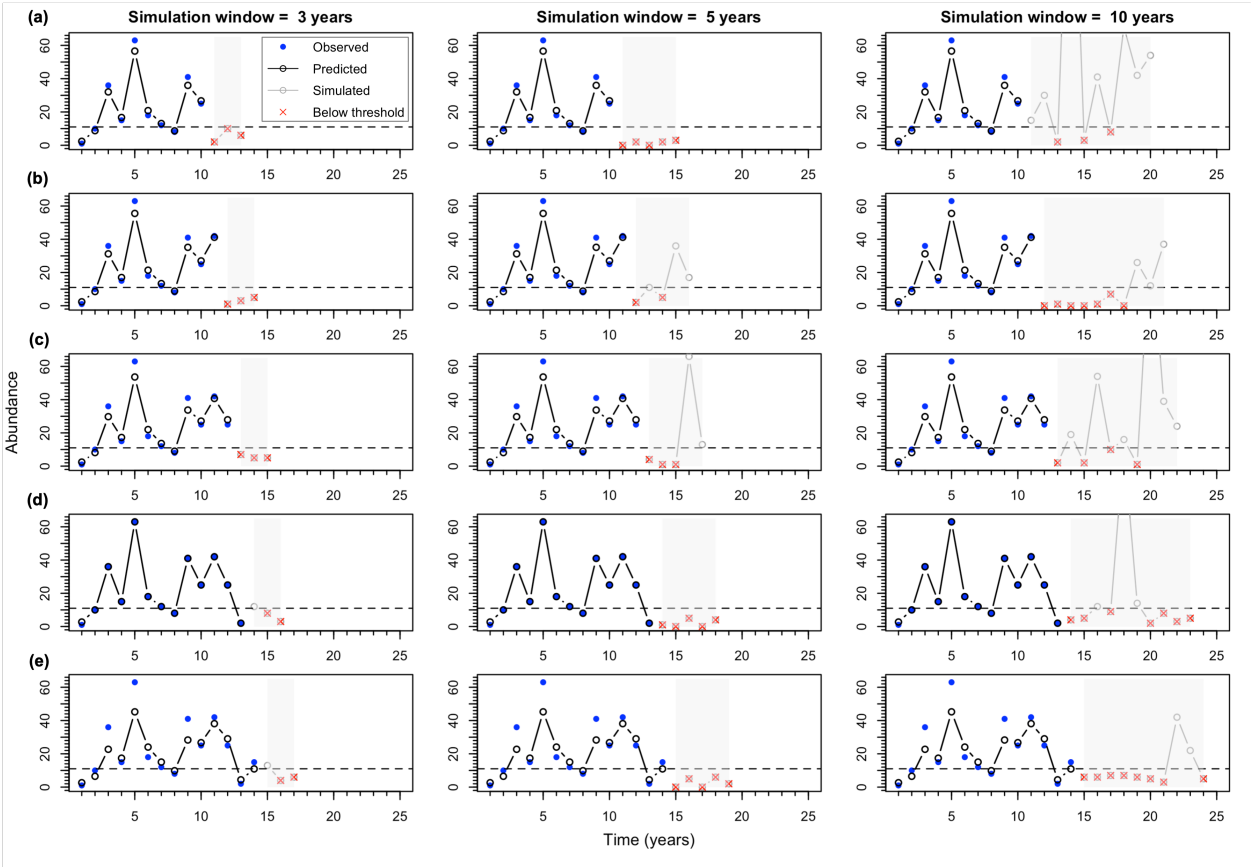


Figure SI-1: The VPM framework - Figure 1 in the main text

This population dynamic can be (and we provide the code for) extended to density-dependent dynamics (Dennis & Ponciano, 2014). We focused on the recent (since 2018) expanded population of Everglade snail kite (*Rostrhamus sociabilis plumbeus*) in north-central Florida, US, a species with standardized monitoring efforts that provide benchmark population trends for comparison (see Section 2.1 in the main text).

Two continuous versions of the discrete-time equal sampling Gompertz State-Space model can be fitted with our approach:

1. The density-dependent Ornstein-Uhlenbeck State-Space model (OUSS)
2. The density-independent Exponential Growth State Space model (EGSS)

We present the risk-based viable population monitoring framework (see Staples *et al.*, 2005), estimating the probability of local persistence ϕ; 1-probability of crashing abundance below a specified threshold given a simulation window in near future.

The statistical properties for our proceedings are based on Dennis *et al.* (1991), Dennis *et al.* (2006), Dennis & Ponciano, (2014), Humbert *et al.*, (2009), and reference therein. We adjusted the functions and code published as supplementary information in Humbert *et al.*, (2009) and Dennis & Ponciano, (2014). Thus, users should first run all the packages and functions in the next section. Then, access eBird data and organize it as time series in spatiotemporal subsamples and estimate local persistence - monitoring population risk with community science data.

2 Packages, models, and functions

2.1 Packages required

```
#R functions and datasets to support "Modern Applied Statistics with S",
  #a book from W.N. Venables and B.D. Ripley
  library(MASS);
#Kernel Density Estimation
  library(kde1d)
#To conduct eBird data filtering and manipulation (see Strimas et al. 2018 and 2023)
  library(auk);
#To data management and visualization - sevent packages in one
  library(tidyverse)
#To conduct Spatiotemporal Subsampling
  library(dggridR)
#Simple features to encode spatial vector data
  library(sf)
#load maps
  library(maps);
#composite figure
  library(gridExtra);
#equation in figure with panels (`ggplot2::ggplot()`; `facet_wrap()`)
  library(ggpubr);
```

2.2 Special cases of the Gompertz state-space population dynamics model

We provide the rationale to use two special cases (continuous, diffusion process) models of the discrete-time Gompertz State Space model (Dennis *et al.*, 2006); the density-dependent version that we heretofore denotes as the GSS model and the density-independent model (EGSS) from Humbert *et al.* (2009).

Let N_t be the latent unobserved abundance population at time t, the ecological process in the GSS model. This abundance is defined as $N_t = N_{t-1}\, e^{a+b*\ln N_{t-1}+E_t}$), where a and b are constants representing population growth rate and strength of density dependence, respectively, and E_t is the environmental stochasticity or process noise, $E_t \sim \text{Normal}(0, \sigma^2)$ (Dennis *et al.*, 2006). On the logarithmic scale ($X_t = \ln\ N_t$), the GSS becomes linear and follows an autoregressive model of order 1: $X_t = X_{t-1} + a + b * X_{t-1} + E_t$, which can be simplify as $X_t = a + c * X_{t-1} + E_t$, where $c = b + 1$ is a constant that represents the strength of density dependence (Dennis *et al.*, 2006; Ponciano *et al.*, 2018; Reddingius, 1971 in Acta Biotheor, 20, 1-208). If $b = 0$ ($c = 1$), the GSS reduces to the density-independent model fully treated in state-space model form by Humbert *et al.* (2009).

The discrete GSS population model has four unknown parameters: a, c, σ^2, and τ^2 (Dennis *et al.*, 2006). The transition probability distribution of this logarithmic abundance model is normal, with mean and variance changing as a function of time. The parameter c represents the strength of density-dependence (Ponciano *et al.*, 2018). If the strength of density dependence (c) ranges $-1 < c < 1$, the long-run probability distribution of log-abundance approaches a time-independent normal stationary distribution ($X_t \to X_\infty$), with mean $\frac{a}{1-c}$ and variance $\frac{\sigma^2}{1-c^2}$. Thus, instead of approaching a single population abundance value, or a carrying capacity like deterministic population models, the density-dependent stochastic GSS model approaches a stationary distribution, a cloud of points around which the population fluctuates (Dennis & Taper, 1994; Wolda, 1989). The mean of this probability distribution, $\frac{a}{1-c}$, represents a long-term expected population size (Dennis *et al.*, 2006). As mentioned above, the density-independent stochastic exponential growth model of Dennis *et al.* (1991) is attained when $b = 0$ ($c = 1$); the state equation then becomes $N_t = N_{t-1}\, e^{a+E_t}$ (Dennis *et al.*, 1991). The state-space model formulation for these two models is completed with the specification of the sampling (observation) error model. Following Dennis *et al.* (2006), we assumed that the log-observation at time t, Y_t, is a sample from the (stochastic) process model according to the equation $Y_t = X_t + F_t$; where the F_t are independent and identically distributed normal random variables, i.e. $F_t \sim \text{Normal}(0, \tau^2)$. During

transition growth dynamics, the EGSS model may more accurately represents population trends (but see Dennis & Ponciano, 2014 for details in non-stationary continuous GSS).

Fitting both state-space models when equally sampled population abundances are available is straightforward (Dennis *et al.*, 2006), yet unequally sampled populations over time tend to be the norm rather than the exception in ecology (Dennis et al., 2010). By exploiting the connection of the EGSS and GSS models to diffusion processes, here we show how to connect these models to unequally sampled data to later extend inferences to the dynamic of local persistence (as mathematical complement of local extinction risks). Indeed, the logarithmic transformation in the GSS discrete model opens the opportunity for estimating the infinitesimal mean and variance under diffusion processes for unequal sampling; Brownian motion diffusion in the EGSS model (Humbert *et al.*, 2009) and Ornstein-Uhlenbeck diffusion in the GSS model (Dennis & Ponciano, 2014). Specific statistical properties can be found in the following section with the specific functions per model.

2.3 Functions

The following functions are required to manipulate data and fit the models correctly.

2.3.1 Miscelaneous functions

Function to convert time observation to hours since midnight in eBird data

```
time_to_decimal <- function(x) {
  x <- hms(x, quiet = TRUE)
  hour(x) + minute(x) / 60 + second(x) / 3600
}
```

Multivariate normal random number generator - State Space models

```
randmvn <- function(n, mu.vec, cov.mat){

  # Save the length of the mean vector of the multivariate normal distribution to sample
  p         <- length(mu.vec);
  # The Cholesky decomposition
    #(factorization of a real symmetric positive-definite sqr matriz)
  Tau       <- chol(cov.mat, pivot=TRUE);
  # generate normal deviates outside loop
  Zmat      <- matrix(rnorm(n=p*n,mean=0,sd=1),nrow=p,ncol=n);

  # empty matrix
  out       <- matrix(0,nrow=p,ncol=n);
  # iterate
  for(i in 1:n){
    Z       <- Zmat[,i];
    out[,i] <- t(Tau)%*%Z + mu.vec
  }

  return(out)
}
```

Function to generate equation of a simple linear model in the figures

```
  lm_eqn <- function(df, x, y){
    m <- lm(y ~ x, df);
    eq <- substitute(italic(y) == a + b %.% italic(x)*","~~italic(R)^2~"="~r2,
         list(a = format(unname(coef(m)[1]), digits = 2),
              b = format(unname(coef(m)[2]), digits = 2),
```

```
            r2 = format(summary(m)$r.squared, digits = 3)))
    as.character(as.expression(eq));
}
```

2.3.2 Functions for the Brownian diffusion EGSS model

The stochastic exponential growth model serve as a null hypothesis of density-dependence models (Dennis *et al.*, 2006). The exponential stochastic process for N_t could be defined as $N_t = N_0\lambda^t + E_t$, where λ^t expresses the finite rate of change ($\lambda^t = e^{a(t)}$); a being the instantaneous, intrinsic, or maximum per capita rate of change), N_0 is the initial population ($N(0)$), and E_t is the environmental stochasticity or process noise at time t, which follows a normal distribution with mean 0 and variance σ^2 ($E_t \sim \text{Normal}(0, \sigma^2)$). On the logarithmic scale ($X_t = \ln(N_t)$), the discrete-time process can be defined as a continuous Brownian diffusion process with the stochastic differential equation:

$$\begin{aligned} dX(t) &= \ln\lambda dt + \beta dW(t) \\ &= \theta\ dt + \beta dW(t) \end{aligned}$$

where $X(t) = \ln N(t)$ or the log-normal population abundance at time t, $\theta = \ln\lambda$ is a constant of population growth rate ($\ln\lambda - (\frac{\sigma^2}{2})$ as in the ***Eq13*** in Dennis *et al.* (1991); named $\mu = \ln\lambda$ in Humbert *et al.* (2009); although this notation is confusing for the OUSS notation), and $dW(t)$ is a random perturbation representing the environmental stochasticity or process noise (Itô log-transformation of E_t from GSS in Dennis *et al.* (2006)), with mean 0 and variance $\sigma^2 dt$, or the intensity of environmental noise scaled by β, with $\beta > 0$ (Humbert *et al.*, 2009).

The EGSS model includes a component in sampling times t_i not equally spaced. Thus, we denoted the latent realized abundance (e.g., weekly high counts in our case) as $n(0), n(t_1), n(t_2), ..., n(t_q)$. Let $x(t) = \ln n(t)$ and let $Y(t_i)$ be a value of $x(t)$ observed with error at time t_i. Then, our log-abundance observation model equation becomes $Y(t_i) = X(t_i) + F_i$, where F_i follows a normal distribution with mean 0 and variance τ^2 ($F_i \sim \text{Normal}(0, \tau^2)$). This state-space model has four unknown parameters: $\ln\lambda = \theta_{density\ independent}$ (the trend parameter or population growth rate under density independence; note the notation differs from Humbert *et al.* (2009)), σ^2 (variability of process noise), τ^2 (variability of observer noise), and x_0 (the initial log-abundance population).

The EGSS model has a multivariate normal log-likelihood function given by (***EqA17*** in Humbert *et al.*, (2009)):

$$\ln L(x_0, \theta, \sigma^2, \tau^2) = -\frac{(q+1)}{2}\ln(2\pi) - \frac{1}{2}\ln(|\mathbf{V}|) - \frac{1}{2}(\mathbf{y}-\mathbf{m})'\mathbf{V}^{-1}(\mathbf{y}-\mathbf{m})$$

For the numerical optimization, we will need three arguments:

1. A vector of time-series log-observed abundances `yt` ($\mathbf{y}$)
2. A vector of observation times `tt`
3. A vector of initial parameters `fguess` (a first guess for the **four** parameters in EGSS), that could be roughly computing by provide the vector of log-abundance observations `yt` ($\mathbf{y}$) and the vector of observation times `tt` with function `guess_eggs()`

```
guess_egss <- function(yt,tt){

  # Time-vector starting in 0.
  t.i       <- tt-tt[1];
  # Number of time-series transitions
  q         <- length(yt)-1;
  # length of time-series
  qp1       <- q+1;
```

```
  # time intervals (named as S.t in H_O and sometimes in DP_E)
  t.s       <- t.i[2:qp1]-t.i[1:q];

  #The Exponential Growth Observation Error (EGOE in H_O) initial values
  # mean of the observations as assumed to arise from stationary distribution
  Ybar      <- mean(yt);
  # mean of the time series
  Tbar      <- mean(t.i)
  # trend parameter for EGOE (theta = ln(lambda) in H_O)
  theta.egoe    <- sum((t.i-Tbar)*(yt-Ybar))/sum((t.i-Tbar)*(t.i-Tbar));
  # Initial population  of EGOE
  x0.egoe   <- Ybar-theta.egoe*Tbar
  # sigma square for EGOE is 0 (assume no ecological process variation)
  ssq.egoe  <- 0
  # estimate of initial population observed under EGOE
  Yhat.egoe <- x0.egoe+theta.egoe*t.i;
  # initial value for tau^2
  tsq.egoe  <- sum((yt-Yhat.egoe)*(yt-Yhat.egoe))/(q-1);

  #The Exponential Growth Process Noise (EGPN in H_O) initial values
  # Square root of time intervals (time trend?)
  Ttr       <- sqrt(t.s);
  # Observed trend?
  Ytr       <- (yt[2:qp1]- yt[1:q])/Ttr;
  # trend parameter for EGPN (mu = ln(lambda) in H_O)
  theta.egpn    <- sum(Ttr*Ytr)/sum(Ttr*Ttr);
  # Trend of observed estimated
  Ytrhat    <- theta.egpn*Ttr;
  # initial value for sigma^2
  ssq.egpn  <- sum((Ytr-Ytrhat)*(Ytr-Ytrhat))/(q-1);
  # tau square for EGPN is 0 (assume no observation variation)
  tsq.egpn  <- 0;
  # Initial population  of EGPN is the first observation
  x0.egpn   <- yt[1];

  #four parameters needed in EGSS and OUSS.NoSt
  theta0    <- (theta.egoe+theta.egpn)/2;
  ssq0      <- ssq.egpn/2;
  tsq0      <- tsq.egoe/2;
  x0.out    <- (x0.egoe+x0.egpn)/2;

  return(c(theta0, ssq0, tsq0, x0.out))
}
```

The numerical optimization for computing the Restricted Maximum Likelihood Estimates for the multivariate normal distribution of parameters in R is:

```
negloglike_egss_remle <- function(fguess,yt,tt){

  sigmasq <- exp(fguess[1]); #in egss_remle, this have only the two parameters
  tausq   <- exp(fguess[2]);
  q       <- length(yt) - 1;
  qp1     <- q+1;
```

```
  ss      <- tt[2:qp1]-tt[1:q];
  wt      <- (yt[2:qp1]-yt[1:q])/ss;
  ut      <- wt[2:q]-wt[1:q-1];
  vx      <- matrix(0,qp1,qp1);
  for(i in 1:q){
    vx[(i+1):qp1,(i+1):qp1] <- matrix(1,(qp1-i),(qp1-i))*tt[i+1];
  }
  Sigma.mat<- sigmasq*vx;
  Itausq  <- matrix(rep(0,(qp1*qp1)), nrow=qp1, ncol=qp1);
  diag(Itausq)<- rep(tausq,qp1);
  V       <- Sigma.mat + Itausq;

  D1mat   <- cbind(-diag(1/ss),matrix(0,q,1))+cbind(matrix(0,q,1),diag(1/ss));
  D2mat   <- cbind(-diag(1,(q-1)),matrix(0,(q-1),1)) + cbind(matrix(0,(q-1),1),diag(1,(q-1)));
  V2      <- D2mat%*%D1mat%*%V%*%t(D1mat)%*%t(D2mat);

  ofn=((q-1)/2)*log(2*pi)+(0.5*log(det(V2))) + (0.5*(ut%*%ginv(V2)%*%ut));

  return(ofn)
}
```

To compute the EGSS-REMLEs we use the function `egss_remle()`

```
egss_remle <- function(yt,tt,fguess){

  #Temporal vectors
  t.i     <- tt-tt[1];
  q       <- length(t.i)-1;
  qp1     <- q+1;
  t.s     <- t.i[2:qp1] - t.i[1:q];

  # initial guesses (sigmasq and tausq at log scale)
  guess.optim <- c(log(fguess[2:3]))
  # numerical optimization
  optim.out   <- optim(par=guess.optim,
                       fn=negloglike_egss_remle,
                       method="Nelder-Mead",
                       yt=yt,
                       tt=t.i)

  #extract parameters estimated by REML
  sigmasq <- exp(optim.out$par)[1]
  tausq <- exp(optim.out$par)[1]

  #to estimate trend parameter (theta) and initial population (x0)
  vx      <- matrix(0,qp1,qp1);
  for(i in 1:q){
    vx[((i+1):qp1),((i+1):qp1)] <- matrix(1,(qp1-i),(qp1-i))*t.i[(i+1)];
  }
  Sigma.mat     <- sigmasq*vx;
  Itausq        <- matrix(rep(0,(qp1*qp1)),
                          nrow=qp1,
                          ncol=qp1);
  diag(Itausq)  <- rep(tausq,qp1);
```

```
  V             <- Sigma.mat + Itausq;
  D1mat=cbind(-diag(1/t.s),
              matrix(0,q,1))+cbind(matrix(0,q,1),
                                   diag(1/t.s));
  V1mat=D1mat%*%V%*%t(D1mat);
  W.t=(yt[2:qp1]-yt[1:q])/t.s;
  j1=matrix(1,q,1);
  V1inv=ginv(V1mat);

  #Trend parameter
  theta.remle=(t(j1)%*%V1inv%*%W.t)/(t(j1)%*%V1inv%*%j1);

  j=matrix(1,qp1,1);
  Vinv=ginv(V);

  #initial population
  x0.remle=(t(j)%*%Vinv%*%(yt-as.numeric(theta.remle)*t.i))/(t(j)%*%Vinv%*%j);

  #Extract REMLEs and AIC
  remles      <- c(theta.remle,exp(optim.out$par[1:2]),x0.remle)
  lnL.hat     <- - optim.out$value[1]
  AIC         <- -2*lnL.hat + 2*2 #where 2 = length(REMLEs)...

  out         <- list(remles=remles,
                      lnL.hat = lnL.hat,
                      AIC=AIC)
  return(out)
}
```

With the EGSS-REMLE values, we can predict the trajectory of the latent ecological process with the function `egss_predict()`

```
egss_predict <- function(yt,tt,parms,plot.it="TRUE"){

  # Time-vector starting in 0.
  t.i     <- tt-tt[1];
  q       <- length(t.i)-1;
  qp1     <- q+1;
  t.s     <- t.i[2:qp1] - t.i[1:q];

  # parameters ()
  theta.remle <- parms[1];
  sigmasq     <- parms[2];
  tausq       <- parms[3];
  x0.remle    <- parms[4];

  #Calculate estimated population size for EGSS model

  m=rep(1,qp1); # Will contain Kalman means for Kalman calculations.
  v=rep(1,qp1); # Will contain variances for Kalman calculations.

  m[1]=x0.remle; # Initial mean of Y(t).
  v[1]=tausq; # Initial variance of Y(t).
```

```
  for (ti in 1:q) # Loop to generate estimated population abundances
  { # using Kalman filter (see equations 6 & 7, # Dennis et al. (2006)).
    m[ti+1]=theta.remle+(m[ti]+((v[ti]-tausq)/v[ti])*(yt[ti]-m[ti]));
    v[ti+1]=tausq*((v[ti]-tausq)/v[ti])+sigmasq+tausq;
  }

  # The following statement calculates exp{E[X(t) | Y(t), Y(t-1),...,Y(0)]};
  # see equation 54 in Dennis et al. (2006).

  Predict.EGSS.REML = exp(m+((v-tausq)/v)*(yt-m));

  if(plot.it=="TRUE"){
    #  Plot the data & model-fitted values
    #X11()
    plot(tt,exp(yt),xlab="Time",ylab="Population abundance",
         type="b",cex=1.5, lwd = 1.5, lty = 1,
         main="Predicted (--) and observed (-o-) abundances");
            # Population data are circles.
    points(tt,Predict.EGSS.REML, type="l", lwd=1, lty = 2);
  }

  return(list(cbind(Time = tt, Predict.EGSS.REML, Observed.y = exp(yt))))
}
```

And also simulate trajectories with the function `egss_sim()`

```
egss_sim <- function(nsims,tt,parms){

  # time and temporal scale
  t.i   <- tt-tt[1];
  q     <- length(t.i)-1;
  qp1   <- q+1;

  # parameters
  theta <- parms[1];
  sigmasq<- parms[2];
  tausq <- parms[3];
  x0    <- parms[4];

  vx    <- matrix(0,qp1,qp1);
  for(i in 1:q){
    vx[((i+1):qp1),((i+1):qp1)] <- matrix(1,(qp1-i),(qp1-i))*t.i[(i+1)];
  }

  Sigma.mat<- sigmasq*vx;
  Itausq<- matrix(rep(0,(qp1*qp1)),
                          nrow=qp1,
                          ncol=qp1);
  diag(Itausq)  <- rep(tausq,qp1);
  V     <- Sigma.mat + Itausq;
  theta.vec     <- matrix((x0+theta*t.i),
                          nrow=qp1,
                          ncol=1);
  out   <- randmvn(n=nsims,
```

```
                    mu.vec=theta.vec,
                    cov.mat=V);

  return(out)
}
```

2.3.3 Functions for the Ornstein-Uhlenbeck diffusion GSS - OUSS model

The key to generalize the GSS for unequal sampling intervals lies in the mathematical insight that the solution of the discrete-time GSS model matches the solution at discrete time points of a diffusion process, which is a continuous time stochastic process. Specifically, the solution of the discrete time Gompertz model in the log scale matches exactly the well-known Ornstein-Uhlenbeck (OU) Gaussian diffusion process (Dennis & Ponciano, 2014). Let N_t represent the population abundance in the Gompertz diffusion with environmental stochasticity and no demographic stochasticity (Dennis & Ponciano, 2014; Ponciano, 2018). This diffusion process is a well-known continuous-time version of an autoregressive process of order 1, characterized by a joint multivariate normal distribution of values across time points. The process is defined by its infinitesimal mean, variance, and covariance parameters (Dennis & Ponciano, 2014; Ponciano, 2018).

The infinitesimal mean and variance of the process are given by: $m_N(n) = \theta n[\ln \kappa - \ln\ n]$ and $\sigma^2_N(n) = \beta^2 n^2$, respectively; here θ represents the speed of equilibration, κ the equilibrium abundance, and β scales the random perturbation by environment stochasticity, dW. The OU diffusion process is usually presented in its stochastic differential equation form $dN_t = \theta N_t[\ln \kappa - \ln N_t)dt + \beta N_t dW_t$. A smooth transformation to N_t, given by $X_t = g(N_t)$ (e.g. $X_t = \ln N_t$), is also a diffusion process whose infinitesimal mean is given by $m_X(x) = m_N(n)g'(n) + 1/2\sigma^2_N(n)g''(n)$ and infinitesimal variance by $\sigma^2_X(x) = \sigma^2_N n[g'(n)]^2$, where $n = g^{-1}(x)$. This result is the well-known Itô-transformation for diffusion processes widely used in stochastic population dynamics modeling (see citations in the Main text). For $g(n) = \ln\ n$, the infinitesimal mean simplifies to $m_X(x) = \theta(\mu - x)$, where $\mu = \ln \kappa - \frac{\beta^2}{2\theta}$, and the infinitesimal variance becomes $\sigma^2_X(x) = \beta^2$. Thus, the stochastic differential equation of the logarithmic process X_t is $dX_t = \theta(\mu - X_t)dt + \beta dW_t$. If the process starts at an initial log-abundance $X_0 = x_0$, the expected value and variance at any time t are given by $\mathrm{E}[X_t|X_0 = x_0] = \mu - (\mu - x_0)e^{\theta - t}$ and $\mathrm{V}[X_t|X_0 = x_0] = \frac{\beta^2}{2\theta}(1 - e^{-2\theta})$ respectively (Dennis & Ponciano, 2014). Over time, the process converges to a stationary distribution with mean μ and variance $\frac{\beta^2}{2\theta}$.

The one-to-one relationships of the discrete equal sampling GSS model parameters to the continuous OUSS model parameters are (Dennis & Ponciano, 2014):

$$\begin{array}{rcc} a & = & \mu(1 - e^{-\theta}) \\ c & = & e^{-\theta} \\ \sigma^2 & = & \frac{(1-e^{-2\theta})\beta^2}{2\theta} \\ \tau^2 & = & \tau^2 \end{array}$$

Thus, the OUSS model has four unknown parameters under stationary distribution: μ (mean stationary log-abundance), $\theta_{density\ dependent}$ (the trend parameter under density dependence, or rate to approach stationarity), β^2 (variability of the process noise), and τ^2 (variability of sampling). The OUSS model also adds a component in sampling times t_i not equally spaced $Y(t_i) = X(t_i) + F_i$, where the observation error keeps a normal distribution and the same unknown parameter ($F_i \sim \mathrm{Normal}(0, \tau^2)$), and the underlying unobserved population $X(t_i)$ follows a continuous-time version of the GSS model. With the strength of density dependence parameter (c) ranging between 0 and 1, the dynamic of the population is stationary.

The inverse relationship between the OUSS and the GSS model parameters are:

$$\begin{array}{rcc} \mu & = & \frac{a}{1-c} \\ \theta & = & -\ln c \\ \beta^2 & = & -\frac{2\sigma^2 \ln c}{1-c^2} \\ \tau^2 & = & \tau^2 \end{array}$$

The normal stationary probability distribution has mean μ and variance $\frac{\beta^2}{2\theta}$ (Dennis & Ponciano, 2014). If the initial log-abundance of the population does not meet this assumption (e.g., it is under transition growth), a nonstationary distribution could be modeled with a different maximum likelihood estimation approach (Dennis & Ponciano, 2014). The normal transition in nonstationary cases has a mean $\mu - (\mu - x_0)e^{-\theta t}$ and variance $(\frac{\beta^2}{2\theta}(1 - e^{-2\theta}))$, adding an extra parameter to estimate (x_0). Given that a restricted maximum likelihood estimation is not available for nonstationary OUSS (Dennis & Ponciano, 2014), we modeled the dynamic of the populations for density independence (including initial nonstationary distributions) as EGSS, while stationary distributions as OUSS.

The multivariate normal log-likelihood for the stationary OUSS model is given by (see ***Eq. 19*** Dennis & Ponciano, (2014)).

$$\ln L(\mu, \theta, \beta^2, \tau^2) = -\frac{(q+1)}{2}\ln(2\pi) - \frac{1}{2}\ln(|\mathbf{V}|) - \frac{1}{2}(\mathbf{y} - \mathbf{m})'\mathbf{V}^{-1}(\mathbf{y} - \mathbf{m})$$

where q is the number of time-series transitions (thus, $q+1$ reflect the length of the time-series, with the initial population estimation y_0 as a realized value of the random variable $Y(0)$), $\mathbf{V}$ is the variance-covariance matrix (with diagonal computed from $\mathrm{V}[Y(t_i)] = \tau^2 + \frac{\beta^2}{2\theta}$; ***Eq. 17*** in Dennis & Ponciano, (2014)), $\mathbf{y}$ is the data values (y_0, y_1, y_2, ..., y_q), and $\mathbf{m}$ is the vector of same μ in all $q+1$ times ($E[Y(t_i)] = \mu$]; ***Eq. 16*** in Dennis & Ponciano, (2014)).

This function requires three arguments for the numerical optimization in `R`:

1. A vector of time-series of log-observed abundances `yt` ($\mathbf{y}$)
2. A vector of observation times `tt`
3. A vector of parameters `fguess` (a first guess for the **four** parameters), that could be roughly computing by provide the vector of log-abundance observations `yt` ($\mathbf{y}$) and the vector of observation times `tt` with the function `guess_ouss()`

```
guess_ouss <- function(yt,tt){

  # Time-vector starting in 0.
  t.i     <- tt-tt[1];
  # Number of time-series transitions
  q       <- length(yt)-1;
  # length of time-series
  qp1     <- q+1;
  # time intervals
  t.s     <- t.i[2:qp1]-t.i[1:q];
  # mean of the observations as assumed to arise from stationary distribution
  Ybar    <- mean(yt);
  # Variance of the observations
  Yvar    <- sum((yt-Ybar)*(yt-Ybar))/q;
  # Initial mu estimate (at stationary distribution)
  mu1     <- Ybar;

  # Kludge an initial value for theta based on mean of Y(t+s) given Y(t).
  th1     <- -mean(log(abs((yt[2:qp1]-mu1)/(yt[1:q]-mu1)))/t.s);
  # Moment estimate using stationary distribution
  bsq1    <- 2*th1*Yvar/(1+2*th1);
  # Observation error variance, assumed as first guess as betasq=tausq.
  tsq1    <- bsq1;

  # What to do if initial guesses is three 0's (or NAs)? Assume arbitrary values
  three0s <- sum(c(th1,bsq1,tsq1))
```

```
  if(three0s==0|is.na(three0s)){
    th1   <- 0.5;
    bsq1  <- 0.09;
    tsq1  <- 0.23;}

  out1    <- c(th1,bsq1,tsq1);

  # What to do if initial guesses are too little? Assume arbitrary values
  if(sum(out1<1e-7)>=1){
    out1  <- c(0.5,0.09,0.23)}

  out     <- c(mu1,out1);

  return(abs(out))
}
```

The numerical optimization for computing the parameters Restricted Maximum Likelihood Estimate within the multivariate log-likelihood for the stationary Ornstein-Uhlenbeck State-Space (OUSS) in R is:

```
negloglike_ouss_remle=function(yt,tt,fguess){
  # Constrains parameters theta, beta^2, and tau^2 > 0

  # speed of equilibration (Eq1 in DP_E)
  theta  <- exp(fguess[2]);
  # variability of process noise
  betasq <- exp(fguess[3]);
  # variability of sampling
  tausq  <- exp(fguess[4]);
  # number of time-series transitions
  q      <- length(yt) - 1;
  # length of time-series
  qp1    <- q+1;
  # Variance (Eq11 in DP_E)
  Var.inf<- betasq/(2*theta);
  # time intervals (not used here?)
  t.s    <- tt[2:qp1] - tt[1:q];
  # part of Eq18 in DP_E
  t.cols <- matrix(rep(tt,each=qp1),
                           nrow=qp1,
                           ncol=qp1,
                           byrow=FALSE);
  # (part of Eq18 in DP_E)
  t.rows <- t(t.cols);
  # (part of Eq18 in DP_E)
  abs.diffs     <- abs(t.rows-t.cols);

  # Covariance of the process (Eq18 in DP_E)
  Sigma.mat     <- Var.inf*exp(-theta*abs.diffs);
  # Create a matrix full of 0s of the length of time series
  Itausq <- matrix(0,qp1,qp1);
  # Repeat the observation error variance guess in the diagonal of the matrix
  diag(Itausq)  <- rep(tausq,qp1);
  # add Covariance with the matrix
  V      <- Sigma.mat+Itausq;
```

```
  # Create the differencing matrix **D**
  Dmat   <- cbind(-diag(1,q),matrix(0,q,1)) + cbind(matrix(0,q,1),diag(1,q));
  # Variance-covariance matrix **Phi** (Eq20 DP_E)
  Phi.mat<- Dmat%*%V%*%t(Dmat);
  # simple differencing of the observations (W_i? )
  wt     <- yt[2:qp1]-yt[1:q];

  # note the signs change because we want here the negative log-likelihood (Eq22*-1)
  neglogl<- (q/2)*log(2*pi) + (1/2)*log(det(Phi.mat)) + (1/2)*wt%*%ginv(Phi.mat)%*%wt;

  # What to do if the `neglogl` is not finite? assign a big number of 50000
  if(is.infinite(neglogl)==TRUE){
    return(50000)}else{
      return(neglogl)}
}
```

To compute the OUSS-REMLEs we implement the function `ouss_remle()`

```
ouss_remle <- function(yt, tt, fguess){

  # Time-vector starting in 0.
  t.i           <- tt-tt[1];
  # Number of time-series transitions
  # length of time-series
  q             <- length(yt)-1;
  qp1           <- q+1;
  # time intervals
  t.s           <- t.i[2:qp1]-t.i[1:q];
  # initial guesses (all, but negloglike.OU.remle will use only fguess[2:4])
  guess.optim   <- c(fguess[1],
                     log(fguess[2:4]));
  # numerical optimization
  optim.out     <- optim(par = guess.optim,
                         fn=negloglike_ouss_remle,
                         method="Nelder-Mead",
                         yt=yt,
                         tt=t.i);
  # Restricted maximum likelihood estimates (REMLE) and lnL.hat
  remles        <- exp(optim.out$par);
  theta.remle   <- remles[2];
  betasq.remle  <- remles[3];
  tausq.remle   <- remles[4];

  lnL.hat       <- -optim.out$value[1];

  # Variance (Eq11 in DP_E)
  Var.inf       <- betasq.remle/(2*theta.remle)
  # creates an matrix full of 1 dim qp1 x qp1
  vx            <- matrix(1,qp1,qp1);
  # iterate to fill the matrix (couldn't find vx in DP_E!)
  for (t.i in 1:q){
    vx[(t.i+1):qp1,t.i]=exp(-theta.remle*cumsum(t.s[t.i:q]));
    vx[t.i,(t.i+1):qp1]=vx[(t.i+1):qp1,t.i];
  }
```

```
  # ?
  Sigma.mat     <- vx*Var.inf;
  # Create a matrix full of 0s of the length of time series
  Itausq        <- matrix(0,qp1,qp1);
  # Repeat the observation error variance remle in the diagonal of the matrix
  diag(Itausq)  <- rep(tausq.remle,qp1);
  # Variance-covariance matrix (V.hat) evaluated with remles to estimate mu.hat
  V.remle       <- Sigma.mat+Itausq;
  # column vector matrix of ones
  j             <- matrix(1,qp1,1);
  # Inverse matrix (part of Eq23 in DP_E)
  Vinv          <- ginv(V.remle);
  # REMLE of mu (mu.hat) with Eq23 in DP_E
  mu.remle      <- (t(j)%*%Vinv%*%yt)/(t(j)%*%Vinv%*%j);
  #AIC
  AIC           <- -2*lnL.hat + 2*4 #where 4 = length(mles)...

  #Results
  out           <- list(remles = c(mu.remle,
                                   theta.remle,
                                   betasq.remle,
                                   tausq.remle),
                        lnLhat = lnL.hat,
                        AIC = AIC)
  return(out)
}
```

With the OUSS-REMLE values, we can predict the trajectory with the function `ouss_predict()`

```
ouss_predict <- function(yt,tt,parms, plot.it="TRUE"){

  t.i             <- tt-tt[1];
  q               <- length(t.i)-1;
  qp1             <- q+1;

  # parameters
  mu              <- parms[1];
  theta           <- parms[2];
  betasq          <- parms[3];
  tausq           <- parms[4];

  Var.inf         <- betasq/(2*theta);
  t.s             <- t.i[2:qp1] - t.i[1:q];
  t.cols          <- matrix(rep(t.i,each=qp1),nrow=qp1,ncol=qp1, byrow=FALSE);
  t.rows          <- t(t.cols);
  abs.diffs       <- abs(t.rows-t.cols);

  nmiss           <- t.s-1;
  long.nmiss      <- c(0,nmiss);
  Nmiss           <- sum(nmiss)

  long.t          <- t.i[1]:max(t.i)
  where.miss      <- which(is.na(match(x=long.t,table=t.i)),
                           arr.ind=TRUE)
```

```
lt.cols          <- matrix(rep(long.t),
                           nrow=(qp1+Nmiss),
                           ncol=(qp1+Nmiss),
                           byrow=FALSE);
lt.rows          <- t(lt.cols);
labs.diffs       <- abs(lt.rows-lt.cols);

Sigma.mat        <- Var.inf*exp(-theta*abs.diffs);
Itausq           <- matrix(0,qp1,qp1);
diag(Itausq)     <- rep(tausq,qp1);
V                <- Sigma.mat+Itausq;

long.V           <- Var.inf*exp(-theta*labs.diffs) + diag(rep(tausq,(qp1+Nmiss)))

Predict.t        <- rep(0,qp1);
Muvec            <- rep(mu,q);
miss.predict     <- list()
Muvec.miss       <- rep(mu,qp1);
start.miss       <- 1
stop.miss        <- 0
for (tj in 1:qp1){
  Y.omitj        <- yt[-tj];     #  Omit observation at time tj.
  V.omitj        <- V[-tj,-tj];  #  Omit row tj and col tj from var-cov matrix.
  V12            <- V[tj,-tj];       #  Submatrix:  row tj without col tj.
  Predict.t[tj] <- mu+V12%*%ginv(V.omitj)%*%(Y.omitj-Muvec);  #  Graybill's 1976 Thm.

  if(long.nmiss[tj]==0){
    miss.predict[[tj]] <- Predict.t[tj]}else
      if(long.nmiss[tj]>0){

        start.miss <- stop.miss+1
        ntjmiss    <- long.nmiss[tj]
        mu.miss    <- rep(mu,ntjmiss);
        ind.tjmiss <- where.miss[start.miss:(start.miss+(ntjmiss-1))]
        stop.miss  <- stop.miss+ntjmiss

        longV12    <- long.V[ind.tjmiss,-where.miss]

        miss.predict[[tj]] <- c(mu.miss + longV12%*%ginv(V)%*%(yt-Muvec.miss),
                                Predict.t[tj])
      }
}

Predict.t <- exp(Predict.t);
LPredict.t <- exp(as.vector(unlist(miss.predict)))

isinf <- sum(is.infinite(Predict.t))
if(isinf>0){
  where.infs <- which(is.infinite(Predict.t)==TRUE, arr.ind=TRUE)
  Predict.t[where.infs] <- .Machine$double.xmax
}

isinf2 <- sum(is.infinite(LPredict.t))
```

```
  if(isinf2>0){
    where.infs <- which(is.infinite(LPredict.t)==TRUE, arr.ind=TRUE)
    LPredict.t[where.infs] <- .Machine$double.xmax
  }

  if(plot.it=="TRUE"){
    #  Plot the data & model-fitted values
    #X11()
    plot(tt,exp(yt),xlab="Time",ylab="Population abundance",type="b",cex=1.5,
         main="Predicted (--) and observed (-o-) abundances");
        # Population data are circles.
    par(lty="dashed"); #  Predicted abundances are dashed line.
    points(tt,Predict.t, type="l", lwd=1);
  }

  return(list(cbind(tt,Predict.t,exp(yt)), cbind(long.t,LPredict.t) ))
}
```

And also simulate trajectories with `ouss_sim()`

```
ouss_sim <- function(nsims,tt,parms){

  # Time-vector starting in 0.
  t.i       <- tt-tt[1];
  # Number of time-series transitions
  q         <- length(t.i)-1;
  # length of time-series
  qp1       <- q+1;

  # parameters
  mu        <- parms[1];
  theta     <- parms[2];
  betasq    <- parms[3];
  tausq     <- parms[4];

  Var.inf   <- betasq/(2*theta);
  t.s       <- t.i[2:qp1] - t.i[1:q];
  t.cols    <- matrix(rep(t.i,each=qp1),
                      nrow=qp1,
                      ncol=qp1,
                      byrow=FALSE);
  t.rows    <- t(t.cols);
  abs.diffs <- abs(t.rows-t.cols);
  V         <- Var.inf*exp(-theta*abs.diffs);
  diag(V)   <- diag(V) + rep(tausq,qp1);
  m.vec     <- rep(mu,qp1);
  out       <- randmvn(n=nsims,
                       mu.vec=m.vec,
                       cov.mat = V)
  return(out)
}
```

3 eBird data organization

Users should download the `ebd` file from eBird. See the supplement to Johnston *et al.* (2021) in Strimas-Mackey *et al.* (2023), which is a key reference for the next section.

3.1 Download eBird data

3.1.1 Go to eBird and sign in

Go to eBird. You have to sign in in eBird:

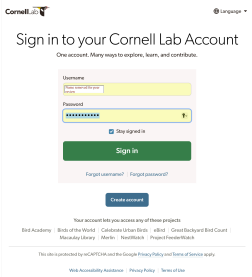


Figure SI-2: Sign in eBird

3.1.2 Request data

If you are in the home page, check you are sign-in and move down on the page to "Request data".

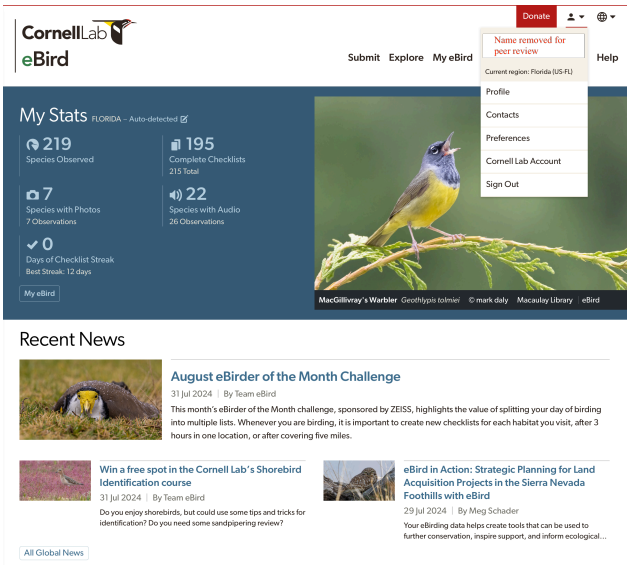


Figure SI-3: eBird home

You have to submit an application to have access to the data. Once you have access, click on "Basic dataset (EBD)"

Then, you can select by Species, region, date. In our case, lets download *Rostrhamus sociabilis* in Florida (US).

In the options, include the sampling event data always is a good recommendation. In snail kite for US it will save the sampling event, but for other Neotropical species tested in preliminary attempts, the sampling

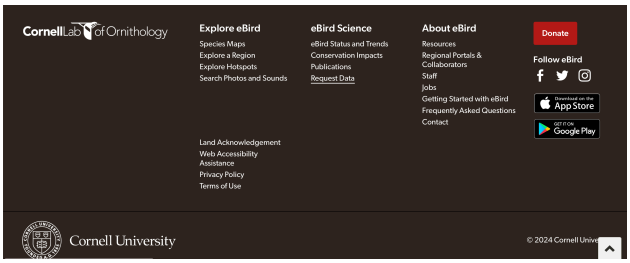


Figure SI-4: eBird home - lower part

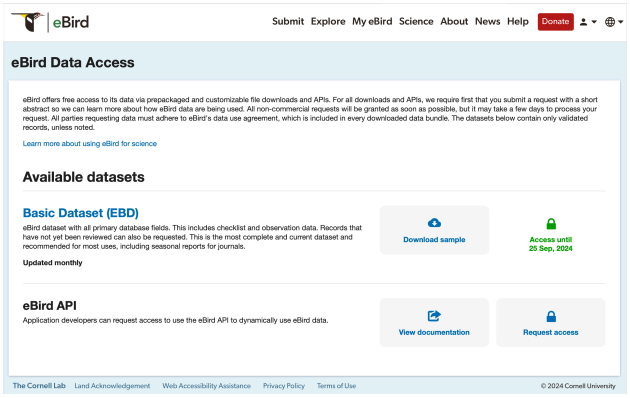


Figure SI-5: eBird data access

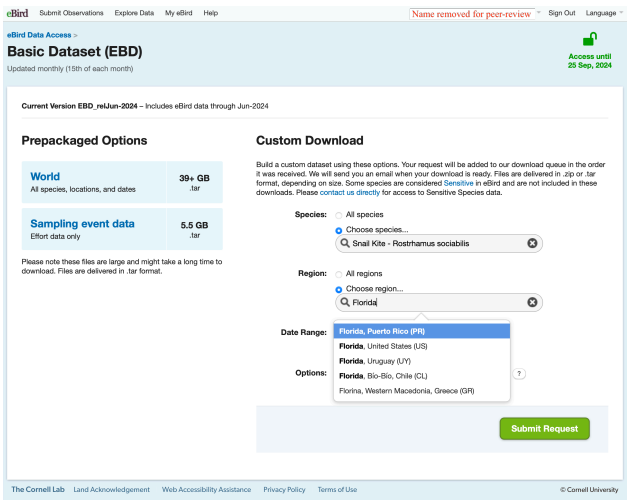


Figure SI-6: Asking Snail Kite in Florida, US - screenshot of a previous download for June 2024

data was not included and the user should download the Sampling event data of 5.5 GB (comprised in `.tar` format). This step is required if you are interested in control for imperfect detection!!

After submitting the request, the link to download will arrive to the email registered in your eBird account. You can deploy the `.txt` files in a `data_raw` directory to be called during the refining process through filtering.

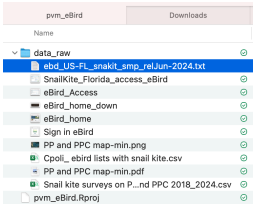


Figure SI-7: Downloaded file

This file will have the detection and observer counts for our species.

3.2 Pre-filtering

We can simplify the eBird data selecting only columns of our interest (it will reduce the size of the dataset)

```
colsE <- c("observer_id", "sampling_event_identifier",
           "group identifier",
           "common_name", "scientific_name",
           "observation_count",
           "country", "state_code", "locality_id", "latitude", "longitude",
           "protocol_type", "all_species_reported",
           "observation_date",
           "time_observations_started",
           "duration_minutes", "effort_distance_km",
           "number_observers")
```

To conduct some filters, we will generate temporal files in our computer. Here we generate only a single temporal file that can be overwritten to assess different species.

```
f_ebd <- "data_tmp/ebd_Examples.txt"
f_sed <- "data_tmp/sed_Examples.txt"
```

The construction of time-series of the individuals counted from eBird will assume spatiotemporal subsampling, selecting a single value with the high counts (assuming to be the minimum number of individuals detected) per week in spatial sampling units of $\sim$ 100 km^2. To construct a discrete global grid system, we can use the function `dgconstruct()` in the package `dgconstruct`; the argument `spacing` indicates the spacing between the center of adjacent cells (related with Characteristic Length Scale - CLS), in our case `spacing = 11` indicates a diameter of ~11 km, representing an area of 95.98 km^2 ($\sim$ 100 km^2).

```
#specify seed for random number generation
dggs_pop <- dgconstruct(spacing = 11)
```

```
## Resolution: 12, Area (km^2): 95.9778454662114, Spacing (km): 9.67579208698331, CLS (km): 11.05453734
```

3.3 Filtering

The package `auk`, in combination with `tidyverse`, allows the filtering of the eBird data (see Strimas-Mackey *et al.* 2023). Note that the first function `auk_ebd()` includes the path of the eBird data downloaded and saved in your working directory, and it is the initial creation of an `auk_ebd` object. Then, different functions

serve to filter the data by the metadata of the checklists. We followed the next order in below, by protocol (`auk_protocol()`; only traveling or stationary), by distance (`auk_distance()`, $\leq$ 5 km), by duration (`auk_duration()`, $\leq$ 5 hours, note the units are minutes: 300), and only complete lists (`auk_complete()`). Then, the filters defined are converted to an AWK script with the function `auk_filter()`, generating a filtered eBird Reference Dataset (ERD), storing in the temporal files `f_ebd` with only the selected columns (defined in the pre-filtering). Finally, the function `read_ebd()` read the filtered file.

```
ebd_filt <- auk_ebd("data_raw/ebd_US-FL_snakit_smp_relSep-2024.txt") %>%
  auk_protocol(c("Traveling", "Stationary")) %>%
  auk_distance(distance = c(0,5)) %>%
  auk_duration(duration = c(0,300))%>%
  auk_complete() %>%
  auk_filter(f_ebd,overwrite=T, keep = colsE) %>%
  read_ebd()
```

```
> head(ebd_filt)
# A tibble: 6 × 19
  checklist_id common_name scientific_name      observation_count country      state_code
  <chr>        <chr>       <chr>                <chr>             <chr>        <chr>
1 G10000894    Snail Kite  Rostrhamus sociabilis 2                 United Stat… US-FL
2 G10001104    Snail Kite  Rostrhamus sociabilis 1                 United Stat… US-FL
3 G10002602    Snail Kite  Rostrhamus sociabilis 4                 United Stat… US-FL
4 G10012337    Snail Kite  Rostrhamus sociabilis 1                 United Stat… US-FL
5 G1001429     Snail Kite  Rostrhamus sociabilis 4                 United Stat… US-FL
6 G10020882    Snail Kite  Rostrhamus sociabilis 2                 United Stat… US-FL
# ℹ 13 more variables: locality_id <chr>, latitude <dbl>, longitude <dbl>,
#   observation_date <date>, time_observations_started <chr>, observer_id <chr>,
#   sampling_event_identifier <chr>, protocol_type <chr>, duration_minutes <int>,
#   effort_distance_km <dbl>, number_observers <int>, all_species_reported <lgl>,
#   group_identifier <chr>
```

Figure SI-8: head of the ebd_filt file

Then, just for the sake of double checking and organization, we can remove the observations without counts, add distance 0 to stationary protocols, modify the time of observations started to decimal, round hour sampling to an integer, extract year, month, week, and day_of_year. Also, we can confirm and filter out by effort, such as observers $\leq$ 10, distance $\leq$ 5 km, duration $\leq$ 5 hours, and only records with counts included.

```
#Some effort extraction and confirmation
ebd_filt <- ebd_filt %>%
  mutate(
    # We don't want here count in 'X', to convert to NA we use `as.integer()`
    observation_count = as.integer(observation_count),
    # effort_distance_km to 0 for non-travelling counts
    effort_distance_km = if_else(protocol_type == "Stationary",
                                 0, effort_distance_km),
    # convert time to decimal hours since midnight
    time_observations_started = time_to_decimal(time_observations_started),
    hour_sampling = round(time_observations_started, 0),
    # split date into year, month, week, and day of year
    year = year(observation_date),
    month = month(observation_date),
    week = week(observation_date),
    day_of_year = yday(observation_date)) %>%
  filter(number_observers <= 10,          #Only list with less than 10 observers
         effort_distance_km <= 5,         #be sure of distance effort
         duration_minutes %in% (0:300),   #be sure of duration effort
         !is.na(observation_count))       #only records with counts reported
```

Now we can add a new variable that identify each `cell` from a grid of hexagons (spatial sampling units), using the `longitude` and `latitude` information of our `ebd_filt` data set and the function `dgGEO_to_SEQNUM()`. With the new variable, we can extract the maximum count of individuals and number of checklists per week per cell.

```
Warning message:
There was 1 warning in `mutate()`.
ℹ In argument: `observation_count = as.integer(observation_count)`.
Caused by warning:
! NAs introduced by coercion
> head(ebd_filt)
# A tibble: 6 × 24
  checklist_id common_name scientific_name       observation_count country      state_code
  <chr>        <chr>       <chr>                 <int> <chr>        <chr>
1 G10000894    Snail Kite  Rostrhamus sociabilis     2 United Stat… US-FL
2 G10001104    Snail Kite  Rostrhamus sociabilis     1 United Stat… US-FL
3 G10002602    Snail Kite  Rostrhamus sociabilis     4 United Stat… US-FL
4 G10012337    Snail Kite  Rostrhamus sociabilis     1 United Stat… US-FL
5 G1001429     Snail Kite  Rostrhamus sociabilis     4 United Stat… US-FL
6 G10020882    Snail Kite  Rostrhamus sociabilis     2 United Stat… US-FL
# ℹ 18 more variables: locality_id <chr>, latitude <dbl>, longitude <dbl>,
#   observation_date <date>, time_observations_started <dbl>, observer_id <chr>,
#   sampling_event_identifier <chr>, protocol_type <chr>, duration_minutes <int>,
#   effort_distance_km <dbl>, number_observers <int>, all_species_reported <lgl>,
#   group_identifier <chr>, hour_sampling <dbl>, year <dbl>, month <dbl>, week <dbl>,
#   day_of_year <dbl>
```

Figure SI-9: head of the ebd_filt file - note new columns were added

```r
SnailKite <- ebd_filt %>%
  mutate(cell = dgGEO_to_SEQNUM(dggs_pop, #id for cells
                                longitude, latitude)$seqnum) %>%
  group_by(cell, year, month, week) %>%
  mutate(max_count = max(observation_count, na.rm = T),
         n_lists = n()) |>
  ungroup()
```

```
> head(SnailKite)
# A tibble: 6 × 27
  checklist_id common_name scientific_name       observation_count country      state_code
  <chr>        <chr>       <chr>                 <int> <chr>        <chr>
1 G10000894    Snail Kite  Rostrhamus sociabilis     2 United Stat… US-FL
2 G10001104    Snail Kite  Rostrhamus sociabilis     1 United Stat… US-FL
3 G10002602    Snail Kite  Rostrhamus sociabilis     4 United Stat… US-FL
4 G10012337    Snail Kite  Rostrhamus sociabilis     1 United Stat… US-FL
5 G1001429     Snail Kite  Rostrhamus sociabilis     4 United Stat… US-FL
6 G10020882    Snail Kite  Rostrhamus sociabilis     2 United Stat… US-FL
# ℹ 21 more variables: locality_id <chr>, latitude <dbl>, longitude <dbl>,
#   observation_date <date>, time_observations_started <dbl>, observer_id <chr>,
#   sampling_event_identifier <chr>, protocol_type <chr>, duration_minutes <int>,
#   effort_distance_km <dbl>, number_observers <int>, all_species_reported <lgl>,
#   group_identifier <chr>, hour_sampling <dbl>, year <dbl>, month <dbl>, week <dbl>,
#   day_of_year <dbl>, cell <dbl>, max_count <int>, n_lists <int>
```

Figure SI-10: head of the ebd_filt file - adding cell

This file is saved as a backup

```r
#and save the filter
saveRDS(SnailKite, "data_tmp/SnailKiteCellsID_filtered.rds")
```

3.4 Time series of eBird weekly high-counts

We adjusted the time-series from the 1st week of 2018 (January) to the last with data of 2024 (September). Since 2018, snail kites reached more than 1000 annualy records.

```r
SnailKite <- readRDS("data_tmp/SnailKiteCellsID_filtered.rds")

png('data_tmp/FigSI-1_HistogramTemporalBias.png',
    width = 10, height = 5, units = "in", res = 300)

hist(SnailKite$year,
     breaks = 50,
     main = "Florida Snail kites, checklists per year",
     xlab = "Year")
abline(h = 1000, v = 2017, col = "red")
dev.off()
```

```
## pdf
##   2
```

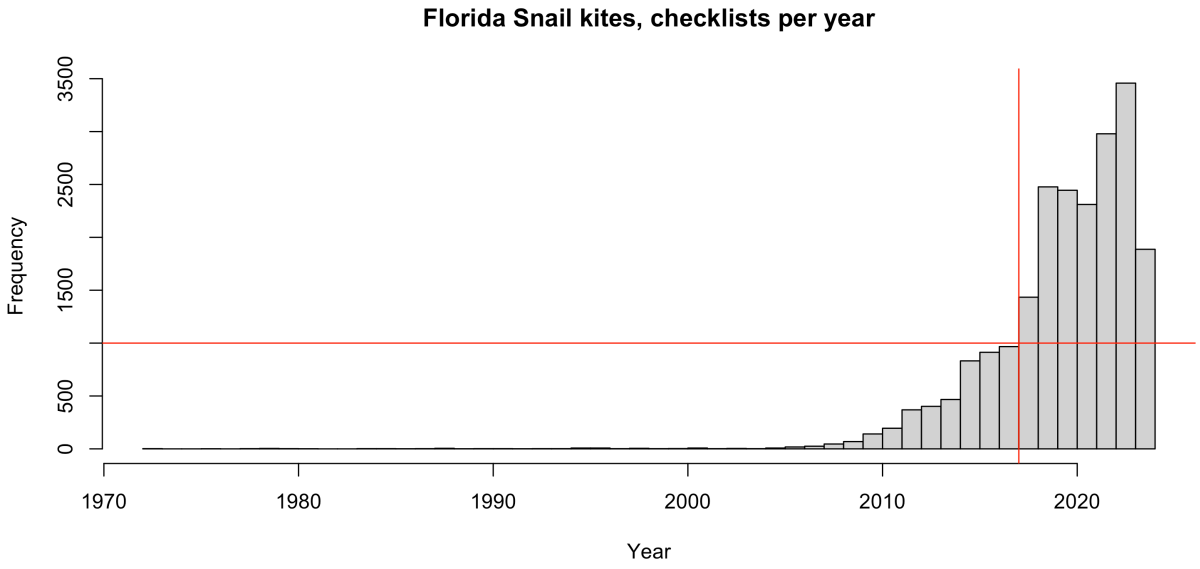


Figure SI-11: Histogram of the number of eBird checklists per year for Florida Snail kites. Red lines indicate our temporal sampling bias threshold (more than 1000 checklists per year), concentrating our sampling between January 2018 and September 2024.

In addition, the spatial cell with higher records overlaps with the Payne's Prairie State Park wetland in Alachua County, north central Florida, where snail kites established in 2018. We extracted the high count per week in the cell of Payne's Prairie to illustrate our method.

But first, we filter by observation data greater than or equal to `2018-01-01`, and generate a new variable called `Time.t`, to have the accumulated id of weeks from 2018 to the end of our time series. We used the function `case_when()` based on `year`.

```
snailkites.week <- SnailKite |>
  filter(observation_date >= "2018-01-01") |>
  mutate(Time.t = case_when(year == 2018 ~ week,
                            year > 2018 ~ week+(52*(year-2018))))
summary(snailkites.week$observation_date)
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.         Max.
## "2018-01-01" "2020-01-18" "2021-12-18" "2021-08-21" "2023-03-02" "2024-09-30"
```

```
summary(snailkites.week$Time.t)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0   107.0   207.0   189.8   269.0   352.0
```

Then, we group by the `cell` and `Time.t`, summarizing the maximum integer count per week, named `Observed.y` in our data set.

```
snailkites.week.counts <- snailkites.week |>
  group_by(cell, Time.t) |>
  summarise(Observed.y = round(max(max_count),0))
```

```
## `summarise()` has grouped output by 'cell'. You can override using the
```

```
## `.groups` argument.
```

```
head(snailkites.week.counts)
```

```
## # A tibble: 6 x 3
## # Groups:   cell [5]
##      cell Time.t Observed.y
##     <dbl>  <dbl>      <dbl>
## 1  529196     12          1
## 2  529925    340          1
## 3  529925    342          1
## 4  531384    289          1
## 5 2657878     12          1
## 6 2658604     97          1
```

And we can save this outcome as a backup.

```
saveRDS(snailkites.week, "data_tmp/SnailKiteCellsWeek.rds")
saveRDS(snailkites.week.counts, "data_tmp/SnailKiteCellsCountsWeek.rds")
```

This file will serve to generate a map figure with the sampling effort after filtering the eBird data following best practices for analysis (see Johnston *et al.*, 2021).

4 Snail Kite in Payne's Prairie from eBird

We focused on the sampling unit with higher number of checklists, which correspond to the Payne's Prairie State Park wetland system in Alachua County. For this locality, we also can access counts published from Poli *et al.* (2020), and 2 areas under current monitoring: Payne's Prairie and Payne's Prairie Central.

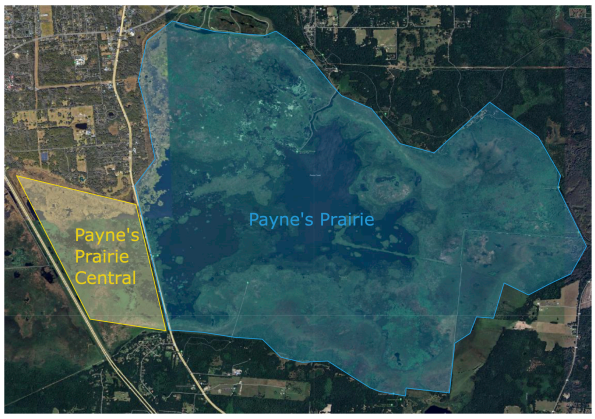


Figure SI-12: Two areas surveyed by the Snail Kite project

This wetland overlap with an hexagonal cell that concentrate most records.

4.1 Map of sampling units

To generate the map of spatiotemporal sampling from eBird, we can use the package `sf`, using the database included in the function `st_as_sf()`.

```
#A global map to make figures ###
world1 <- sf::st_as_sf(maps::map(database = 'world', plot = FALSE, fill = TRUE))
world1
```

```
Simple feature collection with 253 features and 1 field
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -180 ymin: -85.19218 xmax: 190.2708 ymax: 83.59961
Geodetic CRS:  +proj=longlat +ellps=clrk66 +no_defs +type=crs
First 10 features:
                                    ID                           geom
Aruba                             Aruba MULTIPOLYGON (((-69.89912 1...
Afghanistan                 Afghanistan MULTIPOLYGON (((74.89131 37...
Angola                           Angola MULTIPOLYGON (((23.9665 -10...
Anguilla                       Anguilla MULTIPOLYGON (((-63.00122 1...
Albania                         Albania MULTIPOLYGON (((20.06396 42...
Finland                         Finland MULTIPOLYGON (((20.61133 60...
Andorra                         Andorra MULTIPOLYGON (((1.706055 42...
United Arab Emirates United Arab Emirates MULTIPOLYGON (((53.92783 24...
Argentina                     Argentina MULTIPOLYGON (((-64.54916 -...
Armenia                         Armenia MULTIPOLYGON (((45.55235 40...
```

Figure SI-13: world1

We can summarize the filtered ebd-data set `SnailKite` by the number of observations in each hexagonal cell. Below code will generate a tibble with two variables and 444 observations, the count of observations per each cell.

```
#Get the number of observations in each cell
CellObservationsSK   <- SnailKite %>%
  group_by(cell) %>%
  summarise(count=n())
```

And get a grid cell boundaries for cells with the observations counts with the function `dgcellstogrid()`, using the discrete global grid system saved as `dggs_pop`. **CAUTION**, if you generate `dggs_pop` in a different

```
> head(CellObservationsSK)
# A tibble: 6 × 2
     cell count
    <dbl> <int>
1  528466    12
2  529196     1
3  529925     8
4  531384     1
5 2657877     1
6 2657878     1
```

Figure SI-14: Head of Snail Kite observations per cell

computer or session than `SnailKite`, there could be conflict with the `cell` id.

```
gridSnailKite <- dgcellstogrid(dggs_pop,CellObservationsSK$cell)
```

This is an `sf` object with the `cell` named as `seqnum`. We can update the grid cells' properties to include the number of lists in each cell and handling the spatial data with `st_wrap_dateline()`

```
gridSnailKite <- merge(gridSnailKite, CellObservationsSK, by.x="seqnum", by.y="cell")

# Handle cells that cross 180 degrees
wrapped_gridSnailKite = st_wrap_dateline(gridSnailKite,
                                         options = c("WRAPDATELINE=YES",
                                                     "DATELINEOFFSET=180"),
                                         quiet = TRUE)
```

```
> head(wrapped_gridSnailKite)
Simple feature collection with 6 features and 3 fields
Geometry type: POLYGON
Dimension:     XY
Bounding box:  xmin: -84.39284 ymin: 30.01819 xmax: -84.03095 ymax: 30.57365
Geodetic CRS:  WGS 84
  seqnum count.x count.y                       geometry
1 528466      12      12 POLYGON ((-84.39284 30.5301...
2 529196       1       1 POLYGON ((-84.30695 30.4599...
3 529925       8       8 POLYGON ((-84.31262 30.3596...
4 531384       1       1 POLYGON ((-84.23285 30.1893...
5 2657877      1       1 POLYGON ((-84.23882 30.0891...
6 2657878      1       1 POLYGON ((-84.14742 30.1192...
```

Figure SI-15: Head of wrapped grid for Snail kites in Florida

Some aesthetics are defined for the log-scales and arrows

```
my_breaks = c(5, 50, 500, 5000)

arrow1 <- tibble(
  x1 = -82.3,
  x2 = -80.75,
  y1 = 29.6,
  y2 = 29.7
)
```

and the figure is generated with `ggplot()`

```
Fig2a <- ggplot() +
  geom_sf(data = world1)+
  geom_sf(data=wrapped_gridSnailKite,
          aes(color = count,
              fill = count),
          alpha = 0.7) +
  #  geom_point(data = SnailKite, aes(x = longitude, y = latitude), size = 0.1)+
  scale_color_gradient(low="#440154",
                       high="#FDE725",
                       trans = "log10",
                       breaks = my_breaks,
```

```
                       labels = my_breaks)+
  scale_fill_gradient(low="#440154",
                      high="#FDE725",
                      trans = "log10",
                      breaks = my_breaks,
                      labels = my_breaks)+
  coord_sf(xlim = c(-84.5, -79.5),
           ylim =  c(24.1, 30.9)) +
  labs(y = "Latitude",
       x = "Longitude",
       tag = expression(bold("(a)")),
       title = "Snail kites in Florida",
       subtitle = "eBird effort in spatial sampling units",
       color = expression(Log["10"]~"lists"),
       fill = expression(Log["10"]~"lists")) +
  annotate("text", x = -80, y = 29.75, label = "Payne's \n Prairie") +
  geom_curve(data = arrow1, aes(x = x1, y = y1, xend = x2, yend = y2),
             arrow = arrow(length = unit(0.08, "inch")), size = 0.5,
             color = "red", curvature = -0.3) +
  theme_classic()+
  theme(legend.position = c(0.2,0.25),
        legend.direction = "vertical",
        legend.box.background = element_rect(colour = "black"))

#Zoom to Payne's Prairie
arrow2 <- tibble(
  x1 = c(-82.328576, -82.334182, -82.303112, -82.292372),
  x2 = c(-82.3, -82.375, -82.25, -82.2),
  y1 = c(29.619306, 29.574222, 29.606876, 29.549109),
  y2 = c(29.7, 29.475, 29.65, 29.535)
)

Fig2b <- ggplot() +
  geom_sf(data=wrapped_gridSnailKite,
          aes(color = count,
              fill = count)) +
  geom_point(data = SnailKite, aes(x = longitude, y = latitude),
             size = 0.5, alpha = 0.25)+
  scale_color_gradient(low="#440154",
                       high="#FDE725",
                       trans = "log10",
                       breaks = my_breaks,
                       labels = my_breaks)+
  scale_fill_gradient(low = alpha("#440154", 0.25),
                      high = alpha("#FDE725", 0.25),
                      trans = "log10",
                      breaks = my_breaks,
                      labels = my_breaks)+
  coord_sf(xlim = c(-82.475, -82.125),
           ylim =  c(29.45, 29.725),
           expand = T) +
  labs(y = "Latitude",
       x = "Longitude",
```

```
       tag = expression(bold("(b)")),
       title = "Snail kites in Payne's Prairie wetland",
       subtitle = "with eBird records and popular localities") +
  annotate("text", x = -82.25, y = 29.71, label = "Sweetwater Wetlands \n Park") +
  annotate("text", x = -82.4, y = 29.475, label = "US-441") +
  annotate("text", x = -82.2, y = 29.65, label = "La Chua trail") +
  annotate("text", x = -82.2, y = 29.525, label = "Wacahoota trail") +
  geom_curve(data = arrow2, aes(x = x1, y = y1, xend = x2, yend = y2),
             arrow = arrow(length = unit(0.08, "inch")), size = 0.5,
             color = "red", curvature = -0.3) +
  theme_classic()+
  theme(legend.position = "none")

Fig2 <- grid.arrange(Fig2a, Fig2b, ncol = 2, widths = c(1, 2))

ggsave("results/Fig2_SnailKitesMap.pdf",
       plot = Fig2, dpi = 300, width = 10, height = 5, units = "in")

ggsave("results/Fig2_SnailKitesMap.png",
       plot = Fig2, dpi = 300, width = 10, height = 5, units = "in")
```

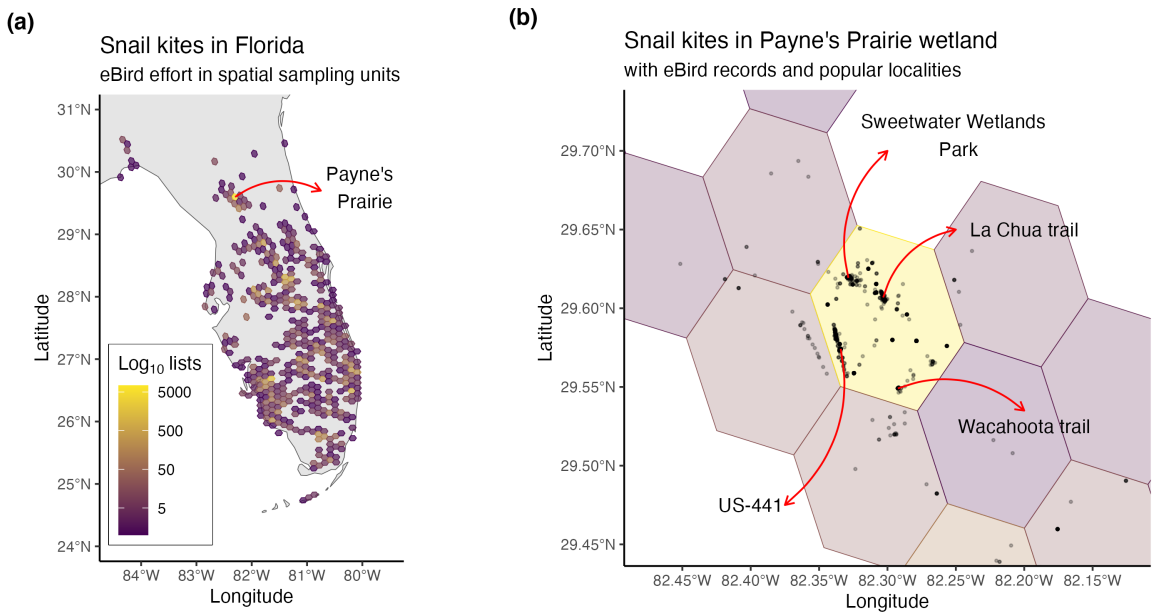


Figure SI-16: *Figure 2 in Main text.* Map of spatial eBird sampling of Snail kites in Florida (a) and zoom to the hexagonal cell with higher records in eBird (b)

4.2 Standardized monitoring - a benchmark to compare eBird

Joining eBird and standardized monitoring (our benchmark) in data set `snailkites.PP`.

First, load saved data from eBird, identifying the cell with more checklists. Recall that different packages in `R` might include same name of functions, that could generate conflict for replicability (e.g., `select()` in packages `MASS` and `dplyr`). To avoid confusion, users can add the name of the package (e.g., `dplyr::select()`)

```r
#load saved data
SnailKite <- readRDS("data_tmp/SnailKiteCellsID_filtered.rds")
snailkites.week <- readRDS("data_tmp/SnailKiteCellsWeek.rds")
snailkites.week.counts <- readRDS("data_tmp/SnailKiteCellsCountsWeek.rds")

#Cell with more values
CellTop <- SnailKite |>
  group_by(cell) |>
  mutate(n_checklists = n()) |>
  ungroup() |>
  filter(n_checklists == max(n_checklists)) |>
  dplyr::select(cell) |>
  unique()
CellTop #ID of the cell with more records - Payne's Prairie
```

```
## # A tibble: 1 x 1
##      cell
##     <dbl>
## 1 2665918
```

Call the data from Poli *et al.* (2020), and organize in the same way that eBird data (the highest value per week since January 2018).

```r
#The monitoring data of Poli et al. (2020)
Poli.etal <- data.frame(observation.date = c("2018-02-19",
                                             "2018-03-12",
                                             "2018-04-09",
                                             "2018-05-31",
                                             "2018-06-04",
                                             "2018-07-16",
                                             "2018-08-07",
                                             "2018-08-27",
                                             "2018-10-15",
                                             "2018-12-17"),
                        abundance.monitored = c(1,
                                                2,
                                                4,
                                                6,
                                                8,
                                                6,
                                                6,
                                                12,
                                                7,
                                                29))

Poli.etal <- Poli.etal |>
  mutate(observation.date = ymd(observation.date),
         year = year(observation.date),
         week = week(observation.date),
         Time.t = case_when(year == 2018 ~ week,
                            year > 2018 ~ week+(52*(year-2018)))) |>
  dplyr::select(!observation.date)
```

Call the data form standardized monitoring in the project Snail Kites (since 2019). This data is also organized by the high count per week, as the eBird data.

```
snail.kite.project.pp <- read_csv("data_raw/Snail kite surveys on PP and PPC 2018_2024.csv") |>
  mutate(observation.date = mdy(date),
         year = year(observation.date),
         week = week(observation.date),
         Time.t = case_when(year == 2018 ~ week,
                            year > 2018 ~ week+(52*(year-2018)))) |>
  group_by(Time.t, year, week) |>
  summarise(abundance.monitored = max(count))
```

```
## Rows: 53 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): location, date, survey_num
## dbl (1): count
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## `summarise()` has grouped output by 'Time.t', 'year'. You can override using the `.groups` argument.
```

Combine with the published data to have a single Standardized Monitoring abundance data set, named `snail.kite.project.pp`. Here is important to be sure that the data set is organized by the variable `Time.t` (week since `2018-01-01` id in our case), applying the function `arrange()`.

```
#including Poli et al.
snail.kite.project.pp <- snail.kite.project.pp |>
  full_join(Poli.etal) |>
  arrange(Time.t)
```

```
## Joining with `by = join_by(Time.t, year, week, abundance.monitored)`
```

Now we can unify the data sets in a single object. First, we filter the `snailkites.week.counts` object by the id of the cell that contains Payne's Prairie (`CellTop$cell`). To avoid conflict with gaps in the time series for `observation.date`, we generated an object of dates from the minimum value of `snailkites.week$observation_date`, and join it to the subset of Snail kites high count per week in Payne's Praire (`snailkites.paynesp`).

```
#Generate the time-series for the cell with more records
snailkites.paynesp <- snailkites.week.counts |>
  filter(cell == CellTop$cell)
head(snailkites.paynesp)
```

```
## # A tibble: 6 x 3
## # Groups:   cell [1]
##      cell Time.t Observed.y
##     <dbl>  <dbl>      <dbl>
## 1 2665918      8          1
## 2 2665918      9          1
## 3 2665918     10          1
## 4 2665918     11          1
## 5 2665918     12          1
## 6 2665918     13          1
```

```
#To return dates from original data
datesPP <- snailkites.week |>
  group_by(Time.t) |>
  summarise(observation.date = min(observation_date))
head(datesPP)
```

```
## # A tibble: 6 x 2
##   Time.t observation.date
##    <dbl> <date>
## 1      1 2018-01-01
## 2      2 2018-01-08
## 3      3 2018-01-15
## 4      4 2018-01-22
## 5      5 2018-01-29
## 6      6 2018-02-05
```

```
snailkites.paynesp <- snailkites.paynesp |>
  left_join(datesPP)
```

```
## Joining with `by = join_by(Time.t)`
```

```
head(snailkites.paynesp)
```

```
## # A tibble: 6 x 4
## # Groups:   cell [1]
##      cell Time.t Observed.y observation.date
##     <dbl>  <dbl>      <dbl> <date>
## 1 2665918      8          1 2018-02-19
## 2 2665918      9          1 2018-02-26
## 3 2665918     10          1 2018-03-05
## 4 2665918     11          1 2018-03-12
## 5 2665918     12          1 2018-03-20
## 6 2665918     13          1 2018-03-26
```

Finally, we add the standardized monitoring (sorted by week id `Time.t`). **CAUTION** if you do not sort by time, some functions will crash and not run correctly, also make sure that you have only one value per time-step selected (in our case, weeks).

```
#Add standardized monitored
snailkites.PP <- snailkites.paynesp |>
  left_join(snail.kite.project.pp, by = "Time.t") |>
  arrange(Time.t)
snailkites.PP
```

```
## # A tibble: 340 x 7
## # Groups:   cell [1]
##       cell Time.t Observed.y observation.date  year  week abundance.monitored
##      <dbl>  <dbl>      <dbl> <date>           <dbl> <dbl>               <dbl>
##  1 2665918      8          1 2018-02-19        2018     8                   1
##  2 2665918      9          1 2018-02-26          NA    NA                  NA
##  3 2665918     10          1 2018-03-05          NA    NA                  NA
##  4 2665918     11          1 2018-03-12        2018    11                   2
##  5 2665918     12          1 2018-03-20          NA    NA                  NA
##  6 2665918     13          1 2018-03-26          NA    NA                  NA
##  7 2665918     14          4 2018-04-03          NA    NA                  NA
##  8 2665918     15          4 2018-04-09        2018    15                   4
##  9 2665918     16          2 2018-04-16          NA    NA                  NA
## 10 2665918     17          2 2018-04-23          NA    NA                  NA
## # i 330 more rows
```

Note that the `abundance.monitored` variable, from the standardized monitored surveys, have many `NA` values when compared with the `Observed.y`, which is the weekly high-counts in eBird.

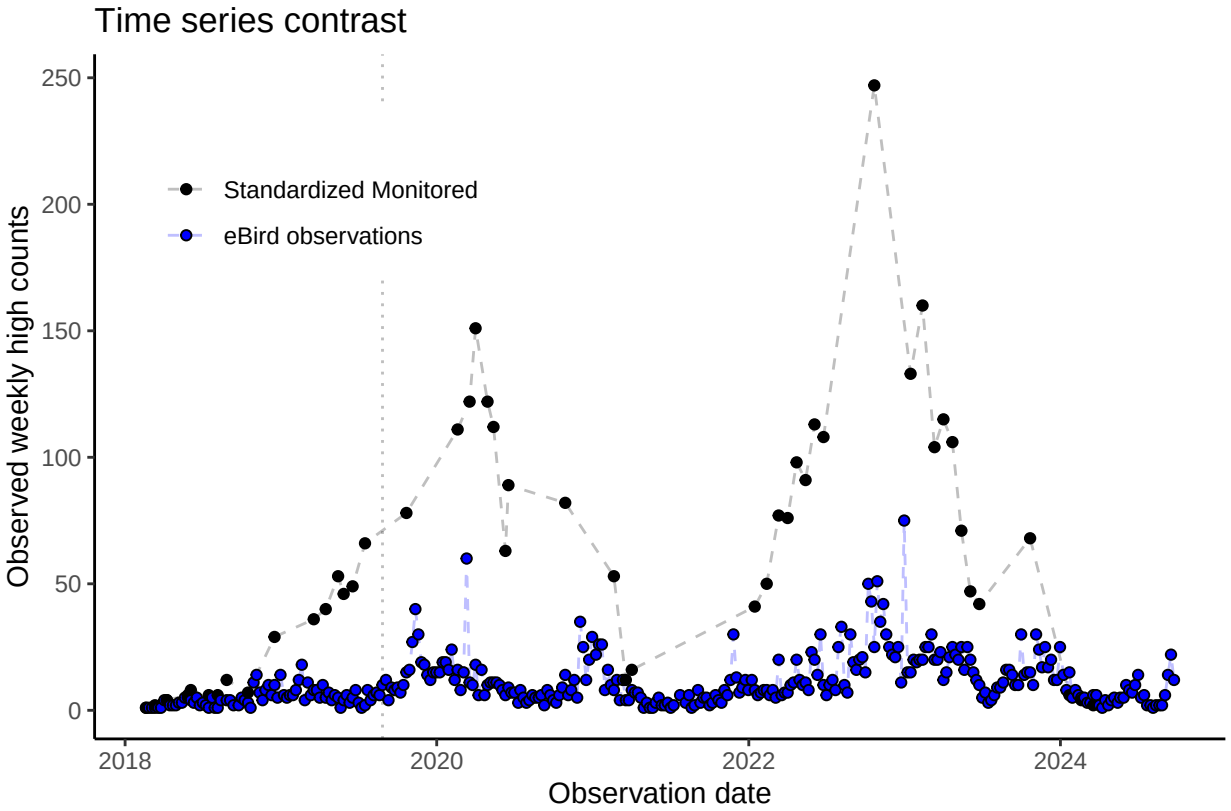
We can save the backup.

```r
#save backup
saveRDS(snailkites.PP, file = "data_tmp/snailkitesPP.rds")
saveRDS(datesPP, file = "data_tmp/datesTimeseriesPaynesPrairie.rds")
```

And we can see both time-series graphically

```r
datesPP <- readRDS("data_tmp/datesTimeseriesPaynesPrairie.rds")
snailkites.PP <- readRDS("data_tmp/snailkitesPP.rds")

#Figure
snailkites.PP |>
  pivot_longer(cols = !c(Time.t, cell, year, week, observation.date),
               names_to = "group",
               values_to = "Abundance") |>
  drop_na(Abundance) |>
  ggplot(aes(x = observation.date, y = Abundance, fill = group))+
    geom_vline(xintercept = snailkites.PP$observation.date[78],
             color = "gray", linetype = "dotted") +
    geom_line(aes(color = group), alpha = 0.25, linetype = "dashed") +
    geom_point(aes(shape = group), color = "black")+
    labs(x = "Observation date",
       y = "Observed weekly high counts",
       title = "Time series contrast",
       tag = "",
       fill = "",
       color = "",
       shape = "")+
    scale_color_manual(values = c("black", "blue"),
                       labels = c("Standardized Monitored",
                                  "eBird observations"))+
    scale_fill_manual(values = c("black", "blue"),
                       labels = c("Standardized Monitored",
                                  "eBird observations"))+
    scale_shape_manual(values = c(19, 21),
                       labels = c("Standardized Monitored",
                                  "eBird observations"))+
    theme_classic()+
    theme(legend.position = c(0.2, 0.8))
```

```
## Warning: A numeric `legend.position` argument in `theme()` was deprecated in ggplot2
## 3.5.0.
## i Please use the `legend.position.inside` argument of `theme()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Assumed initial population establishment is depicted to the left of the vertical dotted gray line (weeks 1-87;January 2018-August 2019). Local persistence probability estimation ($\hat{\phi}$) will correspond to the weeks to the right of the dotted gray line ($n_{eBird} = 262$, $n_{SM} = 33$).

5 Viability Population Monitoring framework and local persistence estimation - $\hat{\phi}$

In the next section, we provide a step by step example and the iterated process for each week estimation of local persistence probability ($\hat{\phi}$).

5.1 Fit population dynamics model to a first part of the time-series

Let's fit an EGSS model for the first two years – weeks from 1 to 87, n = 78, matching Standardized monitoring data, including the published data (Poli *et al.*, 2020).

First, load the data saved and filter `Time.t` between `1:87`.

```
datesPP <- readRDS("data_tmp/datesTimeseriesPaynesPrairie.rds")
snailkites.PP <- readRDS("data_tmp/snailkitesPP.rds")

sk.pp.init <- snailkites.PP |>
  filter(Time.t %in% c(1:87))
```

To fit a first EGSS model for the eBird data observations, we have to adjust the data as required by the functions (vectors of log-abundance and time-steps, in order and with a single value of log-abundance).

```
#Define variables
yt1 = sk.pp.init |>
  ungroup() |>
  arrange(Time.t) |>
  drop_na(Observed.y) |>
  dplyr::select(Observed.y) |>
  log()

#log-abundance estimate as a vector
yt1 <- yt1$Observed.y
yt1
```

```
##  [1] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 1.3862944
##  [8] 1.3862944 0.6931472 0.6931472 0.6931472 1.0986123 1.0986123 1.6094379
## [15] 1.6094379 1.3862944 1.0986123 1.6094379 0.6931472 1.0986123 0.6931472
## [22] 0.0000000 1.6094379 0.0000000 0.0000000 1.3862944 1.3862944 1.3862944
## [29] 0.6931472 0.6931472 1.6094379 1.3862944 1.0986123 0.0000000 2.3978953
## [36] 2.6390573 1.9459101 1.3862944 2.0794415 2.3025851 1.7917595 2.3025851
## [43] 1.6094379 2.6390573 1.7917595 1.6094379 1.7917595 1.7917595 2.0794415
## [50] 2.4849066 2.8903718 1.3862944 2.3978953 1.7917595 2.0794415 2.0794415
## [57] 1.6094379 2.3025851 1.6094379 1.9459101 1.3862944 1.7917595 1.6094379
## [64] 0.0000000 1.3862944 1.7917595 1.0986123 1.6094379 2.0794415 1.0986123
## [71] 0.0000000 0.6931472 2.0794415 1.3862944 1.7917595 1.9459101 1.7917595
## [78] 2.3025851
```

```
tt1 <- sk.pp.init |>
  ungroup() |>
  arrange(Time.t) |>
  drop_na(Observed.y) |>
  dplyr::select(Time.t)

#time vector (week since 2018-01)
tt1 <- tt1$Time.t
tt1
```

```
##  [1]  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
## [26] 33 35 36 37 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59
## [51] 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84
## [76] 85 86 87
```

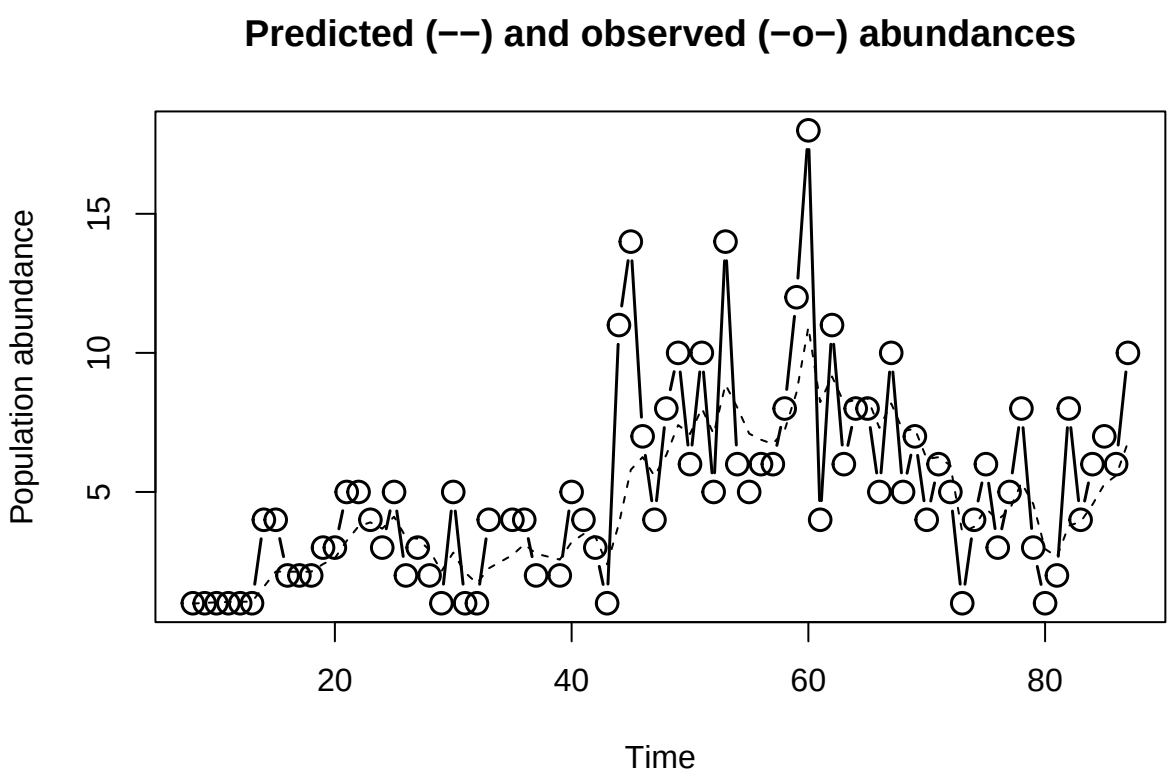
```
#Estimate REML parameters
sk.egss.parms <- egss_reml(yt = yt1,
                           tt = tt1,
                           fguess = guess_egss(yt = yt1,
                                               tt = tt1))
print(sk.egss.parms)
```

```
## $remles
## [1]  0.02713171  0.03440026  0.26582804 -0.01276513
##
## $lnL.hat
## [1] -73.54665
##
## $AIC
## [1] 151.0933
```

The function `egss_reml()` fit an EGSS model and compute the Restricted Maximum Likelihood Estimates, which are stored in the first object of the list (named here `sk.egss.parms`). The values correspond (in order) to the trend parameter ($\hat{\theta}_{eBird} = 0.0271$), the environmental noise ($\widehat{\sigma^2}_{eBird} = 0.0344$), observation error noise ($\widehat{\tau^2}_{eBird} = 0.2658$), and initial population ($\hat{x_0}_{eBird} = -0.0128$; note that $e^{x_0} \approx 1$).

With the `egss_predict()` function, we can predict the trajectory for the EGSS model.

```
sk.egss.predict.init <- egss_predict(yt = yt1,
                                     tt = tt1,
                                     parms = sk.egss.parms$remles,
                                     plot.it = TRUE)
```




```
head(sk.egss.predict.init[[1]])
```

```
##      Time Predict.EGSS.REML Observed.y
## [1,]    8          0.987316          1
## [2,]    9          1.012802          1
## [3,]   10          1.032554          1
## [4,]   11          1.045648          1
## [5,]   12          1.053581          1
## [6,]   13          1.058207          1
```

As we put the `plot.it` argument as `TRUE`, we have a figure of the predicted and observed trajectory. The `head()` of the object shows the vector time (`Time.t`, week since January 2018, in our case), the predicted abundance (`Predict.EGSS.REML`), and the observed vector (`Observed.y`, the eBird weekly high counts in our case). For our example, we want to compare with standardized monitored surveys, so it might be convenient to change the names of this output.

```
#change names to combine
colnames(sk.egss.predict.init[[1]]) <- c("Time.t", "Estimated_eBird_EGSS","eBird.Observed")
```

We can fit the EGSS for the standardized monitored in the same way

```
#Define variables
ytSM1 = sk.pp.init |>
  ungroup() |>
  arrange(Time.t) |>
  drop_na(abundance.monitored) |>
  dplyr::select(abundance.monitored) |>
  log()
```

```
#Standardized abundance monitored as a vector
ytSM1 <- ytSM1$abundance.monitored
ytSM1
```

```
##  [1] 0.0000000 0.6931472 1.3862944 1.7917595 2.0794415 1.7917595 1.7917595
##  [8] 2.4849066 1.9459101 3.3672958 3.5835189 3.6888795 3.9702919 3.8286414
## [15] 3.8918203 4.1896547
```

```
ttSM1 <- sk.pp.init |>
  ungroup() |>
  arrange(Time.t) |>
  drop_na(abundance.monitored) |>
  dplyr::select(Time.t)

#corresponding time vector
ttSM1 <- ttSM1$Time.t
ttSM1
```

```
##  [1]  8 11 15 22 23 29 32 35 42 51 64 68 72 74 77 81
```

```
skSM.egss.parms <- egss_reml(yt = ytSM1,
                             tt = ttSM1,
                             fguess = guess_egss(yt = ytSM1,
                                                 tt = ttSM1))
skSM.egss.parms
```

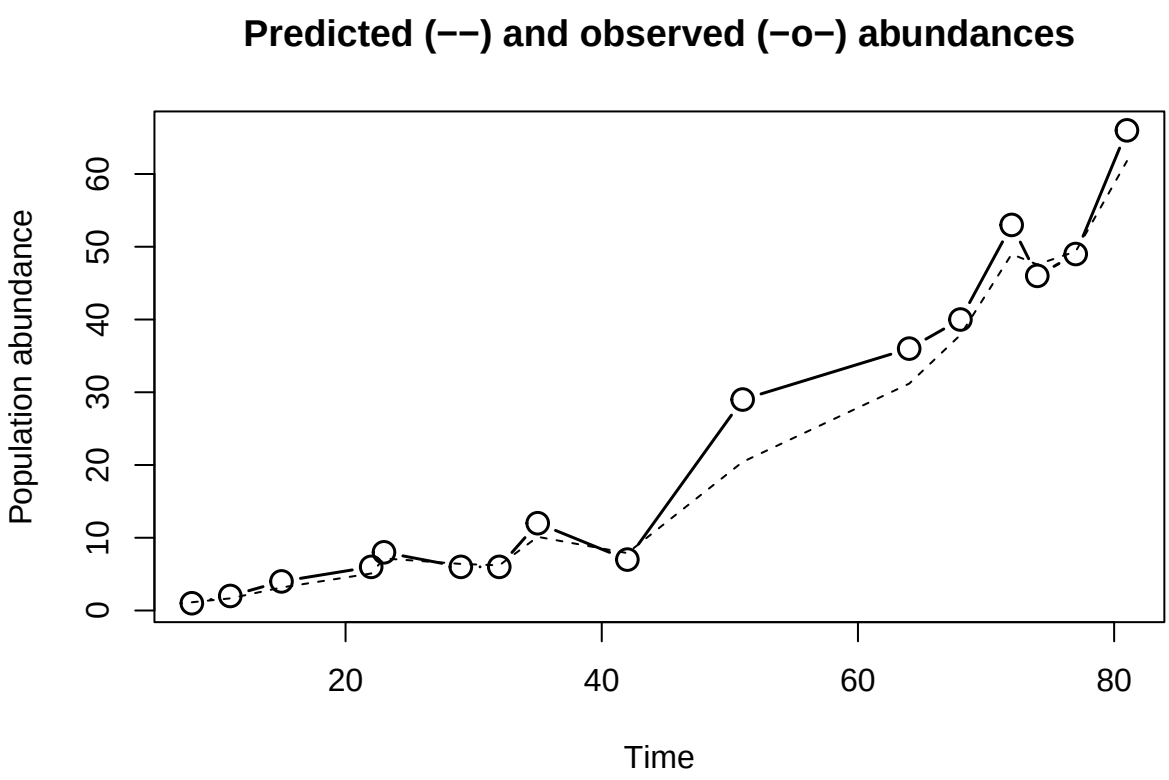
```
## $remles
## [1] 0.05557391 0.03455566 0.01878917 0.12583555
##
## $lnL.hat
## [1] 10.17092
##
## $AIC
## [1] -16.34184
```

Again, the values correspond (in order) to the trend parameter ($\hat{\theta}_{SM} = 0.0556$), the environmental noise ($\hat{\sigma}^2_{SM} = 0.0346$), observation error noise ($\hat{\tau}^2_{SM} = 0.0188$), and initial population ($\hat{x_0} = 0.1258$).

It is interesting that $\hat{\sigma}^2_{SM}$ is very similar (~ 0.03) than $\hat{\sigma}^2_{eBird}$, as well as $e^{x_0=0.126}_{SM} \approx 1$ and $e^{x_0=-0.013}_{eBird} \approx 1$, but the observation error is over an order of magnitude bigger in eBird data with respect to the standardized monitoring ($\hat{\tau}^2_{SM} = 0.018 << \hat{\tau}^2_{eBird} = 0.266$). The expected change in log-abundance per week describe lower change in eBird ($\hat{\theta}_{eBird} = 0.034$) and higher changes in standardized monitoring weekly counts ($\hat{\theta}_{SM} = 0.056$).

Let's predict the trajectory and change the column names for the figure.

```
skSM.egss.predict.init <- egss_predict(yt = ytSM1,
                                       tt = ttSM1,
                                       parms = skSM.egss.parms$remles,
                                       plot.it = T)
```




```
#change names to combine
colnames(skSM.egss.predict.init[[1]]) <- c("Time.t", "Estimated_SKProj_EGSS","abundance.monitored")
```

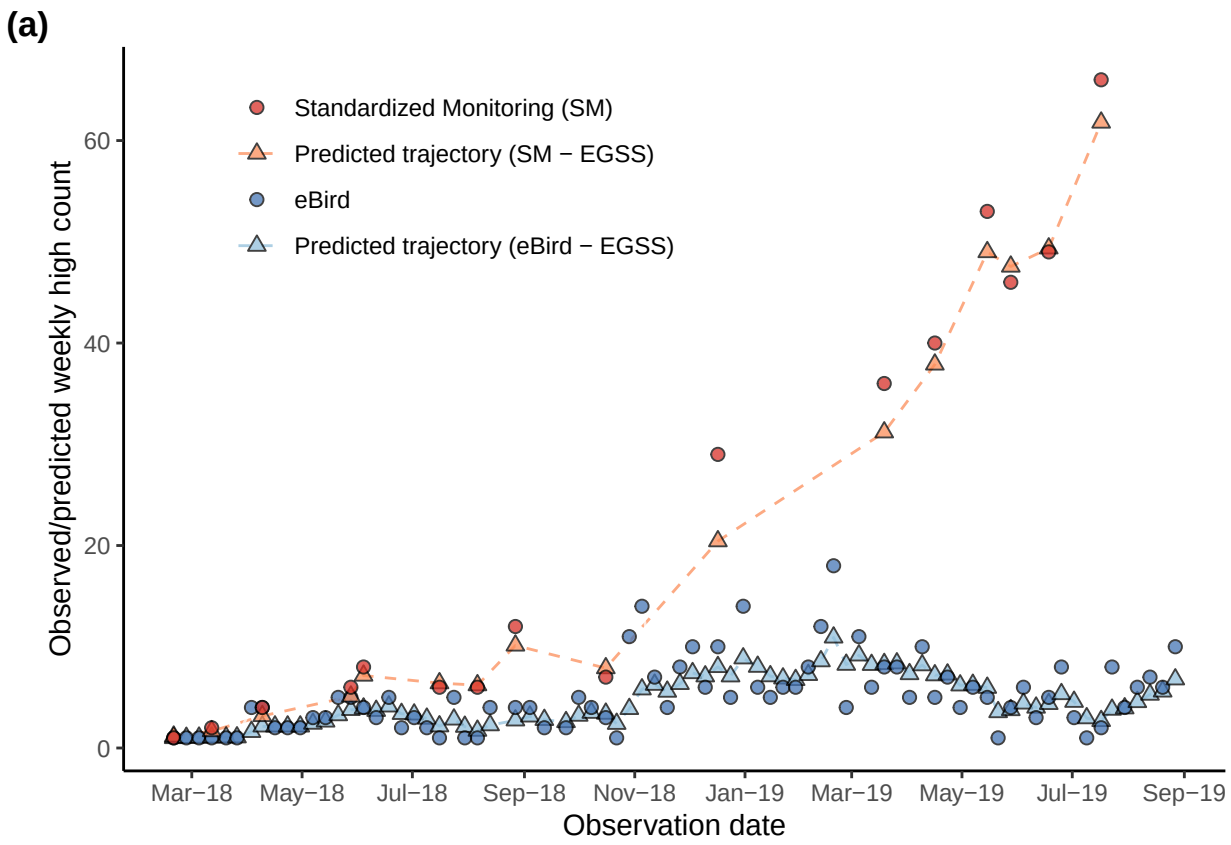
To make the figure 3 of the main text, we combine the predicted trajectories in a single data frame to use `tidyverse` and `ggplot`.

```
sk.Estimated <- data.frame(sk.egss.predict.init) |>
  left_join(data.frame(skSM.egss.predict.init)) |>
  left_join(datesPP) #this recover the observation date per week
```

```
## Joining with `by = join_by(Time.t)`
## Joining with `by = join_by(Time.t)`
```

```
FigS3a <- sk.Estimated |>
  pivot_longer(cols = !c(Time.t,observation.date),
               names_to = "group",
               values_to = "Abundance") |>
  drop_na(Abundance) |>
  ggplot(aes(x = observation.date,
             y = Abundance,
             fill = factor(group,
                           levels = c("abundance.monitored",
                                      "Estimated_SKProj_EGSS",
                                      "eBird.Observed",
                                      "Estimated_eBird_EGSS"))))+
    geom_line(aes(color = factor(group,
                           levels = c("abundance.monitored",
                                      "Estimated_SKProj_EGSS",
```

```
                                    "eBird.Observed",
                                    "Estimated_eBird_EGSS")),
                  linetype = factor(group,
                                    levels = c("abundance.monitored",
                                    "Estimated_SKProj_EGSS",
                                    "eBird.Observed",
                                    "Estimated_eBird_EGSS"))),
              alpha = 0.75) +
    geom_point(aes(shape = factor(group,
                                    levels = c("abundance.monitored",
                                    "Estimated_SKProj_EGSS",
                                    "eBird.Observed",
                                    "Estimated_eBird_EGSS"))),
               color = "black",
               alpha = 0.75, size = 2)+
    labs(x = "Observation date",
       y = "Observed/predicted weekly high count",
       tag = expression(bold("(a)")),
       fill = "",
       color = "",
       shape = "",
       linetype = "")+
  scale_x_date(breaks = seq(as.Date("2018-01-01"), as.Date(Sys.Date( )),
                            by = "2 months"), date_labels="%b-%y")+
    scale_color_manual(values = c("white", "#fc8d59","white", "#91bfdb"),
                       labels = c("Standardized Monitoring (SM)",
                                  "Predicted trajectory (SM - EGSS)",
                                  "eBird",
                                  "Predicted trajectory (eBird - EGSS)"))+
    scale_fill_manual(values = c("#d73027", "#fc8d59", "#4575b4", "#91bfdb"),
                       labels = c("Standardized Monitoring (SM)",
                                  "Predicted trajectory (SM - EGSS)",
                                  "eBird",
                                  "Predicted trajectory (eBird - EGSS)"))+
    scale_shape_manual(values = c(21, 24, 21, 24),
                       labels = c("Standardized Monitoring (SM)",
                                  "Predicted trajectory (SM - EGSS)",
                                  "eBird",
                                  "Predicted trajectory (eBird - EGSS)"))+
  scale_linetype_manual(values = c("solid","dashed","solid","dashed"),
                        labels = c("Standardized Monitoring (SM)",
                                   "Predicted trajectory (SM - EGSS)",
                                   "eBird",
                                   "Predicted trajectory (eBird - EGSS)"))+
    theme_classic()+
    theme(legend.position = c(0.3, 0.85))
FigS3a
```



How much is the prediction out of the observed data in SM? Lets generate Figure 3b (we can draw the equation in the figure with our function `lm_eqn()`, removed for now).

```
#Predicted EGSS from eBird ~ Observed counts SM
lm(Estimated_eBird_EGSS~abundance.monitored, data = sk.Estimated)
```

```
##
## Call:
## lm(formula = Estimated_eBird_EGSS ~ abundance.monitored, data = sk.Estimated)
##
## Coefficients:
##         (Intercept)  abundance.monitored
##             2.58275              0.05593
```

```
#Predicted EGSS from Standardized Monitored ~ Observed counts SM
lm(Estimated_SKProj_EGSS~abundance.monitored, data = sk.Estimated)
```

```
##
## Call:
## lm(formula = Estimated_SKProj_EGSS ~ abundance.monitored, data = sk.Estimated)
##
## Coefficients:
##         (Intercept)  abundance.monitored
##             -0.4086               0.9503
```

```
FigS3b <- ggplot(sk.Estimated)+
  geom_abline(slope = 1)+
  geom_smooth(aes(x=abundance.monitored, y=Estimated_SKProj_EGSS),
```

```
              method = "lm", color = "#fc8d59",
              se = T, fill = "#fc8d5905", linetype = "dashed")+
  geom_point(aes(x=abundance.monitored, y=Estimated_SKProj_EGSS),
             color = "black", fill = "#fc8d59", shape = 24, alpha = 0.75, size = 2)+
# geom_text(x = 25, y = 50,
#             label = lm_eqn(df = sk.Estimated,
#                            x = sk.Estimated$abundance.monitored,
#                            y = sk.Estimated$Estimated_SKProj_EGSS),
#             parse = TRUE, color = "#fc8d59")+
  geom_smooth(aes(x=abundance.monitored, y=Estimated_eBird_EGSS),
              method = "lm", color = "#91bfdb",
              se = T, fill = "#91bfdb05", linetype = "dashed")+
  geom_point(aes(x=abundance.monitored, y=Estimated_eBird_EGSS),
             color = "black", fill = "#91bfdb", shape = 24, alpha = 0.75, size = 2)+
#    geom_text(x = 50, y = 0,
#             label = lm_eqn(df = sk.Estimated,
#                            x = sk.Estimated$abundance.monitored,
#                            y = sk.Estimated$Estimated_eBird_EGSS),
#             parse = TRUE, color = "#91bfdb")+
  scale_y_continuous(limits = c(0,80))+
  labs(x = "Observed weekly high count (SM)",
       y = "Predicted trajectory of weekly high count",
       tag = expression(bold("(b)")))+
  coord_fixed()+
  theme_classic()

#and combine in the figure
FigS3 <- grid.arrange(FigS3a, FigS3b, ncol = 2, widths = c(1.5, 1))
```

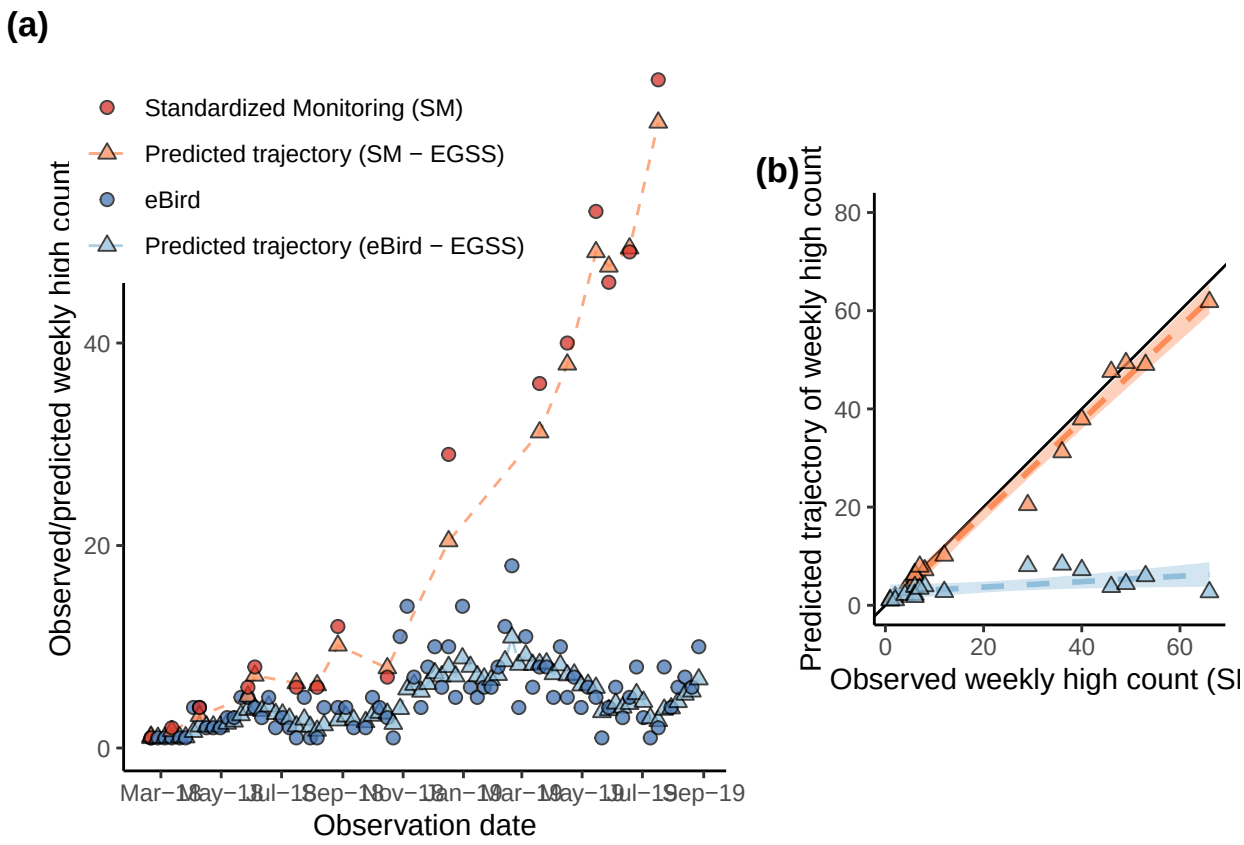
```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning: Removed 62 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning: Removed 62 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 62 rows containing missing values or values outside the scale range
## (`geom_point()`).
## Removed 62 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



It looks not good in the `Rmd` file, but it is saved with good proportions

```
ggsave("results/Figure3ab_EstimatedTrajectory2initdays.pdf",
       plot = FigS3, dpi = 300, width = 10, height = 4, units = "in")

ggsave("results/Figure3ab_EstimatedTrajectory2initdays.png",
       plot = FigS3, dpi = 300, width = 10, height = 4, units = "in")
```

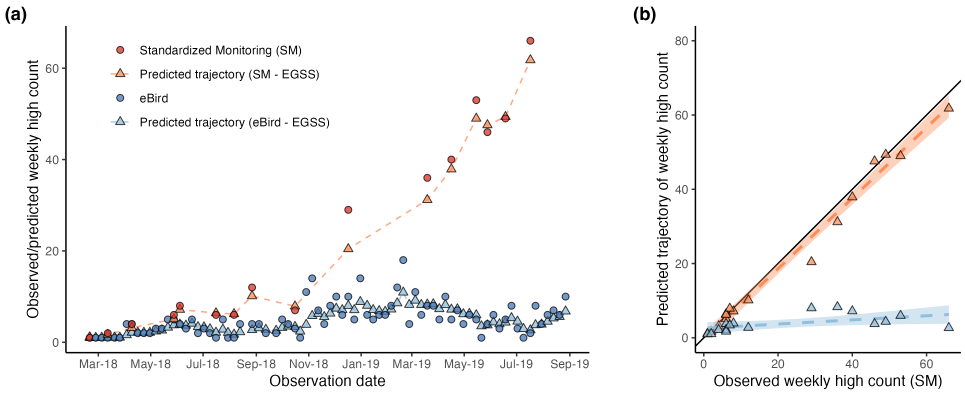


Figure SI-17: *Figure 2 in Main text*Weekly predicted (triangles) and observed (circles) high counts of the number of Snail Kites from standardized monitoring (orange) and eBird (blue).

5.2 Three timeframe (~three, ~five, ~ten years) of first estimation $\hat{\phi}$

With the estimated model parameters and the data during establishment of the population (up to week 87 with observations, $i = 1$ is week 88), we estimated the probability that the population declines to values below a critical threshold ($N_{critical}$, as the critical number of individuals to assume quasi-extinction), in three different periods, given timeframes, or moving simulation windows (~three years 150 weeks, ~five years 250 weeks, and ~ten years 500 weeks).

Let assume $N_{critical\ eBird} = \frac{1}{2}(e^{\bar{y}}) = 5.58$. The probability of persistence in each trajectory m (ϕ_m) will be the mathematical complement of the number of time steps within the three different periods (150, 250, 500) that the population was lower than the threshold. The resulting probability ϕ_m is recorded for each trajectory m ($M = 50,000$), estimating the expected value ($\hat{\phi}$) and standard deviation ($\sqrt{Var(\hat{\phi})}$) as variability of the estimation per each week with data.

```
#variables:
#Log observations for the entire dataset
yt = snailkites.PP |>
  ungroup() |>
  arrange(Time.t) |>
  drop_na(Observed.y) |>
  dplyr::select(Observed.y) |>
  log()

#Vector form
yt <- yt$Observed.y
yt
```

```
##   [1] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 1.3862944
##   [8] 1.3862944 0.6931472 0.6931472 0.6931472 1.0986123 1.0986123 1.6094379
##  [15] 1.6094379 1.3862944 1.0986123 1.6094379 0.6931472 1.0986123 0.6931472
##  [22] 0.0000000 1.6094379 0.0000000 0.0000000 1.3862944 1.3862944 1.3862944
##  [29] 0.6931472 0.6931472 1.6094379 1.3862944 1.0986123 0.0000000 2.3978953
##  [36] 2.6390573 1.9459101 1.3862944 2.0794415 2.3025851 1.7917595 2.3025851
##  [43] 1.6094379 2.6390573 1.7917595 1.6094379 1.7917595 1.7917595 2.0794415
##  [50] 2.4849066 2.8903718 1.3862944 2.3978953 1.7917595 2.0794415 2.0794415
##  [57] 1.6094379 2.3025851 1.6094379 1.9459101 1.3862944 1.7917595 1.6094379
##  [64] 0.0000000 1.3862944 1.7917595 1.0986123 1.6094379 2.0794415 1.0986123
##  [71] 0.0000000 0.6931472 2.0794415 1.3862944 1.7917595 1.9459101 1.7917595
##  [78] 2.3025851 2.4849066 1.3862944 2.1972246 2.0794415 2.1972246 1.9459101
##  [85] 2.3025851 2.7080502 2.7725887 3.2958369 3.6888795 3.4011974 2.9444390
##  [92] 2.8903718 2.6390573 2.4849066 2.7080502 2.7080502 2.7080502 2.9444390
##  [99] 2.9444390 2.7725887 3.1780538 2.4849066 2.7725887 2.0794415 2.7080502
## [106] 4.0943446 2.3978953 2.3025851 2.8903718 1.7917595 2.7725887 1.7917595
## [113] 2.3025851 2.3978953 2.3978953 2.3978953 2.3025851 2.0794415 1.7917595
## [120] 2.1972246 1.9459101 1.9459101 1.0986123 2.0794415 1.6094379 1.0986123
## [127] 1.3862944 1.7917595 1.6094379 1.6094379 1.7917595 0.6931472 2.0794415
## [134] 1.7917595 1.3862944 1.0986123 1.7917595 2.1972246 2.6390573 1.7917595
## [141] 2.0794415 2.4849066 1.6094379 3.5553481 3.2188758 2.4849066 2.9957323
## [148] 3.3672958 3.0910425 3.2580965 3.2580965 2.0794415 2.7725887 2.3025851
## [155] 2.0794415 2.4849066 1.3862944 2.4849066 1.3862944 1.3862944 2.0794415
## [162] 1.9459101 1.9459101 1.6094379 0.0000000 1.0986123 0.0000000 0.0000000
## [169] 1.0986123 1.6094379 0.6931472 0.6931472 1.0986123 0.0000000 0.6931472
## [176] 1.7917595 1.0986123 1.7917595 0.0000000 0.6931472 2.0794415 1.0986123
## [183] 1.6094379 1.6094379 0.6931472 1.0986123 1.7917595 1.3862944 1.0986123
## [190] 2.0794415 1.7917595 2.4849066 3.4011974 2.5649494 1.9459101 2.1972246
```

```
## [197] 2.4849066 2.0794415 2.4849066 2.0794415 1.7917595 1.9459101 2.0794415
## [204] 2.0794415 1.7917595 2.0794415 1.6094379 2.9957323 1.7917595 1.9459101
## [211] 1.9459101 2.3025851 2.3978953 2.9957323 2.4849066 2.3025851 2.3978953
## [218] 2.0794415 3.1354942 2.9957323 2.6390573 3.4011974 2.3025851 1.7917595
## [225] 2.3025851 2.4849066 2.0794415 3.2188758 3.4965076 2.3025851 1.9459101
## [232] 3.4011974 2.9444390 2.7725887 2.9957323 3.0445224 2.7080502 3.9120230
## [239] 3.7612001 3.2188758 3.9318256 3.5553481 3.7376696 3.4011974 3.2188758
## [246] 3.0910425 3.0445224 3.2188758 2.3978953 4.3174881 2.7080502 2.7080502
## [253] 2.9957323 2.9444390 2.9957323 2.9957323 3.2188758 3.2188758 3.4011974
## [260] 2.9957323 2.9957323 3.1354942 2.4849066 2.7080502 3.0445224 3.2188758
## [267] 3.0910425 2.9957323 3.2188758 2.7725887 3.2188758 2.9957323 2.7080502
## [274] 2.4849066 2.3025851 1.6094379 1.9459101 1.0986123 1.3862944 1.7917595
## [281] 2.1972246 2.1972246 2.3978953 2.7725887 2.7725887 2.6390573 2.3025851
## [288] 2.3025851 3.4011974 2.6390573 2.7080502 2.7080502 2.3025851 3.4011974
## [295] 3.1780538 2.8332133 3.2188758 2.8332133 2.9957323 2.4849066 2.4849066
## [302] 3.2188758 2.6390573 2.0794415 2.7080502 1.6094379 2.0794415 1.7917595
## [309] 1.6094379 1.6094379 1.0986123 1.0986123 1.7917595 1.7917595 0.6931472
## [316] 0.0000000 1.3862944 0.6931472 1.3862944 1.6094379 1.0986123 1.6094379
## [323] 1.6094379 2.3025851 2.0794415 1.9459101 2.3025851 2.6390573 1.6094379
## [330] 1.7917595 0.6931472 0.6931472 0.0000000 0.6931472 0.6931472 0.6931472
## [337] 1.7917595 2.6390573 3.0910425 2.4849066
```

```
tt <- snailkites.PP |>
  ungroup() |>
  arrange(Time.t) |>
  drop_na(Observed.y) |>
  dplyr::select(Time.t)

#Time vector of the time series
tt <- tt$Time.t
tt
```

```
##   [1]   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25
##  [19]  26  27  28  29  30  31  32  33  35  36  37  39  40  41  42  43  44  45
##  [37]  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63
##  [55]  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81
##  [73]  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99
##  [91] 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117
## [109] 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135
## [127] 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153
## [145] 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171
## [163] 172 173 174 175 176 177 178 179 180 181 182 183 184 186 188 189 190 191
## [181] 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209
## [199] 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227
## [217] 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245
## [235] 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263
## [253] 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281
## [271] 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299
## [289] 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317
## [307] 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335
## [325] 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351
```

```
#the establishment (matching SM data) is going up to the last position
last.tt <- 79
```

```r
#Define N_critical for eBird
N.critical = round(1/2*mean(exp(yt)),0)

#number of simulations
ntraj = 50000

#last observed time (week 87 - August 2019)
l = last(tt[1:last.tt])

#Simulation length
len.sim <- c(151, 251, 501)

#which correspond to three timeframes
timeframes <- c("150 weeks or ~3 years",
                "250 weeks or ~5 years",
                "500 weeks or ~10 years")
```

```r
#eval=FALSE

#save   (SD)
phi_results <- list()

# Set the base filename
base_filename <- "supporting/FigS2_Step2_1stTrajectories_10_5"

#time of starting process
StartTime <- Sys.time()

for (i in seq_along(len.sim)){
  #to save figures in png and call them after
  png_filename <- paste0(base_filename, "_", (len.sim[i]-1), ".png")
  png(png_filename, width = 10, height = 5, units = "in", res = 300)

  #plot the abundance data and model estimation
  plot(tt[1:last.tt],
     exp(yt[1:last.tt]),
     type="b", lwd=2, cex.lab=1.25, col = "#4575b4", pch = 1,
     xlim=c(min(tt), (max(tt)*3)),
     ylim=c(0,250),
     ylab="Observed/predicted weekly high counts",
     xlab="Time (week since January 2018)");

  points(x = sk.Estimated$Time.t,
     y = sk.Estimated$Estimated_eBird_EGSS,
     type = "b", col = "#91bfdb", pch = 2)

  thres.times <- as.numeric(0:(len.sim[i]-1))
  len <- max(thres.times)+1

  sim.mat.eBird <- egss_sim(ntraj,
                            tt = thres.times,
                            parms = sk.egss.parms$remles)
  phi <- rep(0, ntraj)
  last.points <- rep(0, ntraj)
```

```
  for(n in 1:ntraj){
    Pop.sim <- exp(c(yt[last.tt], sim.mat.eBird[-1,n]));
    last.points[n] <- Pop.sim[len]

    #How many times each trajectory go below the threshold?
    below.threshold <- sum(Pop.sim < N.critical)
    phi[n] <- 1-(below.threshold/len)

        if( phi[n]>=0.5){
      lines(l+thres.times, Pop.sim, col="#d3d3d308", lty = "solid")
    }else{lines(l+thres.times, Pop.sim, col="#FF000008", lty = "solid")
    }
  }
    #add critical value line
  abline(h=N.critical, lty=2, lwd=1);

  #Expected value of probability of local persistence, and SD
  phi_mean <- mean(phi)
  phi_SD <- sqrt(var(phi))

  phi_results[[timeframes[i]]] <- cbind(phi_mean,phi_SD)

  #Create a kernel density estimate (KDE) of the `last.points` results (library(ke31d))
  kde.sims <- kde1d(x=last.points);

  #add the distribution to the plot
  #above the threshold
  lines(x = (kde.sims$values*(ntraj/5) +
               l + last(thres.times)),
      y = kde.sims$grid_points,
      col="#d3d3d398", lwd=2, lty=1)

    lines(x = (kde.sims$values[kde.sims$grid_points <= N.critical]*(ntraj/5) +
                 l + last(thres.times)),
      y = kde.sims$grid_points[kde.sims$grid_points <= N.critical],
      col="#FF000098", lwd=2, lty=1)

    title(main=paste0("  = ",
                  signif(phi_mean,2),
                  " (SD: ",
                  signif(phi_SD,2),
                  ")", "; 1st EGSS through week ", tt[last.tt],
                  ", projected for ~",
                  signif((len.sim[i]-1)/52, 1),
                  " year(s)"),
      cex=1.5)
    dev.off();

    print(cbind(tt[last.tt], phi_mean, phi_SD, len.sim[i]-1))
}

#time process finished
EndTime <- Sys.time()
```

```
#difference of time (time lasted in the process)
timelast <- EndTime-StartTime
timelast

#Results for the first week in three timeframes
phi_results
```

We can see graphically the trajectories and the difference of ϕ estimated for the week 88. The figures show the observed weekly high counts (blue circles), the fitted model (light blue triangles), 50,000 trajectories (lines; those with $\phi < 0.5$ in reddish, else in gray), $N_{critical}$ as horizontal dashed line, and the kernel density estimates of the last point of the trajectories (reddish below $N_{critical}$, else gray).

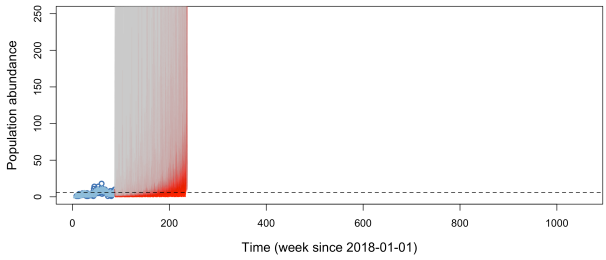


Figure SI-18: First round of trajectories - ~3 years

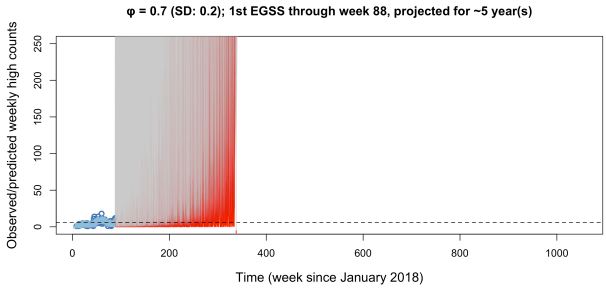


Figure SI-19: First round of trajectories - ~5 years

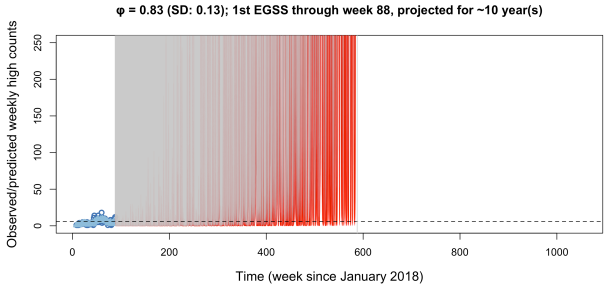


Figure SI-20: First round of trajectories - ~10 years

5.3 Add the next observations, repeat model fit, projection, and $\hat{\phi}$

We can add the next week of observations (week 89; second week of September 2019) to the time-series, re-estimate model parameters as well as the probability of crashing below the $N_{critical}$ during the three timeframes (150, 250, 500 weeks). Here, users can try to fit the OUSS model the following weeks, and decided by the density dependence parameter $\hat{\theta}$ to fit the EGSS model instead. We provide below the example, but the main results focused on EGSS, which is the expected dynamic in this expanded population.

```
#The next week: 88+1
last.tt <- which(tt==89)

yt[1:last.tt] #It was already converted to the log-abundance
```

```
##  [1] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 1.3862944
##  [8] 1.3862944 0.6931472 0.6931472 0.6931472 1.0986123 1.0986123 1.6094379
## [15] 1.6094379 1.3862944 1.0986123 1.6094379 0.6931472 1.0986123 0.6931472
## [22] 0.0000000 1.6094379 0.0000000 0.0000000 1.3862944 1.3862944 1.3862944
## [29] 0.6931472 0.6931472 1.6094379 1.3862944 1.0986123 0.0000000 2.3978953
## [36] 2.6390573 1.9459101 1.3862944 2.0794415 2.3025851 1.7917595 2.3025851
## [43] 1.6094379 2.6390573 1.7917595 1.6094379 1.7917595 1.7917595 2.0794415
## [50] 2.4849066 2.8903718 1.3862944 2.3978953 1.7917595 2.0794415 2.0794415
## [57] 1.6094379 2.3025851 1.6094379 1.9459101 1.3862944 1.7917595 1.6094379
## [64] 0.0000000 1.3862944 1.7917595 1.0986123 1.6094379 2.0794415 1.0986123
## [71] 0.0000000 0.6931472 2.0794415 1.3862944 1.7917595 1.9459101 1.7917595
## [78] 2.3025851 2.4849066 1.3862944
```

```
tt[1:last.tt]
```

```
##  [1]  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
## [26] 33 35 36 37 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59
## [51] 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84
## [76] 85 86 87 88 89
```

```
#fit the OUSS
OUSS.partial <- ouss_reml(yt = yt[1:last.tt],
                          tt = tt[1:last.tt],
                          fguess = guess_ouss(yt = yt[1:last.tt],
                                              tt = tt[1:last.tt]))

OUSS.partial$remles
```

```
## [1] 9.918641e-01 1.669843e-07 3.265292e-02 2.699545e-01
```

The function `ouss_reml()` fit the OUSS model and estimate parameters. In order, the values correspond to the mean stationary distribution log-abundance ($\hat{\mu}_{eBird} = 0.99186$); the trend, speed of equilibration, rate to approach stationary distribution, or density dependence parameter ($\hat{\theta}_{eBird} = 0.0000002$); the environmental noise ($\hat{\sigma}^2_{SM} = 0.03265$); and the observation error noise ($\hat{\tau}^2_{SM} = 0.26995$). Note that the density dependence parameter $\hat{\theta} < 0.025$, which suggests density independence and the use of the EGSS model instead.

```
model <- if(OUSS.partial$remles[2] < 0.025){"EGSS"}else{"OUSS"}

model
```

```
## [1] "EGSS"
```

```
#Lets fit EGSS model partial for second observation (week 89)
EGSS.partial <- egss_reml(yt = yt[1:last.tt],
                          tt = tt[1:last.tt],
```

```
                          fguess = guess_egss(yt = yt[1:last.tt],
                                              tt = tt[1:last.tt]))
EGSS.partial
```

```
## $remles
## [1]  0.021912040  0.032929682  0.268895829 -0.009539194
##
## $lnL.hat
## [1] -75.54
##
## $AIC
## [1] 155.08
```

```
sk.egss.predict1 <- egss_predict(yt = yt[1:last.tt],
                                       tt = tt[1:last.tt],
                                       parms = EGSS.partial$remles,
                                  plot.it = F)

head(sk.egss.predict1[[1]])
```

```
##      Time Predict.EGSS.REML Observed.y
## [1,]    8         0.9905062          1
## [2,]    9         1.0110839          1
## [3,]   10         1.0271032          1
## [4,]   11         1.0378248          1
## [5,]   12         1.0443875          1
## [6,]   13         1.0482439          1
```

Remember that `Time` is the week in our case, and `Observed.y` the weekly high count in eBird. Let's change column names to combine with the `datesPP`.

```
colnames(sk.egss.predict1[[1]]) <- c("Time.t", "Estimated_eBird_EGSS","eBird.Observed")

sk.Estimated.1 <- data.frame(sk.egss.predict1) |>
  left_join(datesPP)
```

```
## Joining with `by = join_by(Time.t)`
```

And we can run the simulations with the new model updated, saving the $\hat{\phi}$ values and their standard deviation for three timeframes.

```
#eval=FALSE
l = last(tt[1:last.tt])

#save   (SD)
phi_results <- list()

# Set the base filename
base_filename <- "supporting/FigS3_Step3_2ndTrajectories_10_5"

StartTime <- Sys.time()

for (i in seq_along(len.sim)){

  #to save figures in png and call them after
  png_filename <- paste0(base_filename, "_", len.sim[i]-1, ".png")
```

```r
png(png_filename, width = 10, height = 5, units = "in", res = 300)

#plot the abundance data and model estimation
plot(tt[1:last.tt],
   exp(yt[1:last.tt]),
   type="b", lwd=2, cex.lab=1.25, col = "#4575b4", pch = 1,
   xlim=c(min(tt), (max(tt)*3)),
   ylim=c(0,250),
   ylab="Observed/predicted weekly high counts",
   xlab="Time (week since January 2018)");

points(x = sk.Estimated.1$Time.t,
   y = sk.Estimated.1$Estimated_eBird_EGSS,
   type = "b", col = "#91bfdb", pch = 2)

thres.times <- as.numeric(0:(len.sim[i]-1))
len <- max(thres.times)+1

sim.mat.eBird <- egss_sim(ntraj,
                          tt = thres.times,
                          parms = EGSS.partial$remles)
phi <- rep(0, ntraj)
last.points <- rep(0, ntraj)

for(n in 1:ntraj){
  Pop.sim <- exp(c(yt[last.tt], sim.mat.eBird[-1,n]));
  last.points[n] <- Pop.sim[len]

  #How many times each trajectory go below the threshold?
  below.threshold <- sum(Pop.sim < N.critical)
  phi[n] <- 1-(below.threshold/len)

      if(phi[n]>=0.5){
    lines(l+thres.times, Pop.sim, col="#d3d3d308", lty = "solid")
  }else{lines(l+thres.times, Pop.sim, col="#FF000008", lty = "solid")
  }
}
  #add critical value line
abline(h=N.critical, lty=2, lwd=1);

#Expected value of probability of local persistence, and SD
phi_mean <- mean(phi)
phi_SD <- sqrt(var(phi))

phi_results[[timeframes[i]]] <- cbind(phi_mean,phi_SD)

#Create a kernel density estimate (KDE) of the `last.points` results (library(ke31d))
kde.sims <- kde1d(x=last.points);

  #add the distribution to the plot
#above the threshold
lines(x = (kde.sims$values*(ntraj/5) +
             l +
```

```
                last(thres.times)),
      y = kde.sims$grid_points,
      col="#d3d3d398", lwd=2, lty=1)

    lines(x = (kde.sims$values[kde.sims$grid_points <= N.critical] *
                 ntraj + l + last(thres.times)),
      y = kde.sims$grid_points[kde.sims$grid_points <= N.critical],
      col="#FF000098", lwd=2, lty=1)

    title(main=paste0("  = ",
                  signif(phi_mean,2),
                  " (SD: ",
                  signif(phi_SD,2),
                  ")", "; 2nd EGSS through week ", tt[last.tt],
                  ", projected for ~",
                  signif((len.sim[i]-1)/52, 1),
                  " year(s)"),
      cex=1.5)

    dev.off()
    print(cbind(tt[last.tt], phi_mean, phi_SD, len.sim[i]-1))
}
#time process finished
EndTime <- Sys.time()
#difference of time (time lasted in the process)
timelast <- EndTime-StartTime
timelast

phi_results
```

And we can see the new trajectories projected in three timeframes.

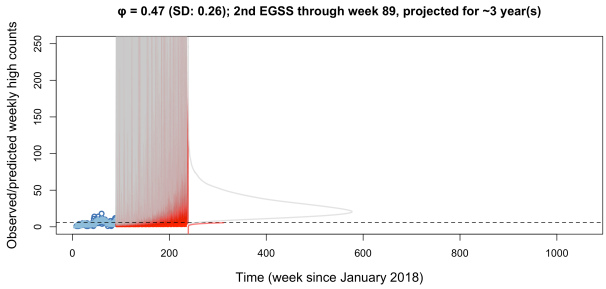


Figure SI-21: Second round of trajectories - ~3 years

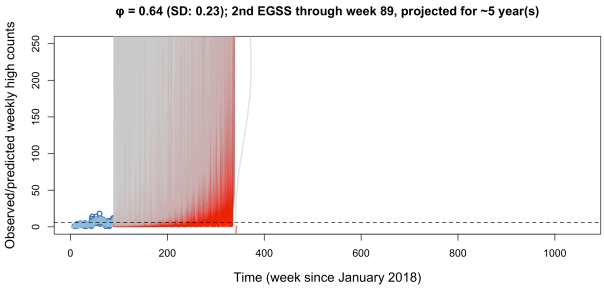


Figure SI-22: Second round of trajectories - ~5 years

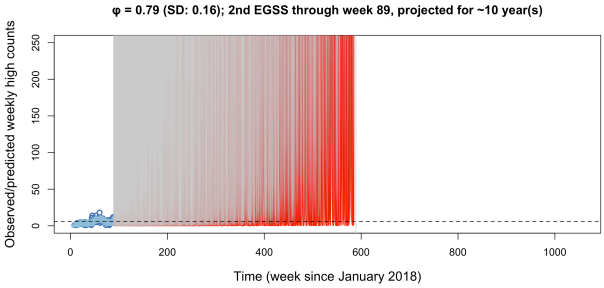


Figure SI-23: Second round of trajectories - ~10 years

5.4 Iterate the process

5.4.1 Iterate the process with eBird contrasting OUSS-EGSS (some figure examples)

Let's iterate the process for some weeks that share data in eBird and Standardized Monitored. This is going to change the ending position; instead of manually change 88 to 89, we use a vector of the "ending positions" and iterate from a sequence of length of that vector.

```
#eval=FALSE
#It last an hour
# Log observations for the entire dataset
yt = snailkites.PP |>
  ungroup() |>
  arrange(Time.t) |>
  drop_na(Observed.y) |>
  dplyr::select(Observed.y) |>
  log()

# Vector form
yt <- yt$Observed.y
yt

tt <- snailkites.PP |>
  ungroup() |>
  arrange(Time.t) |>
  drop_na(Observed.y) |>
  dplyr::select(Time.t)

# Time vector of the time series
```

```r
tt <- tt$Time.t
tt

# Define N_critical for eBird
N.critical = round((1/2)*mean(exp(yt)), 0)

# List of end positions to be used
which(tt == 95) #week 95 - around 2019-10-22 - is position 86 in tt
#which(tt == 148) #week 148 - around 2020-10-28 - is position 139 in tt
which(tt == 211) #week 211 - around 2022-01-15 - is position 200 in tt
#which(tt == 263) #week 263 - around 2023-01-15 - is position 252 in tt
which(tt == 316) #week 316 - around 2024-01-22 - is position 305 in tt

end_positions <- c(86,200,305)

#save   (SD)
phi_results <- vector("list", length = length(end_positions))

#save model used
modelSS <- vector("list", length = length(end_positions))

# Set the base filename for figures
base_filename <- "supporting/FigS4_Step4_IterateTraj_10_5"

# Start process timing
StartTime <- Sys.time()

for (j in seq_along(end_positions)) {

  last.tt <- end_positions[j]
  l <- tt[last.tt]

  for (i in seq_along(len.sim)){
      # Set up plot for all three timeframes
  png_filename <- paste0(base_filename,
                         "_endpos_", l,
                         "_timeframe_",(len.sim[i]-1),
                         ".png")
  png(png_filename, width = 10, height = 5, units = "in", res = 300)

    plot(tt[1:last.tt],
         exp(yt[1:last.tt]),
         type="b", lwd=2, cex.lab=1.25, col = "#4575b4", pch = 1,
         xlim=c(min(tt), (max(tt)*3)),
         ylim=c(0,250),
         ylab="Observed/predicted weekly high counts",
         xlab="Time (week since January 2018)");

    OUSS.partial <- ouss_reml(yt = yt[1:last.tt],
                              tt = tt[1:last.tt],
                              fguess = guess_ouss(yt = yt[1:last.tt],
                                                  tt = tt[1:last.tt]))
```

```
  model <- if(OUSS.partial$remles[2] < 0.025){
    "EGSS"
  }else{
    "OUSS"
  }
  modelSS[[j]][[timeframes[i]]] <- model

  if(model == "OUSS"){
    parcial.predict <- ouss_predict(yt = yt[1:last.tt],
                                    tt = tt[1:last.tt],
                                    parms = OUSS.partial$remles,
                                    plot.it = F)

    points(x = (parcial.predict)[[1]][,1],
           y = (parcial.predict)[[1]][,2],
           type = "b", col = "#91bfdb", pch = 2)

    thres.times <- as.numeric(0:(len.sim[i]-1))
    len <- max(thres.times) + 1

    sim.mat.eBird <- ouss_sim(ntraj,
                              tt = thres.times,
                              parms = OUSS.partial$remles)

    phi <- rep(0, ntraj)
    last.points <- rep(0, ntraj)

    for(n in 1:ntraj){
      Pop.sim <- exp(c(yt[last.tt], sim.mat.eBird[-1,n]));
      last.points[n] <- Pop.sim[len]

    # How many times each trajectory go below the threshold?
      below.threshold <- sum(Pop.sim < N.critical)
      phi[n] <- 1-(below.threshold/len)

      if(phi[n] >= 0.5){
        lines(l + thres.times, Pop.sim, col="#d3d3d308", lty = "solid")
        } else {
          lines(l + thres.times, Pop.sim, col="#FF000008", lty = "solid")
          }
  }
    # Add critical value line
    abline(h=N.critical, lty=2, lwd=1);

    #Expected value of probability of local persistence, and SD
    phi_mean <- mean(phi)
    phi_SD <- sqrt(var(phi))

    phi_results[[j]][[timeframes[i]]] <- cbind(phi_mean,phi_SD)

    # Create a kernel density estimate (KDE) of the `last.points` results (library(ke31d))
    kde.sims <- kde1d(x=last.points);
```

```
    # Add the distribution to the plot
    lines(x = (kde.sims$values*(ntraj/5) + l + last(thres.times)),
          y = kde.sims$grid_points,
          col="#d3d3d398", lwd=2, lty=1)

    lines(x = (kde.sims$values[kde.sims$grid_points <= N.critical]*(ntraj/5) +
                 l + last(thres.times)),
          y = kde.sims$grid_points[kde.sims$grid_points <= N.critical],
          col="#FF000098", lwd=2, lty=1)

    title(main=paste0("  = ",
                      signif(phi_mean,2),
                      " (SD: ",
                      signif(phi_SD,2),
                      "); ", model, "; through week ", tt[last.tt],
                      ", projected for ~",
                      signif((len.sim[i]-1)/52, 1),
                      " year(s)"),
          cex=1.5)
  }

  else{

    EGSS.partial <- egss_reml(yt = yt[1:last.tt],
                              tt = tt[1:last.tt],
                              fguess = guess_egss(yt = yt[1:last.tt],
                                                  tt = tt[1:last.tt]));
    parcial.predict <- egss_predict(yt = yt[1:last.tt],
                                    tt = tt[1:last.tt],
                                    parms = EGSS.partial$remles,
                                    plot.it = F)

    points(x = (parcial.predict)[[1]][,1],
           y = (parcial.predict)[[1]][,2],
           type = "b", col = "#91bfdb", pch = 2)

    thres.times <- as.numeric(0:(len.sim[i]-1))
    len <- max(thres.times) + 1

    sim.mat.eBird <- egss_sim(ntraj,
                              tt = thres.times,
                              parms = EGSS.partial$remles)

    phi <- rep(0, ntraj)
    last.points <- rep(0, ntraj)

    for(n in 1:ntraj){
      Pop.sim <- exp(c(yt[last.tt], sim.mat.eBird[-1,n]));
      last.points[n] <- Pop.sim[len]

    # How many times each trajectory go below the threshold?
      below.threshold <- sum(Pop.sim < N.critical)
      phi[n] <- 1-(below.threshold/len)
```

```
        if(phi[n] >= 0.5){
          lines(l+thres.times, Pop.sim, col="#d3d3d308", lty = "solid")
          } else {
            lines(l+thres.times, Pop.sim, col="#FF000008", lty = "solid")
            }
    }
      # Add critical value line
      abline(h=N.critical, lty=2, lwd=1);

      #Expected value of probability of local persistence, and SD
      phi_mean <- mean(phi)
      phi_SD <- sqrt(var(phi))

      phi_results[[j]][[timeframes[i]]] <- cbind(phi_mean,phi_SD)

      # Create a kernel density estimate (KDE) of the `last.points` results (library(ke31d))
      kde.sims <- kde1d(x=last.points);

      # Add the distribution to the plot
      lines(x = (kde.sims$values*(ntraj/5) + l + last(thres.times)),
            y = kde.sims$grid_points,
            col="#d3d3d398", lwd=2, lty=1)

      lines(x = (kde.sims$values[kde.sims$grid_points <= N.critical]*(ntraj/5) +
                   l + last(thres.times)),
            y = kde.sims$grid_points[kde.sims$grid_points <= N.critical],
            col="#FF000098", lwd=2, lty=1)

      title(main=paste0("  = ",
                        signif(phi_mean,2),
                        " (SD: ",
                        signif(phi_SD,2),
                        "); ", model, "; through week ", tt[last.tt],
                        ", projected for ~",
                        signif((len.sim[i]-1)/52, 1),
                        " year(s)"),
            cex=1.5)
    }

    dev.off()

  }
  print(cbind(l, model, phi_mean, phi_SD, len.sim[i]-1))
}

#time process finished
EndTime <- Sys.time()
#difference of time (time lasted in the process)
timelast <- EndTime-StartTime
timelast
```

```
     l    model  phi_mean           phi_SD
[1,] "95" "EGSS" "0.847376087824351" "0.120321511926854" "500"
     l     model  phi_mean            phi_SD
[1,] "211" "OUSS" "0.433565109780439" "0.130684439548197" "500"
     l     model  phi_mean            phi_SD
[1,] "316" "EGSS" "0.542928622754491" "0.304126980008073" "500"
```

Figure SI-24: iterating some weeks

5.4.1.1 Trajectories in timeframe of ~3 years.

Let's see the examples.

How is changing in three moments $\hat{\phi}_{eBird\ \sim three\ years\ simulated}$?

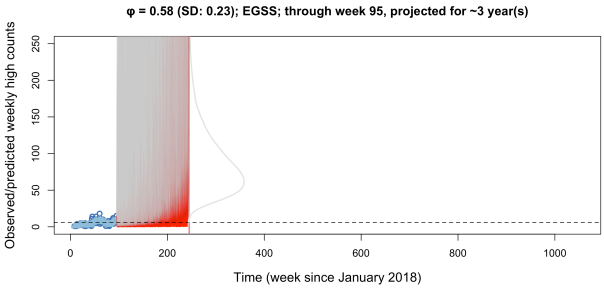


Figure SI-25: week 95 - October 2019 simulated for ~3 years

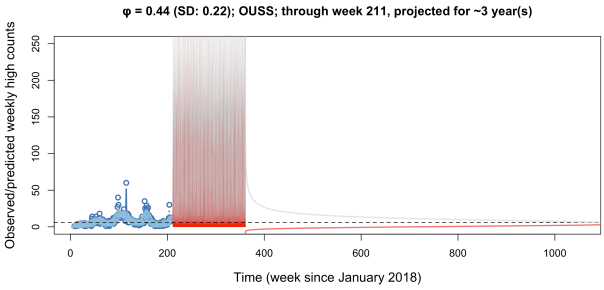


Figure SI-26: week 211 - January 2022 simulated for ~3 years

Note that for the week 211 (January 2022) the population model fitted is the density dependence model (OUSS). However, this might be do to the fact that lower and less variable weekly high counts are similar to the initial population, misidentifying density dependence dynamic in this expanding population. We include this example to show the users how to apply the contrast for established populations (see also Section 8 - Other populations of Snail Kite).

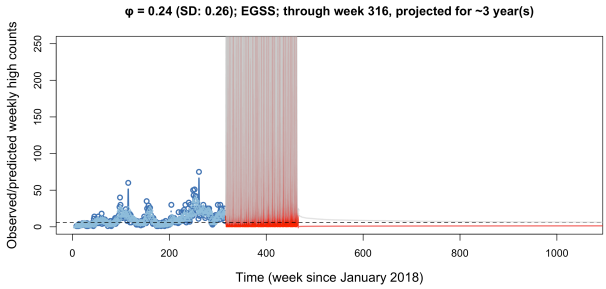


Figure SI-27: week 316 - January 2024 simulated for ~3 years

5.4.1.2 Trajectories in timeframe of ~5 years.

How is changing in 5 moments $\hat{\phi}_{eBird\ \sim five\ years\ simulated}$?

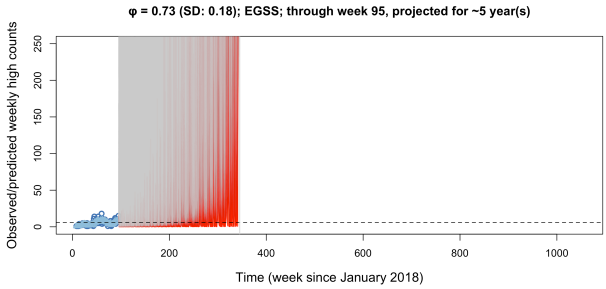


Figure SI-28: week 95 - October 2019 simulated for ~5 years

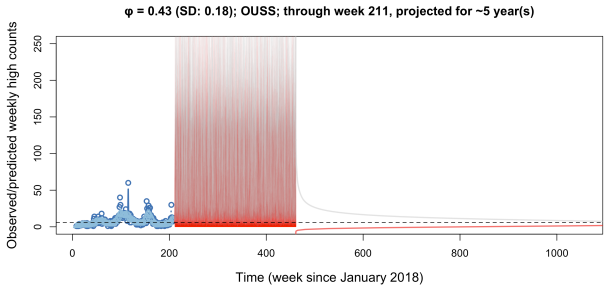


Figure SI-29: week 211 - January 2022 simulated for ~5 years

Again, it is not expected for the population to be at stationary distribution, unless we adjust a new initial population of the time series (see Dennis & Ponciano for more details on the OUSS model).

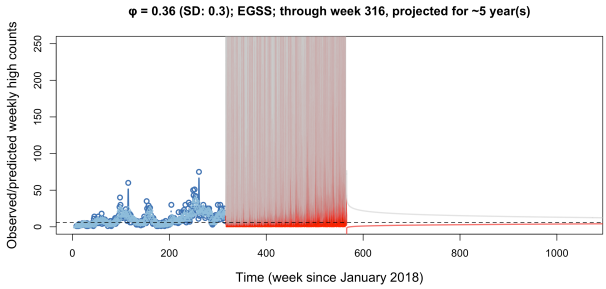


Figure SI-30: week 316 - January 2024 simulated for ~5 years

5.4.1.3 Trajectories in timeframe of ~10 years.

How is changing in 5 moments $\hat{\phi}_{eBird\ \sim ten\ years\ simulated}$?

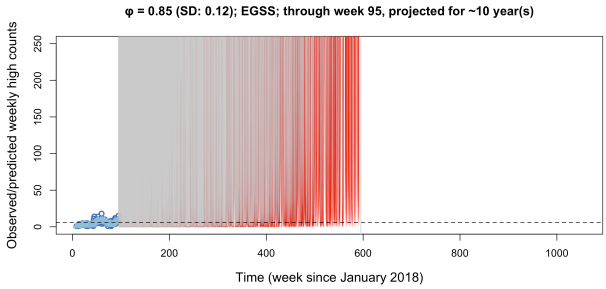


Figure SI-31: week 95 - October 2019 simulated for ~10 years

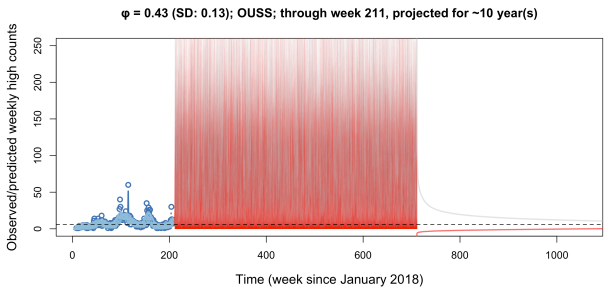


Figure SI-32: week 211 - January 2022 simulated for ~10 years

The pattern selected on week 211 keeps selecting OUSS, likely misidentifying density dependence from the eBird data.

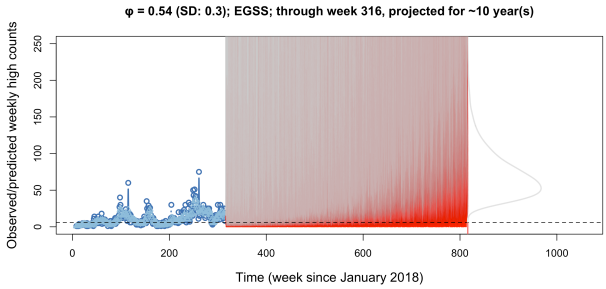


Figure SI-33: week 316 - January 2024 simulated for ~10 years

5.4.2 Iterate each week to extract $\hat{\phi}$ and SD from eBird contrasting OUSS-EGSS (no figures)

```
#Change at the end to `{r eval=FALSE}`; it might take long time

#the data
snailkites.PP <- readRDS("data_tmp/snailkitesPP.rds")

#Log observations for the entire data set
yt = snailkites.PP |>
  ungroup() |>
  arrange(Time.t) |>
  drop_na(Observed.y) |>
  dplyr::select(Observed.y) |>
  log()

#Vector form
yt <- yt$Observed.y
yt

# Define N_critical for eBird
N.critical = round((1/2)*mean(exp(yt)),0)

tt <- snailkites.PP |>
  ungroup() |>
  arrange(Time.t) |>
  drop_na(Observed.y) |>
  dplyr::select(Time.t)

#Time vector of the time series
tt <- tt$Time.t
tt

#the first two years length
last.tt <- 87

#Number of trajectories and bootstraps (CIs)
ntraj = 50000

# End positions to modeling (steps)
end_positions <- which(tt > last.tt)
```

```r
#check weeks ID selected as `end_positions`
tt[end_positions]

#save   (SD)
phi_results <- vector("list", length = length(end_positions))

#save model used
modelSS <- vector("list", length = length(end_positions))

# Start process timing
StartTime <- Sys.time()

for (j in seq_along(end_positions)) {

  last.tt <- end_positions[j]
  l <- tt[last.tt]

  for (i in seq_along(len.sim)){

    OUSS.partial <- ouss_reml(yt = yt[1:last.tt],
                              tt = tt[1:last.tt],
                              fguess = guess_ouss(yt = yt[1:last.tt],
                                                  tt = tt[1:last.tt]))

    model <- if(OUSS.partial$remles[2] < 0.025){
      "EGSS"
    }else{
      "OUSS"
    }
    modelSS[[j]][[timeframes[i]]] <- model

    if(model == "OUSS"){

      thres.times <- as.numeric(0:(len.sim[i]-1))
      len <- max(thres.times) + 1

      sim.mat.eBird <- ouss_sim(ntraj,
                                tt = thres.times,
                                parms = OUSS.partial$remles)

      phi <- rep(0, ntraj)
      last.points <- rep(0, ntraj)

      for(n in 1:ntraj){
        Pop.sim <- exp(c(yt[last.tt], sim.mat.eBird[-1,n]));
        last.points[n] <- Pop.sim[len]

      # How many times each trajectory go below the threshold?
        below.threshold <- sum(Pop.sim < N.critical)
        phi[n] <- 1-(below.threshold/len)
      }

      #Expected value of probability of local persistence, and SD
```

```
      phi_mean <- mean(phi)
      phi_SD <- sqrt(var(phi))

      phi_results[[j]][[timeframes[i]]] <- cbind(phi_mean,phi_SD)

    }
    else{

      EGSS.partial <- egss_remle(yt = yt[1:last.tt],
                                 tt = tt[1:last.tt],
                                 fguess = guess_egss(yt = yt[1:last.tt],
                                                     tt = tt[1:last.tt]));

      thres.times <- as.numeric(0:(len.sim[i]-1))
      len <- max(thres.times) + 1

      sim.mat.eBird <- egss_sim(ntraj,
                                tt = thres.times,
                                parms = EGSS.partial$remles)

      phi <- rep(0, ntraj)
      last.points <- rep(0, ntraj)

      for(n in 1:ntraj){
        Pop.sim <- exp(c(yt[last.tt], sim.mat.eBird[-1,n]));
        last.points[n] <- Pop.sim[len]

      # How many times each trajectory go below the threshold?
        below.threshold <- sum(Pop.sim < N.critical)
        phi[n] <- 1-(below.threshold/len)
      }

      #Expected value of probability of local persistence, and SD
      phi_mean <- mean(phi)
      phi_SD <- sqrt(var(phi))

      phi_results[[j]][[timeframes[i]]] <- cbind(phi_mean,phi_SD)

    }
  }
  print(cbind(l, model,
              phi_mean,
              phi_SD,
              len.sim[i]-1))
}

#time process finished
EndTime <- Sys.time()
#difference of time (time lasted in the process)
timelast <- EndTime-StartTime
timelast
```

...

```
     l    model  phi_mean           phi_SD
[1,] "88" "EGSS" "0.84454626746507" "0.126617086262298" "500"
     l    model  phi_mean            phi_SD
[1,] "89" "EGSS" "0.792343632734531" "0.161768189801998" "500"
     l    model  phi_mean            phi_SD
[1,] "90" "EGSS" "0.820494131736527" "0.142522226583426" "500"
     l    model  phi_mean           phi_SD
[1,] "91" "EGSS" "0.8223624750499" "0.139970753725444" "500"
     l    model  phi_mean            phi_SD
[1,] "92" "EGSS" "0.827177005988024" "0.137469337975623" "500"
     l    model  phi_mean            phi_SD
[1,] "93" "EGSS" "0.814261077844311" "0.145887989455012" "500"
     l    model  phi_mean           phi_SD
[1,] "94" "EGSS" "0.82705381237525" "0.135215899541732" "500"
     l    model  phi_mean            phi_SD
[1,] "95" "EGSS" "0.846947704590818" "0.120689470204239" "500"
     l    model  phi_mean            phi_SD
```

Figure SI-34: Iterating all

```
     l     model  phi_mean            phi_SD
[1,] "347" "EGSS" "0.325836407185629" "0.319505756664078" "500"
     l     model  phi_mean            phi_SD
[1,] "348" "EGSS" "0.398255289421158" "0.328577122859422" "500"
     l     model  phi_mean            phi_SD
[1,] "349" "EGSS" "0.478135968063872" "0.326428000195004" "500"
     l     model  phi_mean            phi_SD
[1,] "350" "EGSS" "0.526691497005988" "0.318771170864887" "500"
     l     model  phi_mean            phi_SD
[1,] "351" "EGSS" "0.514594211576846" "0.322386473028937" "500"
> #time process finished
> EndTime <- Sys.time()
> #difference of time (time lasted in the process)
> timelast <- EndTime-StartTime
> timelast
Time difference of 8.668951 hours
```

Figure SI-35: Iterating all

This is a longer process ($\sim$ 9 hours; currently is `{r eval=FALSE}`), so we save the `phi_results` (with `phi_mean` and `phi_SD`), and `modelSS` vectors as data frames to make a plot.

```r
#creating data frames for phi_mean, phi_SD, and modelSS
phi_eBird_df <- do.call(rbind, lapply(seq_along(phi_results), function(j) {
  do.call(rbind, lapply(seq_along(phi_results[[j]]), function(i) {
    data.frame(
      Time.t = tt[end_positions[j]],
      Timeframe = timeframes[i],
      phi.hat = phi_results[[j]][[i]][1],
      SD.phi = phi_results[[j]][[i]][2],
      Model = modelSS[[j]][[i]]
    )
  }))
}))

# Convert to factors for better plotting
phi_eBird_df$Timeframe <- factor(phi_eBird_df$Timeframe, levels = timeframes)
head(phi_eBird_df, n = 12)

saveRDS(phi_eBird_df, "results/phi_hat_SD_eBird_OUSS_EGSS.rds")
```

```
   Time.t             Timeframe   phi.hat    SD.phi Model
1      88  150 weeks or ~3 years 0.5862160 0.2360549  EGSS
2      88  250 weeks or ~5 years 0.7263161 0.1894415  EGSS
3      88 500 weeks or ~10 years 0.8445463 0.1266171  EGSS
4      89  150 weeks or ~3 years 0.4708873 0.2599956  EGSS
5      89  250 weeks or ~5 years 0.6395306 0.2297969  EGSS
6      89 500 weeks or ~10 years 0.7923436 0.1617682  EGSS
7      90  150 weeks or ~3 years 0.5262428 0.2492840  EGSS
8      90  250 weeks or ~5 years 0.6835844 0.2087211  EGSS
9      90 500 weeks or ~10 years 0.8204941 0.1425222  EGSS
10     91  150 weeks or ~3 years 0.5284556 0.2476024  EGSS
11     91  250 weeks or ~5 years 0.6847880 0.2070146  EGSS
12     91 500 weeks or ~10 years 0.8223625 0.1399708  EGSS
```

Figure SI-36: head screenshot of eBird phi estimations

5.4.3 Iterate each week to extract $\hat{\phi}$ and SD from eBird only EGSS (no figures)

```
#Change at the end to `{r eval=FALSE}`; it might take long time

#the data
snailkites.PP <- readRDS("data_tmp/snailkitesPP.rds")

#Log observations for the entire data set
yt = snailkites.PP |>
  ungroup() |>
  arrange(Time.t) |>
  drop_na(Observed.y) |>
  dplyr::select(Observed.y) |>
  log()

#Vector form
yt <- yt$Observed.y
yt

# Define N_critical for eBird
N.critical = round((1/2)*mean(exp(yt)),0)

tt <- snailkites.PP |>
  ungroup() |>
  arrange(Time.t) |>
  drop_na(Observed.y) |>
  dplyr::select(Time.t)

#Time vector of the time series
tt <- tt$Time.t
tt

#the first two years length
last.tt <- 87

#Number of trajectories and bootstraps (CIs)
ntraj = 50000

# End positions to modeling (steps)
end_positions <- which(tt > last.tt)

#check weeks ID selected as `end_positions`
tt[end_positions]

#save   (SD)
phi_results <- vector("list", length = length(end_positions))

#save model used
modelSS <- vector("list", length = length(end_positions))

# Start process timing
StartTime <- Sys.time()

for (j in seq_along(end_positions)) {
```

```
  last.tt <- end_positions[j]
  l <- tt[last.tt]

  for (i in seq_along(len.sim)){

    model <- "EGSS"

    modelSS[[j]][[timeframes[i]]] <- model

      EGSS.partial <- egss_reml(yt = yt[1:last.tt],
                                tt = tt[1:last.tt],
                                fguess = guess_egss(yt = yt[1:last.tt],
                                                    tt = tt[1:last.tt]));

      thres.times <- as.numeric(0:(len.sim[i]-1))
      len <- max(thres.times) + 1

      sim.mat.eBird <- egss_sim(ntraj,
                                tt = thres.times,
                                parms = EGSS.partial$remles)

      phi <- rep(0, ntraj)
      last.points <- rep(0, ntraj)

      for(n in 1:ntraj){
        Pop.sim <- exp(c(yt[last.tt], sim.mat.eBird[-1,n]));
        last.points[n] <- Pop.sim[len]

      # How many times each trajectory go below the threshold?
        below.threshold <- sum(Pop.sim < N.critical)
        phi[n] <- 1-(below.threshold/len)
      }

      #Expected value of probability of local persistence, and SD
      phi_mean <- mean(phi)
      phi_SD <- sqrt(var(phi))

      phi_results[[j]][[timeframes[i]]] <- cbind(phi_mean,phi_SD)


  }
  print(cbind(l, model,
              phi_mean,
              phi_SD,
              len.sim[i]-1))
}

#time process finished
EndTime <- Sys.time()
#difference of time (time lasted in the process)
timelast <- EndTime-StartTime
timelast
```

```
     l    model  phi_mean           phi_SD
[1,] "88" "EGSS" "0.844360798403194" "0.126583050557561" "500"
     l    model  phi_mean           phi_SD
[1,] "89" "EGSS" "0.79439125748503" "0.160991658893347" "500"
     l    model  phi_mean           phi_SD
[1,] "90" "EGSS" "0.821543872255489" "0.141191542369354" "500"
     l    model  phi_mean           phi_SD
[1,] "91" "EGSS" "0.822791736526946" "0.140320217752644" "500"
     l    model  phi_mean           phi_SD
[1,] "92" "EGSS" "0.825983233532934" "0.137832794108836" "500"
     l    model  phi_mean           phi_SD
[1,] "93" "EGSS" "0.815964750499002" "0.144741603712934" "500"
     l    model  phi_mean           phi_SD
[1,] "94" "EGSS" "0.827703552894212" "0.135275603452382" "500"
     l    model  phi_mean           phi_SD
[1,] "95" "EGSS" "0.846463552894212" "0.121276786669318" "500"
```

Figure SI-37: Iterating all

...

```
[1,] "347" "EGSS" "0.327647864271457" "0.320445153268953" "500"
     l     model  phi_mean           phi_SD
[1,] "348" "EGSS" "0.395122395209581" "0.327522032970388" "500"
     l     model  phi_mean           phi_SD
[1,] "349" "EGSS" "0.476552375249501" "0.324731276677594" "500"
     l     model  phi_mean           phi_SD
[1,] "350" "EGSS" "0.529132894211577" "0.318205289232656" "500"
     l     model  phi_mean           phi_SD
[1,] "351" "EGSS" "0.514361596806387" "0.322353211779331" "500"
> #time process finished
> EndTime <- Sys.time()
> #difference of time (time lasted in the process)
> timelast <- EndTime-StartTime
> timelast
Time difference of 10.0862 hours
```

Figure SI-38: Iterating all

This is a longer process ($\sim$ 10 hours; currently is `{r eval=FALSE}`), so we saved the `phi_results` (with `phi_mean` and `phi_SD`), and `modelSS` vectors as data frames to make a plot.

```r
#creating data frames for phi_mean, phi_SD, and modelSS
phi_eBird_df <- do.call(rbind, lapply(seq_along(phi_results), function(j) {
  do.call(rbind, lapply(seq_along(phi_results[[j]]), function(i) {
    data.frame(
      Time.t = tt[end_positions[j]],
      Timeframe = timeframes[i],
      phi.hat = phi_results[[j]][[i]][1],
      SD.phi = phi_results[[j]][[i]][2],
      Model = modelSS[[j]][[i]]
    )
  }))
}))

# Convert to factors for better plotting
phi_eBird_df$Timeframe <- factor(phi_eBird_df$Timeframe, levels = timeframes)
head(phi_eBird_df, n = 12)

saveRDS(phi_eBird_df, "results/phi_hat_SD_eBird.rds")
```

```
> head(phi_eBird_df, n = 12)
   Time.t           Timeframe   phi.hat    SD.phi Model
1      88  150 weeks or ~3 years 0.5852791 0.2343896  EGSS
2      88  250 weeks or ~5 years 0.7257916 0.1886146  EGSS
3      88 500 weeks or ~10 years 0.8443608 0.1265831  EGSS
4      89  150 weeks or ~3 years 0.4711426 0.2601445  EGSS
5      89  250 weeks or ~5 years 0.6397819 0.2294043  EGSS
6      89 500 weeks or ~10 years 0.7943913 0.1609917  EGSS
7      90  150 weeks or ~3 years 0.5275387 0.2488712  EGSS
8      90  250 weeks or ~5 years 0.6858692 0.2087568  EGSS
9      90 500 weeks or ~10 years 0.8215439 0.1411915  EGSS
10     91  150 weeks or ~3 years 0.5285050 0.2474488  EGSS
11     91  250 weeks or ~5 years 0.6853514 0.2065132  EGSS
12     91 500 weeks or ~10 years 0.8227917 0.1403202  EGSS
```

Figure SI-39: head screenshot of eBird phi estimations

5.4.4 Iterate the process with Standardized Monitoring data (figure examples)

Let iterates for the periods that share data in eBird and Standardized Monitored (data available in standardized monitored)

```
#Standardized monitored data
#Log observations for the entire dataset
ytSM = snailkites.PP |>
  ungroup() |>
  arrange(Time.t) |>
  drop_na(abundance.monitored) |>
  dplyr::select(abundance.monitored) |>
  log()

#Vector form
ytSM <- ytSM$abundance.monitored
ytSM
```

```
##  [1] 0.0000000 0.6931472 1.3862944 1.7917595 2.0794415 1.7917595 1.7917595
##  [8] 2.4849066 1.9459101 3.3672958 3.5835189 3.6888795 3.9702919 3.8286414
## [15] 3.8918203 4.1896547 4.3567088 4.7095302 4.8040210 5.0172798 4.8040210
## [22] 4.7184989 4.1431347 4.4886364 4.4067192 3.9702919 2.4849066 2.7725887
## [29] 3.7135721 3.9120230 4.3438054 4.3307333 4.5849675 4.5108595 4.7273878
## [36] 4.6821312 5.5093883 4.8903491 5.0751738 4.6443909 4.7449321 4.6634391
## [43] 4.2626799 3.8501476 3.7376696 4.2195077 1.7917595 1.3862944 0.6931472
```

```
#N crticial
N.criticalSM <- round((1/2)*mean(exp(ytSM)),0)

ttSM <- snailkites.PP |>
  ungroup() |>
  arrange(Time.t) |>
  drop_na(abundance.monitored) |>
  dplyr::select(Time.t)

#Time vector of the time series
ttSM <- ttSM$Time.t
ttSM
```

```
##  [1]   8  11  15  22  23  29  32  35  42  51  64  68  72  74  77  81  95 112 116
## [20] 118 122 124 128 129 148 164 168 170 211 215 219 222 225 228 231 234 251 263
## [39] 267 271 274 277 280 283 286 303 316 320 324
```

And run the model

```
#This will take ~2 hours, consider change `{r eval=FALSE}`

ntraj =50000

# End positions to modeling (steps)
end_positions <- which(ttSM >= 88)

#actual weeks
ttSM[end_positions]

#save   (SD)
phi_results <- vector("list", length = length(end_positions))
```

```r
#save model used
modelSS <- vector("list", length = length(end_positions))

# Set the base filename for figures
base_filename <- "supporting/FigS5_IterateTrajSM_10_5"

# Start process timing
StartTime <- Sys.time()

for (j in seq_along(end_positions)) {

  last.tt <- end_positions[j]
  l <- ttSM[last.tt]

  for (i in seq_along(len.sim)){
      # Set up plot for all three timeframes
  png_filename <- paste0(base_filename,
                         "_endpos_",l,
                         "_timeframe_",(len.sim[i]-1),
                         ".png")
  png(png_filename, width = 10, height = 5, units = "in", res = 300)


    plot(ttSM[1:last.tt],
         exp(ytSM[1:last.tt]),
         type="b", lwd=2, cex.lab=1.25, col = "#d73027", pch = 1,
         xlim=c(min(tt), (max(tt)*3)),
         ylim=c(0,250),
         ylab="Observed/predicted weekly high counts",
         xlab="Time (week since January 2018)");

    OUSS.partial <- ouss_reml(yt = ytSM[1:last.tt],
                              tt = ttSM[1:last.tt],
                              fguess = guess_ouss(yt = ytSM[1:last.tt],
                                                  tt = ttSM[1:last.tt]))

    model <- if(OUSS.partial$remles[2] < 0.025){
      "EGSS"
    }else{
      "OUSS"
    }
    modelSS[[j]][[timeframes[i]]] <- model

    if(model == "OUSS"){
      parcial.predict <- ouss_predict(yt = ytSM[1:last.tt],
                                      tt = ttSM[1:last.tt],
                                      parms = OUSS.partial$remles,
                                      plot.it = F)

      points(x = (parcial.predict)[[1]][,1],
             y = (parcial.predict)[[1]][,2],
             type = "b", col = "#fc8d59", pch = 2)
```

```
    thres.times <- as.numeric(0:(len.sim[i]-1))
    len <- max(thres.times) + 1

    sim.mat.eBird <- ouss_sim(ntraj,
                              tt = thres.times,
                              parms = OUSS.partial$remles)

    phi <- rep(0, ntraj)
    last.points <- rep(0, ntraj)

    for(n in 1:ntraj){
      Pop.sim <- exp(c(ytSM[last.tt], sim.mat.eBird[-1,n]));
      last.points[n] <- Pop.sim[len]

    # How many times each trajectory go below the threshold?
      below.threshold <- sum(Pop.sim < N.criticalSM)
      phi[n] <- 1-(below.threshold/len)

      if(phi[n] >= 0.5){
        lines(l+thres.times, Pop.sim, col="#d3d3d308", lty = "solid")
        } else {
          lines(l+thres.times, Pop.sim, col="#FF000008", lty = "solid")
          }
  }
    # Add critical value line
    abline(h=N.criticalSM, lty=2, lwd=1);

    #Expected value of probability of local persistence, and SD
    phi_mean <- mean(phi)
    phi_SD <- sqrt(var(phi))

    phi_results[[j]][[timeframes[i]]] <- cbind(phi_mean,phi_SD)

    # Create a kernel density estimate (KDE) of the `last.points` results (library(ke31d))
    kde.sims <- kde1d(x=last.points);

    # Add the distribution to the plot
    lines(x = (kde.sims$values*(ntraj/5) + l + last(thres.times)),
          y = kde.sims$grid_points,
          col="#d3d3d398", lwd=2, lty=1)

    lines(x = (kde.sims$values[kde.sims$grid_points <= N.criticalSM]*(ntraj/5) +
                 l + last(thres.times)),
          y = kde.sims$grid_points[kde.sims$grid_points <= N.criticalSM],
          col="#FF000098", lwd=2, lty=1)

    title(main=paste0("  = ",
                      signif(phi_mean,2),
                      " (SD: ",
                      signif(phi_SD,2),
                      "); ", model, "; through week ", ttSM[last.tt],
                      ", projected for ~",
                      signif((len.sim[i]-1)/52, 2),
```

```
                    " year(s)"),
       cex=1.5)
}

else{

  EGSS.partial <- egss_reml(yt = ytSM[1:last.tt],
                            tt = ttSM[1:last.tt],
                            fguess = guess_egss(yt = ytSM[1:last.tt],
                                                tt = ttSM[1:last.tt]));
  parcial.predict <- egss_predict(yt = ytSM[1:last.tt],
                                  tt = ttSM[1:last.tt],
                                  parms = EGSS.partial$remles,
                                  plot.it = F)

  points(x = (parcial.predict)[[1]][,1],
         y = (parcial.predict)[[1]][,2],
         type = "b", col = "#fc8d59", pch = 2)

  thres.times <- as.numeric(0:(len.sim[i]-1))
  len <- max(thres.times) + 1

  sim.mat.eBird <- egss_sim(ntraj,
                            tt = thres.times,
                            parms = EGSS.partial$remles)

  phi <- rep(0, ntraj)
  last.points <- rep(0, ntraj)

  for(n in 1:ntraj){
    Pop.sim <- exp(c(ytSM[last.tt], sim.mat.eBird[-1,n]));
    last.points[n] <- Pop.sim[len]

  # How many times each trajectory go below the threshold?
    below.threshold <- sum(Pop.sim < N.criticalSM)
    phi[n] <- 1-(below.threshold/len)

    if(phi[n] >= 0.5){
      lines(l+thres.times, Pop.sim, col="#d3d3d308", lty = "solid")
      } else {
        lines(l+thres.times, Pop.sim, col="#FF000008", lty = "solid")
        }
}
  # Add critical value line
  abline(h=N.criticalSM, lty=2, lwd=1);

  #Expected value of probability of local persistence, and SD
  phi_mean <- mean(phi)
  phi_SD <- sqrt(var(phi))

  phi_results[[j]][[timeframes[i]]] <- cbind(phi_mean,phi_SD)

  # Create a kernel density estimate (KDE) of the `last.points` results (library(ke31d))
```

```
      kde.sims <- kde1d(x=last.points);

      # Add the distribution to the plot
      lines(x = (kde.sims$values*(ntraj/5) + l + last(thres.times)),
            y = kde.sims$grid_points,
            col="#d3d3d398", lwd=2, lty=1)

      lines(x = (kde.sims$values[kde.sims$grid_points <= N.criticalSM]*(ntraj/5) +
                   l + last(thres.times)),
            y = kde.sims$grid_points[kde.sims$grid_points <= N.criticalSM],
            col="#FF000098", lwd=2, lty=1)

      title(main=paste0("  = ",
                        signif(phi_mean,2),
                        " (SD: ",
                        signif(phi_SD,2),
                        "); ", model, "; through week ", ttSM[last.tt],
                        ", projected for ~",
                        signif((len.sim[i]-1)/52, 2),
                        " year(s)"),
            cex=1.5)
    }

    dev.off()

  }
  print(cbind(l, model,
              phi_mean,
              phi_SD,
              len.sim[i]-1))
}
#time process finished
EndTime <- Sys.time()
#difference of time (time lasted in the process)
timelast <- EndTime-StartTime
timelast
```

```
     l    model  phi_mean           phi_SD
[1,] "95" "EGSS" "0.858389860279441" "0.0799620981115019" "500"
     l     model  phi_mean           phi_SD
[1,] "112" "EGSS" "0.844352854291417" "0.0837005626598 67" "500"
     l     model  phi_mean           phi_SD
[1,] "116" "EGSS" "0.840836047904192" "0.0846694371562266" "500"
     l     model  phi_mean           phi_SD
[1,] "118" "EGSS" "0.844634650698603" "0.0850581262080492" "500"
     l     model  phi_mean           phi_SD
[1,] "122" "EGSS" "0.834552614770459" "0.0875630876208279" "500"
```

Figure SI-40: head screenshot of eBird phi estimations

```
     l     model  phi_mean           phi_SD
[1,] "320" "EGSS" "0.244276526946108" "0.288755858167907" "500"
     l     model  phi_mean           phi_SD
[1,] "324" "EGSS" "0.204686147704591" "0.273290275785204" "500"
> #time process finished
> EndTime <- Sys.time()
> #difference of time (time lasted in the process)
> timelast <- EndTime-StartTime
> timelast
Time difference of 4.569424 hours
```

Figure SI-41: head screenshot of eBird phi estimations

Now we save the `phi_results` and `modelSS` vectors as data frames to make a plot.

```r
#creating data frames for phi_mean, phi_SD, and modelSS
phi_SM_df <- do.call(rbind, lapply(seq_along(phi_results), function(j) {
  do.call(rbind, lapply(seq_along(phi_results[[j]]), function(i) {
    data.frame(
      Time.t = ttSM[end_positions[j]],
      Timeframe = timeframes[i],
      phi.hatSM = phi_results[[j]][[i]][1],
      SD.phiSM = phi_results[[j]][[i]][2],
      Model = modelSS[[j]][[i]]
    )
  }))
}))

# Convert to factors for better plotting
phi_SM_df$Timeframe <- factor(phi_SM_df$Timeframe, levels = timeframes)
head(phi_SM_df, n = 12)

saveRDS(phi_SM_df, "results/phi_hat_SD_SM.rds")
```

5.4.4.1 Trajectories in timeframe of ~3 years.

Let's see the examples.

How is changing in 3 moments $\hat{\phi}_{SM\ \sim three\ years\ simulated}$?

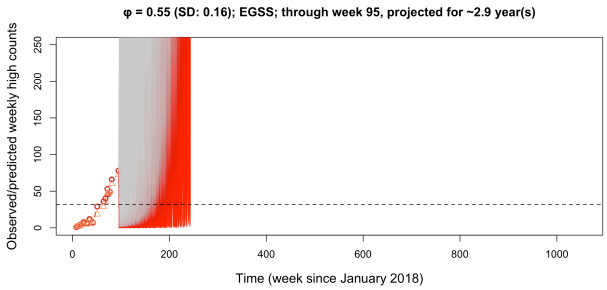


Figure SI-42: Standardized monitoring - week 95 - October 2019 simulated for ~3 years

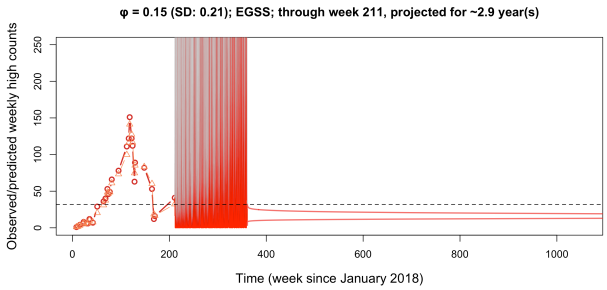


Figure SI-43: Standardized monitoring - week 211 - January 2022 simulated for ~3 years

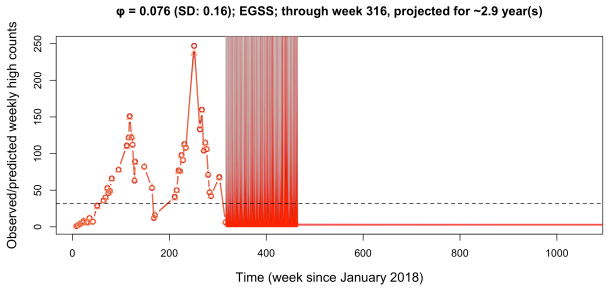


Figure SI-44: Standardized monitoring - week 316 - January 2024 simulated for ~3 years

5.4.4.2 Trajectories in timeframe of ~5 years.

How is changing in 3 moments $\hat{\phi}_{SM\ \sim five\ years\ simulated}$?

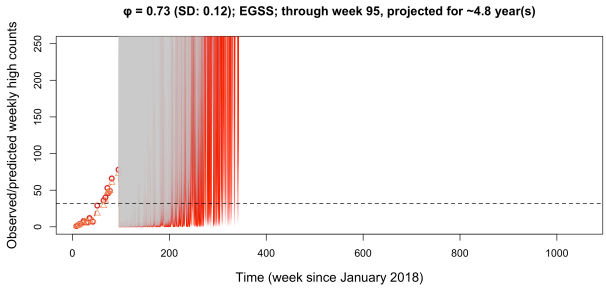


Figure SI-45: Standardized monitoring - week 95 - October 2019 simulated for ~5 years

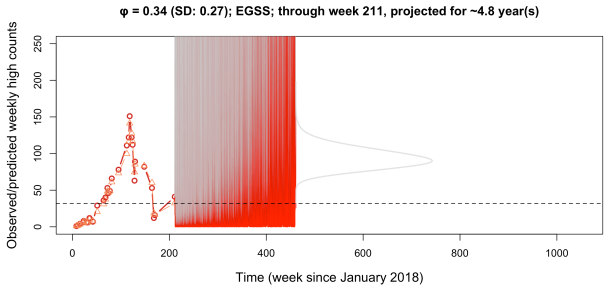


Figure SI-46: Standardized monitoring - week 211 - January 2022 simulated for ~5 years

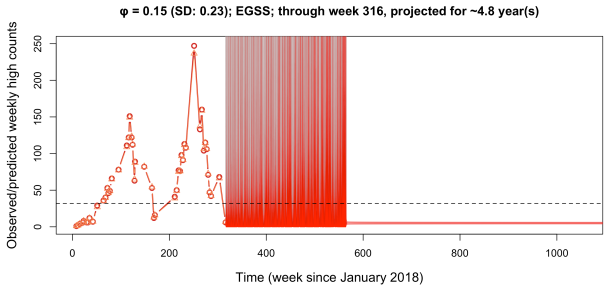


Figure SI-47: Standardized monitoring - week 316 - January 2024 simulated for ~5 years

5.4.4.3 Trajectories in timeframe of ~10 years.

How is changing in 3 moments $\hat{\phi}_{SM\ \sim ten\ years\ simulated}$?

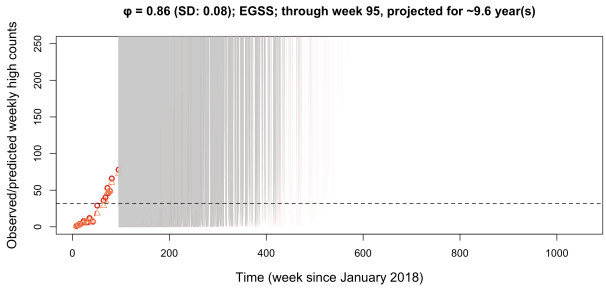


Figure SI-48: Standardized monitoring - week 95 - October 2019 simulated for ~10 years

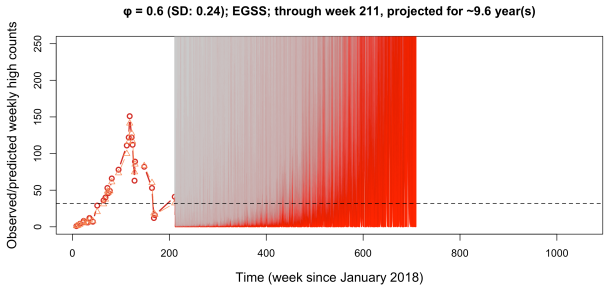


Figure SI-49: Standardized monitoring - week 211 - January 2022 simulated for ~10 years

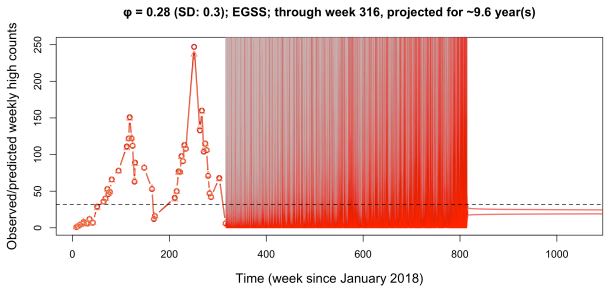


Figure SI-50: Standardized monitoring - week 316 - January 2024 simulated for ~10 years

6 Results - $\hat{\phi}$

Call the data and establish transformation of second Y-axis (supplementary figure to include time series counts and ϕ).

```
#Load data saved
  #dates
  datesPP <- readRDS("data_tmp/datesTimeseriesPaynesPrairie.rds")
  #data organized
  snailkites.PP <- readRDS("data_tmp/snailkitesPP.rds")
  #local persistence from eBird - OUSS or EGSS
  phi.eBird.OUSS.EGSS <- readRDS("results/phi_hat_SD_eBird_OUSS_EGSS.rds")
  #local persistence from eBird - EGSS
  phi.eBird <- readRDS("results/phi_hat_SD_eBird.rds")
  #local persistence from Standardized Monitored
  phi.SM <-readRDS("results/phi_hat_SD_SM.rds")

  N.critical <- 6 # 1/2*empirical K = 1/2 mean(exp(yt))
  N.criticalSM <- 32 # 1/2*empirical K = 1/2 mean(exp(ytSM))

#coefficient to convert second Y-axis
coeff <- 100
coeffSM <- 250
```

We can compare the overlap of ϕ estimation. First we combine the `phi.eBird` and `phi.SM` results.

```
head(phi.eBird)
```

```
##   Time.t              Timeframe   phi.hat    SD.phi Model
## 1     88  150 weeks or ~3 years 0.5852791 0.2343896  EGSS
## 2     88  250 weeks or ~5 years 0.7257916 0.1886146  EGSS
## 3     88 500 weeks or ~10 years 0.8443608 0.1265831  EGSS
## 4     89  150 weeks or ~3 years 0.4711426 0.2601445  EGSS
## 5     89  250 weeks or ~5 years 0.6397819 0.2294043  EGSS
## 6     89 500 weeks or ~10 years 0.7943913 0.1609917  EGSS
```

```
head(phi.eBird.OUSS.EGSS)
```

```
##   Time.t              Timeframe   phi.hat    SD.phi Model
## 1     88  150 weeks or ~3 years 0.5862160 0.2360549  EGSS
## 2     88  250 weeks or ~5 years 0.7263161 0.1894415  EGSS
## 3     88 500 weeks or ~10 years 0.8445463 0.1266171  EGSS
## 4     89  150 weeks or ~3 years 0.4708873 0.2599956  EGSS
## 5     89  250 weeks or ~5 years 0.6395306 0.2297969  EGSS
## 6     89 500 weeks or ~10 years 0.7923436 0.1617682  EGSS
```

```
head(phi.SM)
```

```
##   Time.t              Timeframe phi.hatSM   SD.phiSM Model
## 1     95  150 weeks or ~3 years 0.5504385 0.15961408  EGSS
## 2     95  250 weeks or ~5 years 0.7258154 0.12059628  EGSS
## 3     95 500 weeks or ~10 years 0.8583899 0.07996210  EGSS
## 4    112  150 weeks or ~3 years 0.5058521 0.16321447  EGSS
## 5    112  250 weeks or ~5 years 0.6982037 0.12548544  EGSS
## 6    112 500 weeks or ~10 years 0.8443529 0.08370056  EGSS
```

```r
#combine phi dataframes
phi_combined <- phi.eBird |>
  bind_rows(.id = "dataset", phi.eBird.OUSS.EGSS) |>
  mutate(dataset = case_when(dataset == 1~"eBird.EGSS",
                             dataset != 1~"eBird.OUSS.EGSS")) |>
  bind_rows(.id = "original.dataset", phi.SM) |>
  mutate(original.dataset = case_when(original.dataset == 1~"eBird",
                             original.dataset != 1~"SM"),
         dataset = case_when(original.dataset == "SM"~"SM",
                                  original.dataset != "SM"~dataset))

head(phi_combined)
```

```
##   original.dataset    dataset Time.t              Timeframe   phi.hat    SD.phi
## 1            eBird eBird.EGSS     88  150 weeks or ~3 years 0.5852791 0.2343896
## 2            eBird eBird.EGSS     88  250 weeks or ~5 years 0.7257916 0.1886146
## 3            eBird eBird.EGSS     88 500 weeks or ~10 years 0.8443608 0.1265831
## 4            eBird eBird.EGSS     89  150 weeks or ~3 years 0.4711426 0.2601445
## 5            eBird eBird.EGSS     89  250 weeks or ~5 years 0.6397819 0.2294043
## 6            eBird eBird.EGSS     89 500 weeks or ~10 years 0.7943913 0.1609917
##   Model phi.hatSM SD.phiSM
## 1  EGSS        NA       NA
## 2  EGSS        NA       NA
## 3  EGSS        NA       NA
## 4  EGSS        NA       NA
## 5  EGSS        NA       NA
## 6  EGSS        NA       NA
```

Then, we add the dates and filter by the three Timeframes

```r
#add dates while filtering Timeframe
phi_data_plot <- datesPP |>
  left_join(phi_combined |>
              filter(Timeframe == "150 weeks or ~3 years")) |>
  ungroup() |>
  pivot_longer(cols = !c(Timeframe,
                         Time.t,
                         observation.date,
                         Model,
                         original.dataset,
                         dataset),
               names_to = "group",
               values_to = "phi_values") |>
  drop_na(phi_values)
```

```
## Joining with `by = join_by(Time.t)`
```

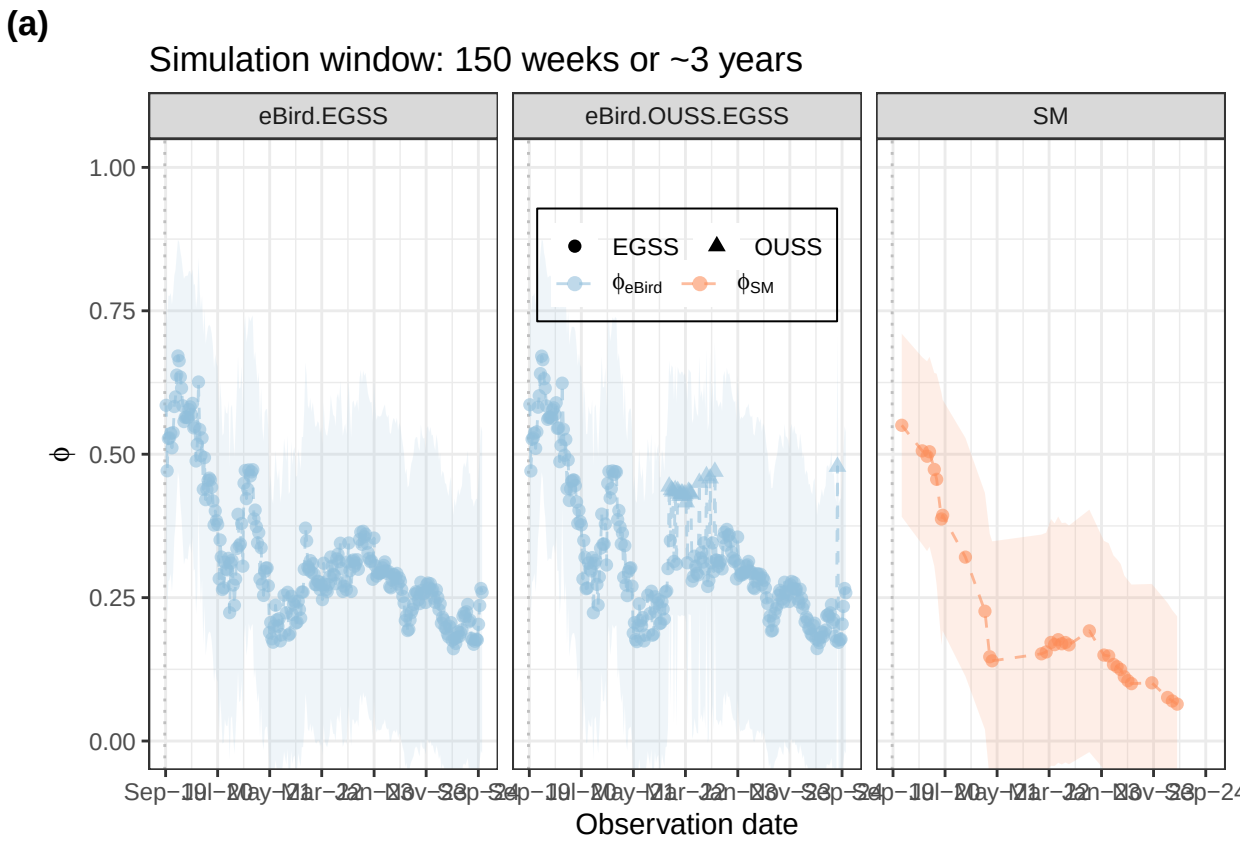
```r
phi_data_plot
```

```
## # A tibble: 1,114 x 8
##    Time.t observation.date original.dataset dataset        Timeframe Model group
##     <dbl> <date>           <chr>            <chr>          <fct>     <chr> <chr>
##  1     88 2019-09-04       eBird            eBird.EGSS     150 week~ EGSS  phi.~
##  2     88 2019-09-04       eBird            eBird.EGSS     150 week~ EGSS  SD.p~
##  3     88 2019-09-04       eBird            eBird.OUSS.EG~ 150 week~ EGSS  phi.~
##  4     88 2019-09-04       eBird            eBird.OUSS.EG~ 150 week~ EGSS  SD.p~
```

```
##  5     89 2019-09-10       eBird            eBird.EGSS     150 week~ EGSS  phi.~
##  6     89 2019-09-10       eBird            eBird.EGSS     150 week~ EGSS  SD.p~
##  7     89 2019-09-10       eBird            eBird.OUSS.EG~ 150 week~ EGSS  phi.~
##  8     89 2019-09-10       eBird            eBird.OUSS.EG~ 150 week~ EGSS  SD.p~
##  9     90 2019-09-17       eBird            eBird.EGSS     150 week~ EGSS  phi.~
## 10     90 2019-09-17       eBird            eBird.EGSS     150 week~ EGSS  SD.p~
## # i 1,104 more rows
## # i 1 more variable: phi_values <dbl>
```

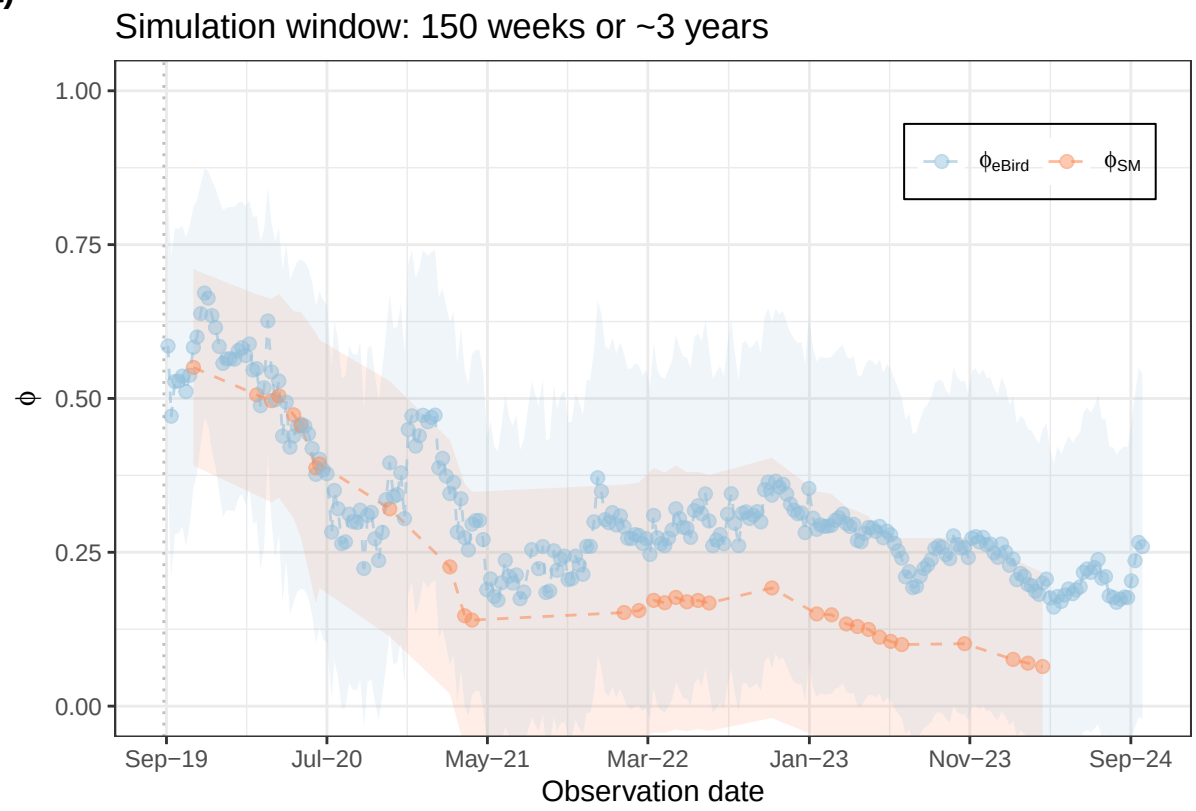
```
#Ribbon lo-hi phi (SD) for the two datasets
#eBird EGSS
ribbon_phi_eBird <- phi_data_plot |>
  filter(dataset == "eBird.EGSS",
         group %in% c("phi.hat", "SD.phi")) |>
  pivot_wider(names_from = group, values_from = phi_values, values_fn = {mean}) |>
  mutate(phi_values = phi.hat)

#eBird OUSS vs EGSS
ribbon_phi_eBird_OUSSEGSS <- phi_data_plot |>
  filter(dataset == "eBird.OUSS.EGSS",
         group %in% c("phi.hat", "SD.phi")) |>
  pivot_wider(names_from = group, values_from = phi_values, values_fn = {mean}) |>
  mutate(phi_values = phi.hat)

ribbon_phi_SM <- phi_data_plot |>
  filter(dataset == "SM",
         group %in% c("phi.hatSM", "SD.phiSM")) |>
  pivot_wider(names_from = group, values_from = phi_values, values_fn = {mean})|>
  mutate(phi_values = phi.hatSM)

#three trends
FigOUSS_A <- ggplot(phi_data_plot,
       aes(x = observation.date, y = phi_values)) +
  facet_grid(~dataset)+
  geom_vline(xintercept = snailkites.PP$observation.date[78],
             color = "gray", linetype = "dotted") +
  geom_ribbon(data = ribbon_phi_SM,
              aes(ymin = phi_values+SD.phiSM,
                  ymax = phi_values-SD.phiSM),
              fill = "#fc8d5925") +
  geom_ribbon(data = ribbon_phi_eBird,
              aes(ymin = phi_values+SD.phi,
                  ymax = phi_values-SD.phi),
              fill = "#91bfdb25")+
  geom_ribbon(data = ribbon_phi_eBird_OUSSEGSS,
              aes(ymin = phi_values+SD.phi,
                  ymax = phi_values-SD.phi),
              fill = "#91bfdb25")+
  geom_line(data = phi_data_plot |>
              filter(group %in% c("phi.hat", "phi.hatSM")),
            aes(color = group), linetype = "dashed") +
  geom_point(data = phi_data_plot |>
              filter(group %in% c("phi.hat", "phi.hatSM")),
             aes(shape = Model,
```

```
                color = group),
            size = 2) +
  labs(x = "Observation date",
       y = expression(italic(phi)),
       tag = expression(bold("(a)")),
       title = "Simulation window: 150 weeks or ~3 years")  +
  coord_cartesian(ylim = c(0, 1))+
  scale_x_date(limits = c(snailkites.PP$observation.date[78],
                          max(snailkites.PP$observation.date)),
               breaks = seq(as.Date("2018-01-01"), as.Date(Sys.Date( )),
                            by = "10 months"), date_labels="%b-%y")+
  scale_color_manual(values = c("#91bfdb95",
                                "#fc8d5995"),
                     labels = c(expression(italic(phi)[eBird]),
                                expression(italic(phi)[SM])))+
  theme_bw() +
  theme(legend.title = element_blank(),
        legend.position = c(0.5, 0.8),
        legend.spacing.y = unit(-0.5, "cm"),
        legend.background = element_blank(),
        legend.box.background = element_rect(colour = "black"))+
  guides(color = guide_legend(ncol=2),
         shape = guide_legend(ncol=2))

FigOUSS_A
```



```r
#Only EGSS comparison

#Figure overlap phi
Figure4a <- ggplot(phi_data_plot,
                   aes(x = observation.date, y = phi_values)) +
  geom_vline(xintercept = snailkites.PP$observation.date[78],
             color = "gray", linetype = "dotted") +
  geom_ribbon(data = ribbon_phi_SM,
              aes(ymin = phi_values+SD.phiSM,
                  ymax = phi_values-SD.phiSM),
              fill = "#fc8d5925") +
  geom_ribbon(data = ribbon_phi_eBird,
              aes(ymin = phi_values+SD.phi,
                  ymax = phi_values-SD.phi),
              fill = "#91bfdb25") +
  geom_line(data = phi_data_plot |>
               filter(dataset != "eBird.OUSS.EGSS",
                      group %in% c("phi.hat", "phi.hatSM")),
            aes(color = group), linetype = "dashed") +
  geom_point(data = phi_data_plot |>
               filter(dataset != "eBird.OUSS.EGSS",
                      group %in% c("phi.hat", "phi.hatSM")),
             aes(color = group, fill = group),
             size = 2, shape = 21) +
  labs(x = "Observation date",
       y = expression(italic(phi)),
       tag = expression(bold("(a)")),
       title = "Simulation window: 150 weeks or ~3 years")  +
  coord_cartesian(ylim = c(0, 1))+
  scale_x_date(limits = c(snailkites.PP$observation.date[78],
                          max(snailkites.PP$observation.date)),
               breaks = seq(as.Date("2018-01-01"), as.Date(Sys.Date( )),
                            by = "10 months"), date_labels="%b-%y")+
  scale_color_manual(values = c("#91bfdb95",
                                "#fc8d5995"),
                     labels = c(expression(italic(phi)[eBird]),
                                expression(italic(phi)[SM])))+
  scale_fill_manual(values = c("#91bfdb75",
                                "#fc8d5975"),
                     labels = c(expression(italic(phi)[eBird]),
                                expression(italic(phi)[SM])))+
  theme_bw() +
  theme(legend.title = element_blank(),
        legend.position = c(0.85, 0.85),
        legend.spacing.y = unit(-0.5, "cm"),
        legend.background = element_blank(),
        legend.box.background = element_rect(colour = "black"))+
  guides(color = guide_legend(ncol=2),
         shape = guide_legend(ncol=2))

Figure4a
```

(a)




```
#add dates while filtering Timeframe
phi_data_plot <- datesPP |>
  left_join(phi_combined|> filter(Timeframe == "250 weeks or ~5 years")) |>
  ungroup() |>
  pivot_longer(cols = !c(Timeframe,
                         Time.t,
                         observation.date,
                         Model,
                         original.dataset,
                         dataset),
               names_to = "group",
               values_to = "phi_values") |>
  drop_na(phi_values)
```

```
## Joining with `by = join_by(Time.t)`
```

```
phi_data_plot
```

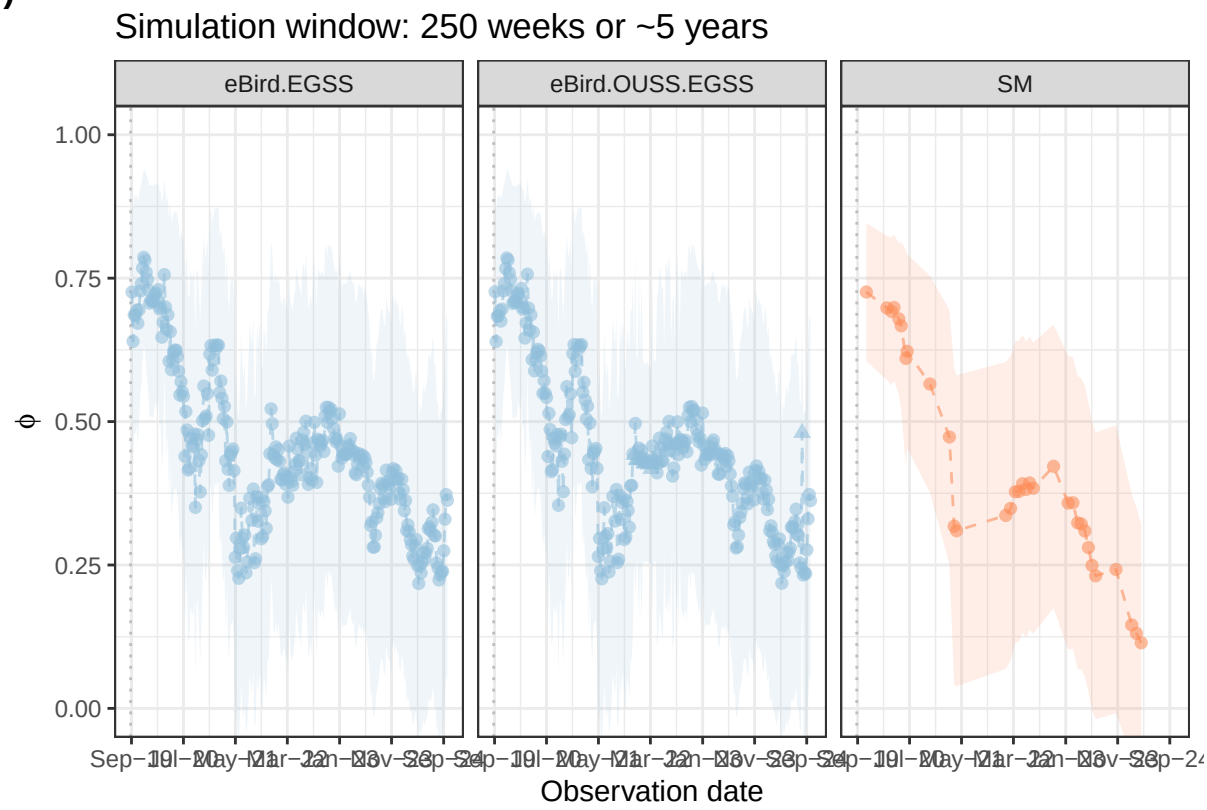
```
## # A tibble: 1,114 x 8
##    Time.t observation.date original.dataset dataset        Timeframe Model group
##     <dbl> <date>           <chr>            <chr>          <fct>     <chr> <chr>
##  1     88 2019-09-04       eBird            eBird.EGSS     250 week~ EGSS  phi.~
##  2     88 2019-09-04       eBird            eBird.EGSS     250 week~ EGSS  SD.p~
##  3     88 2019-09-04       eBird            eBird.OUSS.EG~ 250 week~ EGSS  phi.~
##  4     88 2019-09-04       eBird            eBird.OUSS.EG~ 250 week~ EGSS  SD.p~
##  5     89 2019-09-10       eBird            eBird.EGSS     250 week~ EGSS  phi.~
##  6     89 2019-09-10       eBird            eBird.EGSS     250 week~ EGSS  SD.p~
```

```
##  7     89 2019-09-10       eBird            eBird.OUSS.EG~ 250 week~ EGSS  phi.~
##  8     89 2019-09-10       eBird            eBird.OUSS.EG~ 250 week~ EGSS  SD.p~
##  9     90 2019-09-17       eBird            eBird.EGSS     250 week~ EGSS  phi.~
## 10     90 2019-09-17       eBird            eBird.EGSS     250 week~ EGSS  SD.p~
## # i 1,104 more rows
## # i 1 more variable: phi_values <dbl>
```

```
#Ribbon lo-hi phi (SD) for the two datasets
#eBird EGSS
ribbon_phi_eBird <- phi_data_plot |>
  filter(dataset == "eBird.EGSS",
         group %in% c("phi.hat", "SD.phi")) |>
  pivot_wider(names_from = group, values_from = phi_values, values_fn = {mean}) |>
  mutate(phi_values = phi.hat)

#eBird OUSS vs EGSS
ribbon_phi_eBird_OUSSEGSS <- phi_data_plot |>
  filter(dataset == "eBird.OUSS.EGSS",
         group %in% c("phi.hat", "SD.phi")) |>
  pivot_wider(names_from = group, values_from = phi_values, values_fn = {mean}) |>
  mutate(phi_values = phi.hat)

ribbon_phi_SM <- phi_data_plot |>
  filter(dataset == "SM",
         group %in% c("phi.hatSM", "SD.phiSM")) |>
  pivot_wider(names_from = group, values_from = phi_values, values_fn = {mean})|>
  mutate(phi_values = phi.hatSM)

#three trends
FigOUSS_B <- ggplot(phi_data_plot,
       aes(x = observation.date, y = phi_values)) +
  facet_grid(~dataset)+
  geom_vline(xintercept = snailkites.PP$observation.date[78],
             color = "gray", linetype = "dotted") +
  geom_ribbon(data = ribbon_phi_SM,
              aes(ymin = phi_values+SD.phiSM,
                  ymax = phi_values-SD.phiSM),
              fill = "#fc8d5925") +
  geom_ribbon(data = ribbon_phi_eBird,
              aes(ymin = phi_values+SD.phi,
                  ymax = phi_values-SD.phi),
              fill = "#91bfdb25")+
  geom_ribbon(data = ribbon_phi_eBird_OUSSEGSS,
              aes(ymin = phi_values+SD.phi,
                  ymax = phi_values-SD.phi),
              fill = "#91bfdb25")+
  geom_line(data = phi_data_plot |>
               filter(group %in% c("phi.hat", "phi.hatSM")),
            aes(color = group), linetype = "dashed") +
  geom_point(data = phi_data_plot |>
               filter(group %in% c("phi.hat", "phi.hatSM")),
             aes(shape = Model,
                 color = group),
             size = 2) +
```

```
  labs(x = "Observation date",
       y = expression(italic(phi)),
       tag = expression(bold("(b)")),
       title = "Simulation window: 250 weeks or ~5 years")  +
  coord_cartesian(ylim = c(0, 1))+
  scale_x_date(limits = c(snailkites.PP$observation.date[78],
                          max(snailkites.PP$observation.date)),
               breaks = seq(as.Date("2018-01-01"), as.Date(Sys.Date( )),
                            by = "10 months"), date_labels="%b-%y")+
  scale_color_manual(values = c("#91bfdb95",
                                "#fc8d5995"),
                     labels = c(expression(italic(phi)[eBird]),
                                expression(italic(phi)[SM])))+
  theme_bw() +
  theme(legend.title = element_blank(),
        legend.position = "none",
        legend.spacing.y = unit(-0.5, "cm"),
        legend.background = element_blank(),
        legend.box.background = element_rect(colour = "black"))+
  guides(color = guide_legend(ncol=2),
         shape = guide_legend(ncol=2))
FigOUSS_B
```

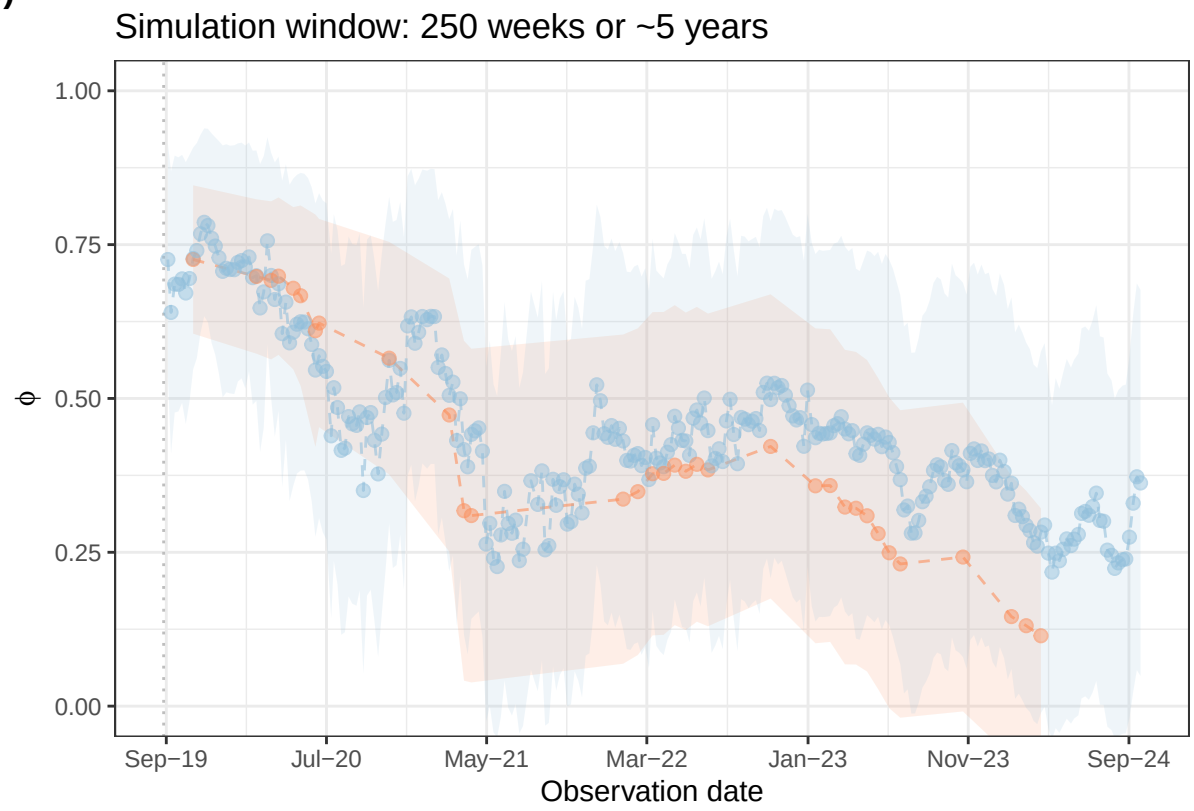
(b)




```
#Only EGSS comparison

#Figure overlap phi
```

```r
Figure4b <- ggplot(phi_data_plot,
                   aes(x = observation.date, y = phi_values)) +
  geom_vline(xintercept = snailkites.PP$observation.date[78],
             color = "gray", linetype = "dotted") +
  geom_ribbon(data = ribbon_phi_SM,
              aes(ymin = phi_values+SD.phiSM,
                  ymax = phi_values-SD.phiSM),
              fill = "#fc8d5925") +
  geom_ribbon(data = ribbon_phi_eBird,
              aes(ymin = phi_values+SD.phi,
                  ymax = phi_values-SD.phi),
              fill = "#91bfdb25") +
  geom_line(data = phi_data_plot |>
              filter(dataset != "eBird.OUSS.EGSS",
                     group %in% c("phi.hat", "phi.hatSM")),
            aes(color = group), linetype = "dashed") +
  geom_point(data = phi_data_plot |>
              filter(dataset != "eBird.OUSS.EGSS",
                     group %in% c("phi.hat", "phi.hatSM")),
             aes(color = group, fill = group),
             size = 2, shape = 21) +
  labs(x = "Observation date",
       y = expression(italic(phi)),
       tag = expression(bold("(b)")),
       title = "Simulation window: 250 weeks or ~5 years",
       color = "Dataset",
       fill = "Dataset")  +
  coord_cartesian(ylim = c(0, 1))+
  scale_x_date(limits = c(snailkites.PP$observation.date[78],
                          max(snailkites.PP$observation.date)),
               breaks = seq(as.Date("2018-01-01"), as.Date(Sys.Date( )),
                            by = "10 months"), date_labels="%b-%y")+
  scale_color_manual(values = c("#91bfdb95",
                                "#fc8d5995"),
                     labels = c(expression(italic(phi)[eBird]),
                                expression(italic(phi)[SM])))+
  scale_fill_manual(values = c("#91bfdb75",
                                "#fc8d5975"),
                     labels = c(expression(italic(phi)[eBird]),
                                expression(italic(phi)[SM])))+
  theme_bw() +
  theme(legend.position = "none")

Figure4b
```

(b)




```
#add dates while filtering Timeframe
phi_data_plot <- datesPP |>
  left_join(phi_combined|> filter(Timeframe == "500 weeks or ~10 years")) |>
  ungroup() |>
  pivot_longer(cols = !c(Timeframe,
                         Time.t,
                         observation.date,
                         Model,
                         original.dataset,
                         dataset),
               names_to = "group",
               values_to = "phi_values") |>
  drop_na(phi_values)
```

```
## Joining with `by = join_by(Time.t)`
```

```
phi_data_plot
```

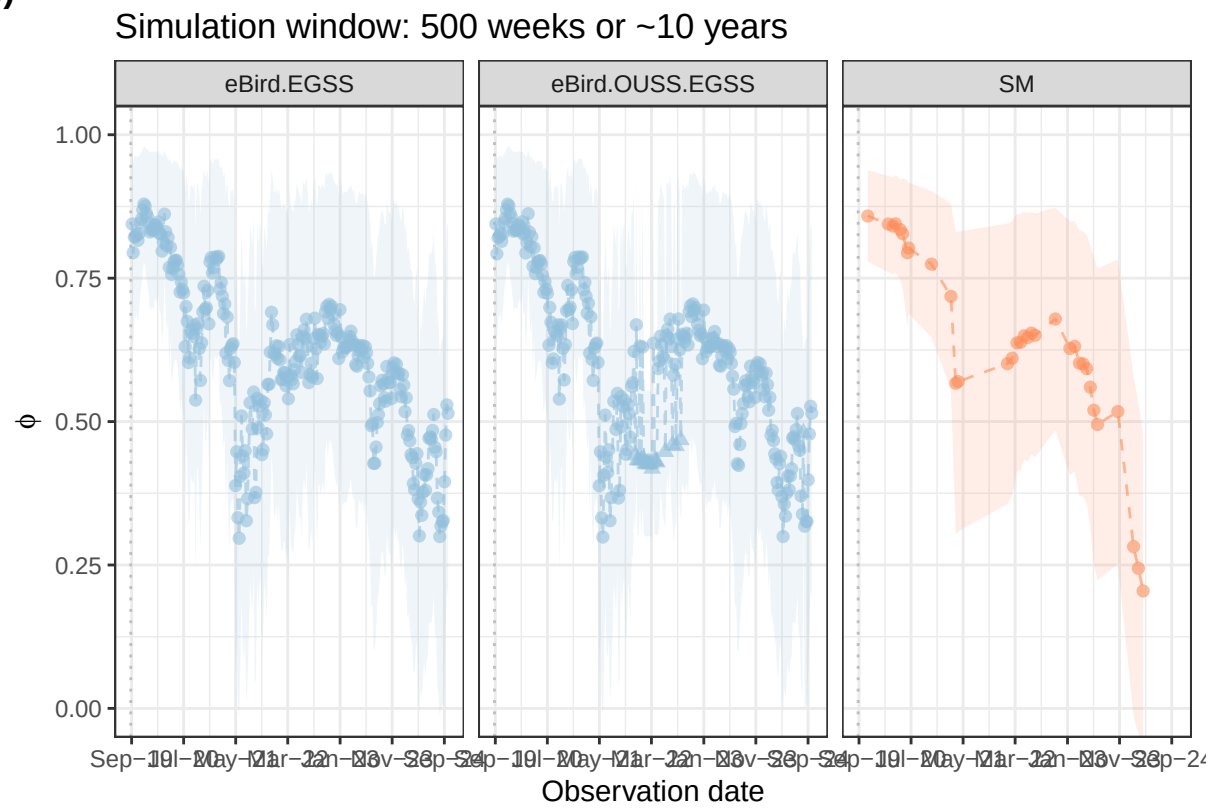
```
## # A tibble: 1,114 x 8
##    Time.t observation.date original.dataset dataset        Timeframe Model group
##     <dbl> <date>           <chr>            <chr>          <fct>     <chr> <chr>
##  1     88 2019-09-04       eBird            eBird.EGSS     500 week~ EGSS  phi.~
##  2     88 2019-09-04       eBird            eBird.EGSS     500 week~ EGSS  SD.p~
##  3     88 2019-09-04       eBird            eBird.OUSS.EG~ 500 week~ EGSS  phi.~
##  4     88 2019-09-04       eBird            eBird.OUSS.EG~ 500 week~ EGSS  SD.p~
##  5     89 2019-09-10       eBird            eBird.EGSS     500 week~ EGSS  phi.~
##  6     89 2019-09-10       eBird            eBird.EGSS     500 week~ EGSS  SD.p~
```

```
##  7     89 2019-09-10       eBird            eBird.OUSS.EG~ 500 week~ EGSS  phi.~
##  8     89 2019-09-10       eBird            eBird.OUSS.EG~ 500 week~ EGSS  SD.p~
##  9     90 2019-09-17       eBird            eBird.EGSS     500 week~ EGSS  phi.~
## 10     90 2019-09-17       eBird            eBird.EGSS     500 week~ EGSS  SD.p~
## # i 1,104 more rows
## # i 1 more variable: phi_values <dbl>
```

```r
#Ribbon lo-hi phi (SD) for the two datasets
#eBird EGSS
ribbon_phi_eBird <- phi_data_plot |>
  filter(dataset == "eBird.EGSS",
         group %in% c("phi.hat", "SD.phi")) |>
  pivot_wider(names_from = group, values_from = phi_values, values_fn = {mean}) |>
  mutate(phi_values = phi.hat)

#eBird OUSS vs EGSS
ribbon_phi_eBird_OUSSEGSS <- phi_data_plot |>
  filter(dataset == "eBird.OUSS.EGSS",
         group %in% c("phi.hat", "SD.phi")) |>
  pivot_wider(names_from = group, values_from = phi_values, values_fn = {mean}) |>
  mutate(phi_values = phi.hat)

ribbon_phi_SM <- phi_data_plot |>
  filter(dataset == "SM",
         group %in% c("phi.hatSM", "SD.phiSM")) |>
  pivot_wider(names_from = group, values_from = phi_values, values_fn = {mean})|>
  mutate(phi_values = phi.hatSM)

#three trends
FigOUSS_C <- ggplot(phi_data_plot,
       aes(x = observation.date, y = phi_values)) +
  facet_grid(~dataset)+
  geom_vline(xintercept = snailkites.PP$observation.date[78],
             color = "gray", linetype = "dotted") +
  geom_ribbon(data = ribbon_phi_SM,
              aes(ymin = phi_values+SD.phiSM,
                  ymax = phi_values-SD.phiSM),
              fill = "#fc8d5925") +
  geom_ribbon(data = ribbon_phi_eBird,
              aes(ymin = phi_values+SD.phi,
                  ymax = phi_values-SD.phi),
              fill = "#91bfdb25")+
  geom_ribbon(data = ribbon_phi_eBird_OUSSEGSS,
              aes(ymin = phi_values+SD.phi,
                  ymax = phi_values-SD.phi),
              fill = "#91bfdb25")+
  geom_line(data = phi_data_plot |>
               filter(group %in% c("phi.hat", "phi.hatSM")),
            aes(color = group), linetype = "dashed") +
  geom_point(data = phi_data_plot |>
               filter(group %in% c("phi.hat", "phi.hatSM")),
             aes(shape = Model,
                 color = group),
             size = 2) +
```

```
  labs(x = "Observation date",
       y = expression(italic(phi)),
       tag = expression(bold("(c)")),
       title = "Simulation window: 500 weeks or ~10 years")  +
  coord_cartesian(ylim = c(0, 1))+
  scale_x_date(limits = c(snailkites.PP$observation.date[78],
                          max(snailkites.PP$observation.date)),
               breaks = seq(as.Date("2018-01-01"), as.Date(Sys.Date( )),
                            by = "10 months"), date_labels="%b-%y")+
  scale_color_manual(values = c("#91bfdb95",
                                "#fc8d5995"),
                     labels = c(expression(italic(phi)[eBird]),
                                expression(italic(phi)[SM])))+
  theme_bw() +
  theme(legend.title = element_blank(),
        legend.position = "none",
        legend.spacing.y = unit(-0.5, "cm"),
        legend.background = element_blank(),
        legend.box.background = element_rect(colour = "black"))+
  guides(color = guide_legend(ncol=2),
         shape = guide_legend(ncol=2))
FigOUSS_C
```

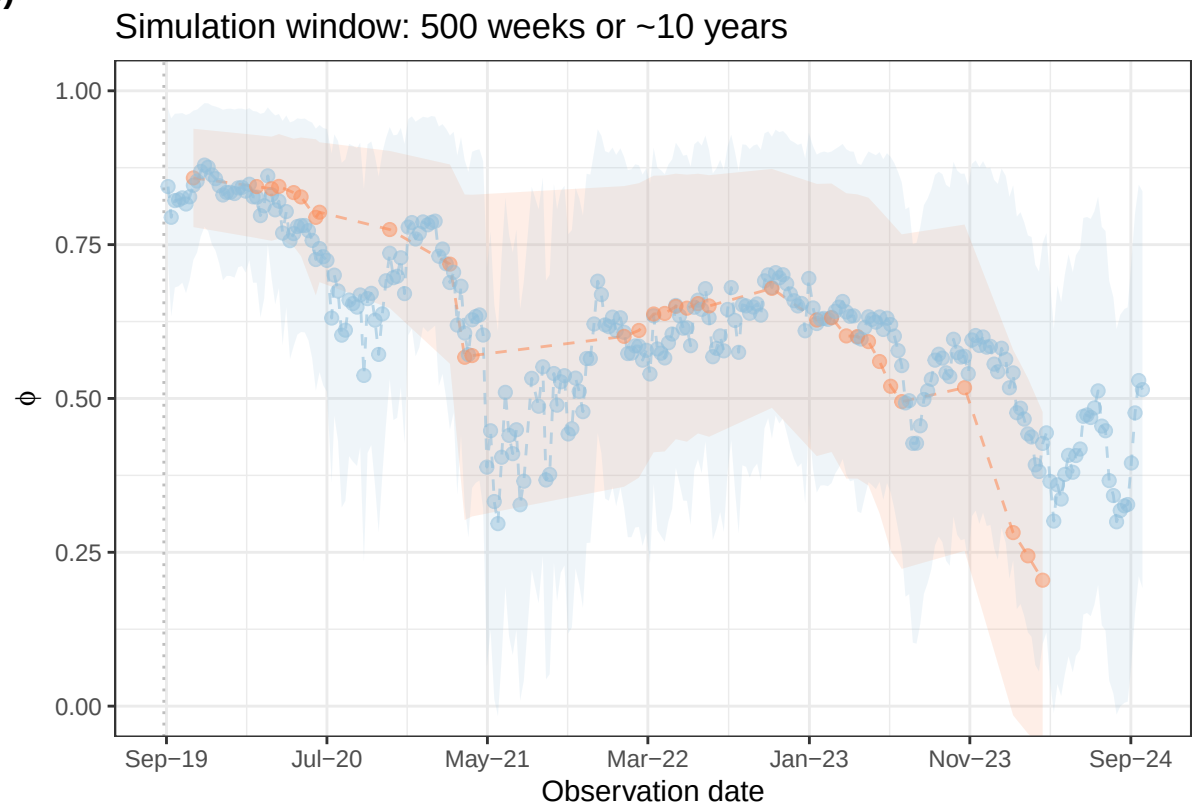
(c)




```
#Only EGSS comparison

#Figure overlap phi
```

```r
Figure4c <- ggplot(phi_data_plot,
                   aes(x = observation.date, y = phi_values)) +
  geom_vline(xintercept = snailkites.PP$observation.date[78],
             color = "gray", linetype = "dotted") +
  geom_ribbon(data = ribbon_phi_SM,
              aes(ymin = phi_values+SD.phiSM,
                  ymax = phi_values-SD.phiSM),
              fill = "#fc8d5925") +
  geom_ribbon(data = ribbon_phi_eBird,
              aes(ymin = phi_values+SD.phi,
                  ymax = phi_values-SD.phi),
              fill = "#91bfdb25") +
  geom_line(data = phi_data_plot |>
               filter(dataset != "eBird.OUSS.EGSS",
                      group %in% c("phi.hat", "phi.hatSM")),
            aes(color = group), linetype = "dashed") +
  geom_point(data = phi_data_plot |>
               filter(dataset != "eBird.OUSS.EGSS",
                      group %in% c("phi.hat", "phi.hatSM")),
             aes(color = group, fill = group),
             size = 2, shape = 21) +
  labs(x = "Observation date",
       y = expression(italic(phi)),
       tag = expression(bold("(c)")),
       title = "Simulation window: 500 weeks or ~10 years")  +
  coord_cartesian(ylim = c(0, 1))+
  scale_x_date(limits = c(snailkites.PP$observation.date[78],
                          max(snailkites.PP$observation.date)),
               breaks = seq(as.Date("2018-01-01"), as.Date(Sys.Date( )),
                            by = "10 months"), date_labels="%b-%y")+
  scale_color_manual(values = c("#91bfdb95",
                                "#fc8d5995"),
                     labels = c(expression(italic(phi)[eBird]),
                                expression(italic(phi)[SM])))+
  scale_fill_manual(values = c("#91bfdb75",
                               "#fc8d5975"),
                    labels = c(expression(italic(phi)[eBird]),
                               expression(italic(phi)[SM])))+
  theme_bw() +
  theme(legend.position = "none")

Figure4c
```

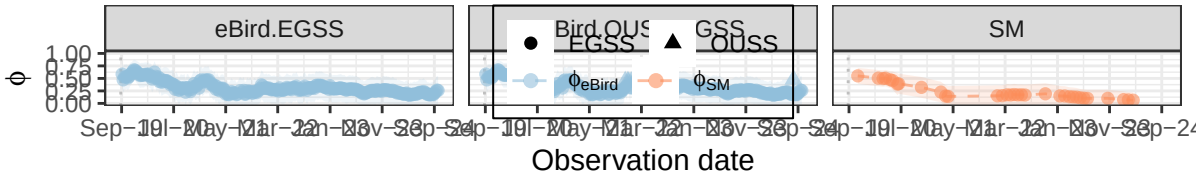
(c)



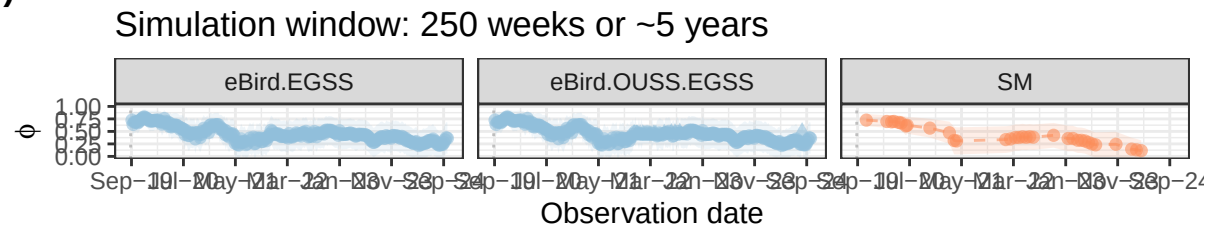
And combine the figures

```
#and combine in the figure
Figure4Ext <- grid.arrange(FigOUSS_A, FigOUSS_B, FigOUSS_C,
                           ncol = 1)
```

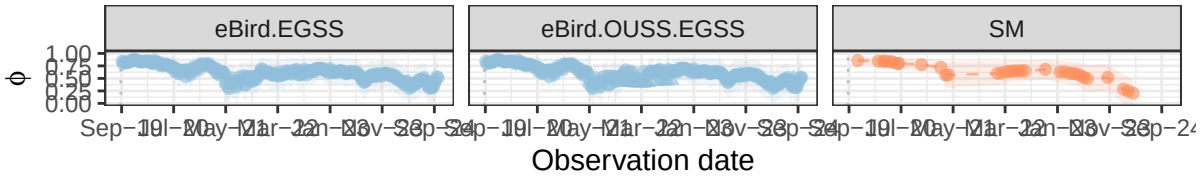
(a)



(b)



(c)




```
#lets saved it with good proportions
ggsave("results/Figure4_Extended_OUSS_EGSS_3-5-10yrsSimulated.pdf",
       plot = Figure4Ext, dpi = 300, width = 12, height = 9, units = "in")

ggsave("results/Figure4_Extended_OUSS_EGSS_3-5-10yrsSimulated.png",
       plot = Figure4Ext, dpi = 300, width = 12, height = 9, units = "in")
```

```
#and combine in the figure
Figure4 <- grid.arrange(Figure4a, Figure4b, Figure4c,
                        ncol = 1)
```

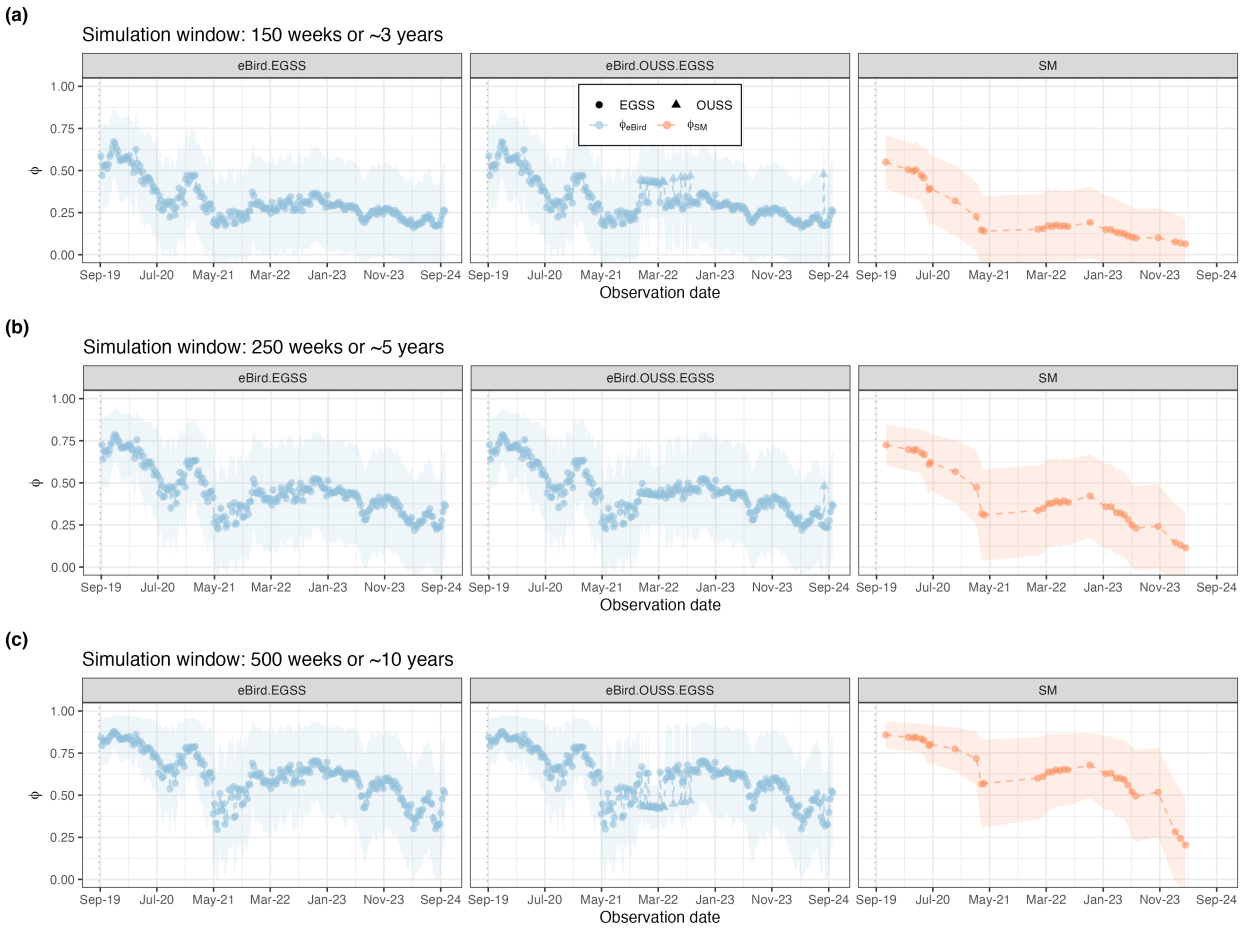
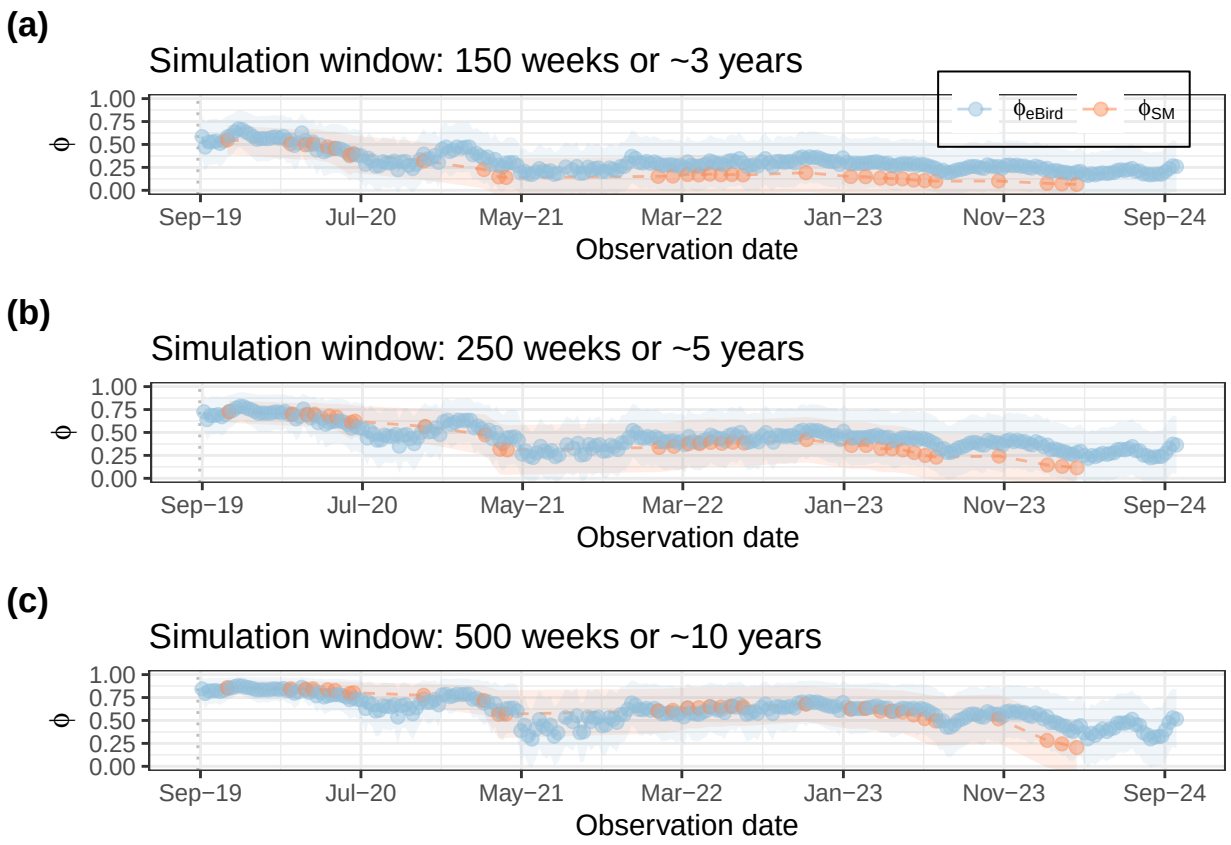


Figure SI-51: Estimated local persistence probability ($\hat{\phi}$ +/- standard deviation) during ~5 years of monitoring (September 2019 to September 2024) after 50,000 population trajectories simulated for three timeframes (a-c). The shape of the points in the central column shows the model selected in each week of estimation (EGSS and OUSS, the preference of the latter by the model suggestion of density dependence). Model selection was conducted for the benchmark (right column), but all suggested density-independent dynamic (EGSS). Higher temporal resolution evident in eBird (n = 262) when compared with standardized monitored data (n = 33).




```
#lets saved it with good proportions
ggsave("results/Figure4_PLocalPersi_3-5-10yrsSimulated.pdf",
       plot = Figure4, dpi = 300, width = 5.5, height = 8, units = "in")

ggsave("results/Figure4_PLocalPersi_3-5-10yrsSimulated.png",
       plot = Figure4, dpi = 300, width = 5.5, height = 8, units = "in")
```

How correlated are the $\hat{\phi}$?

```
correlations <- phi.eBird |>
  full_join(phi.SM) |>
  group_by(Timeframe) |>
  summarise(correlation = cor(phi.hatSM,
                              phi.hat,
                              use = "pairwise.complete.obs"))
```

```
## Joining with `by = join_by(Time.t, Timeframe, Model)`
```

```
correlations
```

```
## # A tibble: 3 x 2
##   Timeframe              correlation
##   <fct>                        <dbl>
## 1 150 weeks or ~3 years        0.965
## 2 250 weeks or ~5 years        0.961
## 3 500 weeks or ~10 years       0.933
```

Time frame of ~5 years had the higher Pearson correlation among the three time frames.

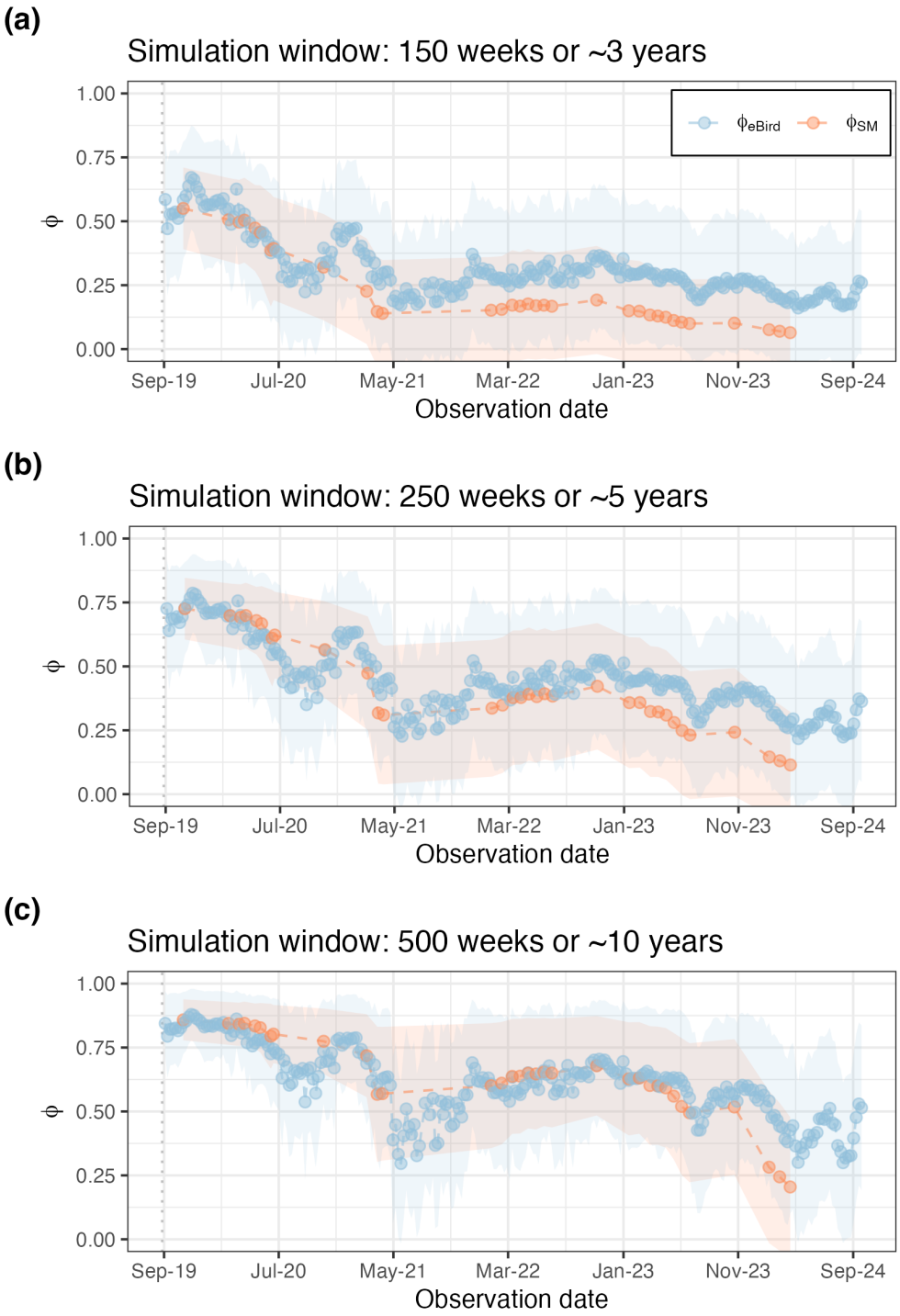


Figure SI-52: *Figure 4 in Main text.* Estimated local persistence probability ($\hat{\phi}$ +/- standard deviation) during ~5 years of monitoring (October 2019 to July 2024) after 50,000 population trajectories simulated for three timeframes (a-c).

6.1 Results for ~3 years (150 weeks)

```
#Extended figures

phi_data_plot <- snailkites.PP |>
  left_join(phi.eBird |>
              full_join(phi.SM) |>
              filter(Timeframe == "150 weeks or ~3 years")) |>
  ungroup() |>
    mutate(phi.hat = phi.hat * coeff,
           SD.phi = SD.phi * coeff,
           phi.hatSM = phi.hatSM * coeffSM,
           SD.phiSM = SD.phiSM * coeffSM) |>
  pivot_longer(cols = !c(cell, year, week, Timeframe, Time.t, observation.date, Model),
               names_to = "group",
               values_to = "phi_values") |>
  drop_na(phi_values)
```

```
## Joining with `by = join_by(Time.t, Timeframe, Model)`
## Joining with `by = join_by(Time.t)`
```

```
phi_data_plot
```

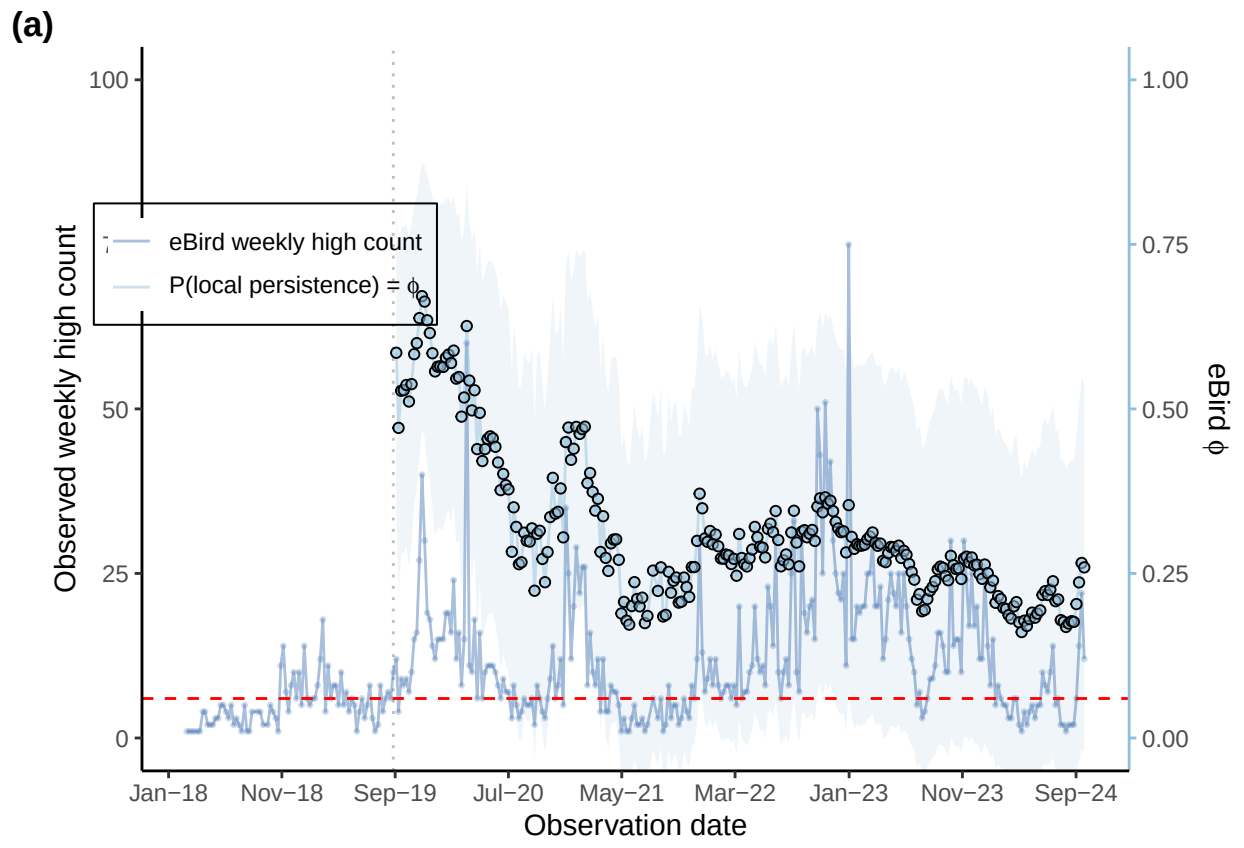
```
## # A tibble: 979 x 9
##       cell Time.t observation.date  year  week Timeframe Model group  phi_values
##      <dbl>  <dbl> <date>           <dbl> <dbl> <fct>     <chr> <chr>       <dbl>
##  1 2665918      8 2018-02-19        2018     8 <NA>      <NA>  Obser~          1
##  2 2665918      8 2018-02-19        2018     8 <NA>      <NA>  abund~          1
##  3 2665918      9 2018-02-26          NA    NA <NA>      <NA>  Obser~          1
##  4 2665918     10 2018-03-05          NA    NA <NA>      <NA>  Obser~          1
##  5 2665918     11 2018-03-12        2018    11 <NA>      <NA>  Obser~          1
##  6 2665918     11 2018-03-12        2018    11 <NA>      <NA>  abund~          2
##  7 2665918     12 2018-03-20          NA    NA <NA>      <NA>  Obser~          1
##  8 2665918     13 2018-03-26          NA    NA <NA>      <NA>  Obser~          1
##  9 2665918     14 2018-04-03          NA    NA <NA>      <NA>  Obser~          4
## 10 2665918     15 2018-04-09        2018    15 <NA>      <NA>  Obser~          4
## # i 969 more rows
```

```
#Ribbon lo-hi phi (SD) for the two datasets
ribbon_phi_eBird <- phi_data_plot |>
  filter(group %in% c("phi.hat", "SD.phi")) |>
  pivot_wider(names_from = group, values_from = phi_values, values_fn = {mean}) |>
  mutate(phi_values = phi.hat)

ribbon_phi_SM <- phi_data_plot |>
  filter(group %in% c("phi.hatSM", "SD.phiSM")) |>
  pivot_wider(names_from = group, values_from = phi_values, values_fn = {mean})|>
  mutate(phi_values = phi.hatSM)

  #figurea
Fig3aExt <- ggplot(phi_data_plot |>
                     filter(group %in% c("phi.hat", "Observed.y")),
                   aes(x = observation.date,
                       y = phi_values,
                       fill = group)) +
```

```
  geom_vline(xintercept = snailkites.PP$observation.date[78],
             color = "gray", linetype = "dotted") +
  geom_ribbon(data = ribbon_phi_eBird,
              aes(ymin = phi.hat-SD.phi, ymax = phi.hat+SD.phi),
              fill = "#91bfdb25") +
  geom_line(aes(color = group, linetype = group)) +
  geom_point(data = phi_data_plot |>
                  filter(group %in% c("Observed.y")),
             color = "#4575b445", size = 0.5, fill = "#4575b445", shape = 21)+
  geom_point(data = phi_data_plot |>
                  filter(group %in% c("phi.hat")) |>
               drop_na(Model),
              color = "black",
             shape = 21, fill = "#91bfdb95") +
  labs(x = "Observation date",
       y = "Observed weekly high count",
       tag = expression(bold("(a)")),
       fill = "",
       color = "",
       linetype = "",
       shape = "") +
  scale_y_continuous(sec.axis = sec_axis(trans = ~./coeff,
                                         name = expression("eBird " * italic(phi)))) +
  coord_cartesian(ylim = c(0, coeff))+
  scale_x_date(limits = c(min(snailkites.PP$observation.date),
                          max(snailkites.PP$observation.date)),
               breaks = seq(as.Date("2018-01-01"), as.Date(Sys.Date( )),
                            by = "10 months"), date_labels="%b-%y")+
  scale_linetype_manual(values = c("solid",
                                   "solid"),
                        labels = c("eBird weekly high count",
                                   expression("P(local persistence) = " * italic(phi))))+
  scale_color_manual(values = c("#4575b475",
                                "#91bfdb75"),
                     labels = c("eBird weekly high count",
                                expression("P(local persistence) = " * italic(phi)))) +
  geom_hline(yintercept = N.critical, linetype = "dashed", color = "red")+
  theme_classic() +
  theme(legend.title = element_blank(),
        legend.position = c(0.125,0.7),
        legend.background = element_blank(),
        legend.box.background = element_rect(colour = "black"),
        axis.line.y.right = element_line(color = "#91bfdb"),
        axis.ticks.y.right = element_line(color = "#91bfdb"))
```

```
## Warning: The `trans` argument of `sec_axis()` is deprecated as of ggplot2 3.5.0.
## i Please use the `transform` argument instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

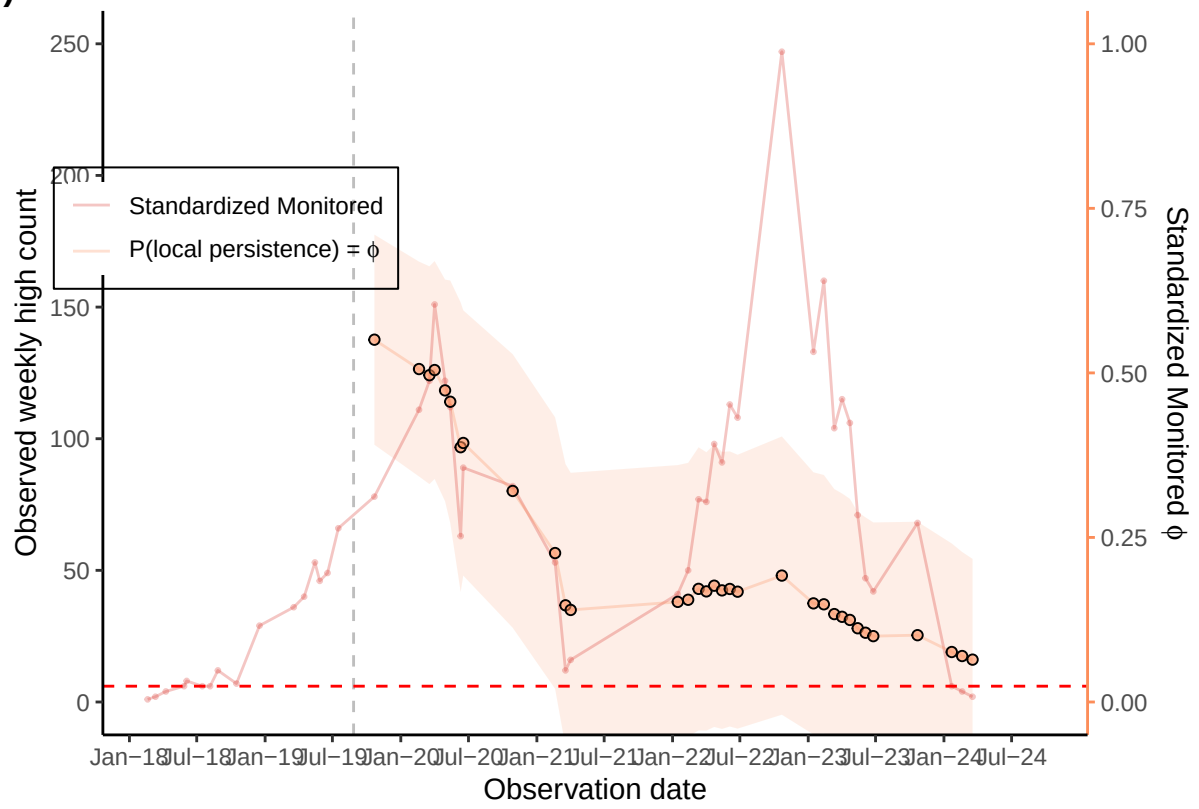
```
Fig3aExt
```




```
#SM figure

  #figurea
Fig3bExt <- ggplot(phi_data_plot |>
                     filter(group %in% c("phi.hatSM", "abundance.monitored")),
                   aes(x = observation.date,
                       y = phi_values,
                       fill = group)) +
  geom_vline(xintercept = snailkites.PP$observation.date[78],
             color = "gray", linetype = "dashed") +
  geom_ribbon(data = ribbon_phi_SM,
              aes(ymin = phi.hatSM-SD.phiSM, ymax = phi.hatSM+SD.phiSM),
              fill = "#fc8d5925") +
  geom_line(aes(color = group, linetype = group)) +
  geom_point(data = phi_data_plot |>
                     filter(group %in% c("abundance.monitored")),
             color = "#d7302745", size = 0.5, fill = "#d7302745", shape = 21)+
  geom_point(data = phi_data_plot |>
                     filter(group %in% c("phi.hatSM")) |>
                drop_na(Model),
             shape = 21, color = "black",
             fill = "#fc8d5995") +
  labs(x = "Observation date",
       y = "Observed weekly high count",
       tag = expression(bold("(b)")),
       fill = "",
```

```
      color = "",
      linetype = "",
      shape = "") +
  scale_y_continuous(sec.axis = sec_axis(trans = ~./coeffSM,
                                         name = expression("Standardized Monitored " * italic(phi)))) +
  coord_cartesian(ylim = c(0, coeffSM))+
  scale_x_date(limits = c(min(snailkites.PP$observation.date),
                          max(snailkites.PP$observation.date)),
               breaks = seq(as.Date("2018-01-01"), as.Date(Sys.Date( )),
                            by = "6 months"), date_labels="%b-%y")+
  scale_linetype_manual(values = c("solid",
                                   "solid"),
                        labels = c("Standardized Monitored",
                                   expression("P(local persistence) = " * italic(phi))))+
  scale_color_manual(values = c("#d7302745",
                                "#fc8d5945"),
                        labels = c("Standardized Monitored",
                                   expression("P(local persistence) = " * italic(phi)))) +
  geom_hline(yintercept = N.critical, linetype = "dashed", color = "red")+
  theme_classic() +
  theme(legend.title = element_blank(),
        legend.position = c(0.125,0.7),
        legend.background = element_blank(),
        legend.box.background = element_rect(colour = "black"),
        axis.line.y.right = element_line(color = "#fc8d59"),
        axis.ticks.y.right = element_line(color = "#fc8d59"))
Fig3bExt
```

(b)




```
#Performance comparison
phidata <- phi.eBird |>
             full_join(phi.SM) |>
  filter(Timeframe == "150 weeks or ~3 years")
```

```
## Joining with `by = join_by(Time.t, Timeframe, Model)`
```

```
#relationship?
lm(phi.hat~phi.hatSM, data = phidata)
```

```
##
## Call:
## lm(formula = phi.hat ~ phi.hatSM, data = phidata)
##
## Coefficients:
## (Intercept)    phi.hatSM
##      0.1901       0.6421
```

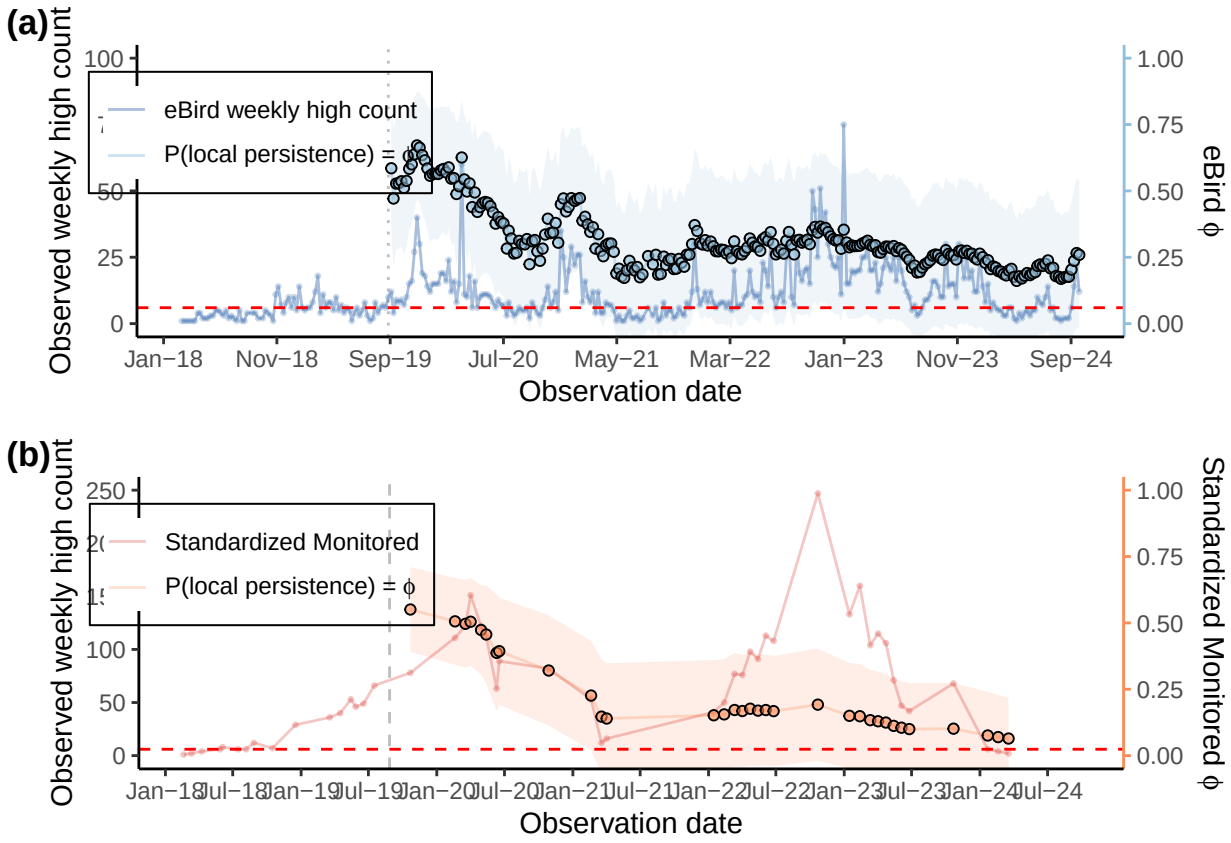
```
Fig3cExt <- ggplot(phidata)+
    geom_abline(slope = 1)+
    geom_pointrange(aes(x = phi.hatSM, y = phi.hat,
                        ymin = phi.hat-SD.phi,
                        ymax = phi.hat+SD.phi),
                    shape = 23, fill = "gray", color = "gray",
                    alpha = 0.25) +
    geom_errorbarh(aes(x = phi.hatSM, y = phi.hat,
                       xmax = phi.hatSM+SD.phiSM,
                       xmin = phi.hatSM-SD.phiSM, height = 0),
```

```
                  color = "gray",
                  alpha = 0.25)+
   geom_point(aes(x = phi.hatSM, y = phi.hat),
                   shape = 23, fill = "gray",
              color = "black", size = 2, alpha = 0.5)+
   geom_text(x = 0.5, y = 0.05,
           label = lm_eqn(df = phidata,
                          x = phidata$phi.hatSM,
                          y = phidata$phi.hat),
           parse = TRUE, color = "black")+
     geom_smooth(aes(x=phi.hatSM, y=phi.hat),
             method = "lm", fullrange=TRUE,
             color = "black", se = F, linetype = "dashed", size = 1)+
   scale_y_continuous(limits = c(0,1))+
   scale_x_continuous(limits = c(0,1))+
   labs(x = expression("Standardized Monitored "*italic(phi)),
        y = expression("eBird "*italic(phi)),
        tag = expression(bold("(c)")),
        title = "150 weeks or ~3 years")+
  coord_fixed()+
  theme_classic()
```

```
## Warning in geom_errorbarh(aes(x = phi.hatSM, y = phi.hat, xmax = phi.hatSM + :
## Ignoring unknown aesthetics: x
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
#and combine in the figure
Fig3ab <- grid.arrange(Fig3aExt, Fig3bExt, ncol = 1)
```




```
Fig3 <- grid.arrange(Fig3ab, Fig3cExt, ncol = 2, widths = c(1.5,1))
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

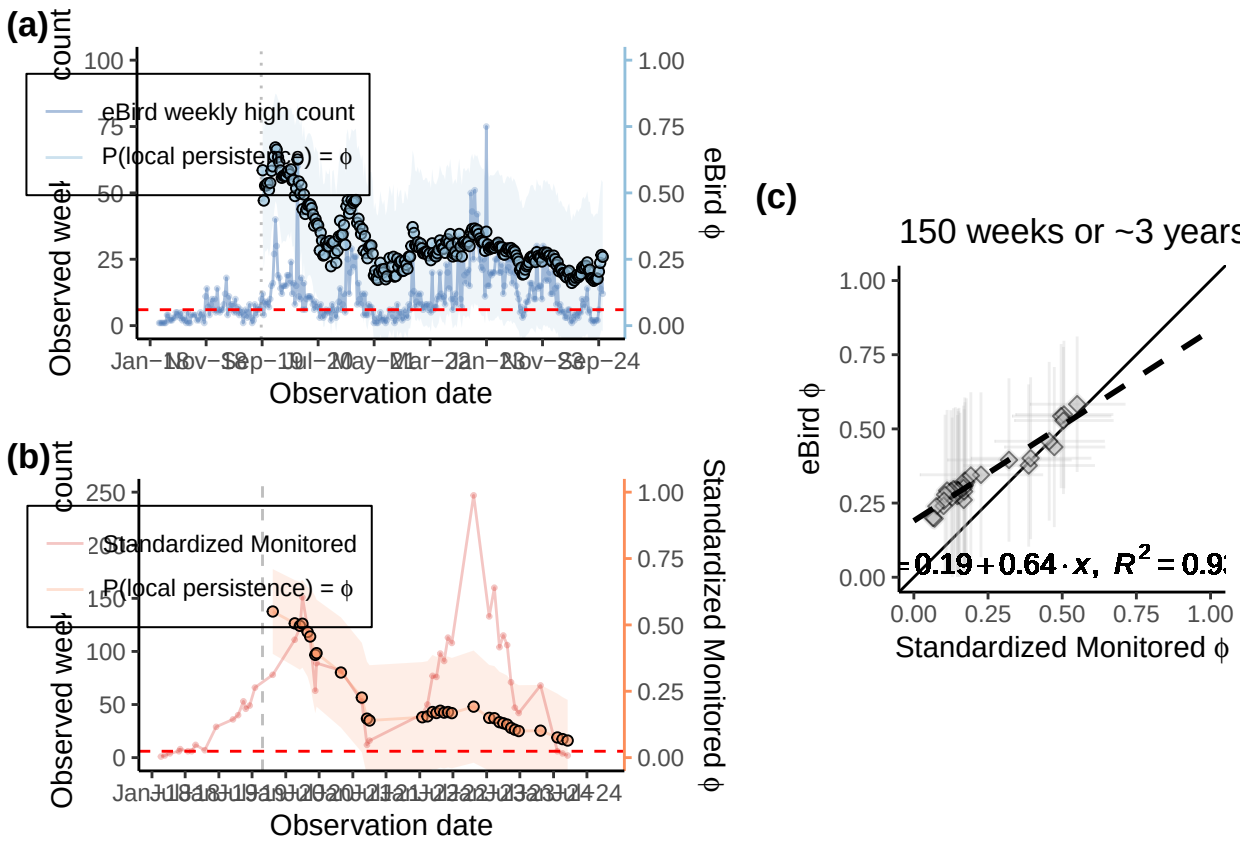
```
## Warning: Removed 229 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 229 rows containing missing values or values outside the scale range
## (`geom_pointrange()`).
```

```
## Warning: Removed 6 rows containing missing values or values outside the scale range
## (`geom_segment()`).
```

```
## Warning: Removed 252 rows containing missing values or values outside the scale range
## (`geom_errorbarh()`).
```

```
## Warning: Removed 229 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
#It looks not good in the `Rmd`, but it is saved with good proportions
ggsave("results/Figure3_timeseries_PLocalPersi_150weeks.pdf",
       plot = Fig3, dpi = 300, width = 16, height = 6, units = "in")

ggsave("results/Figure3_timeseries_PLocalPersi150weeks.png",
       plot = Fig3, dpi = 300, width = 16, height = 6, units = "in")
```

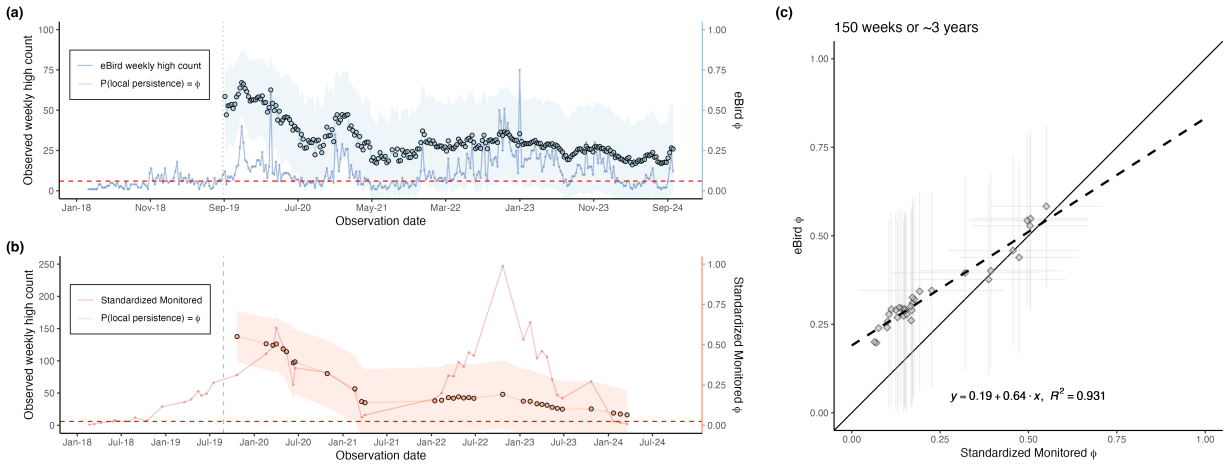


Figure SI-53: Time series of weekly high counts of Snail kites in Payne's Prairie wetland from eBird (a) and standardized monitoring surveys (b), with local persistence probability (ϕ) +/- standard deviation estimated for each dataset after 50,000 population trajectories simulated. (c) Local persistence probability extimated from eBird data plotted against local persistence probability estimated from standardized monitored, with standard deviation. Estimation of ϕ from 50,000 trajectories in a timeframe of ~3 years (150 weeks). Solid black line is the line of identity, while dashed black line is linear regression (equation in gray).

6.2 Results for ~5 years (250 weeks) - selected

```
#Extended figures

phi_data_plot <- snailkites.PP |>
  left_join(phi.eBird |>
              full_join(phi.SM)  |>
              filter(Timeframe == "250 weeks or ~5 years")) |>
  ungroup() |>
    mutate(phi.hat = phi.hat * coeff,
           SD.phi = SD.phi * coeff,
           phi.hatSM = phi.hatSM * coeffSM,
           SD.phiSM = SD.phiSM * coeffSM) |>
  pivot_longer(cols = !c(cell, year, week, Timeframe, Time.t, observation.date, Model),
               names_to = "group",
               values_to = "phi_values") |>
  drop_na(phi_values)
```

```
## Joining with `by = join_by(Time.t, Timeframe, Model)`
## Joining with `by = join_by(Time.t)`
```

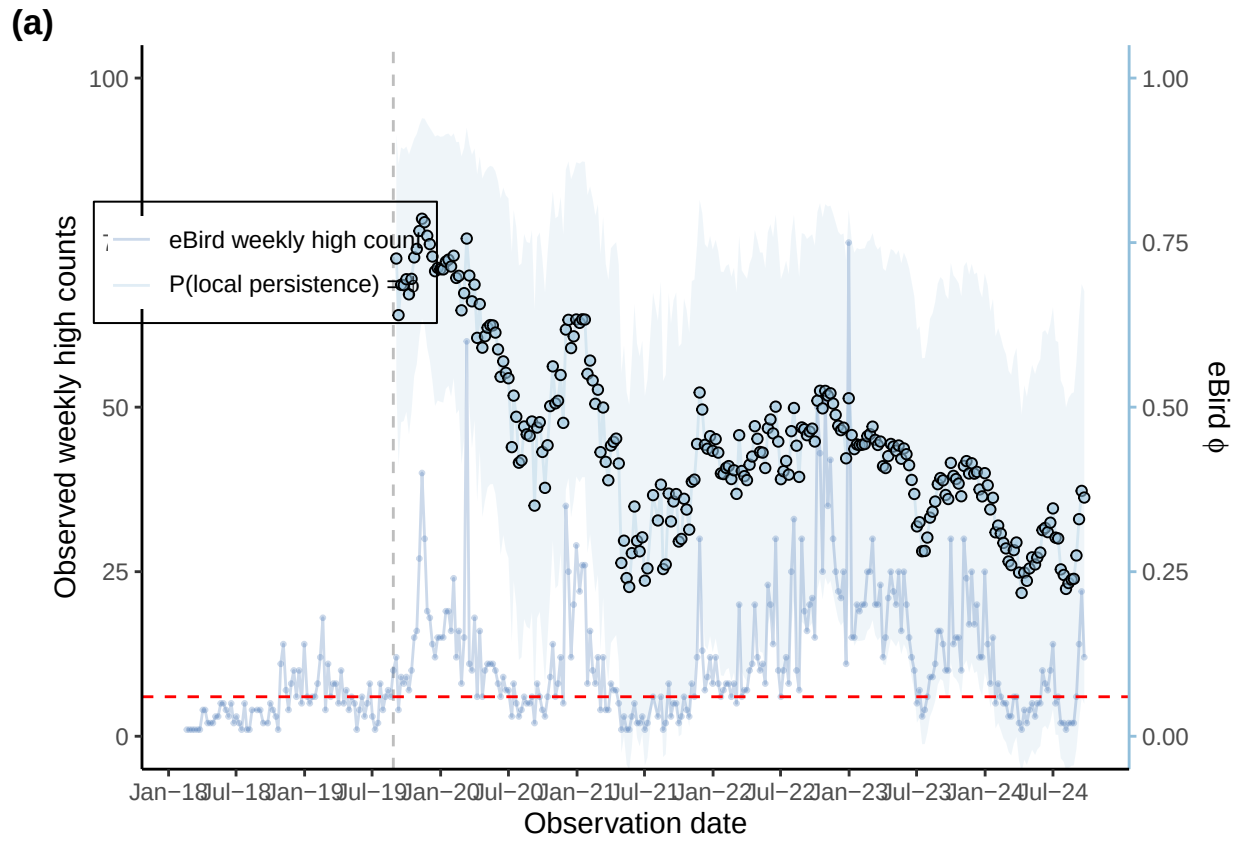
```
phi_data_plot
```

```
## # A tibble: 979 x 9
##       cell Time.t observation.date  year  week Timeframe Model group  phi_values
##      <dbl>  <dbl> <date>           <dbl> <dbl> <fct>     <chr> <chr>       <dbl>
##  1 2665918      8 2018-02-19        2018     8 <NA>      <NA>  Obser~          1
##  2 2665918      8 2018-02-19        2018     8 <NA>      <NA>  abund~          1
##  3 2665918      9 2018-02-26          NA    NA <NA>      <NA>  Obser~          1
##  4 2665918     10 2018-03-05          NA    NA <NA>      <NA>  Obser~          1
##  5 2665918     11 2018-03-12        2018    11 <NA>      <NA>  Obser~          1
##  6 2665918     11 2018-03-12        2018    11 <NA>      <NA>  abund~          2
```

```
##  7 2665918     12 2018-03-20          NA    NA <NA>      <NA>  Obser~          1
##  8 2665918     13 2018-03-26          NA    NA <NA>      <NA>  Obser~          1
##  9 2665918     14 2018-04-03          NA    NA <NA>      <NA>  Obser~          4
## 10 2665918     15 2018-04-09        2018    15 <NA>      <NA>  Obser~          4
## # i 969 more rows
```

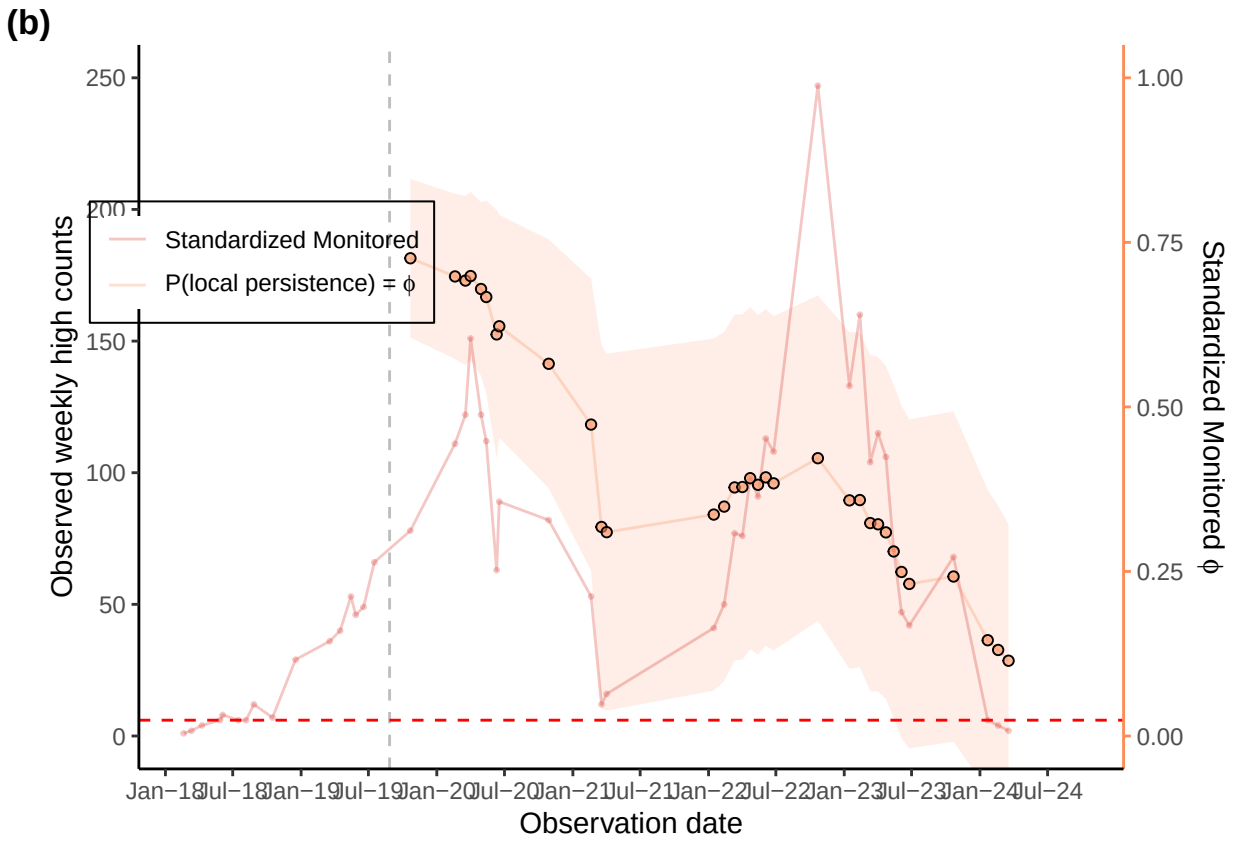
```
#Ribbon lo-hi phi (SD) for the two datasets
ribbon_phi_eBird <- phi_data_plot |>
  filter(group %in% c("phi.hat", "SD.phi")) |>
  pivot_wider(names_from = group, values_from = phi_values, values_fn = {mean}) |>
  mutate(phi_values = phi.hat)

ribbon_phi_SM <- phi_data_plot |>
  filter(group %in% c("phi.hatSM", "SD.phiSM")) |>
  pivot_wider(names_from = group, values_from = phi_values, values_fn = {mean})|>
  mutate(phi_values = phi.hatSM)

  #figurea
Fig3aExt <- ggplot(phi_data_plot |>
                     filter(group %in% c("phi.hat", "Observed.y")),
                   aes(x = observation.date,
                       y = phi_values,
                       fill = group)) +
  geom_vline(xintercept = snailkites.PP$observation.date[78],
             color = "gray", linetype = "dashed") +
  geom_ribbon(data = ribbon_phi_eBird,
              aes(ymin = phi.hat-SD.phi, ymax = phi.hat+SD.phi),
              fill = "#91bfdb25") +
  geom_line(aes(color = group, linetype = group)) +
  geom_point(data = phi_data_plot |>
                     filter(group %in% c("Observed.y")),
             color = "#4575b445", size = 0.5, fill = "#4575b445", shape = 21)+
  geom_point(data = phi_data_plot |>
                     filter(group %in% c("phi.hat")) |>
               drop_na(Model),
             shape = 21, fill = "#91bfdb95", color = "black") +
  labs(x = "Observation date",
       y = "Observed weekly high counts",
       tag = expression(bold("(a)")),
       fill = "",
       color = "",
       linetype = "",
       shape = "") +
  scale_y_continuous(sec.axis = sec_axis(trans = ~./coeff,
                                         name = expression("eBird " * italic(phi)))) +
  coord_cartesian(ylim = c(0, coeff))+
  scale_x_date(limits = c(min(snailkites.PP$observation.date),
                          max(snailkites.PP$observation.date)),
               breaks = seq(as.Date("2018-01-01"), as.Date(Sys.Date( )),
                            by = "6 months"), date_labels="%b-%y")+
  scale_linetype_manual(values = c("solid",
                                   "solid"),
                        labels = c("eBird weekly high count",
                                   expression("P(local persistence) = " * italic(phi))))+
```

```
  scale_color_manual(values = c("#4575b445",
                                "#91bfdb45"),
                     labels = c("eBird weekly high count",
                                expression("P(local persistence) = " * italic(phi)))) +
  geom_hline(yintercept = N.critical, linetype = "dashed", color = "red")+
  theme_classic() +
  theme(legend.title = element_blank(),
        legend.position = c(0.125,0.7),
        legend.background = element_blank(),
        legend.box.background = element_rect(colour = "black"),
        axis.line.y.right = element_line(color = "#91bfdb"),
        axis.ticks.y.right = element_line(color = "#91bfdb"))
Fig3aExt
```




```
#SM figure

  #figurea
Fig3bExt <- ggplot(phi_data_plot |>
                     filter(group %in% c("phi.hatSM", "abundance.monitored")),
                   aes(x = observation.date,
                       y = phi_values,
                       fill = group)) +
  geom_vline(xintercept = snailkites.PP$observation.date[78],
             color = "gray", linetype = "dashed") +
  geom_ribbon(data = ribbon_phi_SM,
              aes(ymin = phi.hatSM-SD.phiSM, ymax = phi.hatSM+SD.phiSM),
```

```
              fill = "#fc8d5925") +
  geom_line(aes(color = group, linetype = group)) +
  geom_point(data = phi_data_plot |>
                  filter(group %in% c("abundance.monitored")),
             color = "#d7302745", size = 0.5, fill = "#d7302745", shape = 21)+
  geom_point(data = phi_data_plot |>
                  filter(group %in% c("phi.hatSM")) |>
               drop_na(Model),
             shape = 21, fill = "#fc8d5995", color = "black") +
  labs(x = "Observation date",
       y = "Observed weekly high counts",
       tag = expression(bold("(b)")),
       fill = "",
       color = "",
       linetype = "",
       shape = "") +
  scale_y_continuous(sec.axis = sec_axis(trans = ~./coeffSM,
                                         name = expression("Standardized Monitored " * italic(phi)))) +
  coord_cartesian(ylim = c(0, coeffSM))+
  scale_x_date(limits = c(min(snailkites.PP$observation.date),
                          max(snailkites.PP$observation.date)),
               breaks = seq(as.Date("2018-01-01"), as.Date(Sys.Date( )),
                            by = "6 months"), date_labels="%b-%y")+
  scale_linetype_manual(values = c("solid",
                                   "solid"),
                        labels = c("Standardized Monitored",
                                   expression("P(local persistence) = " * italic(phi))))+
  scale_color_manual(values = c("#d7302745",
                                "#fc8d5945"),
                     labels = c("Standardized Monitored",
                                expression("P(local persistence) = " * italic(phi)))) +
  geom_hline(yintercept = N.critical, linetype = "dashed", color = "red")+
  theme_classic() +
  theme(legend.title = element_blank(),
        legend.position = c(0.125,0.7),
        legend.background = element_blank(),
        legend.box.background = element_rect(colour = "black"),
        axis.line.y.right = element_line(color = "#fc8d59"),
        axis.ticks.y.right = element_line(color = "#fc8d59"))
Fig3bExt
```




```
#Performance comparison
phidata <- phi.eBird |>
              full_join(phi.SM)  |>
  filter(Timeframe == "250 weeks or ~5 years")
```

```
## Joining with `by = join_by(Time.t, Timeframe, Model)`
```

```
#relationship?
lm(phi.hat~phi.hatSM, data = phidata)
```

```
##
## Call:
## lm(formula = phi.hat ~ phi.hatSM, data = phidata)
##
## Coefficients:
## (Intercept)    phi.hatSM
##      0.2288       0.6131
```

```
Fig3cExt <- ggplot(phidata)+
    geom_abline(slope = 1)+
    geom_pointrange(aes(x = phi.hatSM, y = phi.hat,
                        ymin = phi.hat-SD.phi,
                        ymax = phi.hat+SD.phi),
                    shape = 23, fill = "gray", color = "gray",
                    alpha = 0.25) +
    geom_errorbarh(aes(x = phi.hatSM, y = phi.hat,
                       xmax = phi.hatSM+SD.phiSM,
                       xmin = phi.hatSM-SD.phiSM, height = 0),
```

```
                  color = "gray",
                  alpha = 0.25)+
   geom_point(aes(x = phi.hatSM, y = phi.hat),
                   shape = 23, fill = "gray",
              color = "black", size = 2, alpha = 0.5)+
   geom_text(x = 0.5, y = 0.05,
           label = lm_eqn(df = phidata,
                          x = phidata$phi.hatSM,
                          y = phidata$phi.hat),
           parse = TRUE, color = "black")+
     geom_smooth(aes(x=phi.hatSM, y=phi.hat),
             method = "lm", fullrange=TRUE,
             color = "black", se = F, linetype = "dashed", size = 1)+
   scale_y_continuous(limits = c(0,1))+
   scale_x_continuous(limits = c(0,1))+
   labs(x = expression("Standardized Monitored "*italic(phi)),
        y = expression("eBird "*italic(phi)),
        tag = expression(bold("(c)")),
        title = "250 weeks or ~5 years")+
  coord_fixed()+
  theme_classic()
```

```
## Warning in geom_errorbarh(aes(x = phi.hatSM, y = phi.hat, xmax = phi.hatSM + :
## Ignoring unknown aesthetics: x
```

```
Fig3cExt
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning: Removed 229 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

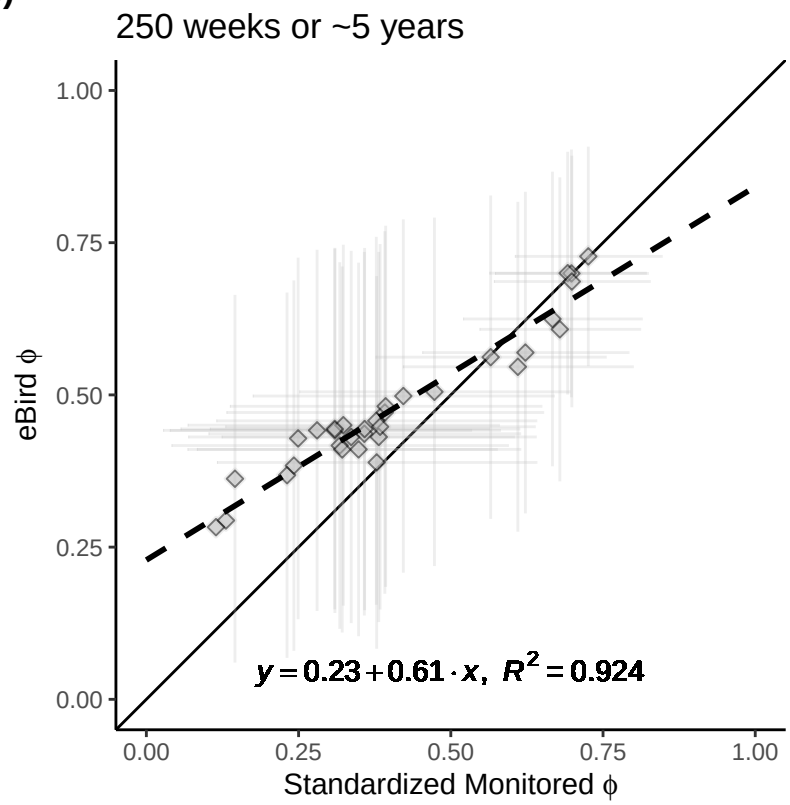
```
## Warning: Removed 229 rows containing missing values or values outside the scale range
## (`geom_pointrange()`).
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_segment()`).
```

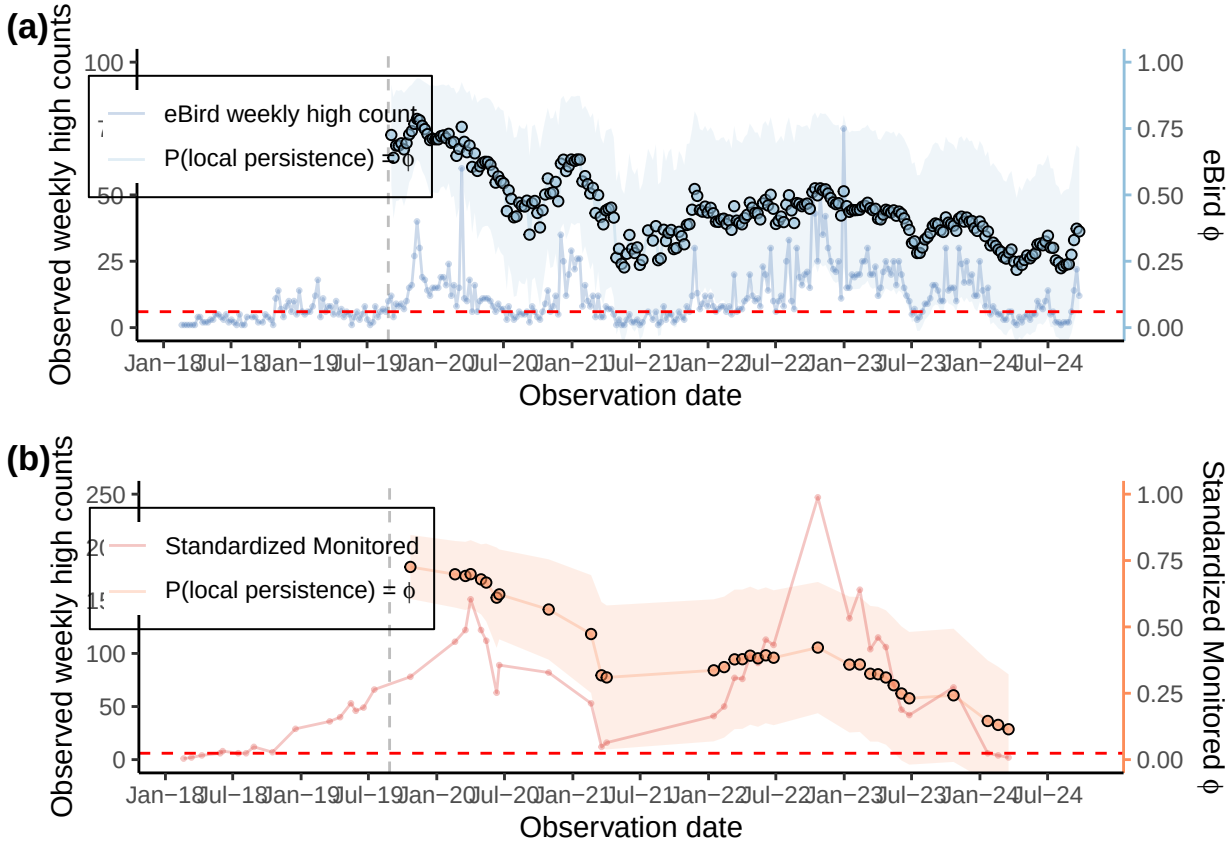
```
## Warning: Removed 235 rows containing missing values or values outside the scale range
## (`geom_errorbarh()`).
```

```
## Warning: Removed 229 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

(c)




```
#and combine in the figure
Fig3ab <- grid.arrange(Fig3aExt, Fig3bExt, ncol = 1)
```




```
Fig3 <- grid.arrange(Fig3ab, Fig3cExt, ncol = 2, widths = c(1.5,1))
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

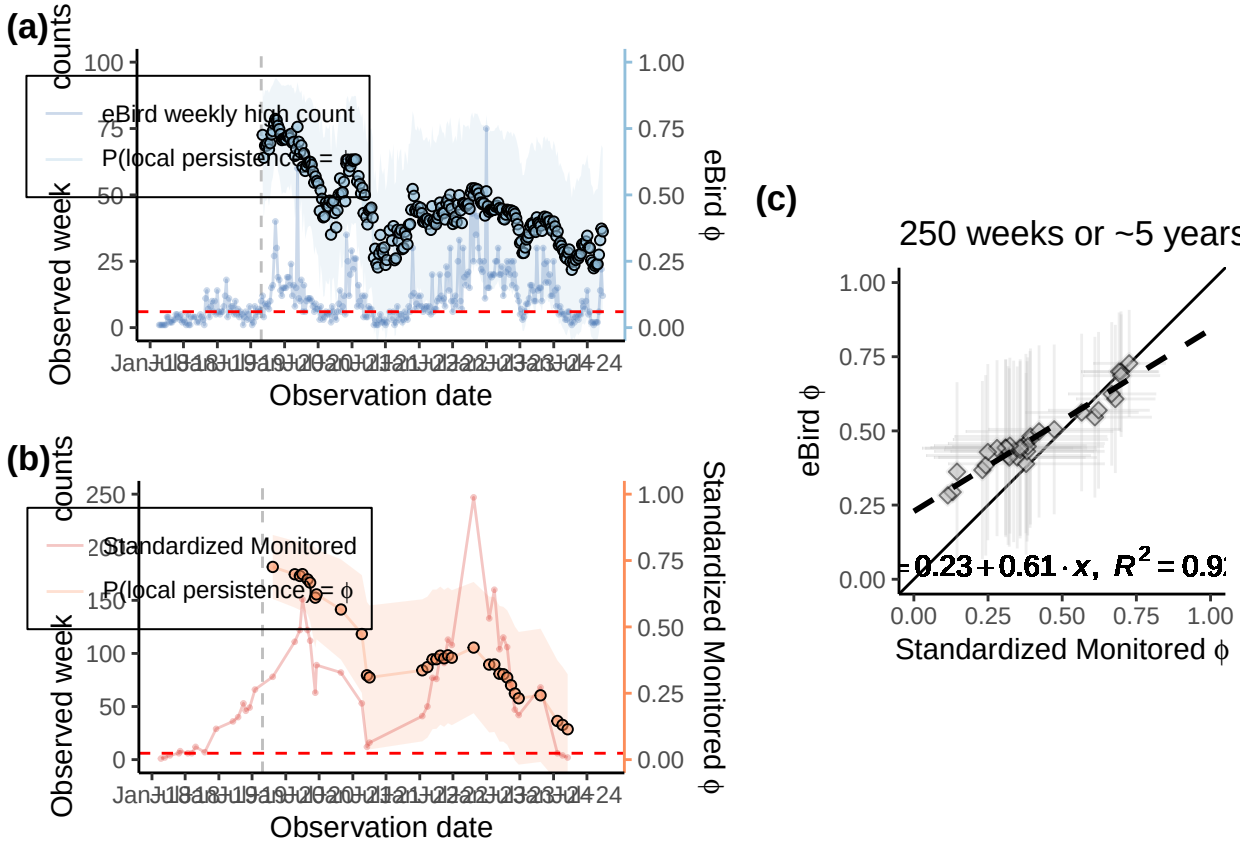
```
## Warning: Removed 229 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 229 rows containing missing values or values outside the scale range
## (`geom_pointrange()`).
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_segment()`).
```

```
## Warning: Removed 235 rows containing missing values or values outside the scale range
## (`geom_errorbarh()`).
```

```
## Warning: Removed 229 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
#It looks not good in the `Rmd`, but it is saved with good proportions
ggsave("results/Figure3_timeseries_PLocalPersi_250weeks.pdf",
       plot = Fig3, dpi = 300, width = 16, height = 6, units = "in")

ggsave("results/Figure3_timeseries_PLocalPersi250weeks.png",
       plot = Fig3, dpi = 300, width = 16, height = 6, units = "in")
```

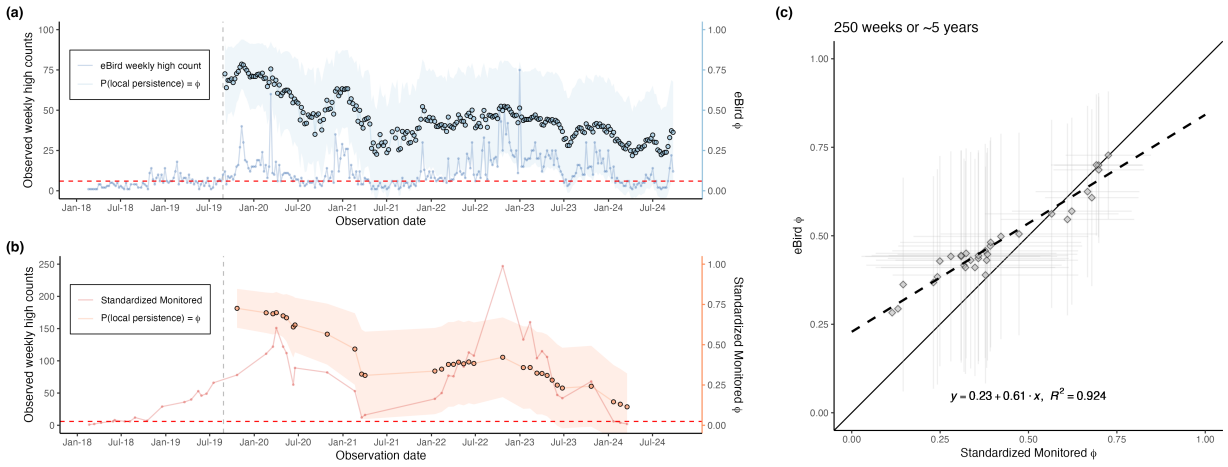


Figure SI-54: Time series of weekly high counts of Snail kites in Payne's Prairie wetland from eBird (a) and standardized monitoring surveys (b), with local persistence probability (ϕ) +/- standard deviation estimated for each dataset after 50,000 population trajectories simulated. (c) Local persistence probability extimated from eBird data plotted against local persistence probability estimated from standardized monitored, with standard deviation. Estimation of ϕ from 50,000 trajectories in a timeframe of ~5 years (250 weeks). Solid black line is the line of identity, while dashed black line is linear regression (equation in gray).

6.3 Results for ~10 years (500 weeks)

```
#Extended figures

phi_data_plot <- snailkites.PP |>
  left_join(phi.eBird |>
              full_join(phi.SM) |>
              filter(Timeframe == "500 weeks or ~10 years")) |>
  ungroup() |>
    mutate(phi.hat = phi.hat * coeff,
           SD.phi = SD.phi * coeff,
           phi.hatSM = phi.hatSM * coeffSM,
           SD.phiSM = SD.phiSM * coeffSM) |>
  pivot_longer(cols = !c(cell, year, week, Timeframe, Time.t, observation.date, Model),
               names_to = "group",
               values_to = "phi_values") |>
  drop_na(phi_values)
```

```
## Joining with `by = join_by(Time.t, Timeframe, Model)`
## Joining with `by = join_by(Time.t)`
```

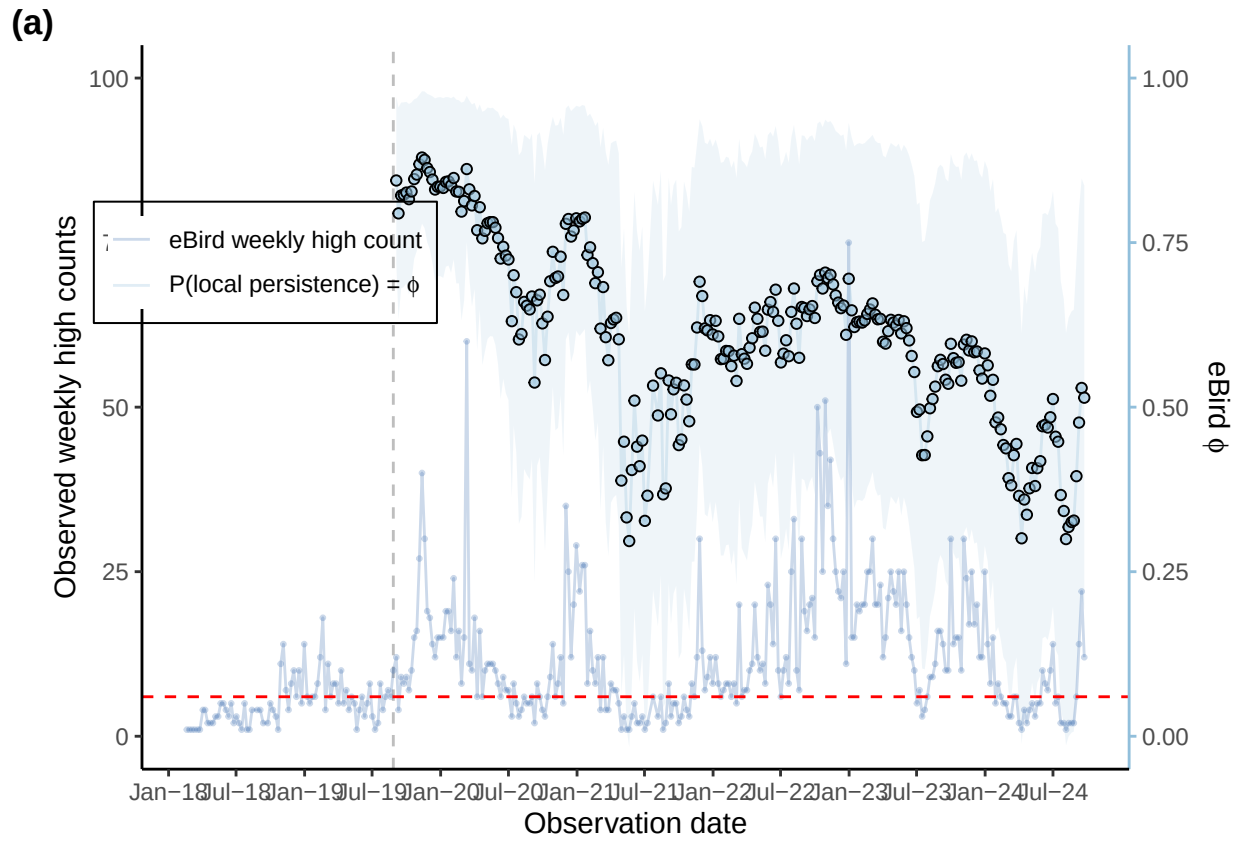
```
phi_data_plot
```

```
## # A tibble: 979 x 9
##       cell Time.t observation.date  year  week Timeframe Model group  phi_values
##      <dbl>  <dbl> <date>           <dbl> <dbl> <fct>     <chr> <chr>       <dbl>
##  1 2665918      8 2018-02-19        2018     8 <NA>      <NA>  Obser~          1
##  2 2665918      8 2018-02-19        2018     8 <NA>      <NA>  abund~          1
##  3 2665918      9 2018-02-26          NA    NA <NA>      <NA>  Obser~          1
##  4 2665918     10 2018-03-05          NA    NA <NA>      <NA>  Obser~          1
##  5 2665918     11 2018-03-12        2018    11 <NA>      <NA>  Obser~          1
##  6 2665918     11 2018-03-12        2018    11 <NA>      <NA>  abund~          2
```

```
##  7 2665918     12 2018-03-20          NA    NA <NA>      <NA>  Obser~          1
##  8 2665918     13 2018-03-26          NA    NA <NA>      <NA>  Obser~          1
##  9 2665918     14 2018-04-03          NA    NA <NA>      <NA>  Obser~          4
## 10 2665918     15 2018-04-09        2018    15 <NA>      <NA>  Obser~          4
## # i 969 more rows
```

```
#Ribbon lo-hi phi (SD) for the two datasets
ribbon_phi_eBird <- phi_data_plot |>
  filter(group %in% c("phi.hat", "SD.phi")) |>
  pivot_wider(names_from = group, values_from = phi_values, values_fn = {mean}) |>
  mutate(phi_values = phi.hat)

ribbon_phi_SM <- phi_data_plot |>
  filter(group %in% c("phi.hatSM", "SD.phiSM")) |>
  pivot_wider(names_from = group, values_from = phi_values, values_fn = {mean})|>
  mutate(phi_values = phi.hatSM)

  #figurea
Fig3aExt <- ggplot(phi_data_plot |>
                     filter(group %in% c("phi.hat", "Observed.y")),
                   aes(x = observation.date,
                       y = phi_values,
                       fill = group)) +
  geom_vline(xintercept = snailkites.PP$observation.date[78],
             color = "gray", linetype = "dashed") +
  geom_ribbon(data = ribbon_phi_eBird,
              aes(ymin = phi.hat-SD.phi, ymax = phi.hat+SD.phi),
              fill = "#91bfdb25") +
  geom_line(aes(color = group, linetype = group)) +
  geom_point(data = phi_data_plot |>
                     filter(group %in% c("Observed.y")),
             color = "#4575b445", size = 0.5, fill = "#4575b445", shape = 21)+
  geom_point(data = phi_data_plot |>
                     filter(group %in% c("phi.hat")) |>
               drop_na(Model),
             shape = 21, fill = "#91bfdb95", color = "black") +
  labs(x = "Observation date",
       y = "Observed weekly high counts",
       tag = expression(bold("(a)")),
       fill = "",
       color = "",
       linetype = "",
       shape = "") +
  scale_y_continuous(sec.axis = sec_axis(trans = ~./coeff,
                                         name = expression("eBird " * italic(phi)))) +
  coord_cartesian(ylim = c(0, coeff))+
  scale_x_date(limits = c(min(snailkites.PP$observation.date),
                          max(snailkites.PP$observation.date)),
               breaks = seq(as.Date("2018-01-01"), as.Date(Sys.Date( )),
                            by = "6 months"), date_labels="%b-%y")+
  scale_linetype_manual(values = c("solid",
                                   "solid"),
                        labels = c("eBird weekly high count",
                                   expression("P(local persistence) = " * italic(phi))))+
```

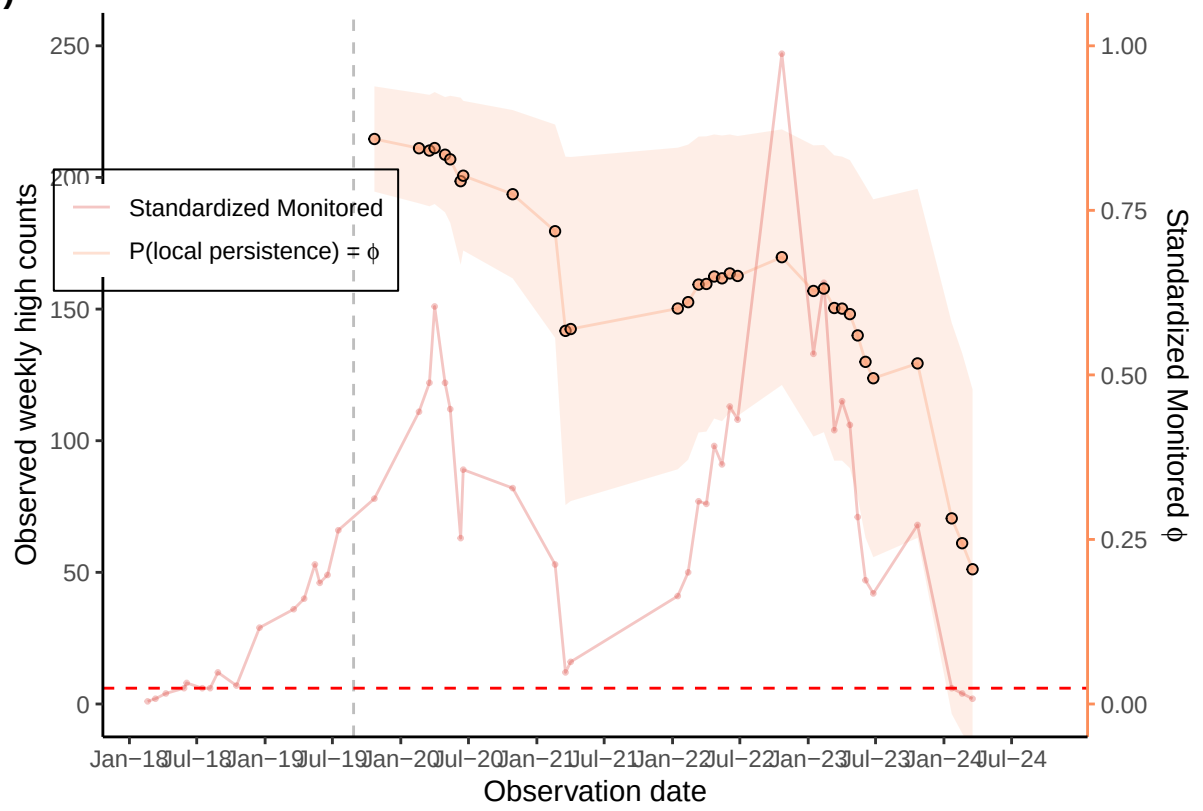
```
  scale_color_manual(values = c("#4575b445",
                                "#91bfdb45"),
                     labels = c("eBird weekly high count",
                                expression("P(local persistence) = " * italic(phi)))) +
  geom_hline(yintercept = N.critical, linetype = "dashed", color = "red")+
  theme_classic() +
  theme(legend.title = element_blank(),
        legend.position = c(0.125,0.7),
        legend.background = element_blank(),
        legend.box.background = element_rect(colour = "black"),
        axis.line.y.right = element_line(color = "#91bfdb"),
        axis.ticks.y.right = element_line(color = "#91bfdb"))
Fig3aExt
```




```
#SM figure

  #figurea
Fig3bExt <- ggplot(phi_data_plot |>
                     filter(group %in% c("phi.hatSM", "abundance.monitored")),
                   aes(x = observation.date,
                       y = phi_values,
                       fill = group)) +
  geom_vline(xintercept = snailkites.PP$observation.date[78],
             color = "gray", linetype = "dashed") +
  geom_ribbon(data = ribbon_phi_SM,
              aes(ymin = phi.hatSM-SD.phiSM, ymax = phi.hatSM+SD.phiSM),
```

```
              fill = "#fc8d5925") +
  geom_line(aes(color = group, linetype = group)) +
  geom_point(data = phi_data_plot |>
                  filter(group %in% c("abundance.monitored")),
             color = "#d7302745", size = 0.5, fill = "#d7302745", shape = 21)+
  geom_point(data = phi_data_plot |>
                  filter(group %in% c("phi.hatSM")) |>
               drop_na(Model),
             shape = 21, fill = "#fc8d5995", color = "black") +
  labs(x = "Observation date",
       y = "Observed weekly high counts",
       tag = expression(bold("(b)")),
       fill = "",
       color = "",
       linetype = "",
       shape = "") +
  scale_y_continuous(sec.axis = sec_axis(trans = ~./coeffSM,
                                         name = expression("Standardized Monitored " * italic(phi)))) +
  coord_cartesian(ylim = c(0, coeffSM))+
  scale_x_date(limits = c(min(snailkites.PP$observation.date),
                          max(snailkites.PP$observation.date)),
               breaks = seq(as.Date("2018-01-01"), as.Date(Sys.Date( )),
                            by = "6 months"), date_labels="%b-%y")+
  scale_linetype_manual(values = c("solid",
                                   "solid"),
                        labels = c("Standardized Monitored",
                                   expression("P(local persistence) = " * italic(phi))))+
  scale_color_manual(values = c("#d7302745",
                                "#fc8d5945"),
                     labels = c("Standardized Monitored",
                                expression("P(local persistence) = " * italic(phi)))) +
  geom_hline(yintercept = N.critical, linetype = "dashed", color = "red")+
  theme_classic() +
  theme(legend.title = element_blank(),
        legend.position = c(0.125,0.7),
        legend.background = element_blank(),
        legend.box.background = element_rect(colour = "black"),
        axis.line.y.right = element_line(color = "#fc8d59"),
        axis.ticks.y.right = element_line(color = "#fc8d59"))
Fig3bExt
```

(b)




```
#Performance comparison
phidata <- phi.eBird |>
             full_join(phi.SM) |>
  filter(Timeframe == "500 weeks or ~10 years")
```

```
## Joining with `by = join_by(Time.t, Timeframe, Model)`
```

```
#relationship?
lm(phi.hat~phi.hatSM, data = phidata)
```

```
##
## Call:
## lm(formula = phi.hat ~ phi.hatSM, data = phidata)
##
## Coefficients:
## (Intercept)    phi.hatSM
##      0.2881       0.5754
```

```
Fig3cExt <- ggplot(phidata)+
    geom_abline(slope = 1)+
    geom_pointrange(aes(x = phi.hatSM, y = phi.hat,
                        ymin = phi.hat-SD.phi,
                        ymax = phi.hat+SD.phi),
                    shape = 23, fill = "gray", color = "gray",
                    alpha = 0.25) +
    geom_errorbarh(aes(x = phi.hatSM, y = phi.hat,
                       xmax = phi.hatSM+SD.phiSM,
                       xmin = phi.hatSM-SD.phiSM, height = 0),
```

```
                  color = "gray",
                  alpha = 0.25)+
   geom_point(aes(x = phi.hatSM, y = phi.hat),
                   shape = 23, fill = "gray",
              color = "black", size = 2, alpha = 0.5)+
   geom_text(x = 0.5, y = 0.05,
           label = lm_eqn(df = phidata,
                          x = phidata$phi.hatSM,
                          y = phidata$phi.hat),
           parse = TRUE, color = "black")+
     geom_smooth(aes(x=phi.hatSM, y=phi.hat),
             method = "lm", fullrange=TRUE,
             color = "black", se = F, linetype = "dashed", size = 1)+
   scale_y_continuous(limits = c(0,1))+
   scale_x_continuous(limits = c(0,1))+
   labs(x = expression("Standardized Monitored "*italic(phi)),
        y = expression("eBird "*italic(phi)),
        tag = expression(bold("(c)")),
        title = "500 weeks or ~10 years")+
  coord_fixed()+
  theme_classic()
```

```
## Warning in geom_errorbarh(aes(x = phi.hatSM, y = phi.hat, xmax = phi.hatSM + :
## Ignoring unknown aesthetics: x
```

```
Fig3cExt
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

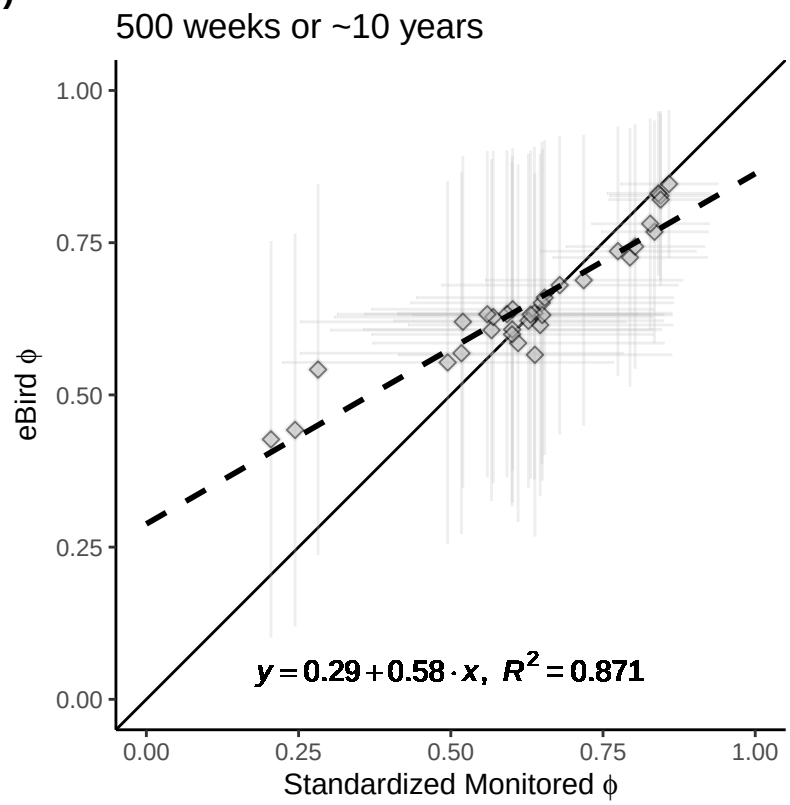
```
## Warning: Removed 229 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 229 rows containing missing values or values outside the scale range
## (`geom_pointrange()`).
```

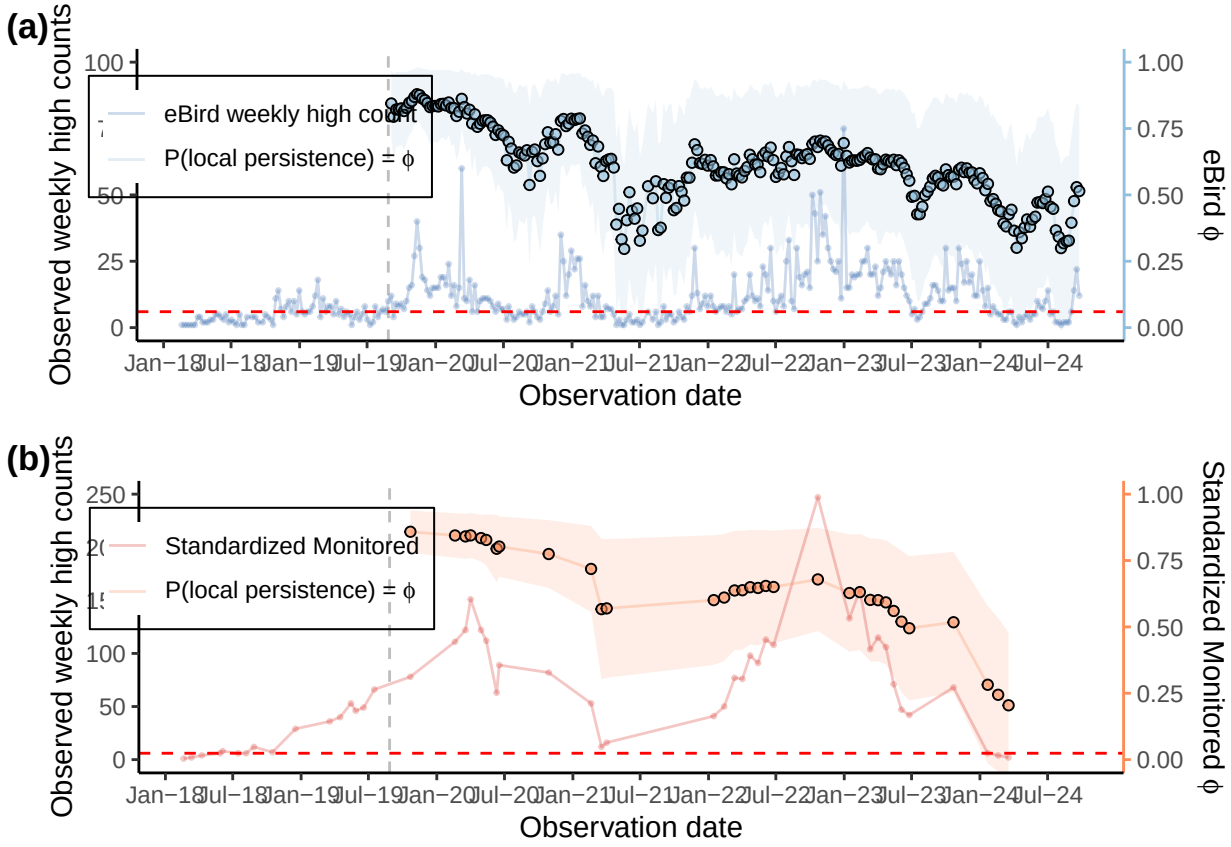
```
## Warning: Removed 232 rows containing missing values or values outside the scale range
## (`geom_errorbarh()`).
```

```
## Warning: Removed 229 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

(c)




```
#and combine in the figure
Fig3ab <- grid.arrange(Fig3aExt, Fig3bExt, ncol = 1)
```




```
Fig3 <- grid.arrange(Fig3ab, Fig3cExt, ncol = 2, widths = c(1.5,1))
```

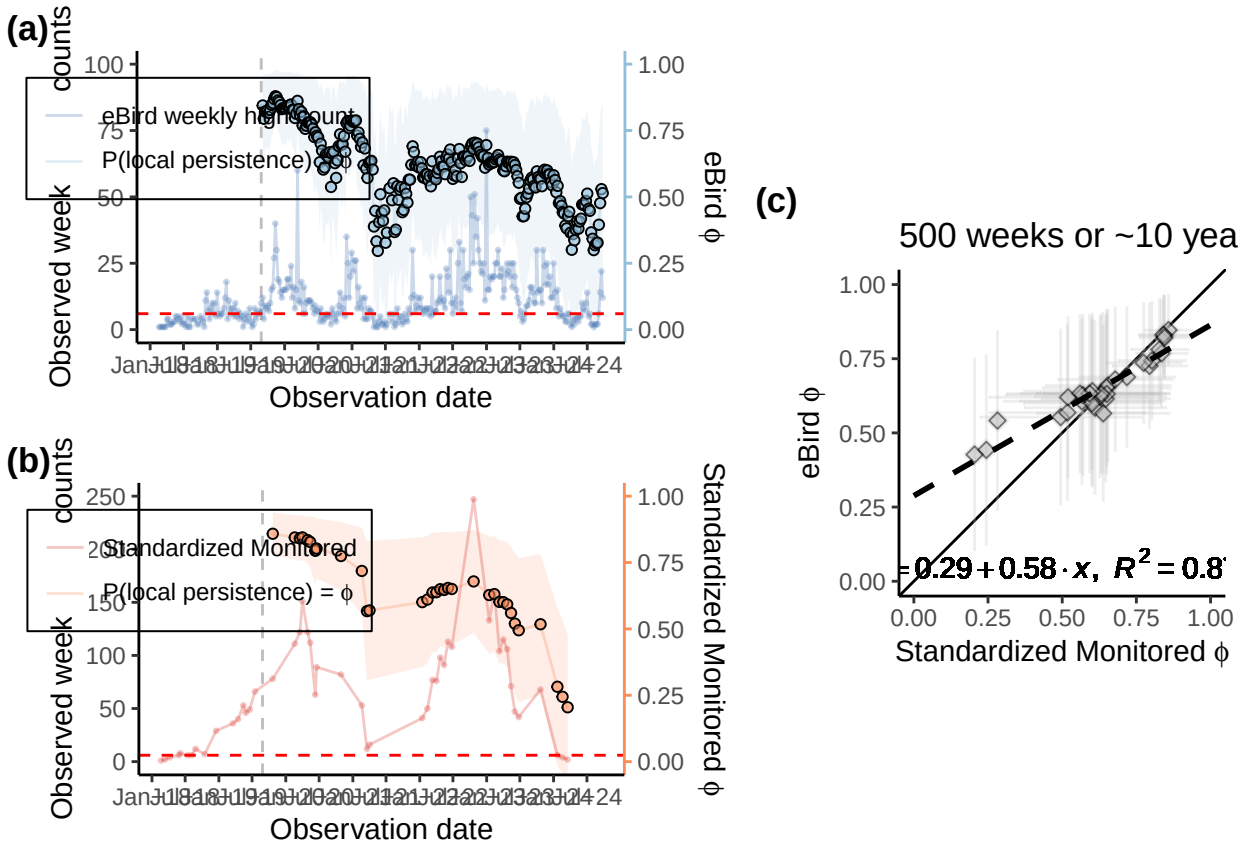
```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning: Removed 229 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 229 rows containing missing values or values outside the scale range
## (`geom_pointrange()`).
```

```
## Warning: Removed 232 rows containing missing values or values outside the scale range
## (`geom_errorbarh()`).
```

```
## Warning: Removed 229 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
#It looks not good in the `Rmd`, but it is saved with good proportions
ggsave("results/Figure3_timeseries_PLocalPersi_500weeks.pdf",
       plot = Fig3, dpi = 300, width = 16, height = 6, units = "in")

ggsave("results/Figure3_timeseries_PLocalPersi500weeks.png",
       plot = Fig3, dpi = 300, width = 16, height = 6, units = "in")
```

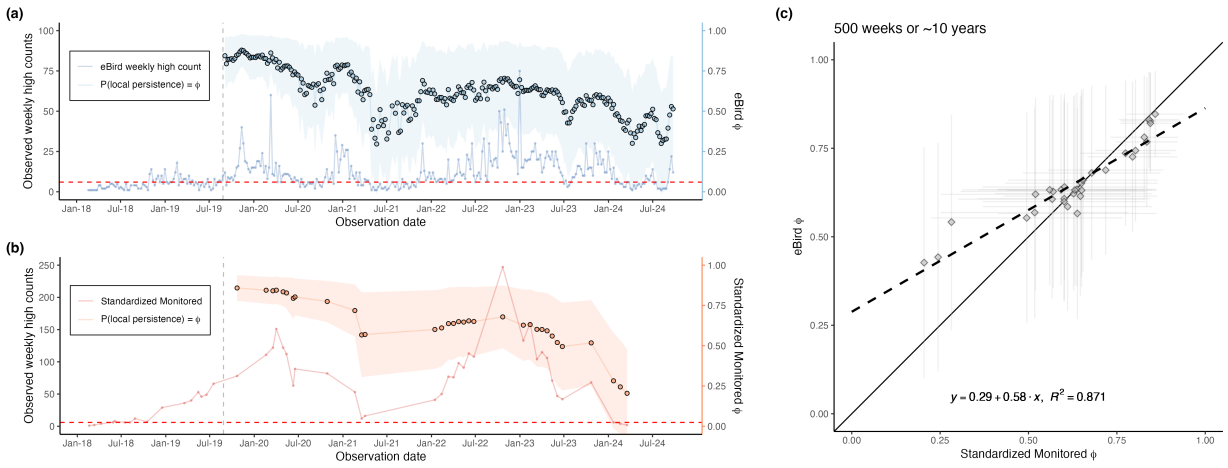


Figure SI-55: Time series of weekly high counts of Snail kites in Payne's Prairie wetland from eBird (a) and standardized monitoring surveys (b), with local persistence probability (ϕ) +/- standard deviation estimated for each dataset after 50,000 population trajectories simulated. (c) Local persistence probability extimated from eBird data plotted against local persistence probability estimated from standardized monitored, with standard deviation. Estimation of ϕ from 50,000 trajectories in a timeframe of ~10 years (500 weeks). Solid black line is the line of identity, while dashed black line is linear regression (equation in gray).

7 Sensitivity analysis

As a sensitivity analysis, we randomly removed 5% of the data in eBird, and projected under the timeframe of ~5 years (250 weeks). This process last ~13 hours.

```
#the original data
snailkites.PP <- readRDS("data_tmp/snailkitesPP.rds")

#recall the number of trajectories
ntraj =50000

# Define percentages for sensitivity analysis
percentages <- seq(0.95, 0.05, -0.05)

# Initialize lists to store results of the sensitivity analysis
results <- list()

# Loop over each percentage
for (p in percentages) {

  # Start timing for each percentage
  StartTime <- Sys.time()

  # Sample tt and yt for the current percentage p
  sampled_data <- snailkites.PP |>
    ungroup() |>
    drop_na(Observed.y) |>
    sample_frac(p) |>
    arrange(Time.t)

  # Extract tt and yt
  tt_sampled <- sampled_data$Time.t
  yt_sampled <- log(sampled_data$Observed.y)

  # Define N_critical
  N.critical <- round((1/2) * mean(exp(yt_sampled)),0)

  # End positions to modeling
  end_positions <- which(tt_sampled >= 88) #having the initial phi fixed

  #save   (SD)
  phi_results <- vector("list", length = length(end_positions))

  #save model used
  modelSS <- vector("list", length = length(end_positions))

  for (j in seq_along(end_positions)) {
    last.tt <- end_positions[j]
    m <- tt_sampled[last.tt]

    #only for ~five years
    #Forcing EGSS
    model <- "EGSS"
    modelSS[[j]] <- model
```

```
    EGSS.partial <- egss_reml(yt = yt_sampled[1:last.tt],
                              tt = tt_sampled[1:last.tt],
                              fguess = guess_egss(yt = yt_sampled[1:last.tt],
                                                  tt = tt_sampled[1:last.tt]));

      thres.times <- as.numeric(0:(250)) #~5 years
      len <- max(thres.times) + 1

      sim.mat.eBird <- egss_sim(ntraj,
                              tt = thres.times,
                              parms = EGSS.partial$remles)

      phi <- rep(0, ntraj)
      last.points <- rep(0, ntraj)

      for(n in 1:ntraj){
        Pop.sim <- exp(c(yt_sampled[last.tt], sim.mat.eBird[-1,n]));
        last.points[n] <- Pop.sim[len]

        # How many times each trajectory go below the threshold?
        below.threshold <- sum(Pop.sim < N.critical)
        phi[n] <- 1-(below.threshold/len)
      }

      #Expected value of probability of local persistence, and SD
      phi_mean <- mean(phi)
      phi_SD <- sqrt(var(phi))

      phi_results[[j]] <- cbind(phi_mean,phi_SD)

    #see advance by printing results
    print(cbind(c(p*100),model,
                (phi_mean-phi_SD),
                phi_mean,
                (phi_mean+phi_SD)))

    # Store results in lists for each percentage
    phi_results[[j]] <- cbind(phi_mean,phi_SD)
    modelSS[[j]] <- model
  }

  # End timing for this percentage
  EndTime <- Sys.time()
  timelast <- EndTime - StartTime

  # Store results for this percentage
  results[[paste0("p", p*100)]] <- list(
    Time.t = tt_sampled[end_positions],
    phi_sensitivity = phi_results,
    modelSS = modelSS,
    time_taken = timelast
  )
}
```

Recover and save the results as data frame for comparison (with an iterative process for each percentage).

```r
# Initialize an empty data frame to store results
phiSD_df_sensitivity <- data.frame(
  Percentage = numeric(),
  Time.t = numeric(),
  phi.hat = numeric(),
  SD.phi = numeric(),
  Model = character(),
  Time_Taken = character()
)

# Loop through each percentage in the results
for (p in names(results)) {
  for (j in seq_along(results[[p]]$phi_sensitivity)) {
    # Extract data for this specific percentage and end position
    phi_hat <- results[[p]]$phi_sensitivity[[j]][1]
    SD_phi <- results[[p]]$phi_sensitivity[[j]][2]
    model <- results[[p]]$modelSS[[j]]
    time_t <- results[[p]]$Time.t[j]
    time_taken <- results[[p]]$time_taken

    # Combine the extracted data into a data frame
    temp_df <- data.frame(
      Percentage = p,
      Time.t = time_t,
      phi.hat = phi_hat,
      SD.phi = SD_phi,
      Model = model,
      Time_Taken = time_taken
    )

    # Bind the temp_df to the final data frame
    phiSD_df_sensitivity <- rbind(phiSD_df_sensitivity, temp_df)
  }
}

head(phiSD_df_sensitivity)
tail(phiSD_df_sensitivity)

# Save the data frame to a file
saveRDS(phiSD_df_sensitivity, "results/phi_SD_Sensitivity.rds")
```

```
> head(phiSD_df_sensitivity)
  Percentage Time.t   phi.hat    SD.phi Model     Time_Taken
1        p95     88 0.7426800 0.1916598  EGSS 1.665098 hours
2        p95     89 0.6602792 0.2320728  EGSS 1.665098 hours
3        p95     90 0.7030531 0.2102578  EGSS 1.665098 hours
4        p95     91 0.7043085 0.2093648  EGSS 1.665098 hours
5        p95     93 0.6891339 0.2168744  EGSS 1.665098 hours
6        p95     94 0.7111683 0.2023171  EGSS 1.665098 hours
> tail(phiSD_df_sensitivity)
     Percentage Time.t   phi.hat    SD.phi Model       Time_Taken
2506         p5    224 0.6911869 0.2498712  EGSS 0.05632068 hours
2507         p5    313 0.8460566 0.0205282  EGSS 0.05632068 hours
2508         p5    318 0.6643653 0.1126319  EGSS 0.05632068 hours
2509         p5    342 0.3942388 0.3087640  EGSS 0.05632068 hours
2510         p5    346 0.4055800 0.3278764  EGSS 0.05632068 hours
2511         p5    349 0.5856061 0.2560301  EGSS 0.05632068 hours
```

Figure SI-56: head and tail of the sensitivity data frame

With the data saved, we can compare how the reduction of data affect the inference.

```
#Load data saved
  #data organized
  datesPP <- readRDS("data_tmp/datesTimeseriesPaynesPrairie.rds")
  #local persistence from Standardized Monitored
  phi.SM <-readRDS("results/phi_hat_SD_SM.rds") |>
    left_join(datesPP) |>
    filter(Timeframe == "250 weeks or ~5 years")
```

```
## Joining with `by = join_by(Time.t)`
```

```
  #local persistence from eBird
  phi.eBird <- readRDS("results/phi_hat_SD_eBird.rds") |>
    left_join(datesPP) |>
    filter(Timeframe == "250 weeks or ~5 years")
```

```
## Joining with `by = join_by(Time.t)`
```

```
  #data reduced
  phiSD_df_sensitivity <- readRDS("results/phi_SD_Sensitivity.rds") |>
    left_join(datesPP) |>
    full_join(phi.eBird |>
                mutate(Percentage = "All")) |>
    mutate(Percentage = case_when(Percentage == "p9.99999999999999"~"p10",
                                  Percentage != "p9.99999999999999"~Percentage))
```

```
## Joining with `by = join_by(Time.t)`
## Joining with `by = join_by(Percentage, Time.t, phi.hat, SD.phi, Model,
## observation.date)`
```

For example, using Pearson correlation coefficient

```
#generate a dataframe to estimate correlation and make a figure
cor_sen <- phiSD_df_sensitivity |>
  left_join(phi.SM, by = c("Time.t", "observation.date"))

#correlation estimation for each percentage
cor_sen |>
  group_by(Percentage) |>
  summarise(correlation = cor(phi.hatSM,
                              phi.hat,
                              use = "pairwise.complete.obs"),
            n_obs = n())
```

```
## # A tibble: 20 x 3
##    Percentage correlation n_obs
##    <chr>            <dbl> <int>
##  1 All              0.961   262
##  2 p10              0.782    25
##  3 p15              0.988    39
##  4 p20              1        50
##  5 p25              0.861    65
##  6 p30              0.933    77
##  7 p35              0.797    99
##  8 p40              0.964   111
##  9 p45              0.971   120
## 10 p5              NA        12
## 11 p50              0.954   128
```

```
## 12 p55              0.965   146
## 13 p60              0.945   169
## 14 p65              0.975   171
## 15 p70              0.961   184
## 16 p75              0.967   197
## 17 p80              0.966   210
## 18 p85              0.956   226
## 19 p90              0.961   232
## 20 p95              0.959   250
```

```
#overlapping weeks
cor_sen |>
  filter(Timeframe.y == "250 weeks or ~5 years") |>
  group_by(Percentage) |>
  summarise(correlation = cor(phi.hatSM,
                              phi.hat,
                              use = "pairwise.complete.obs"),
            n_obs = n())
```

```
## # A tibble: 20 x 3
##    Percentage correlation n_obs
##    <chr>            <dbl> <int>
##  1 All              0.961    33
##  2 p10              0.782     5
##  3 p15              0.988     3
##  4 p20              1         2
##  5 p25              0.861     6
##  6 p30              0.933     8
##  7 p35              0.797    10
##  8 p40              0.964    20
##  9 p45              0.971    17
## 10 p5              NA         1
## 11 p50              0.954    21
## 12 p55              0.965    19
## 13 p60              0.945    23
## 14 p65              0.975    21
## 15 p70              0.961    23
## 16 p75              0.967    23
## 17 p80              0.966    28
## 18 p85              0.956    29
## 19 p90              0.961    32
## 20 p95              0.959    33
```

```
#labeller object for the `facet_wrap()`
sensilab <- c(
  'All' = "Complete (n = 262; 33)",
  'p95' = "95% dataset (n = 250; 33)",
  'p90' = "90% dataset (n = 232; 32)",
  'p85' = "85% dataset (n = 226; 29)",
  'p80' = "80% dataset (n = 210; 28)",
  'p75' = "75% dataset (n = 197; 23)",
  'p70' = "70% dataset (n = 184; 23)",
  'p65' = "65% dataset (n = 171; 21)",
  'p60' = "60% dataset (n = 169; 23)",
  'p55' = "55% dataset (n = 146; 19)",
```

```
  'p50' = "50% dataset (n = 128; 21)",
  'p45' = "45% dataset (n = 120; 17)",
  'p40' = "40% dataset (n = 111; 20)",
  'p35' = "35% dataset (n = 99; 10)",
  'p30' = "30% dataset (n = 77; 8)",
  'p25' = "25% dataset (n = 65; 6)",
  'p20' = "20% dataset (n = 50; 2)",
  'p15' = "15% dataset (n = 39; 3)",
  'p10' = "10% dataset (n = 25; 5)",
  'p5' = "5% dataset (n = 12; 1)"
)

Figure5 <- ggplot(cor_sen,
       aes(x = phi.hatSM, y = phi.hat))+
  geom_abline(slope = 1)+
  facet_wrap(~factor(Percentage,
                     levels = c("All","p95","p90","p85","p80",
                                "p75","p70","p65","p60","p55",
                                "p50","p45","p40","p35","p30",
                                "p25","p20","p15","p10","p5")),
             ncol = 5,
             labeller = as_labeller(sensilab))+
  geom_point()+
  geom_smooth(method = "lm", fullrange = T)+
  stat_regline_equation(label.x = 0.35, label.y = 0.2, size = 2.5) +
  stat_cor(method = "pearson",
         aes(label = paste("rho == ", ..r.., "*','~~p == ", ..p..)),
         label.x = 0.25, label.y = 0.1, size = 2.5, parse = TRUE)+
  labs(x = expression(phi[SM]),
       y = expression(phi[eBird]),
       title = expression("Persistence probability ("~phi~") comparison"))+
  scale_x_continuous(limits = c(0,1))+
  scale_y_continuous(limits = c(0,1))+
  coord_equal(ratio = 1)+
  theme_classic()
```

```
## Warning: Duplicated aesthetics after name standardisation: parse
```

```
Figure5
```

```
## Warning: The dot-dot notation (`..r..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(r)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

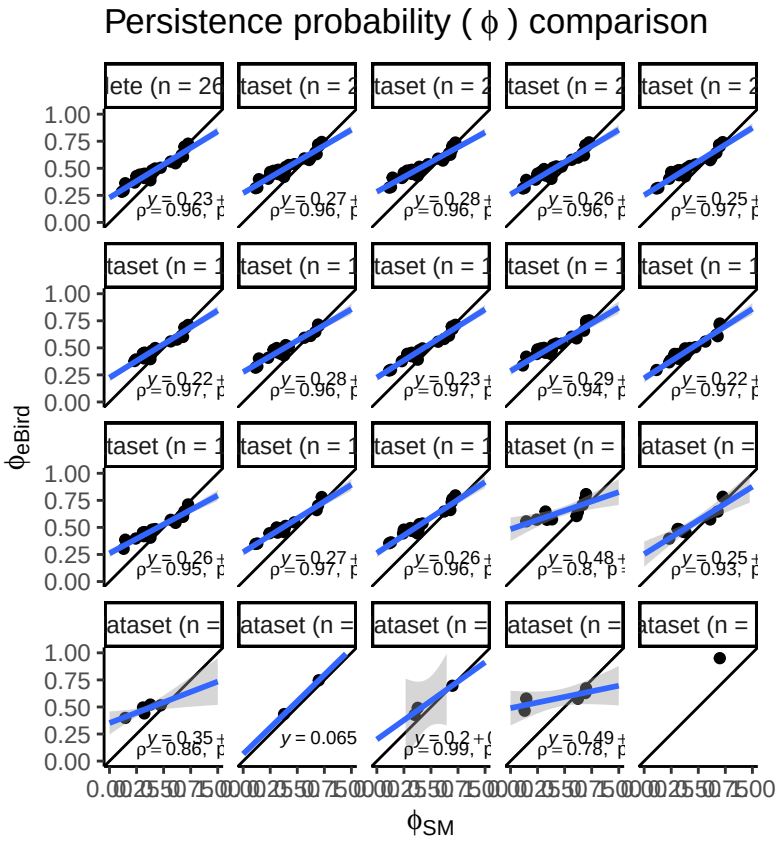
```
## Warning: Removed 2416 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning in qt((1 - level)/2, df): NaNs produced
```

```
## Warning: Removed 2416 rows containing non-finite outside the scale range
## (`stat_regline_equation()`).
```

```
## Warning: Removed 2416 rows containing non-finite outside the scale range
```

```
## (`stat_cor()`).

## Warning: Computation failed in `stat_cor()`.
## Caused by error in `cor.test.default()`:
## ! not enough finite observations

## Warning: Removed 2416 rows containing missing values or values outside the scale range
## (`geom_point()`).

## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_smooth()`).

## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```




```
#lets saved it with good proportions
ggsave("results/Figure5_Sensitivity_correlations.pdf",
       plot = Figure5, dpi = 300, width = 9, height = 8, units = "in")
```

```
## `geom_smooth()` using formula = 'y ~ x'

## Warning: Removed 2416 rows containing non-finite outside the scale range
## (`stat_smooth()`).

## Warning in qt((1 - level)/2, df): NaNs produced

## Warning: Removed 2416 rows containing non-finite outside the scale range
## (`stat_regline_equation()`).

## Warning: Removed 2416 rows containing non-finite outside the scale range
## (`stat_cor()`).
```

```
## Warning: Computation failed in `stat_cor()`.
## Caused by error in `cor.test.default()`:
## ! not enough finite observations
```

```
## Warning: Removed 2416 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_smooth()`).
```

```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```

```
ggsave("results/Figure5_Sensitivity_correlations.png",
       plot = Figure5, dpi = 300, width = 9, height = 8, units = "in")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning: Removed 2416 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning in qt((1 - level)/2, df): NaNs produced
```

```
## Warning: Removed 2416 rows containing non-finite outside the scale range
## (`stat_regline_equation()`).
```

```
## Warning: Removed 2416 rows containing non-finite outside the scale range
## (`stat_cor()`).
```

```
## Warning: Computation failed in `stat_cor()`.
## Caused by error in `cor.test.default()`:
## ! not enough finite observations
```

```
## Warning: Removed 2416 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_smooth()`).
```

```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```

And we can see the differences in a figure

```
Fig5Ext <- ggplot(phiSD_df_sensitivity, aes(x = observation.date,
                                    y = phi.hat))+
  geom_ribbon(aes(ymin = phi.hat-SD.phi,
                      ymax = phi.hat+SD.phi),
              fill = "#91bfdb25")+
  geom_ribbon(data = phi.SM,
              aes(x = observation.date,
                  y = phi.hatSM,
                  ymin = phi.hatSM-SD.phiSM,
                  ymax = phi.hatSM+SD.phiSM),
              fill = "#fc8d5925")+
  geom_line(color = "#91bfdb95")+
  geom_point(aes(shape = Model),
              size = 2, color = "#91bfdb95")+
  geom_line(data = phi.SM,
              aes(x = observation.date,
                  y = phi.hatSM),
```

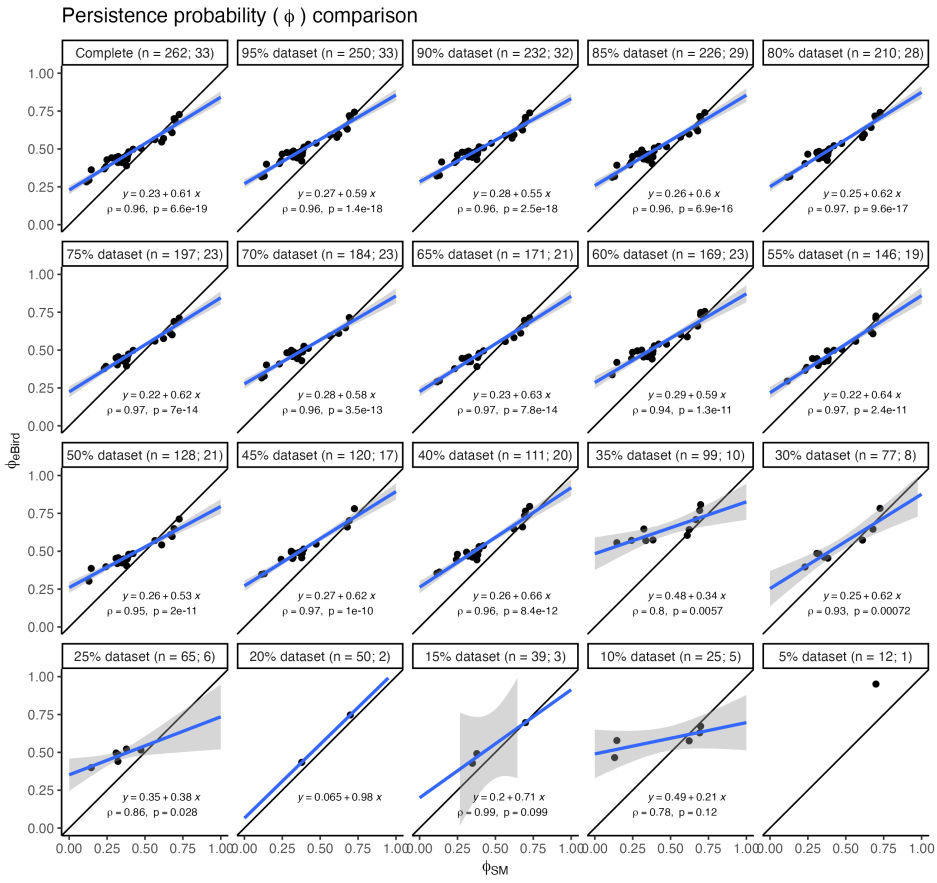
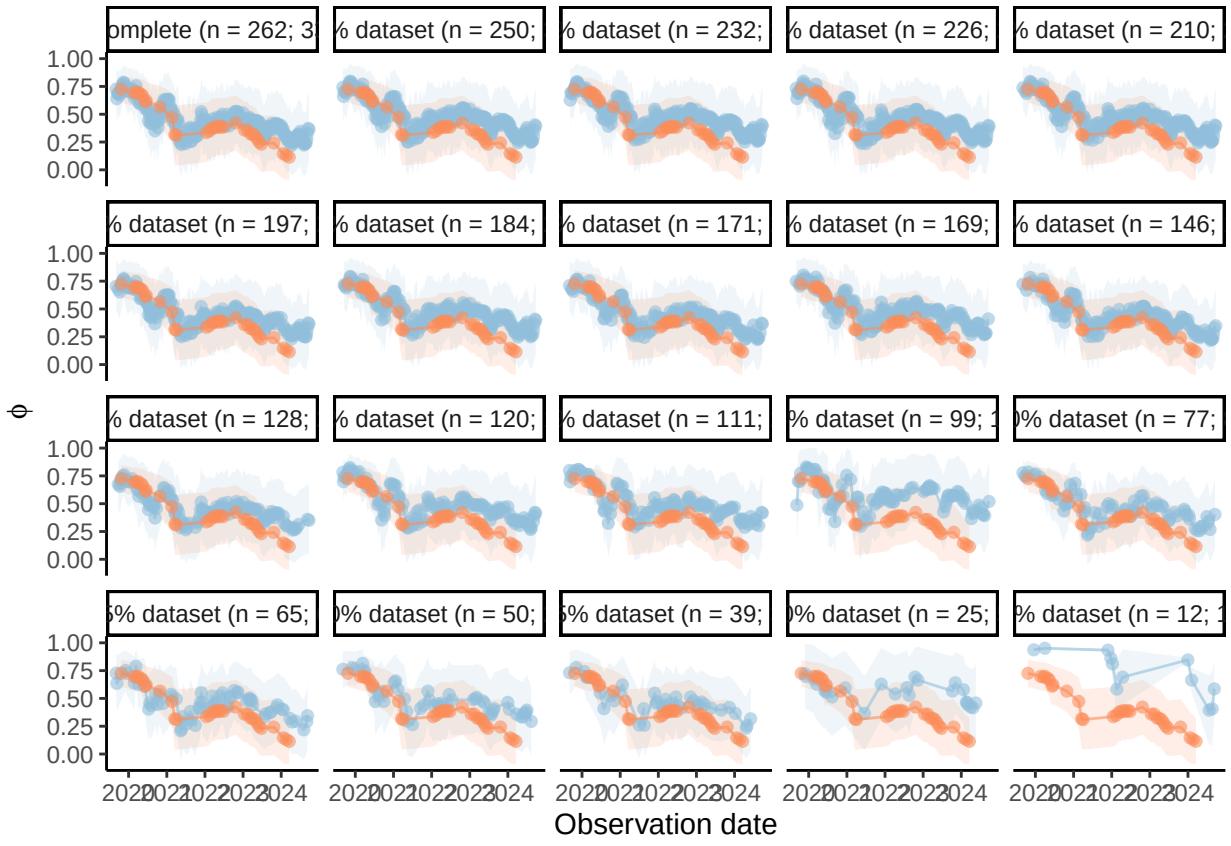


Figure SI-57: *Figure 5 in Main text.* Relationship of estimated local persistence probability ($\hat{\phi}$) from eBird (ordinate) and standardized monitoring (abscissa) under data reduction as sensitivity analysis.

```
            color = "#fc8d5995")+
  geom_point(data = phi.SM,
             aes(x = observation.date,
                 y = phi.hatSM,
                 shape = Model),
             size = 2, color = "#fc8d5995")+
  facet_wrap(~factor(Percentage,
                     levels = c("All","p95","p90","p85","p80",
                                "p75","p70","p65","p60","p55",
                                "p50","p45","p40","p35","p30",
                                "p25","p20","p15","p10","p5")),
             ncol = 5,
             labeller = as_labeller(sensilab))+
  labs(x = "Observation date",
       y = expression(phi))+
  theme_classic() +
  theme(legend.position = "none")

Fig5Ext
```




```
#lets saved it with good proportions
ggsave("results/Figure5ext_sensitivity_datareduction.pdf",
       plot = Fig5Ext, dpi = 300, width = 10, height = 7, units = "in")

ggsave("results/Figure5ext_sensitivity_datareduction.png",
       plot = Fig5Ext, dpi = 300, width = 10, height = 7, units = "in")
```

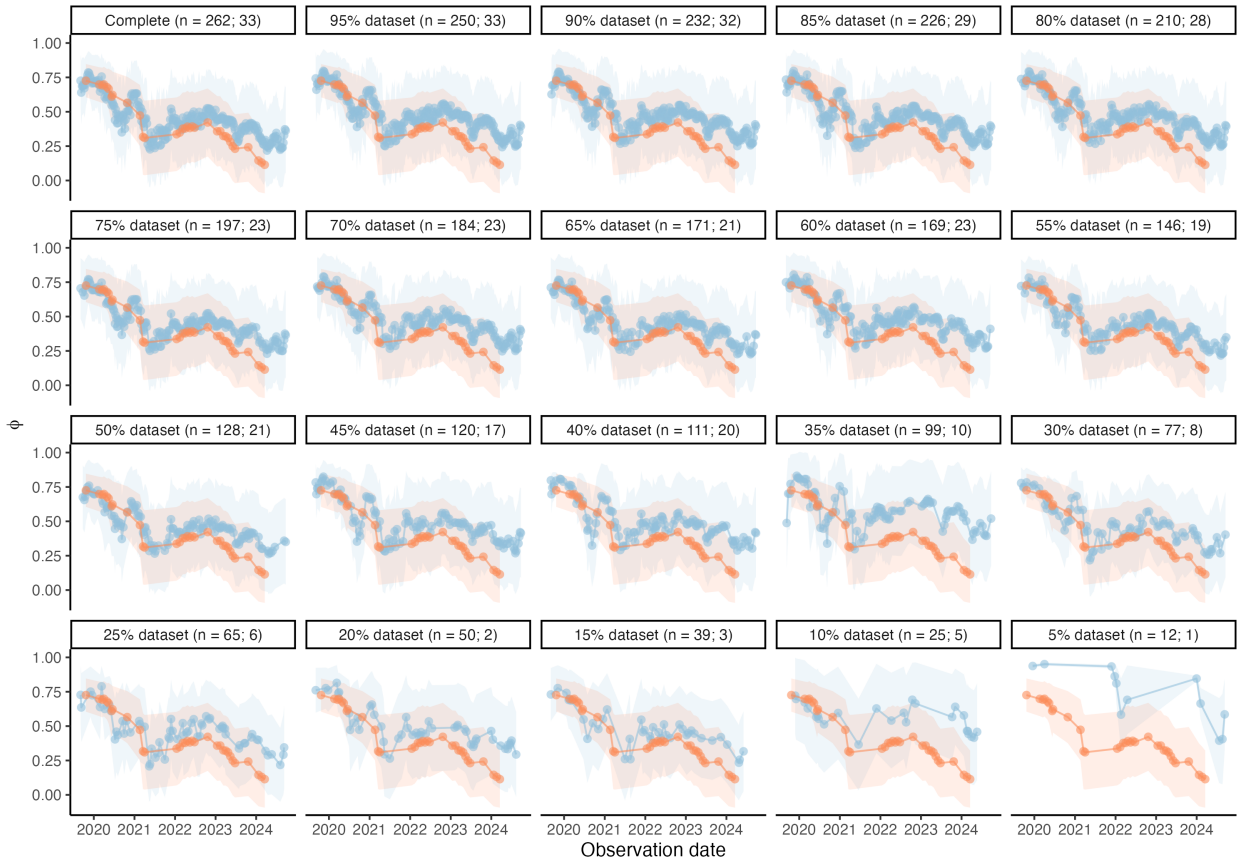


Figure SI-58: *Extended figure for Figure 5 in the Main text.* Data reduction as sensitivity analysis. Estimation of local persistence probability (ϕ) for standardized monitoring (orange) is replicated in each panel. Sequence of panels represent reduction from the complete dataset (same as Figure 4b *in the Main text*) to different percentages of randomly subsampled data, from top-left to bottom-right. Point shapes represent fitted model, the EGSS (circle) or OUSS (triangle).

8 Other populations of Snail Kite

```
#load saved data
SnailKite <- readRDS("data_tmp/SnailKiteCellsID_filtered.rds")

SnailKite |>
  group_by(cell) |>
  filter(year >= 2018) |>
  summarise(count = n(),
            mu_lat = mean(latitude),
            mu_lon = mean(longitude)) |>
  filter(count >= 100) |>
  arrange(mu_lat)
```

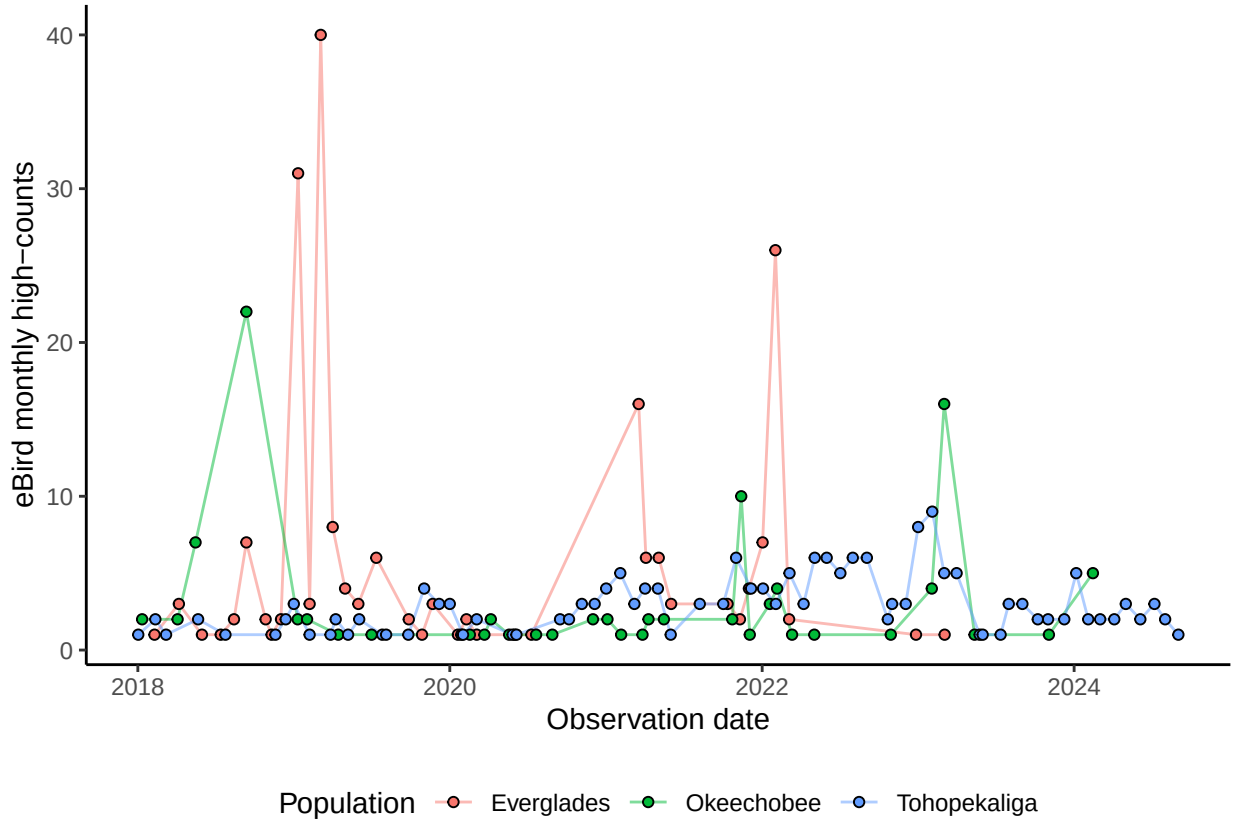
```
## # A tibble: 26 x 4
##       cell count mu_lat mu_lon
##      <dbl> <int>  <dbl>  <dbl>
##  1 2702386   152   25.8  -80.7
##  2 2696545   254   26.0  -81.6
##  3 2695817   109   26.1  -81.5
##  4 2698017   436   26.5  -80.2
##  5 2691441   105   26.6  -81.7
##  6 2690712   147   26.6  -81.7
##  7 2690713   884   26.6  -81.7
##  8 2694367   300   26.7  -80.7
##  9 2695831   250   26.8  -80.1
## 10 2695101   261   26.8  -80.2
## # i 16 more rows
```

```
sk.others <- SnailKite |>
  filter(cell %in% c("2702386", #Central Everglades
                     "2690719", #West site of Lake Okeechobee
                     "2677594")) |> #North Lake Tohopekaliga
  mutate(cell = case_when(cell == "2702386"~"Everglades",
                          cell == "2690719"~"Okeechobee",
                          cell == "2677594"~"Tohopekaliga"))

sk.others.ts <- sk.others |>
  group_by(cell) |>
  filter(observation_date >= "2018-01-01") |>
  mutate(Time.t = case_when(year == 2018 ~ month,
                            year > 2018 ~ month+(12*(year-2018)))) |>
  group_by(cell, Time.t) |>
  summarise(Observed.y = round(max(max_count),0),
            observation_date = min(observation_date))
```

```
## `summarise()` has grouped output by 'cell'. You can override using the
## `.groups` argument.
```

```
ggplot(sk.others.ts, aes(x = observation_date,
                         y = Observed.y,
                         fill = cell))+
  geom_line(aes(color = cell), alpha = 0.5)+
  geom_point(shape = 21)+
  labs(x = "Observation date",
```

```
      y = "eBird monthly high-counts",
      color = "Population",
      fill = "Population")+
  theme_classic() +
  theme(legend.position = "bottom")
```



And iterate for each cell and each week with data of eBird high count

```
# Initialize lists to store results of the sensitivity analysis
sk.others.results <- list()

ntraj = 50000

cells <- c(unique(sk.others.ts$cell))

# Loop over each percentage
for (k in cells) {

  # Start timing for this percentage
  StartTime <- Sys.time()

  # Sample tt and yt for the current percentage p
  cell_data <- sk.others.ts |>
    ungroup() |>
    drop_na(Observed.y) |>
    filter(cell == k) |>
```

```
  arrange(Time.t)

# Extract tt and yt
tt_sampled <- cell_data$Time.t
yt_sampled <- log(cell_data$Observed.y)

# Define N_critical
N.critical <- (1/2) * mean(exp(yt_sampled))

#Initial time vector (for fist model fitting)
init.tt <- min(tt_sampled) + 12

# End positions to modeling
end_positions <- which(tt_sampled >= init.tt)

#save   (SD)
phi_results <- vector("list", length = length(end_positions))

#save model used
modelSS <- vector("list", length = length(end_positions))

#save trend parameter
pop.growth.rate <- vector("list", length = length(end_positions))

for (j in seq_along(end_positions)) {
  last.tt <- end_positions[j]
  m <- tt_sampled[last.tt]

  #only for ~5 years

  OUSS.partial <- ouss_reml(yt = yt_sampled[1:last.tt],
                            tt = tt_sampled[1:last.tt],
                            fguess = guess_ouss(yt = yt_sampled[1:last.tt],
                                                tt = tt_sampled[1:last.tt]))

  model <- if(OUSS.partial$remles[2] < 0.025){
    "EGSS"
  }else{
    "OUSS"
  }
  modelSS[[j]] <- model

  if(model == "OUSS"){

    #Transformation in Dennis & Ponciano (Ecology 2014. pg 2072):
      #a = mu(1-e^(-theta))
    trendP <- (OUSS.partial$remles[1])*(1-exp(-OUSS.partial$remles[2]))

    thres.times <- as.numeric(0:(60)) #~5 years
    len <- max(thres.times) + 1

    sim.mat.eBird <- ouss_sim(ntraj,
                              tt = thres.times,
```

```
                              parms = OUSS.partial$remles)

  phi <- rep(0, ntraj)
  last.points <- rep(0, ntraj)

  for(n in 1:ntraj){
    Pop.sim <- exp(c(yt_sampled[last.tt], sim.mat.eBird[-1,n]));
    last.points[n] <- Pop.sim[len]

  # How many times each trajectory go below the threshold?
    below.threshold <- sum(Pop.sim < N.critical)
    phi[n] <- 1-(below.threshold/len)
  }

  #Expected value of probability of local persistence, and SD
  phi_mean <- mean(phi)
  phi_SD <- sqrt(var(phi))

}
else{

  EGSS.partial <- egss_reml(yt = yt_sampled[1:last.tt],
                            tt = tt_sampled[1:last.tt],
                            fguess = guess_egss(yt = yt_sampled[1:last.tt],
                                                tt = tt_sampled[1:last.tt]));

  trendP <- EGSS.partial$remles[1]

  thres.times <- as.numeric(0:(60)) #~5 years
  len <- max(thres.times) + 1

  sim.mat.eBird <- egss_sim(ntraj,
                            tt = thres.times,
                            parms = EGSS.partial$remles)

  phi <- rep(0, ntraj)
  last.points <- rep(0, ntraj)

  for(n in 1:ntraj){
    Pop.sim <- exp(c(yt_sampled[last.tt], sim.mat.eBird[-1,n]));
    last.points[n] <- Pop.sim[len]

  # How many times each trajectory go below the threshold?
    below.threshold <- sum(Pop.sim < N.critical)
    phi[n] <- 1-(below.threshold/len)
  }

  #Expected value of probability of local persistence, and SD
  phi_mean <- mean(phi)
  phi_SD <- sqrt(var(phi))

}
```

```
    #see advance by printing results
    print(cbind(k, #cell ID
                m, #week modeled
                model, #model selected
                round(trendP, 3), #trend parameter
                (phi_mean-phi_SD), #lower ribbon
                phi_mean, #Expected value
                (phi_mean+phi_SD))) #higher ribbon

      # Store results in lists for each percentage
      phi_results[[j]] <- cbind(phi_mean,phi_SD)
      modelSS[[j]] <- model
      pop.growth.rate[[j]] <- trendP
  }

  # End timing for this percentage
  EndTime <- Sys.time()
  timelast <- EndTime - StartTime

  # Store results for this percentage
  sk.others.results[[paste0("cell - ", k)]] <- list(
    Time.t = tt_sampled[end_positions],
    phi_others = phi_results,
    modelSS = modelSS,
    trendParm = trendP,
    time_taken = timelast
  )
}
```

```
##      k            m    model                              phi_mean
## [1,] "Everglades" "14" "EGSS" "0.129" "0.843256940256938" "0.874449508196721"
##
## [1,] "0.905642076136504"
##      k            m    model                             phi_mean
## [1,] "Everglades" "15" "EGSS" "0.23" "0.898751405291848" "0.926163278688525"
##
## [1,] "0.953575152085201"
##      k            m    model                              phi_mean
## [1,] "Everglades" "16" "EGSS" "0.173" "0.876875308651254" "0.904223606557377"
##
## [1,] "0.9315719044635"
##      k            m    model                              phi_mean
## [1,] "Everglades" "17" "EGSS" "0.115" "0.824301470531849" "0.857550163934426"
##
## [1,] "0.890798857337003"
##      k            m    model                              phi_mean
## [1,] "Everglades" "18" "EGSS" "0.079" "0.752569628124086" "0.793214754098361"
##
## [1,] "0.833859880072635"
##      k            m    model                              phi_mean
## [1,] "Everglades" "19" "EGSS" "0.091" "0.784930534853857" "0.822203606557377"
##
## [1,] "0.859476678260897"
##      k            m    model                              phi_mean
```

```
## [1,] "Everglades" "21" "OUSS" "0.089" "0.315307530803166" "0.473008524590164"
##
## [1,] "0.630709518377162"
##      k            m    model                              phi_mean
## [1,] "Everglades" "22" "OUSS" "0.148" "0.320988001805674" "0.443425573770492"
##
## [1,] "0.565863145735309"
##      k            m    model                             phi_mean
## [1,] "Everglades" "23" "OUSS" "0.157" "0.35452973106361" "0.473224918032787"
##
## [1,] "0.591920105001964"
##      k            m    model                              phi_mean
## [1,] "Everglades" "25" "OUSS" "0.126" "0.294159989393744" "0.419992459016393"
##
## [1,] "0.545824928639043"
##      k            m    model                              phi_mean
## [1,] "Everglades" "26" "OUSS" "0.137" "0.304562134970829" "0.425315409836066"
##
## [1,] "0.546068684701302"
##      k            m    model                              phi_mean
## [1,] "Everglades" "27" "OUSS" "0.107" "0.266247149621334" "0.397240327868852"
##
## [1,] "0.528233506116371"
##      k            m    model                              phi_mean
## [1,] "Everglades" "31" "OUSS" "0.079" "0.226994721204978" "0.367028852459016"
##
## [1,] "0.507062983713055"
##      k            m    model                              phi_mean
## [1,] "Everglades" "39" "OUSS" "0.176" "0.368264935209992" "0.49003606557377"
##
## [1,] "0.611807195937549"
##      k            m    model                              phi_mean
## [1,] "Everglades" "40" "OUSS" "0.183" "0.371426986966802" "0.493893442622951"
##
## [1,] "0.6163598982791"
##      k            m    model                              phi_mean
## [1,] "Everglades" "41" "OUSS" "0.181" "0.375341770202069" "0.498869180327869"
##
## [1,] "0.622396590453669"
##      k            m    model                              phi_mean
## [1,] "Everglades" "42" "OUSS" "0.196" "0.371736969680368" "0.488064918032787"
##
## [1,] "0.604392866385205"
##      k            m    model                              phi_mean
## [1,] "Everglades" "46" "OUSS" "0.205" "0.373115105393377" "0.485810819672131"
##
## [1,] "0.598506533950885"
##      k            m    model                             phi_mean
## [1,] "Everglades" "47" "OUSS" "0.21" "0.348504179426606" "0.459067540983607"
##
## [1,] "0.569630902540608"
##      k            m    model                              phi_mean
## [1,] "Everglades" "48" "OUSS" "0.213" "0.373113324183274" "0.484130819672131"
##
```

```
## [1,] "0.595148315160988"
##      k            m    model                              phi_mean
## [1,] "Everglades" "49" "OUSS" "0.22" "0.390731279558298" "0.502624918032787"
##
## [1,] "0.614518556507276"
##      k            m    model                              phi_mean
## [1,] "Everglades" "50" "OUSS" "0.23" "0.417831353455256" "0.536927540983607"
##
## [1,] "0.656023728511957"
##      k            m    model                               phi_mean
## [1,] "Everglades" "51" "OUSS" "0.237" "0.396375306756379" "0.502982295081967"
##
## [1,] "0.609589283407556"
##      k            m    model                               phi_mean
## [1,] "Everglades" "60" "OUSS" "0.237" "0.367197025703765" "0.471666229508197"
##
## [1,] "0.576135433312629"
##      k            m    model                               phi_mean
## [1,] "Everglades" "63" "OUSS" "0.204" "0.340164724285613" "0.450664918032787"
##
## [1,] "0.561165111779961"
##      k            m    model                               phi_mean
## [1,] "Okeechobee" "13" "OUSS" "0.549" "0.723493598598719" "0.801693770491803"
##
## [1,] "0.879893942384888"
##      k            m    model                               phi_mean
## [1,] "Okeechobee" "14" "OUSS" "0.337" "0.654214232715302" "0.765807213114754"
##
## [1,] "0.877400193514206"
##      k            m    model                               phi_mean
## [1,] "Okeechobee" "16" "OUSS" "0.165" "0.486367132853891" "0.64513868852459"
##
## [1,] "0.803910244195289"
##      k            m    model                               phi_mean
## [1,] "Okeechobee" "19" "OUSS" "0.124" "0.437109840947291" "0.611865245901639"
##
## [1,] "0.786620650855988"
##      k            m    model                              phi_mean
## [1,] "Okeechobee" "25" "OUSS" "0.107" "0.40513606861614" "0.580229180327869"
##
## [1,] "0.755322292039598"
##      k            m    model                              phi_mean
## [1,] "Okeechobee" "26" "OUSS" "0.09" "0.381886827972809" "0.569708196721311"
##
## [1,] "0.757529565469814"
##      k            m    model                               phi_mean
## [1,] "Okeechobee" "27" "OUSS" "0.077" "0.364000804224696" "0.56163606557377"
##
## [1,] "0.759271326922845"
##      k            m    model                              phi_mean
## [1,] "Okeechobee" "28" "OUSS" "0.11" "0.447255014164314" "0.626013442622951"
##
## [1,] "0.804771871081587"
##      k            m    model                               phi_mean
```

```
## [1,] "Okeechobee" "29" "OUSS" "0.085" "0.377966206089625" "0.565299672131148"
##
## [1,] "0.75263313817267"
##      k            m    model                              phi_mean
## [1,] "Okeechobee" "30" "OUSS" "0.082" "0.371588588325 42" "0.561466229508197"
##
## [1,] "0.751343870690973"
##      k            m    model                               phi_mean
## [1,] "Okeechobee" "31" "OUSS" "0.075" "0.359089668604894" "0.554732786885246"
##
## [1,] "0.750375905165598"
##      k            m    model                               phi_mean
## [1,] "Okeechobee" "32" "OUSS" "0.068" "0.348882473027475" "0.548512786885246"
##
## [1,] "0.748143100743017"
##      k            m    model                               phi_mean
## [1,] "Okeechobee" "36" "OUSS" "0.093" "0.419561215399437" "0.603290819672131"
##
## [1,] "0.787020423944825"
##      k            m    model                              phi_mean
## [1,] "Okeechobee" "37" "OUSS" "0.09" "0.419727059565397" "0.605851803278688"
##
## [1,] "0.79197654699198"
##      k            m    model                               phi_mean
## [1,] "Okeechobee" "38" "OUSS" "0.078" "0.361570328037942" "0.548960983606557"
##
## [1,] "0.736351639175173"
##      k            m    model                               phi_mean
## [1,] "Okeechobee" "39" "OUSS" "0.073" "0.354601597519152" "0.545861967213115"
##
## [1,] "0.737122336907078"
##      k            m    model                               phi_mean
## [1,] "Okeechobee" "40" "OUSS" "0.096" "0.420875671336348" "0.599034754098361"
##
## [1,] "0.777193836860373"
##      k            m    model                               phi_mean
## [1,] "Okeechobee" "41" "OUSS" "0.094" "0.420479816116024" "0.600199344262295"
##
## [1,] "0.779918872408566"
##      k            m    model                               phi_mean
## [1,] "Okeechobee" "46" "OUSS" "0.103" "0.434138055977668" "0.606022950819672"
##
## [1,] "0.777907845661676"
##      k            m    model                              phi_mean
## [1,] "Okeechobee" "47" "OUSS" "0.13" "0.500026786182777" "0.660737049180328"
##
## [1,] "0.821447312177878"
##      k            m    model                               phi_mean
## [1,] "Okeechobee" "48" "OUSS" "0.081" "0.450911832093844" "0.605063606557377"
##
## [1,] "0.75921538102091"
##      k            m    model                               phi_mean
## [1,] "Okeechobee" "49" "OUSS" "0.078" "0.477165954251504" "0.634105573770492"
##
```

```
## [1,] "0.79104519328948"
##      k            m    model                              phi_mean
## [1,] "Okeechobee" "50" "OUSS" "0.078" "0.491856388592468" "0.65118262295082"
##
## [1,] "0.810508857309171"
##      k            m    model                              phi_mean
## [1,] "Okeechobee" "51" "OUSS" "0.075" "0.456284723176559" "0.606924590163934"
##
## [1,] "0.75756445715131"
##      k            m    model                              phi_mean
## [1,] "Okeechobee" "53" "OUSS" "0.078" "0.434706044273888" "0.583180327868852"
##
## [1,] "0.731654611463817"
##      k            m    model                              phi_mean
## [1,] "Okeechobee" "58" "OUSS" "0.071" "0.410127610129191" "0.561147868852459"
##
## [1,] "0.712168127575727"
##      k            m    model                              phi_mean
## [1,] "Okeechobee" "62" "OUSS" "0.095" "0.484328308170467" "0.618865573770492"
##
## [1,] "0.753402839370516"
##      k            m    model                              phi_mean
## [1,] "Okeechobee" "63" "OUSS" "0.119" "0.514005328864203" "0.653107213114754"
##
## [1,] "0.792209097365305"
##      k            m    model                              phi_mean
## [1,] "Okeechobee" "65" "OUSS" "0.133" "0.490384759204791" "0.602644918032787"
##
## [1,] "0.714905076860783"
##      k            m    model                              phi_mean
## [1,] "Okeechobee" "71" "OUSS" "0.138" "0.473875852860964" "0.583243278688525"
##
## [1,] "0.692610704516086"
##      k            m    model                              phi_mean
## [1,] "Okeechobee" "74" "OUSS" "0.156" "0.521326620483507" "0.624293442622951"
##
## [1,] "0.727260264762395"
##      k              m    model                               phi_mean
## [1,] "Tohopekaliga" "13" "OUSS" "0.397" "0.476003834841173" "0.53942393442623"
##
## [1,] "0.602844034011286"
##      k              m    model                               phi_mean
## [1,] "Tohopekaliga" "14" "OUSS" "0.353" "0.420566088582158" "0.48401737704918"
##
## [1,] "0.547468665516202"
##      k              m    model                               phi_mean
## [1,] "Tohopekaliga" "15" "OUSS" "0.318" "0.388174406409546" "0.451198360655738"
##
## [1,] "0.51422231490193"
##      k              m    model                               phi_mean
## [1,] "Tohopekaliga" "16" "EGSS" "0.021" "0.738984682426412" "0.784907213114754"
##
## [1,] "0.830829743803096"
##      k              m    model                               phi_mean
```

```
## [1,] "Tohopekaliga" "17" "OUSS" "0.323" "0.391837777645061" "0.455134754098361"
##
## [1,] "0.51843173055166"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "18" "OUSS" "0.236" "0.434739955298226" "0.498218032786885"
##
## [1,] "0.561696110275544"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "19" "OUSS" "0.326" "0.394151593645484" "0.45726131147541"
##
## [1,] "0.520371029305336"
##      k              m    model                             phi_mean
## [1,] "Tohopekaliga" "20" "OUSS" "0.304" "0.37222931716757" "0.43538262295082"
##
## [1,] "0.498535928734069"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "21" "OUSS" "0.285" "0.353834768454185" "0.416781639344262"
##
## [1,] "0.47972851023434"
##      k              m    model                            phi_mean
## [1,] "Tohopekaliga" "23" "OUSS" "0.35" "0.43486826542113" "0.49867737704918"
##
## [1,] "0.562486488677231"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "24" "OUSS" "0.342" "0.462466370020044" "0.531260655737705"
##
## [1,] "0.600054941455366"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "25" "OUSS" "0.317" "0.480888911748267" "0.555413114754098"
##
## [1,] "0.629937317759929"
##      k              m    model                             phi_mean
## [1,] "Tohopekaliga" "26" "EGSS" "0.01" "0.515182212887825" "0.575519672131148"
##
## [1,] "0.63585713137447"
##      k              m    model                             phi_mean
## [1,] "Tohopekaliga" "27" "EGSS" "0.016" "0.66034878414791" "0.713287213114754"
##
## [1,] "0.766225642081598"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "29" "OUSS" "0.126" "0.461249631579206" "0.524719016393443"
##
## [1,] "0.588188401207679"
##      k              m    model                             phi_mean
## [1,] "Tohopekaliga" "30" "OUSS" "0.08" "0.446919854612136" "0.510481639344262"
##
## [1,] "0.574043424076388"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "33" "EGSS" "0.012" "0.587064160637638" "0.644707213114754"
##
## [1,] "0.70235026559187"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "34" "EGSS" "0.014" "0.634434895298302" "0.688847868852459"
##
```

```
## [1,] "0.743260842406616"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "35" "OUSS" "0.355" "0.492649950343949" "0.563994426229508"
##
## [1,] "0.635338902115067"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "36" "OUSS" "0.337" "0.504493881757777" "0.578663278688525"
##
## [1,] "0.652832675619272"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "37" "OUSS" "0.239" "0.514058376037489" "0.597671803278689"
##
## [1,] "0.681285230519888"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "38" "OUSS" "0.158" "0.517055942150968" "0.620910163934426"
##
## [1,] "0.724764385717884"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "39" "OUSS" "0.156" "0.519437097151573" "0.623793442622951"
##
## [1,] "0.728149788094329"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "40" "OUSS" "0.098" "0.514705914745085" "0.644874098360656"
##
## [1,] "0.775042281976227"
##      k              m    model                             phi_mean
## [1,] "Tohopekaliga" "41" "OUSS" "0.05" "0.479638247574997" "0.662589508196721"
##
## [1,] "0.845540768818446"
##      k              m    model                            phi_mean
## [1,] "Tohopekaliga" "42" "OUSS" "0.159" "0.5010724327498" "0.60248262295082"
##
## [1,] "0.70389281315184"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "44" "OUSS" "0.119" "0.533441910662241" "0.643666885245902"
##
## [1,] "0.753891859829562"
##      k              m    model                             phi_mean
## [1,] "Tohopekaliga" "46" "OUSS" "0.086" "0.53479304233164" "0.659300983606557"
##
## [1,] "0.783808924881475"
##      k              m    model                            phi_mean
## [1,] "Tohopekaliga" "47" "EGSS" "0.03" "0.73767796339661" "0.82698131147541"
##
## [1,] "0.91628465955421"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "48" "EGSS" "0.027" "0.713734987582971" "0.812132459016393"
##
## [1,] "0.910529930449816"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "49" "EGSS" "0.026" "0.702897432436224" "0.805072131147541"
##
## [1,] "0.907246829858858"
##      k              m    model                              phi_mean
```

```
## [1,] "Tohopekaliga" "50" "EGSS" "0.022" "0.677649465370262" "0.773598360655738"
##
## [1,] "0.869547255941213"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "51" "EGSS" "0.026" "0.716718967772359" "0.810186885245902"
##
## [1,] "0.903654802719444"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "52" "EGSS" "0.021" "0.688652768473658" "0.774075737704918"
##
## [1,] "0.859498706936178"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "53" "EGSS" "0.027" "0.737489262736559" "0.821160655737705"
##
## [1,] "0.904832048738851"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "54" "EGSS" "0.029" "0.742003995526338" "0.830889180327869"
##
## [1,] "0.9197743651294"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "55" "EGSS" "0.028" "0.726728775160844" "0.820991475409836"
##
## [1,] "0.915254175658828"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "56" "EGSS" "0.029" "0.728331631650276" "0.826545901639344"
##
## [1,] "0.924760171628412"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "57" "EGSS" "0.029" "0.726127151845576" "0.827085573770492"
##
## [1,] "0.928043995695407"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "58" "EGSS" "0.016" "0.636274031378794" "0.720828196721311"
##
## [1,] "0.805382362063829"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "59" "EGSS" "0.016" "0.643045588523863" "0.715850819672131"
##
## [1,] "0.788656050820399"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "60" "EGSS" "0.016" "0.609080130789592" "0.710266557377049"
##
## [1,] "0.811452983964506"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "61" "EGSS" "0.026" "0.775884299741094" "0.818383278688525"
##
## [1,] "0.860882257635955"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "62" "EGSS" "0.031" "0.805760290599212" "0.844994426229508"
##
## [1,] "0.884228561859805"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "63" "EGSS" "0.026" "0.777037096735061" "0.819184918032787"
##
```

```
## [1,] "0.861332739330513"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "64" "EGSS" "0.024" "0.762390039688675" "0.806052786885246"
##
## [1,] "0.849715534081817"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "65" "EGSS" "0.007" "0.452775521939774" "0.521415737704918"
##
## [1,] "0.590055953470062"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "66" "OUSS" "0.132" "0.561714756820621" "0.69211606557377"
##
## [1,] "0.82251737432692"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "67" "OUSS" "0.134" "0.553286625785775" "0.684687213114754"
##
## [1,] "0.816087800443733"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "68" "EGSS" "0.008" "0.222893175163488" "0.53379737704918"
##
## [1,] "0.844701578934873"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "69" "OUSS" "0.178" "0.625290948564593" "0.738668196721311"
##
## [1,] "0.85204544487803"
##      k              m    model                             phi_mean
## [1,] "Tohopekaliga" "70" "OUSS" "0.17" "0.620510976606347" "0.73618393442623"
##
## [1,] "0.851856892246112"
##      k              m    model                             phi_mean
## [1,] "Tohopekaliga" "71" "OUSS" "0.17" "0.620107620416813" "0.735428524590164"
##
## [1,] "0.850749428763515"
##      k              m    model                             phi_mean
## [1,] "Tohopekaliga" "72" "OUSS" "0.17" "0.621535634368547" "0.736485573770492"
##
## [1,] "0.851435513172436"
##      k              m    model                             phi_mean
## [1,] "Tohopekaliga" "73" "OUSS" "0.18" "0.641066893531927" "0.752216393442623"
##
## [1,] "0.863365893353319"
##      k              m    model                             phi_mean
## [1,] "Tohopekaliga" "74" "EGSS" "0.01" "0.276608332817889" "0.566246885245902"
##
## [1,] "0.855885437673915"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "75" "OUSS" "0.165" "0.629223538767845" "0.743170163934426"
##
## [1,] "0.857116789101007"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "76" "OUSS" "0.166" "0.630210608957113" "0.743516393442623"
##
## [1,] "0.856822177928133"
##      k              m    model                             phi_mean
```

```
## [1,] "Tohopekaliga" "77" "EGSS" "0.01" "0.284496644461207" "0.570940655737705"
##
## [1,] "0.857384667014203"
##      k              m    model                              phi_mean
## [1,] "Tohopekaliga" "78" "OUSS" "0.163" "0.636238433287531" "0.749388524590164"
##
## [1,] "0.862538615892797"
##      k              m    model                             phi_mean
## [1,] "Tohopekaliga" "79" "EGSS" "0.01" "0.288854373044288" "0.572157704918033"
##
## [1,] "0.855461036791778"
##      k              m    model                             phi_mean
## [1,] "Tohopekaliga" "80" "OUSS" "0.159" "0.64048912846131" "0.752907540983607"
##
## [1,] "0.865325953505903"
##      k              m    model                             phi_mean
## [1,] "Tohopekaliga" "81" "OUSS" "0.163" "0.60601670023054" "0.719972131147541"
##
## [1,] "0.833927562064542"
```

Recover and save the results as data frame (with an iterative process for each location).

```
# Initialize an empty data frame to store results
phiSD_df_pops <- data.frame(
  Population = numeric(),
  Time.t = numeric(),
  phi.hat = numeric(),
  SD.phi = numeric(),
  Model = character(),
  Time_Taken = character()
)

# Loop through each percentage in the results
for (k in seq_along(cells)) {
  for (j in seq_along(sk.others.results[[k]]$phi_others)) {
    # Extract data for this specific percentage and end position
    phi_hat <- sk.others.results[[k]]$phi_others[[j]][1]
    SD_phi <- sk.others.results[[k]]$phi_others[[j]][2]
    model <- sk.others.results[[k]]$modelSS[[j]]
    time_t <- sk.others.results[[k]]$Time.t[j]
    time_taken <- sk.others.results[[k]]$time_taken

    # Combine the extracted data into a data frame
    temp_df <- data.frame(
      Population = cells[k],
      Time.t = time_t,
      phi.hat = phi_hat,
      SD.phi = SD_phi,
      Model = model,
      Time_Taken = time_taken
    )

    # Bind the temp_df to the final data frame
    phiSD_df_pops <- rbind(phiSD_df_pops, temp_df)
  }
```

```
}

head(phiSD_df_pops)
```

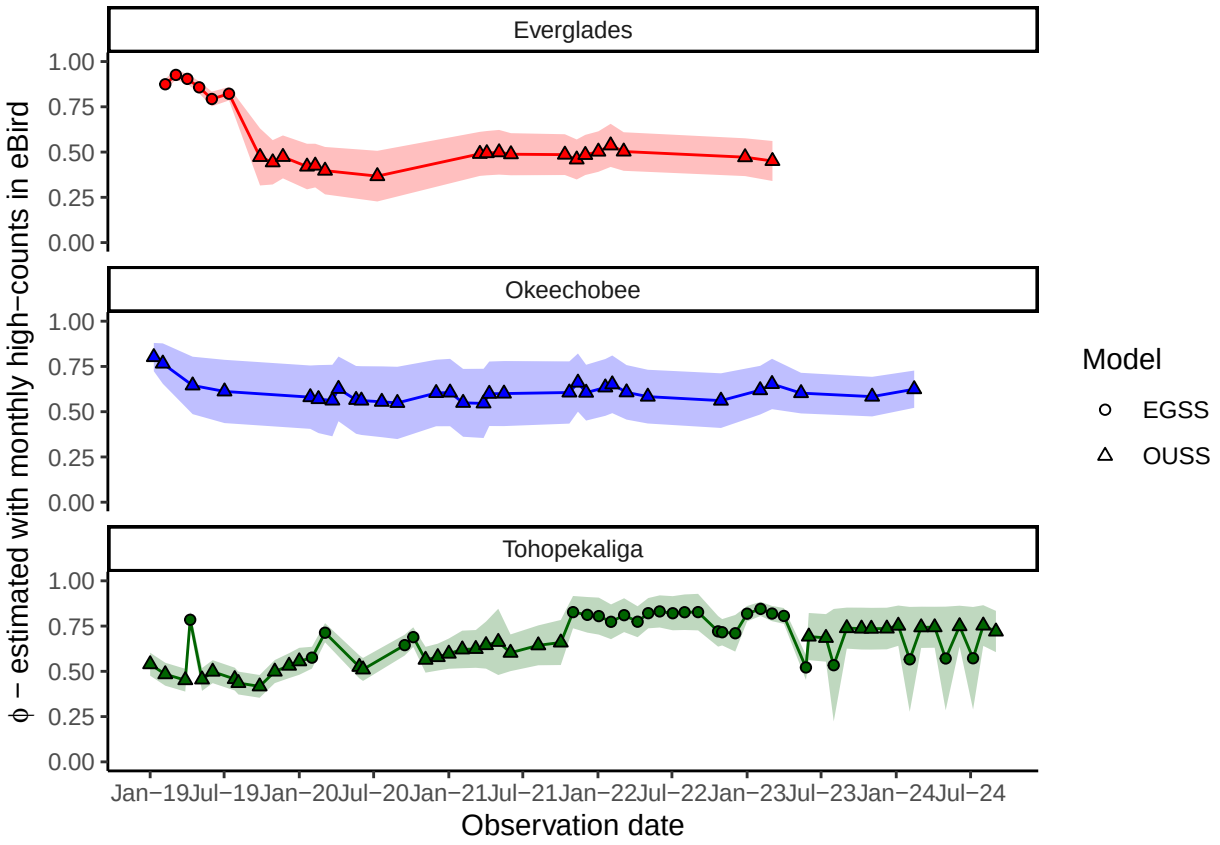
```
##   Population Time.t   phi.hat     SD.phi Model    Time_Taken
## 1 Everglades     14 0.8744495 0.03119257  EGSS 25.38323 secs
## 2 Everglades     15 0.9261633 0.02741187  EGSS 25.38323 secs
## 3 Everglades     16 0.9042236 0.02734830  EGSS 25.38323 secs
## 4 Everglades     17 0.8575502 0.03324869  EGSS 25.38323 secs
## 5 Everglades     18 0.7932148 0.04064513  EGSS 25.38323 secs
## 6 Everglades     19 0.8222036 0.03727307  EGSS 25.38323 secs
```

```
tail(phiSD_df_pops)
```

```
##       Population Time.t   phi.hat    SD.phi Model    Time_Taken
## 114 Tohopekaliga     76 0.7435164 0.1133058  OUSS 71.73449 secs
## 115 Tohopekaliga     77 0.5709407 0.2864440  EGSS 71.73449 secs
## 116 Tohopekaliga     78 0.7493885 0.1131501  OUSS 71.73449 secs
## 117 Tohopekaliga     79 0.5721577 0.2833033  EGSS 71.73449 secs
## 118 Tohopekaliga     80 0.7529075 0.1124184  OUSS 71.73449 secs
## 119 Tohopekaliga     81 0.7199721 0.1139554  OUSS 71.73449 secs
```

```
# Save the data frame to a file
saveRDS(phiSD_df_pops, "results/phi_SD_OtherPopulations.rds")
```

```
phiSD_df_pops |>
  left_join(sk.others.ts, by = c("Population" = "cell",
                                 "Time.t")) |>
  ggplot(aes(x = observation_date, y = phi.hat))+
    geom_line(aes(color = Population))+
    geom_ribbon(aes(ymin = phi.hat-SD.phi,
                    ymax = phi.hat+SD.phi,
                    fill = Population),
                alpha = 0.25) +
    geom_point(aes(fill = Population, shape = Model),
               color = "black")+
  coord_cartesian(ylim = c(0, 1))+
  scale_x_date(breaks = seq(as.Date("2018-01-01"), as.Date(Sys.Date( )),
                            by = "6 months"), date_labels="%b-%y")+
  scale_color_manual(values = c("red", "blue", "darkgreen"))+
  scale_fill_manual(values = c("red", "blue", "darkgreen"))+
  scale_shape_manual(values = c(21,24))+
  labs(x = "Observation date",
       y = expression(phi*" - estimated with monthly high-counts in eBird"))+
  facet_wrap(~Population, ncol = 1)+
  theme_classic() +
  theme(legend.position = "right") +
  guides(color = "none",
         fill = "none")
```



The mean coordinates of the points for these three populations are:

```
sk.others |>
  group_by(cell) |>
  summarise(count = n(),
            mu_lat = mean(latitude),
            mu_lon = mean(longitude))
```

```
## # A tibble: 3 x 4
##   cell         count mu_lat mu_lon
##   <chr>        <int>  <dbl>  <dbl>
## 1 Everglades     166   25.8  -80.7
## 2 Okeechobee     147   27.0  -81.1
## 3 Tohopekaliga   815   28.3  -81.4
```