Oikos **OIK-00916**

Ward, E. J., Holmes, E. E., Thorson, J. T. and Collen, B. 2013. Complexity is costly: a meta-analysis of parametric and non-parametric methods for short-term population forecasting. – Oikos doi: 10.1111/j.1600-0706.2013.00916.x

Appendix 1

Summary of models included

1. Random walk with drift

$Y_t = Y_{t-1} + u + e_t;\ e_t \sim Normal(0, \sigma)$

The drift term is u. This is a process error only model, with errors that are temporally independent.

2. Random walk with autocorrelated errors

$Y_t = Y_{t-1} + u + e_t;\ e_t \sim Normal(\rho \cdot e_{t-1}, \sqrt{1-\rho^2}\sigma)$

This is a process error only model, with errors that are temporally correlated ($-1 < \rho < 1$).

3. State space random walk model

Process equation: $X_t = X_{t-1} + u + e_t;\ e_t \sim Normal(0, \sigma)$

Observation (or 'data model') equation: $Y_t = X_t + \delta_t;\ \delta_t \sim Normal(0, \gamma)$

While the process model is a random walk, the total variance is broken up into a process component (representing natural stochasticity) and observation error component (resulting from imperfect observations and sampling error) (Lindley 2003).

4. Generalized additive models (GAMs)

Our implementation of GAMs only used time as a covariate, so the model was not autoregressive. The basic form is

$g(E[Y]) = B_0 + f(time)$

where the function *g*() is a link function (we used log), B_0 is an intercept, and the function *f*() is a smoothing function, or set of polynomial regression splines. The degree of smoothness was selected by cross validation (Wood 2006).

5. Neural network model

The neural network time series model is autoregressive, but non-linear,

$$Y_{t+d} = B_0 + \sum_{j=1}^{d} B_j g\left(\gamma_{0,j} + \sum_{i=1}^{m} \gamma_{1,j} \cdot Y_{t-(i-1)d}\right)$$

where the structure of the network is controlled by the embedding dimension (*m*) and time delay (*d*). The activation function *g*() was assumed linear, and all other parameters represent weights or coefficients. Because of relatively short time series, we constrained *m* = 1:3, and *d* = 1:2.

6. ARIMA models

AR models treat x_t as autoregressive. The *p* term is the degree of lag included in the model:

AR: $Y_t = b_1 Y_{t-1} + b_2 Y_{t-2} + \ldots + b_q Y_{t-p} + e_t;\ e_t \sim Normal(0, \sigma)$

MA models have treat the errors, e_t, as autoregressive. The *q* term is the degree of lag included in the autoregressive model for the errors. A MA model with no AR component would be:

MA: $Y_t = e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \ldots + \theta_q e_{t-q};\ e_t \sim Normal(0, \sigma)$

An ARMA model is a time series model with both the AR and MA components. ARMA models may also include a constant. For example, AR(1) with constant would be

AR(1)+constant: $Y_t = b_1 x_{t-1} + \mu + e_t;\ e_t \sim Normal(0, \sigma)$

If b_1 is set to 1, this is a random walk with drift.

An ARIMA model includes both the AR and MA components but also specifies whether the raw data, Y_t, or lag-*d* differences are being modeled. An ARIMA model is denoted ARIMA(*p*, *d*, *q*). Thus a ARIMA(0,2,1) model would mean:

ARIMA(0,2,1): $Y_t - Y_{t-2} = e_t + \theta_1 e_{t-1};\ e_t \sim Normal(0, \sigma)$

It should be noted that most ARIMA models---the random walk with drift model being a major exception---are stationary, meaning they do not have a long-term temporal trend. When the time series has a trend, ARIMA models are used to model the residuals of a regression of that time series. We used the `Arima()` function in the forecast package in R which takes care of estimating the linear trend and fitting the residuals with the specified stationary ARIMA model. This can also be done using the base `arima()` function in R by passing in `xreg=1:n` as a covariate.

7. Exponentially smoothed time series

The most basic exponentially smoothed (or weighted) moving average time series models are ARIMA(*p* = 0, *d* = 1, *q* = 1),

$z_t = \sum_{j=1}^{\infty}(1-\lambda)\lambda^{j-1} z_{t-j} + e_t;\ e_t \sim Normal(0, \sigma);\ |\lambda| < 1$ (Shumway & Stoffer 2006)

Where z_t is the detrended data, Y_t-(*a*-*bt*), and *a*+*bt* is the linear trend (estimated simultaneously with the ARIMA model for the residuals).

8. Local regression

Local regression represents a linear model that is fit piecewise, in a moving window procedure, through a time series, and the prediction at a given time point is a function of data in the past and future,

$$\hat{Y}_t = f(Y) + e_t;\ \ e_t \sim Normal(0, \sigma)$$

The function *f*() typically takes two arguments: a nearest neighbor or bandwidth argument, specifying how much of the dataset to use (0-100%), and a parameter or function controlling the exponential decay between points. For each dataset in our analysis, we used cross validation to select the nearest neighbors and polynomial (1:3). The parametric version of this model was implemented using locfit(), and a non-parametric version of the model was implemented with a kernel regression estimator using the npreg() function.

9. Gaussian process regression

The objective of Gaussian process regression is to make prediction while conditioning on a covariance matrix, **Σ**, and previously observed residuals.

$$\hat{Y}_t = f(Y) + e_t;\ \ e_t \sim Multivariate\ normal(0, \mathbf{\Sigma})$$

All data points are assumed to have arisen from an unknown covariance function, and unlike other methods (e.g. local or non-parametric bandwidth regression), the correlation between points is not modeled as a function of the distance between them in time, but in terms of their relative values (e.g. biomass or abundance at time *t* and *t*+1).

10. Random forest regression

Random forest uses an ensemble prediction from n_{trees} different regression trees (we have used $n_{trees} = 500$). Each tree uses a bootstrap of the data, and a randomly chosen subset of the predictor variables. This is done to minimize the correlation among predictions from different trees, which will tend to decrease predictive error for ensemble forecasting methods. For predictor variables we have used a basis-expansion using the lag-operator, and lags 1-10.

$$\hat{Y}_t = \frac{1}{n_{trees}} \sum_{i=1}^{n_{trees}} \hat{Y}_{t,i}$$

where $\hat{Y}_{t,i}$ is the prediction from the *i*-th tree. Each tree starts with the following prediction:

$$\hat{Y}_t = \frac{1}{n} \sum_{j=1}^{n} Y_j$$

The tree then searches among available variables and finds the variable and split that maximizes the reduction in root-mean-squared error. This process is repeated until a particular node has 5 or fewer observations.

11. Simplex

The goal of simplex is to predict the dynamics of a variable without using a parametric equation, and hence potentially avoiding problems associated with parametric models that occur when dynamics are highly state-dependent. Simplex does this by identifying nearest neighbors using a Euclidean distance metric defined in a *d*-dimension space generated using the lag-operator.

$$\hat{Y}_t = \frac{1}{d+1} \sum_{i=d+1}^{t-f} I(D_i) \cdot Y_i$$

where d is the embedding dimension, f is the prediction interval, D_i is a Euclidean distance in d-dimensional lag-space:

$$D_i = \sqrt{\sum_{j=1}^{d} \left(Y_{i-j} - Y_{t-j}\right)^2}$$

and $I(Y_{i-d},...,Y_{i-1})$ is an indicator variable that identifies $d + 1$ nearest neighbors in the Euclidean distance D_i, i.e., equals one if distance D_i is one of the $d + 1$ lowest distances. The embedding dimension d is then selected using cross-validation.

12. S-MAP

S-MAP has a similar goal to Simplex, and typically uses the embedding dimension previously selected using Simplex. However, it has an additional parameter θ representing the degree of state-dependent dynamics in a time series. Instead of nearest neighbors, it calculates a weight γ_i for each point i using the distance defined for Simplex:

$$\gamma_i = \theta \cdot \frac{D_i}{\sum_{j=1}^{n} D_i}$$

This weight is then used to take a weighted average of the dynamics of all points.

$$\hat{Y}_t = \langle 1, Y_{t-f}, \dots, Y_{t-f-d} \rangle \times \boldsymbol{C}$$

where × is the matrix multiplicative operator and $\boldsymbol{C}$ is the solution to a weighted linear model:

$$\boldsymbol{C} = \boldsymbol{A}^{-1} \times \boldsymbol{B}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are formed from the lagged variables, and the inverse of A is accomplished using the singular-value decomposition:

$$\boldsymbol{B} = \boldsymbol{\gamma} \cdot \boldsymbol{x}_{-t}$$

where · is the pairwise multiplication operator and $\boldsymbol{x}_{-t}$ is the vector of the time series excluding observation x_t, and

$$\boldsymbol{A} = \langle \boldsymbol{\gamma} \cdot 1, \boldsymbol{\gamma} \cdot l_f(\boldsymbol{Y}_{-t}), \dots, \boldsymbol{\gamma} \cdot l_{f-d+1}(\boldsymbol{Y}_{-t}) \rangle$$

and $l_f(\boldsymbol{Y}_{-t})$ is the lag operator of order f for the vector $\boldsymbol{Y}_{-t}$.

Table A1. Model summary and the code / functions used to fit them in existing packages in the R programming environment.

Model	R package (R function in package)	Parametric
Random walk	forecast (rwf)	Y
State-space random walk	stats (StructTS), MARSS (MARSS)	Y
GAMs	mgcv (gam)	Y
Neural network time series	tsDyn (nnetTs)	N
Exponentially smoothed time series	forecast (ets)	Y
Local regression	locfit (locfit)	Y
Kernel / bandwidth regression	np (npreg)	N
ARIMA	forecast (Arima), stats (arima)	Y
Gaussian process	kernlab (gausspr)	N
Random Forest	randomForest (randomForest)	N
SMAP, Simplex	Code by Jim Thorson <https://r-forge.r-project.org/R/?group_id=1316>	N

Table A2. Table of 1-step ahead MASE statistics for 49 models in our analysis. R packages and functions used are listed in Table A1. Stationary ARIMA models (those not denoted RW), are fit to detrended data, but the forecast from those models includes the trend.

Model	Marine fish productivity	Salmon	Birds	Mammals
GAM (gam)	1.768	1.040	0.969	1.087
neural network (1,1)	1.850	1.152	1.420	2.191
neural network (1,2)	1.736	1.222	1.197	1.560
neural network (2,1)	1.729	1.171	1.418	2.258
neural network (2,2)	2.109	1.273	1.217	1.451
neural network (3,1)	1.788	1.199	1.434	1.815
neural network (3,2)	2.093	1.413	1.297	1.720
RW no drift - ARIMA(0,1,0) without constant	1.431	0.982	0.976	1.062
RW with drift - ARIMA(0,1,0) with constant	1.449	0.994	0.994	1.159
Exp smooth with trend, ARIMA(0,1,1)	1.471	0.957	0.932	1.277
Exp smooth without trend, ARIMA(0,1,1)	1.473	0.966	0.940	1.277
Structural time series (freq=1)	1.429	0.905	0.904	1.136
Structural time series (freq=2)	1.474	0.962	0.940	1.151
Local regression	2.490	2.333	1.940	2.356
Kernel/bandwidth regression	1.545	1.018	0.961	1.146
ARIMA(1,0,1)	1.414	0.965	0.986	1.175
Gompertz; ARIMA(1,0,0)	1.381	0.976	1.037	1.091
ARIMA(2,0,1)	1.430	0.997	1.000	1.212
ARIMA(1,0,2)	1.478	1.027	1.009	1.136
ARIMA(2,0,2)	1.481	1.021	1.005	1.212
MA model; ARIMA(0,0,1)	1.731	1.118	2.112	1.711
ARIMA(0,0,2)	1.695	1.068	1.715	1.477
ARIMA(2,0,0)	1.386	0.993	1.005	1.175
ARIMA(1,1,1)	1.414	0.913	0.915	1.164
ARIMA(1,1,0)	1.399	0.942	0.933	1.103
ARIMA(2,1,1)	1.407	0.936	0.920	1.214
ARIMA(1,1,2)	1.426	0.935	0.923	1.215
ARIMA(2,1,2)	1.445	0.981	0.951	1.217
ARIMA(0,1,1)	1.422	0.893	0.911	1.174
ARIMA(0,1,2)	1.455	0.934	0.934	1.205
ARIMA(2,1,0)	1.402	0.940	0.923	1.208
ARIMA(1,2,1)	1.421	0.958	0.907	1.189
ARIMA(1,2,0)	1.731	1.279	1.208	1.290
ARIMA(2,2,1)	1.422	0.965	0.910	1.173
ARIMA(1,2,2)	1.445	0.950	0.901	1.295
ARIMA(2,2,2)	1.452	0.963	0.936	1.183
ARIMA(0,2,1)	1.435	0.994	0.967	1.191
ARIMA(0,2,2)	1.476	0.901	0.897	1.240
ARIMA(2,2,0)	1.626	1.183	1.107	1.269
Gaussian process (freq=1)	1.691	1.042	1.730	1.597
Gaussian process (freq=2)	1.716	1.014	1.706	1.570
Gaussian process (freq=3)	1.749	1.014	1.731	1.396
Gaussian process (freq=4)	1.743	1.029	1.706	1.586
State-space RW with drift	1.482	0.928	0.966	1.295
State-space RW no drift	1.464	0.909	0.915	1.155

Simplex	1.578	0.990	1.337	1.321
S-MAP	1.658	1.291	1.483	2.156
Random Forest regression	1.562	0.988	1.124	1.197
linear regression	1.886	1.094	1.549	1.925