Random Walk of the Penguins Competition Solution – Using Imputation and Ensemble Methods

Ambarish Ganguly

https://www.linkedin.com/in/ambarish-ganguly/
@a_ganguly

Random Walk of the Penguins

HOSTED BY DRIVENDATA

This project is a collaborative effort between Oceanites, Inc., Black Bawks Data Science Ltd., and Dr. Heather Lynch's lab at Stony Brook University. Prize generously provided by NASA (Award NNX14AC32G).

MAPPPD

Stony Brook University

Competitors built hundreds of algorithms to predict changes in Antarctic penguin populations from the most comprehensive counts available. These algorithms give researchers a greater understanding of penguin population dynamics, a leading indicator of climate change.

Quick Facts

PARTICIPANTS	605
NO. OF ENTRIES	675

Why worry about Penguin Population

Penguins are among the most charismatic animals in the world and have captured the imaginations of news-makers, scientists, film producers, and the general public. **Beyond their general intrinsic value, they are considered important ecosystem indicators.**

In other words, **monitoring these beautiful species can tell us a lot about the general health of the Antarctic because penguins are important krill and fish predators, and changes (natural or anthropogenic) that influence prey abundance and environmental conditions will ultimately be detected through changes in distribution or population size**.

Data on penguin populations are limited because most monitored colonies are near permanent research stations and other sites are surveyed only sporadically.

Because the data are so patchy, and time series relatively short, it has been difficult to build statistical models that can explain past dynamics or provide reliable future predictions.

Your goal is to create better models to estimate populations for hard-to-reach sites in the Antarctic, and thereby greatly improve our ability to use penguins to monitor the health of the Southern Ocean!

The counts are made up of raw chick and/or nest counts for sites all around the Antarctic continent for three species: Gentoo, Chinstrap and Adélie penguins.

All three species nest on the Western Antarctic Peninsula (south of South America), and in some cases will nest in the same colony. On the Peninsula, Chinstrap and Gentoo penguins are restricted to the more northerly regions, while the Adélie penguin is the only species of the three that nests around the entire continent.

Different penguin species have different ecologies...That is to say, how and when they breed differ between species, as well as where and when they eat. They also have slightly different diets, with each species eating various amounts of krill or fish species depending on their location

The three different species also rely on sea ice in different ways. Finally, the timing of breeding can change from year to year (which would affect the total counts), and is different between the three species.

This competition is a first (of which we are aware) for ecological time series. Due to the nature of the data, we expect this to be challenging, and metrics of success for these exercises in academia are often quite low.

We provide data in this format for all of the seasons up through 2013. However, the time series for each site are variable in length and patchy due to the expensive nature of doing field work in Antarctica.

Here is a sample from the nest count data:

		2009	2010	2011	2012	2013
site_id	**common_name**					
BENN	**chinstrap penguin**	NaN	NaN	NaN	NaN	NaN
BENT	**adelie penguin**	NaN	NaN	NaN	NaN	NaN
BERK	**adelie penguin**	NaN	13423.0	NaN	NaN	NaN
BERN	**gentoo penguin**	NaN	NaN	NaN	NaN	NaN
BERT	**adelie penguin**	NaN	346.0	NaN	313.0	NaN
BIEN	**adelie penguin**	NaN	35466.0	NaN	NaN	NaN
BISC	**adelie penguin**	594.0	539.0	567.0	522.0	NaN
	gentoo penguin	2401.0	2404.0	3081.0	3197.0	NaN
BLAK	**chinstrap penguin**	NaN	NaN	NaN	NaN	NaN

Performance metric

Performance is evaluated according to an adjusted MAPE calculation. Since some penguin counts have differing accuracies, the percent error is weighted differently based on this expected accuracy.

$$AMAPE = \frac{1}{N}\sum_{n=1}^{N}\frac{\left|\frac{\hat{y}_n - y_n}{\max(1, y_n)}\right|}{e_n}$$

where $\hat{y}_n$ is the predicted penguin count, y_n is the actual penguin count, e_n is the error re-weighting based on our confidence in the penguin count. We also will not divide by zero, so if $y_n = 0$ we use one as our percentage denominator instead.

Note: We provide e_n in the same structured format as the nest count timeseries for ease of calculating this metric.

Your `.csv` file for those predictions would look like:

```
site_id,common_name,2014,2015,2016,2017
ACUN,adelie penguin,0.0,0.0,0.0,0.0
ACUN,chinstrap penguin,0.0,0.0,0.0,0.0
ADAM,adelie penguin,0.0,0.0,0.0,0.0
ADAR,adelie penguin,0.0,0.0,0.0,0.0
AILS,chinstrap penguin,0.0,0.0,0.0,0.0
...
YANK,gentoo penguin,0.0,0.0,0.0,0.0
YOUN,adelie penguin,0.0,0.0,0.0,0.0
YTRE,adelie penguin,0.0,0.0,0.0,0.0
ZAWA,chinstrap penguin,0.0,0.0,0.0,0.0
ZIGZ,chinstrap penguin,0.0,0.0,0.0,0.0
```

In [4]:
```
# How many observations do we have for each species?
observations.common_name.value_counts()
```

Out[4]:
```
adelie penguin       1387
gentoo penguin        791
chinstrap penguin     774
Name: common_name, dtype: int64
```

In [5]:
```
# How many differnet sites do we see each species at?
(observations.groupby("common_name")
             .site_id
             .nunique())
```

Out[5]:
```
common_name
adelie penguin       281
chinstrap penguin    340
gentoo penguin       105
Name: site_id, dtype: int64
```

In [6]:
```
# How many count types do we have for each species?
(observations.groupby("common_name")
             .count_type
             .value_counts())
```

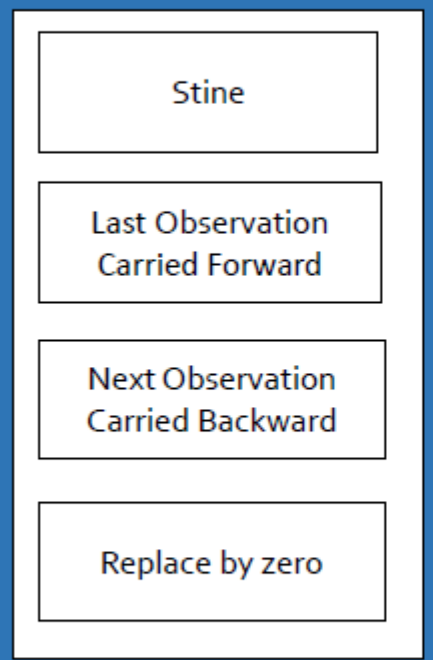
Out[6]:
```
common_name         count_type
adelie penguin      nests         976
                    adults        223
                    chicks        188
chinstrap penguin   nests         608
                    adults         86
                    chicks         80
gentoo penguin      nests         629
                    chicks        161
                    adults          1
Name: count_type, dtype: int64
```

Solution Components

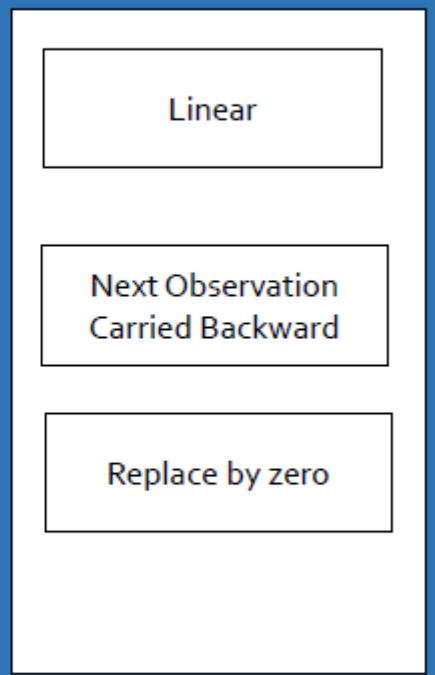
For each site and penguin type, two main components are used

- Imputation
- Model Building

Imputation



R Models
(Arima, ETS , Prophet)



Python Models
(XGBoost and RandomForest)

ELSEVIER

Computers & Geosciences

Volume 17, Issue 6, 1991, Pages 813-820

An interpolation method for estuarine and oceanographic data

Abstract

A relatively new interpolation method (developed by R. W. Stineman) was tested for use in estuarine studies, although it can be employed in a large variety of similar processes. A FORTRAN-77 subroutine STINTV was developed, and several tests were performed by analyzing basic oceanographic variables such as salinity, temperature, current velocities, and tides obtained in the Bahía Blanca Estuary. Stineman's method is between 2 and 6 times faster and more accurate than cubic spline interpolation when compared statistically. Stineman's is as fast as simpler linear interpolation. When the function to be interpolated behaves in a smooth fashion, both methods give reliable results: however, Stineman's is more reliable if the variable presents abrupt changes in slope.

na_locf *Missing Value Imputation by Last Observation Carried Forward*

Description

Replaces each missing value with the most recent present value prior to it (Last Observation Carried Forward- LOCF). Optionally this can also be done starting from the back of the series (Next Observation Carried Backward - NOCB).

Usage

```
na_locf(x, option = "locf", na_remaining = "rev", maxgap = Inf)
```

```r
# Interpolate using the following methods in the order mentioned below:
# Stine
# Last Observation Carried Forward
# Next Observation Carried Backward
# Replace by Zero
# Perform auto.arima using the forecast package
predictTimeSeries = function(tsTrain,i)
{
  tsTrain[i,] = tryCatch({
    na.interpolation(tsTrain[i,], option = "stine")
  },
  error = function(e) {
    tsTrain[i,]
  })

  tsTrain[i,] = tryCatch({
    na.locf(tsTrain[i,])
  },
  error = function(e) {
    tsTrain[i,]
  })

  tsTrain[i,] = tryCatch({
    na.locf(tsTrain[i,],option="nocb")
  },
  error = function(e) {
    tsTrain[i,]
  })

  tsTrain[i,] = na.replace(tsTrain[i,], fill = 0.0)

  fit = auto.arima(tsTrain[i,],stepwise=FALSE, approximation=FALSE)

  return(fit)

}
```

PROPHET

Docs GitHub

Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well.

Prophet is open source software released by Facebook's Core Data Science team. It is available for download on CRAN and PyPI.

Accurate and fast.

Prophet is used in many applications across Facebook for producing reliable forecasts for planning and goal setting. We've found it to perform better than any other approach in the majority of cases. We fit models in Stan so that you get forecasts in just a few seconds.

Fully automatic.

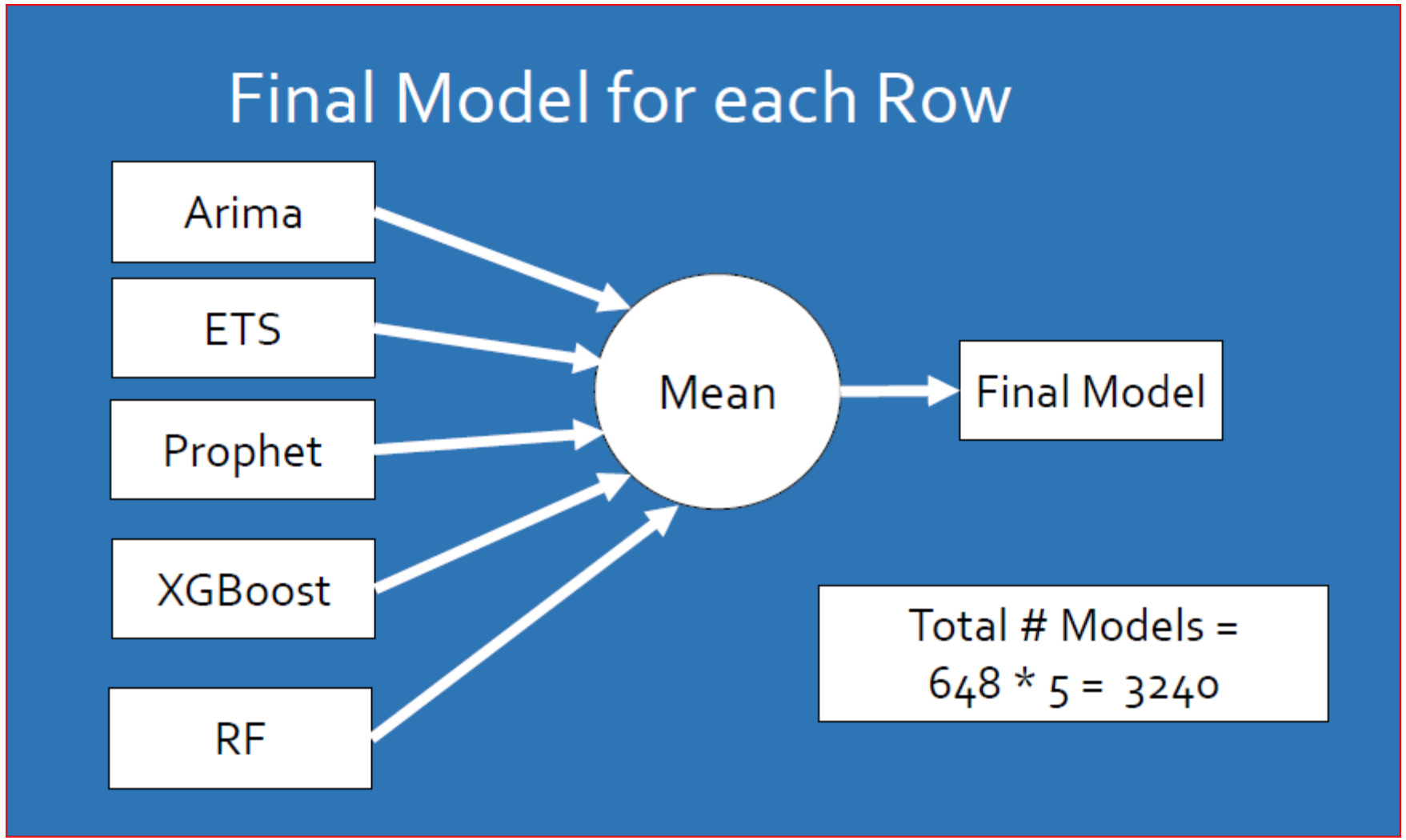
Get a reasonable forecast on messy data with no manual effort. Prophet is robust to outliers, missing data, and dramatic changes in your time series.

Tunable forecasts.

The Prophet procedure includes many possibilities for users to tweak and adjust forecasts. You can use human-interpretable parameters to improve your forecast by adding your domain knowledge.

Available in R or Python.

We've implemented the Prophet procedure in R and Python, but they share the same underlying Stan code for fitting. Use whatever language you're comfortable with to get forecasts.


Final Model for each Row
Arima
ETS
Prophet
XGBoost
RF
Mean
Final Model
Total # Models =
648 * 5 = 3240

Publication
Predicting the future is hard and other lessons from a population time series data science competition
publication date Jul 11, 2018
publication description Elsevier Journal - Ecological Informatics

Blog
https://www.drivendata.co/blog/random-walk-of-the-penguins/

Github
https://github.com/drivendataorg/random-walk-of-the-penguins/tree/823e184b1decaf8f5f1a3810a42619ecd8f848b8

Books
https://otexts.com/fpp2/