

seer: R package for feature-based forecast model selection

Thiyanga S Talagala
Rob J Hyndman
George Athanasopoulos

Monash University, Australia

UseR, 2018

Large collections of time series



- Forecasting demand for thousands of products across multiple warehouses.

Time series features

Objective

Develop a framework that automates the selection of the most appropriate forecasting method for a given time series by using an array of **features** computed from the time series.

Objective

Develop a framework that automates the selection of the most appropriate forecasting method for a given time series by using an array of **features** computed from the time series.

- **Basic idea:**

Transform a given time series $y = \{y_1, y_2, \dots, y_n\}$ to a feature vector $F = (f_1(y), f_2(y), \dots, f_p(y))'$.

Objective

Develop a framework that automates the selection of the most appropriate forecasting method for a given time series by using an array of **features** computed from the time series.

- **Basic idea:**

Transform a given time series $y = \{y_1, y_2, \dots, y_n\}$ to a feature vector $F = (f_1(y), f_2(y), \dots, f_p(y))'$.

- Examples for time series features

Objective

Develop a framework that automates the selection of the most appropriate forecasting method for a given time series by using an array of **features** computed from the time series.

- **Basic idea:**

Transform a given time series $y = \{y_1, y_2, \dots, y_n\}$ to a feature vector $F = (f_1(y), f_2(y), \dots, f_p(y))'$.

- Examples for time series features
 - strength of trend

Objective

Develop a framework that automates the selection of the most appropriate forecasting method for a given time series by using an array of **features** computed from the time series.

- **Basic idea:**

Transform a given time series $y = \{y_1, y_2, \dots, y_n\}$ to a feature vector $F = (f_1(y), f_2(y), \dots, f_p(y))'$.

- Examples for time series features

- strength of trend
- strength of seasonality

Objective

Develop a framework that automates the selection of the most appropriate forecasting method for a given time series by using an array of **features** computed from the time series.

- **Basic idea:**

Transform a given time series $y = \{y_1, y_2, \dots, y_n\}$ to a feature vector $F = (f_1(y), f_2(y), \dots, f_p(y))'$.

- Examples for time series features
 - strength of trend
 - strength of seasonality
 - lag-1 autocorrelation

Objective

Develop a framework that automates the selection of the most appropriate forecasting method for a given time series by using an array of **features** computed from the time series.

- **Basic idea:**

Transform a given time series $y = \{y_1, y_2, \dots, y_n\}$ to a feature vector $F = (f_1(y), f_2(y), \dots, f_p(y))'$.

- Examples for time series features

- strength of trend
- strength of seasonality
- lag-1 autocorrelation
- spectral entropy

Time series features

- length
- strength of seasonality
- strength of trend
- linearity
- curvature
- spikiness
- stability
- lumpiness
- first ACF value of remainder series
- parameter estimates of Holt's linear trend method
- spectral entropy
- Hurst exponent
- nonlinearity
- parameter estimates of Holt-Winters' additive method
- unit root test statistics
- first ACF value of residual series of linear trend model
- ACF and PACF based features - calculated on both the raw and differenced series

FFORMS: Feature-based FORecast Model Selection

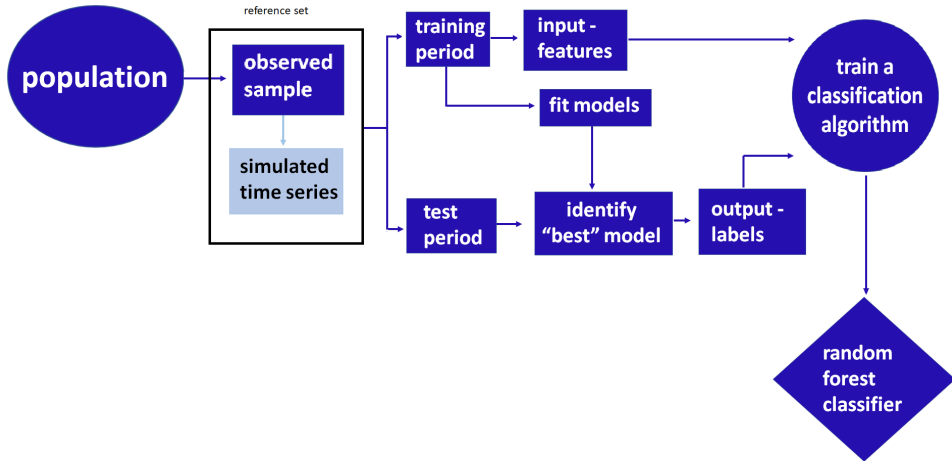
Offline

- A classification algorithm (the meta-learner) is trained.

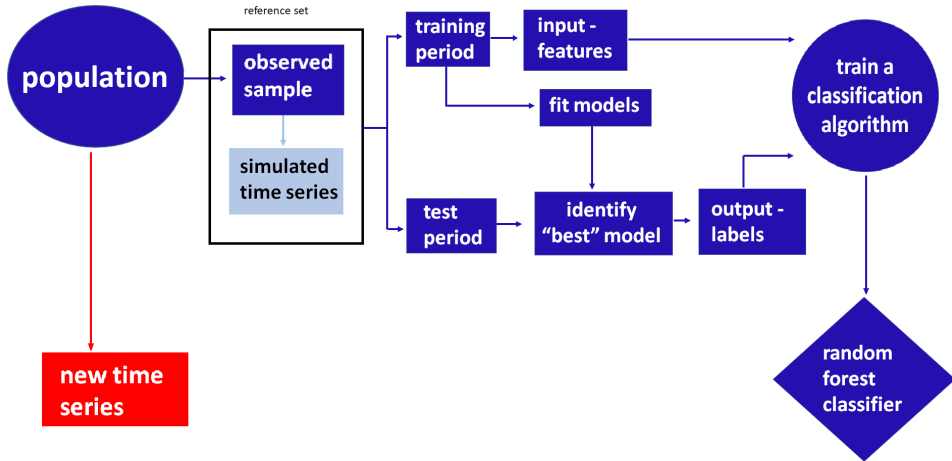
Online

- Calculate the features of a time series and use the pre-trained classifier to identify the best forecasting method.

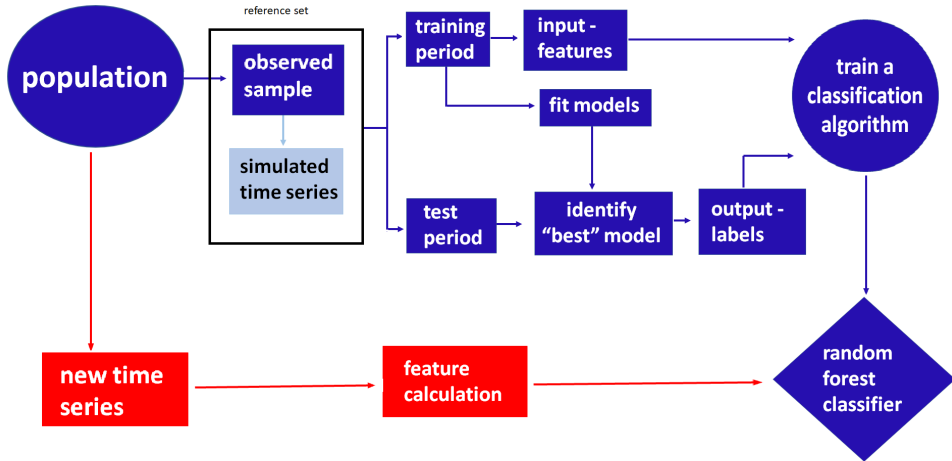
FFORMS: “offline” part of the algorithm



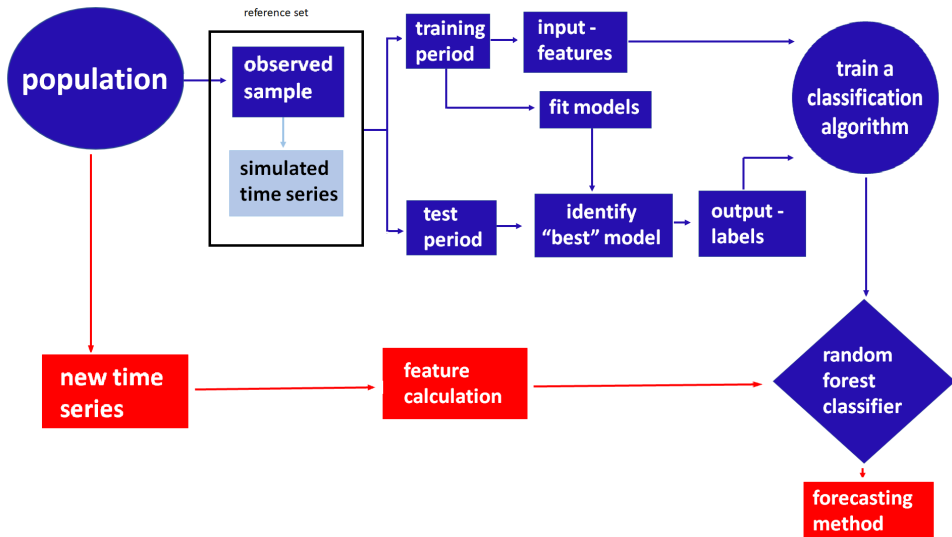
FFORMS: “online” part of the algorithm



FFORMS: “online” part of the algorithm



FFORMS: “online” part of the algorithm



Installation

```
devtools::install_github("thiyanagt/seer")  
library(seer)
```





Installation

```
devtools::install_github("thiyanagt/seer")  
library(seer)
```

Example datasets

observed time series - M1 yearly series (181)

```
library(Mcomp)  
yearlym1 <- subset(M1, "yearly")
```



Installation

```
devtools::install_github("thiyanagt/seer")  
library(seer)
```

Example datasets

observed time series - M1 yearly series (181)

```
library(Mcomp)  
yearlym1 <- subset(M1, "yearly")
```

new time series - M3 yearly series (645)

```
yearlym3 <- subset(M3, "yearly")
```

Input: features

```
cal_features(yearlym1[1:3], database="M3",  
h=6, highfreq=FALSE)
```

```
# A tibble: 3 x 25  
  entropy lumpiness stability hurst trend  spikiness linearity curvature  
    <dbl>    <dbl>    <dbl> <dbl> <dbl>    <dbl>    <dbl>    <dbl>  
1  0.683    0.0400    0.977 0.985 0.985 0.00000132    4.46    0.705  
2  0.711    0.0790    0.894 0.988 0.989 0.00000154    4.47    0.613  
3  0.716    0.0160    0.858 0.987 0.989 0.00000113    4.60    0.695  
# ... with 17 more variables: e_acf1 <dbl>, y_acf1 <dbl>,  
#   diff1y_acf1 <dbl>, diff2y_acf1 <dbl>, y_pacf5 <dbl>,  
#   diff1y_pacf5 <dbl>, diff2y_pacf5 <dbl>, nonlinearity <dbl>,  
#   lmres_acf1 <dbl>, ur_pp <dbl>, ur_kpss <dbl>, N <int>, y_acf5 <dbl>,  
#   diff1y_acf5 <dbl>, diff2y_acf5 <dbl>, alpha <dbl>, beta <dbl>
```

Output: labels

```
fcast_accuracy(yearlym1[1:3],  
  models=c("arima","ets","rw","rwd","theta","nn"),  
  database="M3", cal_MASE, h=6, length_out=1)
```

\$accuracy

	arima	ets	rw	rwd	theta	nn
YAF2	10.527612	10.319029	13.52428	10.527612	12.088375	11.756990
YAF3	5.713867	7.704409	7.78949	5.225965	6.225463	6.700781
YAF4	8.633590	8.091416	11.55633	8.440105	9.952742	10.784962

\$ARIMA

	YAF2	YAF3
"ARIMA(0,1,0) with drift"	"ARIMA(0,1,1) with drift"	
	YAF4	
"ARIMA(0,1,2) with drift"		

\$ETS

	YAF2	YAF3	YAF4
"ETS(A,A,N)"	"ETS(M,A,N)"	"ETS(M,A,N)"	

Reference set

```
accuracy_m1 <- fcast_accuracy(tslist=yearlym1,  
models= c("arima","ets","rw","rwd", "theta", "nn"),  
database ="M1", cal_MASE)
```

```
features_m1 <- cal_features(yearlym1, database="M1", highfreq = FALSE)
```

```
reference_set <- prepare_trainingset(accuracy_set = accuracy_m1,  
feature_set = features_m1)  
head(reference_set$trainingset, 1)
```

```
# A tibble: 1 x 26
```

```
  entropy lumpiness stability hurst trend  spikiness linearity curvature  
    <dbl>    <dbl>    <dbl> <dbl> <dbl>    <dbl>    <dbl>    <dbl>  
1  0.683    0.0400    0.977 0.985 0.985 0.00000132    4.46    0.705  
# ... with 18 more variables: e_acf1 <dbl>, y_acf1 <dbl>,  
#   diff1y_acf1 <dbl>, diff2y_acf1 <dbl>, y_pacf5 <dbl>,  
#   diff1y_pacf5 <dbl>, diff2y_pacf5 <dbl>, nonlinearity <dbl>,  
#   lmres_acf1 <dbl>, ur_pp <dbl>, ur_kpss <dbl>, N <int>, y_acf5 <dbl>,  
#   diff1y_acf5 <dbl>, diff2y_acf5 <dbl>, alpha <dbl>, beta <dbl>,  
#   classlabels <chr>
```

FFORMS classifier

```
ym3_features <- cal_features(yearlym3,  
                             database="M3", highfreq = FALSE)  
  
fforms <- build_rf(training_set = ref_set$trainingset,  
                  testset=ym3_features, rf_type="rcp",  
                  ntree=100, seed=7, import=FALSE)  
  
fforms$predictions %>% head(10)
```

```
##           1           2           3           4           5           6           7  
## ETS-trend      rwd      rwd      rwd      rwd      rwd      rwd  
##           8           9          10          11          12          13          14  
##          rwd      rwd      rwd      rwd ETS-trend      rwd      rwd  
##          15          16          17          18          19          20  
##          nn      rwd      rwd      rwd      rwd      ARIMA  
## 10 Levels: ARIMA ARMA/AR/MA ETS-dampedtrend ... wn
```

Generate point forecasts and 95% prediction intervals

```
rf_forecast(fforms$predictions[1:2],  
tslist=yearlym3[1:2], database="M3",  
function_name="cal_MASE", h=6, accuracy=TRUE)
```

```
## $mean  
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]  
## [1,] 5486.429 6035.865 6585.301 7134.737 7684.173 8233.609  
## [2,] 4402.227 4574.454 4746.681 4918.908 5091.135 5263.362  
##  
## $lower  
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]  
## [1,] 4984.162 4893.098 4629.135 4199.745 3606.858 2848.8735  
## [2,] 2890.401 2366.671 1959.916 1608.186 1288.666  990.2221  
##  
## $upper  
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]  
## [1,] 5988.696 7178.632 8541.467 10069.729 11761.488 13618.344  
## [2,] 5914.053 6782.236 7533.445  8229.629  8893.603  9536.501  
##  
## $accuracy  
## [1] 1.5636089 0.6123443
```

Augmenting the observed sample with simulated time series

```
lapply(yearlym1[1], sim_arimabased, Nsim=2)
```

```
## $YAF2
## $YAF2[[1]]
## Time Series:
## Start = 1972
## End = 1993
## Frequency = 1
## [1] 3600.00 15158.97 38653.24 61436.41 111298.11 85898.46 177282.98
## [8] 171711.97 163492.36 177853.14 216154.50 263424.00 260655.68 265410.40
## [15] 307030.65 371226.22 413602.53 462569.21 460368.35 465142.08 490847.33
## [22] 505928.59
##
## $YAF2[[2]]
## Time Series:
## Start = 1972
## End = 1993
## Frequency = 1
## [1] 3600.00 14657.36 32971.60 80537.55 146425.27 142443.09 190979.40
## [8] 238906.76 266580.96 273224.47 278253.14 282840.62 332403.58 375570.08
## [15] 361558.06 363810.01 377489.10 462165.08 510598.86 476927.75 520308.58
## [22] 600756.76
```


Augmenting the observed sample with simulated time series

```
lapply(yearlym1[1], sim_arimabased, Nsim=2)
```

```
## $YAF2
## $YAF2[[1]]
## Time Series:
## Start = 1972
## End = 1993
## Frequency = 1
## [1] 3600.00 15158.97 38653.24 61436.41 111298.11 85898.46 177282.98
## [8] 171711.97 163492.36 177853.14 216154.50 263424.00 260655.68 265410.40
## [15] 307030.65 371226.22 413602.53 462569.21 460368.35 465142.08 490847.33
## [22] 505928.59
##
## $YAF2[[2]]
## Time Series:
## Start = 1972
## End = 1993
## Frequency = 1
## [1] 3600.00 14657.36 32971.60 80537.55 146425.27 142443.09 190979.40
## [8] 238906.76 266580.96 273224.47 278253.14 282840.62 332403.58 375570.08
## [15] 361558.06 363810.01 377489.10 462165.08 510598.86 476927.75 520308.58
## [22] 600756.76
```

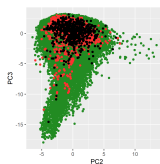
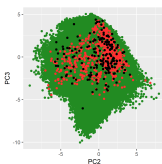
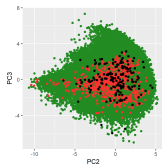
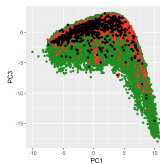
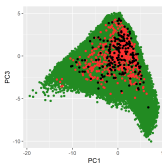
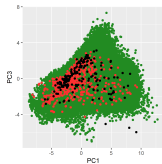
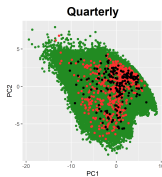
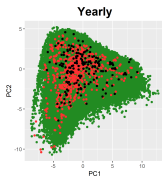
```
lapply(yearlym1[1], sim_etsbased, Nsim=2)
lapply(yearlym1[1], sim_mstlbased, Nsim=2)
```

Application: Distribution of time series in the PCA space

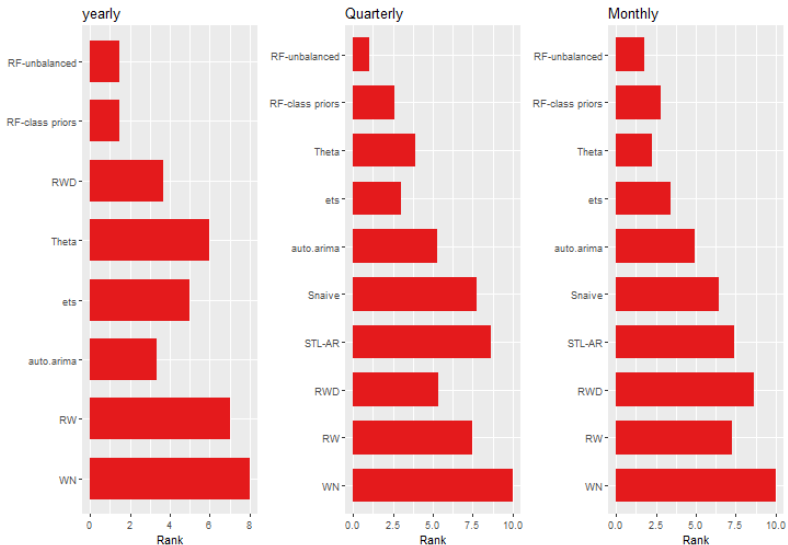
observed - M1

simulated

new - M3



Results



- FFORMS: framework for forecast model selection using meta-learning based on time series features.

- FFORMS: framework for forecast model selection using meta-learning based on time series features.
- FFORMS algorithm uses the knowledge of the past performance of candidate forecast models on a collection of time series in order to identify the best forecasting method for a new series.

- FFORMS: framework for forecast model selection using meta-learning based on time series features.
- FFORMS algorithm uses the knowledge of the past performance of candidate forecast models on a collection of time series in order to identify the best forecasting method for a new series.
- For real-time forecasting, our framework involves only the calculation of features, the selection of a forecast method based on the FFORMS random forest classifier, and the calculation of the forecasts from the chosen model.

- FFORMS: framework for forecast model selection using meta-learning based on time series features.
- FFORMS algorithm uses the knowledge of the past performance of candidate forecast models on a collection of time series in order to identify the best forecasting method for a new series.
- For real-time forecasting, our framework involves only the calculation of features, the selection of a forecast method based on the FFORMS random forest classifier, and the calculation of the forecasts from the chosen model.
- We have also introduced a simple set of time series features that are useful in identifying the "best" forecast method for a given time series.



available at: <https://github.com/thiyanngt/seer>



available at: <https://github.com/thiyanagt/seer>

paper: <https://robjhyndman.com/publications/fforms/>

Email: thiyanga.talagala@monash.edu

twitter: thiyanagt