Forecasting with NNS

Viole, Fred
fred.viole@gmail.com

November 1, 2017

Install NNS

We need the latest version of NNS available on GitHub.

```
> require(devtools); install_github('OVVO-Financial/NNS',ref = "NNS-Beta-Version")
> require(NNS)
```

NNS Forecasting

NNS forecasting is an autoregressive technique that has the following advantages over an ARIMA model:

* no stationarity requirement
* no model identification
* capable of handling nonlinearity

This is accomplished by determining the seasonality present and then regressing the relevant periods.

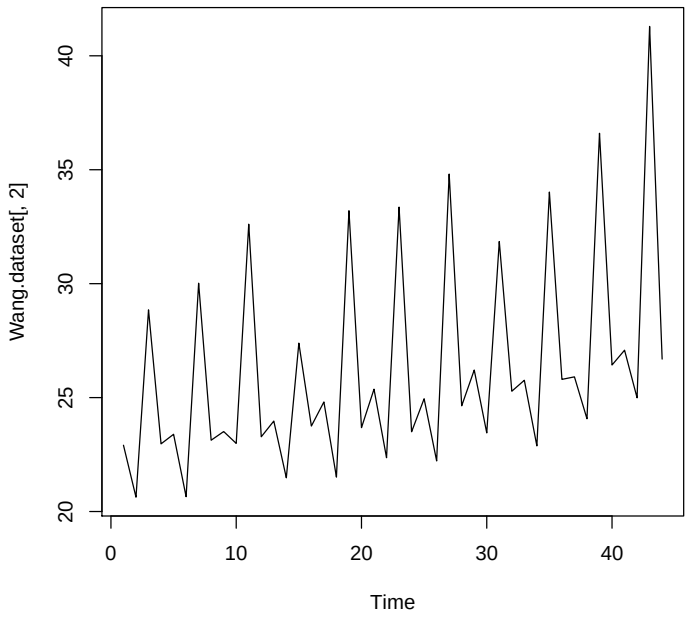
Seasonality

First let's load an example dataset found in Wang [2008], "*A* Guide to Box Jenkins Modeling".

NNS uses a coefficient of variation (CV) ($\frac{\sigma}{\mu}$) for determining seasonality. If a period's CV is less than the overall dataset's CV, that period exhibits seasonality. Large CV's are dismissed, which ameliorates any logical concerns astute readers would have using this statistic.

The variable's overall CV provides an objective cutoff point for significance, and avoids any subjective interpretations to be made from the analogous ACF and PACF plots in the standard ARMA technique. Furthermore, we avoid dependence on a linear technique when trying to determine a relationship in lagged values that may be nonlinear.

```
> Wang.dataset<- read.csv("https://goo.gl/UDk2xB",header=FALSE,sep = '')
> plot.ts(Wang.dataset[,2])
```

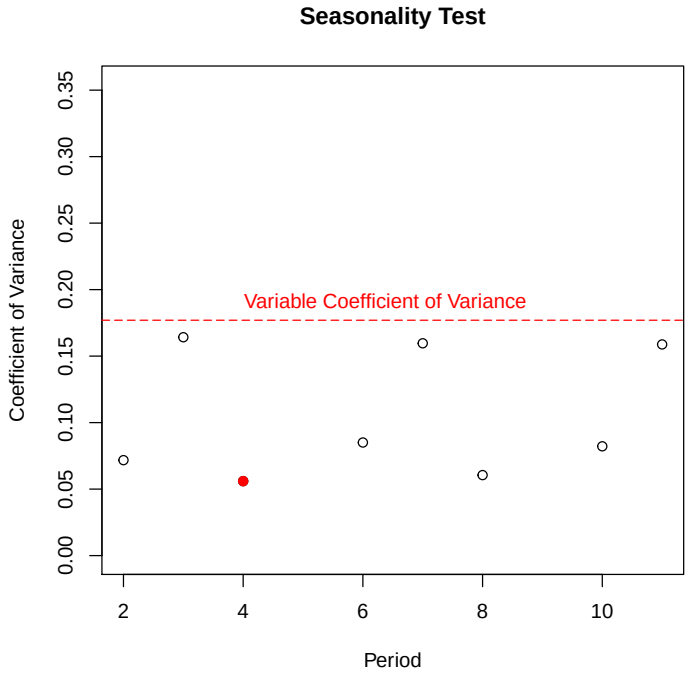


Now we can run a NNS seasonality test on this series.

```
> NNS.seas(Wang.dataset[,2])
```

```
$all.periods
   Period Coefficient.of.Variance Variable.Coefficient.of.Variance
1:      4              0.05599103                          0.1769858
2:      8              0.06053440                          0.1769858
3:      2              0.07176943                          0.1769858
4:     10              0.08217461                          0.1769858
5:      6              0.08503594                          0.1769858
6:     11              0.15878767                          0.1769858
7:      7              0.15964245                          0.1769858
8:      3              0.16419383                          0.1769858

$best.period
[1] 4
```



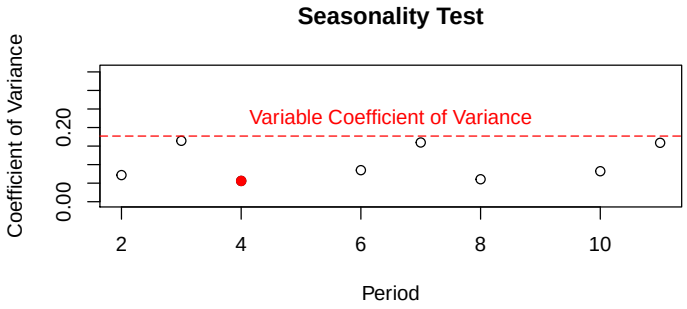
NNS returns all periods that exhibit seasonality, and the period with the highest amount of seasonality. For convenience, NNS also plots all periods with significant seasonality, highlighting the most significant.

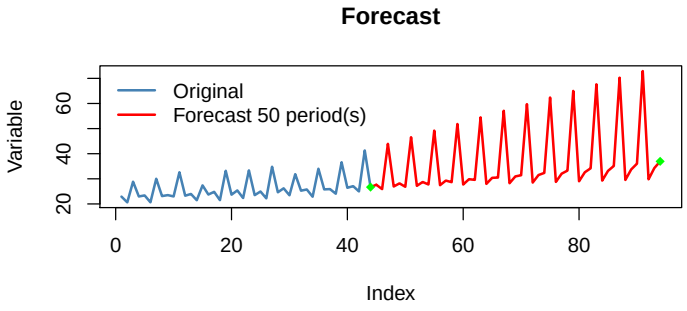
The "best.period" is a 4 period lag, reflective of quarterly periodicity in this monthly time series.

Forecasting

Now that we have our seasonality determined, we can we can forecast our time series using the range extension features of `NNS.reg`.

```
> NNS.ARMA(Wang.dataset[,2],h=50)
```





Options

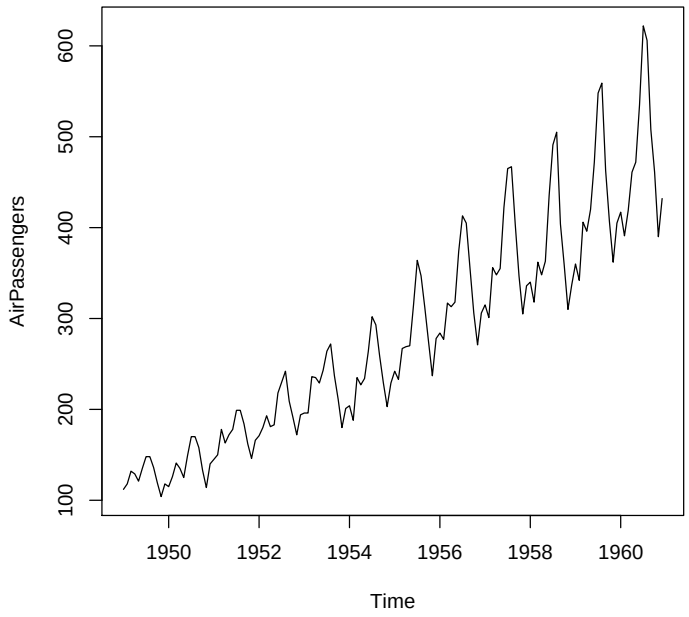
NNS forecasting offers a multitude of options. These options include:

* The ability to use linear or nonlinear regressions on the relevant sub-series
* Specify and test individual lags or all reported seaonality periods
* Use forecast periods to determine new seasonality periods
* Cross-validate fit on sections of original series

AirPassengers Example

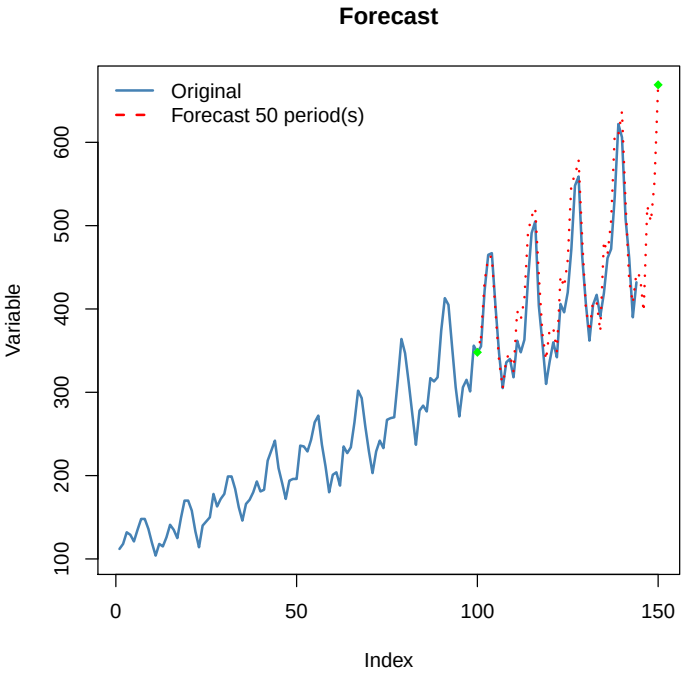
Here is an example on the preloaded **AirPassengers** dataset.

```
> plot.ts(AirPassengers)
> NNS.seas(AirPassengers)
```



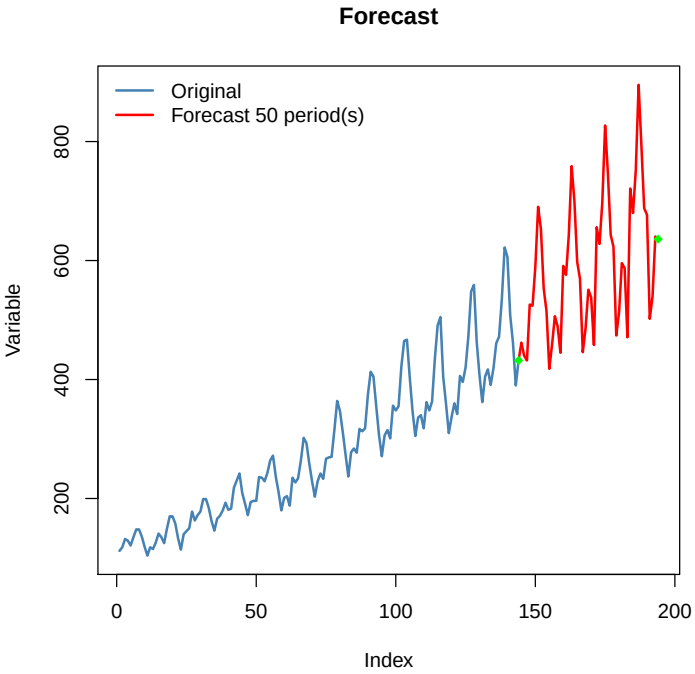
We generate a 20 period forecast using just the first 100 observations; select a seasonal factor of 12 (present in our output above); and a nonlinear regression.

```
> NNS.ARMA(AirPassengers,h=50,training.set = 100,seasonal.factor = 12)
```



And extending the range further...

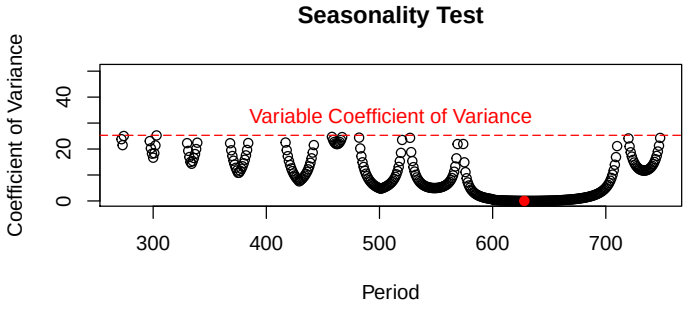
```
> NNS.ARMA(AirPassengers,h=50,seasonal.factor = 12)
```

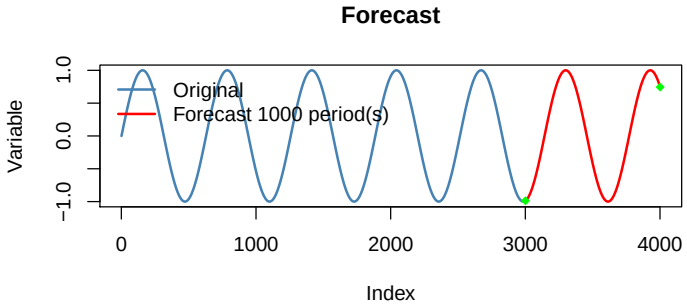

Forecast
Original
Forecast 50 period(s)
Variable
200
400
600
800
0
50
100
150
200
Index

Non-Time Series Data

Here's a sine wave extended well beyond its final observation...

```
> x=seq(0,30,.01);y=sin(x)
> nns.estimate=NNS.ARMA(y,h=1000,negative.values = TRUE)
```





And our accuracy (MSE) to the known values:

```
> mean(c(nns.estimate-sin(seq(30.01,40,.01)))^2)
```

```
[1] 3.491865e-10
```

It's essentially 0!!!

Cross-Validation

It may be helpful to evaluate several seasonal lags against known values and compare a preferred statistic of interest, in this case the MAPE (mean average percentage error). We speed this up by only using the linear method for identification, then 'nonlin' in final analysis.

We are forecasting the last 336 observations of the `taylor` dataset from the forecast package.

Data:
Create `[train]` and `[test]` datasets and then do the same for the reduced `[train]` set in order to avoid any leakage of test information into our training set.

```
> require(forecast);data(taylor)
> test=tail(taylor,336)
> train=head(taylor,(length(taylor)-336))
> in.sample.train=head(train,(length(train)-336))
> in.sample.test=tail(train,336)
```

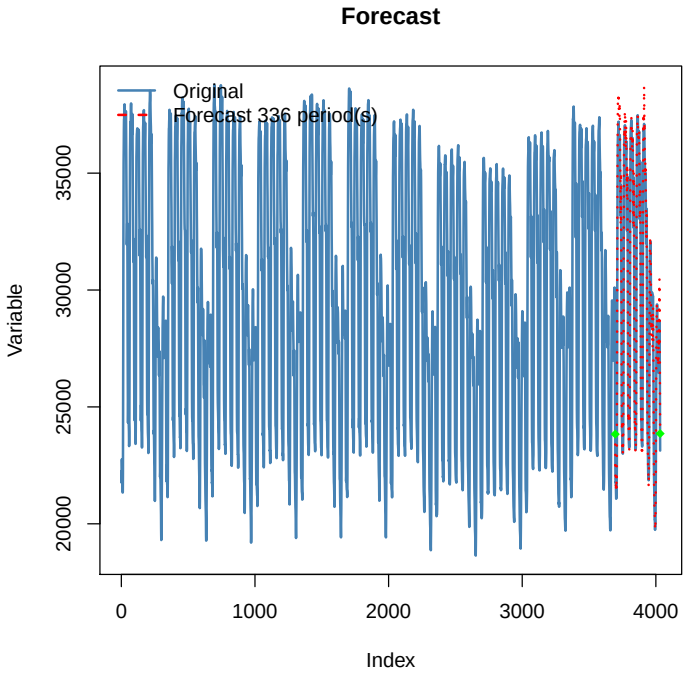
Analysis using the top 10 reported seasonal lags:

```
> a=NNS.seas(in.sample.train,plot=FALSE)
> cv.test=t(sapply(a$all.periods$Period[1:10], function(i)
   c(i,mean(abs((NNS.ARMA(taylor,h=336,training.set=length(taylor)-336,
   seasonal.factor=i,plot=FALSE,method='lin'))-test)/test))))
> colnames(cv.test)=c("Period","MAPE")
> cv.test
```

```
      Period       MAPE
 [1,]    623 0.13829040
 [2,]    336 0.03709453
 [3,]    672 0.04448145
 [4,]    764 0.21249350
 [5,]    767 0.14207245
 [6,]    815 0.11371497
 [7,]    671 0.05032179
 [8,]    335 0.07022057
 [9,]    431 0.11844039
[10,]    621 0.18952096
```

Using our lowest MAPE period from the `cv.test` data frame we have

```
> nns.estimate=NNS.ARMA(taylor,h=336,training.set=length(taylor)-336,
             seasonal.factor = cv.test[which.min(cv.test[,2]),1],plot=TRUE,
             method = 'nonlin')
```



Yielding our final MAPE:

```
> mean(abs(nns.estimate-test)/test)

[1] 0.01466336
```

More NNS:

To learn more about NNS statistics and their theoretical foundations, see "*N*onlinear Nonparametric Statistics: Using Partial Moments" available on Amazon:
http://a.co/5bpHvUg
Check back to see more NNS examples posted on GitHub:

https://github.com/OVVO-Financial/NNS/tree/NNS-Beta-Version/examples