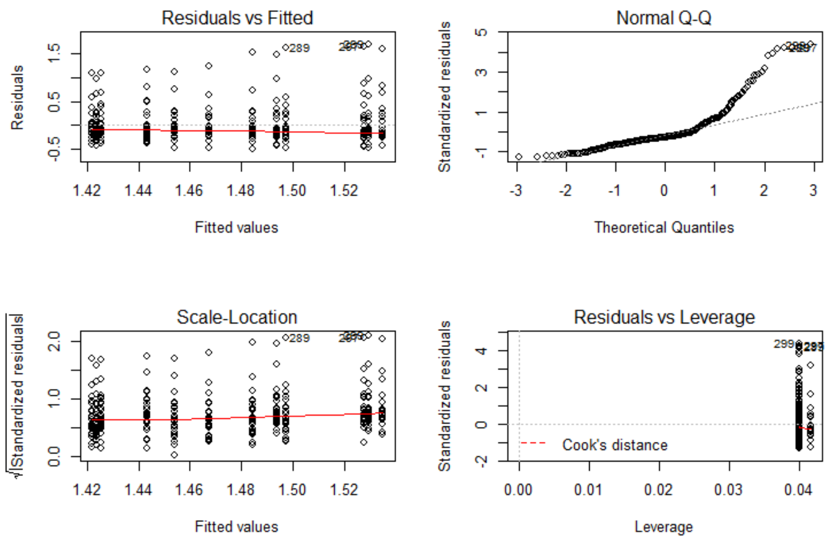
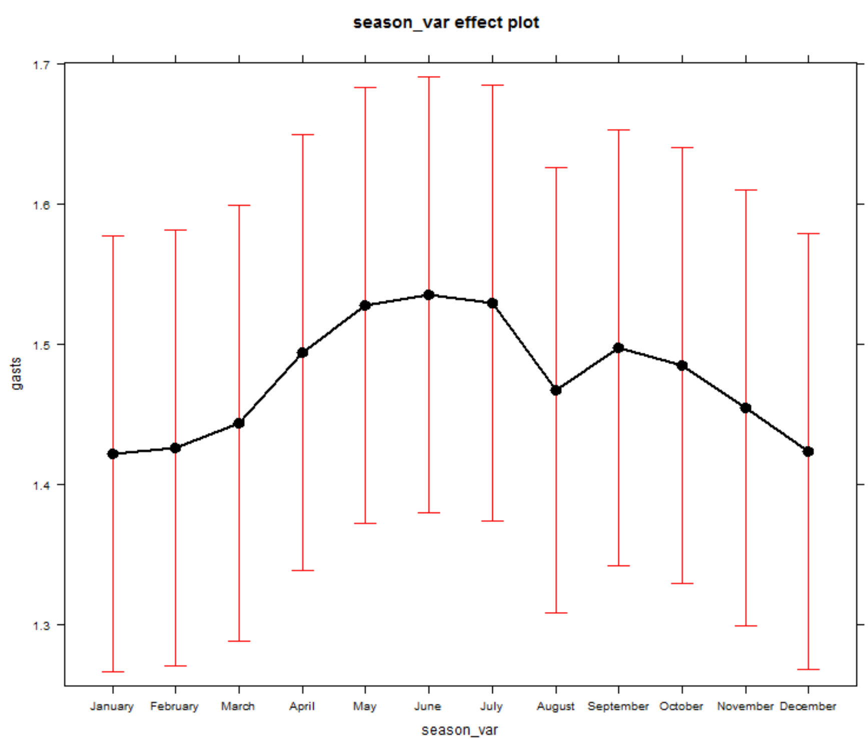




Cosine Trends:

- The seasonal means model is not very efficient if the seasonal pattern is smooth
  - We can define a function that might capture the seasonality using fewer coefficients using a periodic function

- It needs to meet at the end of one cycle and beginning of the next

$\mu_t = \beta\cos(2\pi\, f\, t + \Phi)$

- where β is the amplitude,

- f is the frequency of the oscillation (assumed known for seasonal ts)

- and Φ is the phase of the curve

    - The curve repeats itself every 1/f time units = PERIOD of the curve
        - $f=1/12 \quad \Rightarrow$ Period $=12$

- To estimate β and Φ, we would need to have access to nonlinear regression techniques.
    - Nonlinear regression models can be generally written as:

  - Nonlinear least squares defines the same target as OLS: $(y-f(x))^2$
      - Nonlinear regression models are distinguished from linear models by the derivatives with respect to the parameters (slopes) are functions of the slopes

      - Consider the partial derivatives of a linear model with log(x1), x2, and log(y):

  - NLS uses an iterative weighted least squares algorithm to find the coefficient estimates and an approximate variance-covariance matrix of the coefficients
      - For a brief discussion of `nls` see: http://cran.r-project.org/doc/contrib/Fox-Companion/appendix-nonlinear-regression.pdf
      - The main differences in using `nls` vs `lm` involve writing out all the model components and providing initial estimates of the parameters.
      - An nls version of an SLR model is:

- For the cosine seasonal trend model using NLS:
    - CC assumes that *f* is known initially
    - This corresponds to our definition of seasonality but not cycles
    - As long as we are careful about the parameter initialization, we can consider the nonlinear version using `nls`
        - later we could consider using `gnls` from the `nlme` package

- CC uses a trig identity to re-express (re-parametrize) the previous nonlinear regression model into the following linear model:

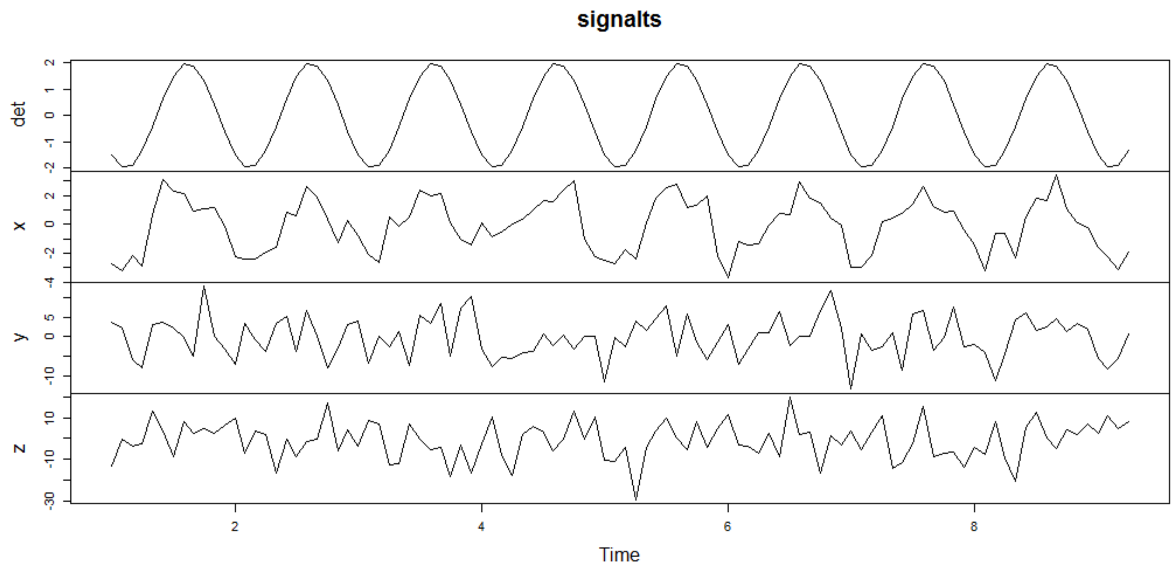
  $\beta\cos(2\pi f t + \Phi) =$

- We really would want to work with the following trend model that incorporates a constant mean (or a cosine at frequency 0):

$\mu_t = \beta_0 + \beta\cos(2\pi\ f\ t + \Phi) =$

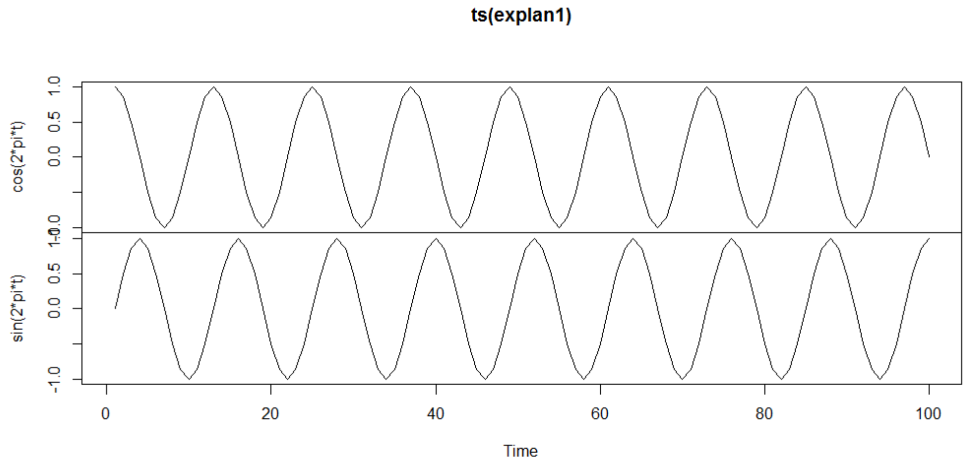
- To generate the sine and cosines, the `TSA` package contains the function `harmonic`
- Also note the very useful advice on p. 34 about how you code "time" and how that coding can change the definition of frequency
- If we continued this process for all the different frequencies then we would decompose the original series into a combination of sines and cosines at different frequencies, which is exactly what a fourier analysis actually does.

```
> time<-seq(1:100)
> det<-2*cos(2*pi*time/12+.6*pi)
> set.seed(13579)
> x<-det+rnorm(100,0,1)
> y<-det+rnorm(100,0,5)
> z<-det+rnorm(100,0,10)
> signalts<-ts(data.frame(det,x,y,z),freq=12)
> plot.ts(signalts)
```



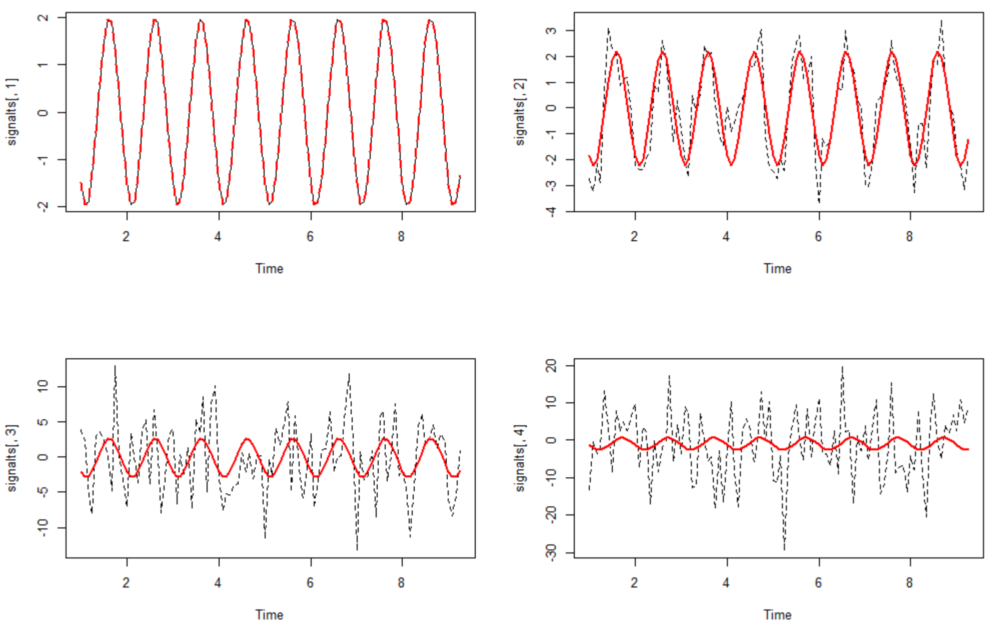

```
> require(TSA)
> explan1<-harmonic(signalts,m=1)
> head(data.frame(Month=season(signalts),round(explan1,2)),16)
       Month cos.2.pi.t. sin.2.pi.t.
1    January        1.00        0.00
2   February        0.87        0.50
3      March        0.50        0.87
4      April        0.00        1.00
5        May       -0.50        0.87
6       June       -0.87        0.50
7       July       -1.00        0.00
8     August       -0.87       -0.50
9  September       -0.50       -0.87
10   October        0.00       -1.00
11  November        0.50       -0.87
12  December        0.87       -0.50
13   January        1.00        0.00
14  February        0.87        0.50
15     March        0.50        0.87
16     April        0.00        1.00
```

```
> plot(ts(explan1))
```




```
> lm1<-lm(det~harmonic(signalts),data=signalts)
> summary(lm1)
                              Estimate Std. Error    t value Pr(>|t|)
(Intercept)                  1.887e-16  6.312e-16  2.990e-01    0.766
harmonic(signalts)cos(2*pi*t) -1.486e+00  8.923e-16 -1.666e+15   <2e-16 ***
harmonic(signalts)sin(2*pi*t) -1.338e+00  8.923e-16 -1.500e+15   <2e-16 ***

Residual standard error: 6.305e-15 on 97 degrees of freedom
Multiple R-squared:      1,    Adjusted R-squared:      1
F-statistic: 2.553e+30 on 2 and 97 DF,  p-value: < 2.2e-16

> anova(lm1)
                   Df Sum Sq Mean Sq    F value    Pr(>F)
harmonic(signalts)  2    203   101.5 2.5529e+30 < 2.2e-16 ***
Residuals          97      0     0.0
Warning message:
In anova.lm(lm1) :
  ANOVA F-tests on an essentially perfect fit are unreliable
> fit1<-fitted(lm1)
> sqrt(sum(lm1$coef[2:3]^2))
[1] 2
>
> lm2<-lm(x~harmonic(signalts),data=signalts)
> summary(lm2)
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                   -0.01511    0.09245  -0.163    0.871
harmonic(signalts)cos(2*pi*t) -1.83420    0.13069 -14.034  < 2e-16 ***
harmonic(signalts)sin(2*pi*t) -1.21600    0.13069  -9.304 4.27e-15 ***

Residual standard error: 0.9235 on 97 degrees of freedom
Multiple R-squared:  0.7479,   Adjusted R-squared:  0.7428
F-statistic: 143.9 on 2 and 97 DF,  p-value: < 2.2e-16

> anova(lm2)
                   Df  Sum Sq Mean Sq F value    Pr(>F)
harmonic(signalts)  2 245.488 122.744  143.92 < 2.2e-16 ***
Residuals          97  82.727   0.853
> fit2<-fitted(lm2)
> sqrt(sum(lm2$coef[2:3]^2))
[1] 2.200664
>
> lm3<-lm(y~harmonic(signalts),data=signalts)
> summary(lm3)
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                    -0.1441     0.4845  -0.297  0.76680
harmonic(signalts)cos(2*pi*t)  -1.9561     0.6849  -2.856  0.00525 **
harmonic(signalts)sin(2*pi*t)  -1.8924     0.6849  -2.763  0.00685 **

Residual standard error: 4.839 on 97 degrees of freedom
Multiple R-squared:  0.142,    Adjusted R-squared:  0.1243
F-statistic: 8.027 on 2 and 97 DF,  p-value: 0.0005946
```

```
> fit3<-fitted(lm3)
> lm4<-lm(z~harmonic(signalts),data=signalts)
> summary(lm4)
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                    -0.9643     0.9041  -1.067    0.289
harmonic(signalts)cos(2*pi*t)  -0.3708     1.2781  -0.290    0.772
harmonic(signalts)sin(2*pi*t)  -1.5176     1.2781  -1.187    0.238

Residual standard error: 9.031 on 97 degrees of freedom
Multiple R-squared:  0.01528, Adjusted R-squared:  -0.00502
F-statistic: 0.7528 on 2 and 97 DF,  p-value: 0.4738
> fit4<-fitted(lm4)
> par(mfrow=c(2,2))
> plot(signalts[,1])
> lines(fit1~as.vector(time(signalts)),col="red",lty=2,lwd=2)
> plot(signalts[,2],lty=2)
> lines(fit2~as.vector(time(signalts)),col="red",lty=1,lwd=2)
> plot(signalts[,3],lty=2)
> lines(fit3~as.vector(time(signalts)),col="red",lty=1,lwd=2)
> plot(signalts[,4],lty=2)
> lines(fit4~as.vector(time(signalts)),col="red",lty=1,lwd=2)
```



Or we can fit the models more directly using `nls`:

```
> 0.6*pi
[1] 1.884956
>
> model1<-nls(signalts[,1]~a+b*cos(2*pi*time/12+c),start=list(a=0,b=2,c=0.6*p
i))
Error in nls(signalts[, 1] ~ a + b * cos(2 * pi * time/12 + c), start = list(
a = 0,  :
  number of iterations exceeded maximum of 50
>
> model2<-nls(signalts[,2]~a+b*cos(2*pi*time/12+c),start=list(a=0,b=2,c=0.6*p
i))
> summary(model2)

Formula: signalts[, 2] ~ a + b * cos(2 * pi * time/12 + c)

Parameters:
  Estimate Std. Error t value Pr(>|t|)
a -0.01511    0.09245  -0.163    0.871
b  2.20066    0.12971  16.966   <2e-16 ***
c  2.03256    0.05983  33.972   <2e-16 ***
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9235 on 97 degrees of freedom

Number of iterations to convergence: 3 
Achieved convergence tolerance: 2.132e-08 

> model3<-nls(signalts[,3]~a+b*cos(2*pi*time/12+c),start=list(a=0,b=2,c=0.6*p
i))
> summary(model3)

Formula: signalts[, 3] ~ a + b * cos(2 * pi * time/12 + c)

Parameters:
  Estimate Std. Error t value Pr(>|t|)    
a  -0.1441     0.4845  -0.297 0.766795    
b   2.7217     0.6793   4.007 0.000121 ***
c   1.8491     0.2537   7.290 8.39e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.839 on 97 degrees of freedom

Number of iterations to convergence: 3 
Achieved convergence tolerance: 9.352e-09 

> 
> model4<-nls(signalts[,4]~a+b*cos(2*pi*time/12+c),start=list(a=0,b=2,c=0.6*p
i))
> summary(model4)

Formula: signalts[, 4] ~ a + b * cos(2 * pi * time/12 + c)

Parameters:
  Estimate Std. Error t value Pr(>|t|)
a  -0.9643     0.9041  -1.067    0.289
b   1.5622     1.2733   1.227    0.223
c   1.2869     0.8212   1.567    0.120

Residual standard error: 9.031 on 97 degrees of freedom

Number of iterations to convergence: 4 
Achieved convergence tolerance: 2.967e-08
```

Note that although one method used nls and the other ls, that the models are actually equivalent here (same likelihood and # parameters):

```
> AIC(lm4,model4)
       df      AIC
lm4     4 728.8845
model4  4 728.8845

> mean(abs(fitted(lm4)-fitted(model4)))
[1] 2.270966e-07
```

- Note that this equivalence may not always hold due to differences in numerical maximization routines and parameter initializations between `nls` and linearized versions of the models.
    - It does suggest that although the initial version of the model is nonlinear it is actually close to the linearized version of the model
    - The inference in the nonlinear model is probably pretty safe
        - Stability of `nls` is related to how linear the model is, with *a* and *b* entering the nonlinear model linearly, the only coefficient with "nonlinearity" attached to it is the phase coefficient, *c*.