- And for the stationary moving average process from the text where current observation includes the current innovation – 0.5*previous innovation

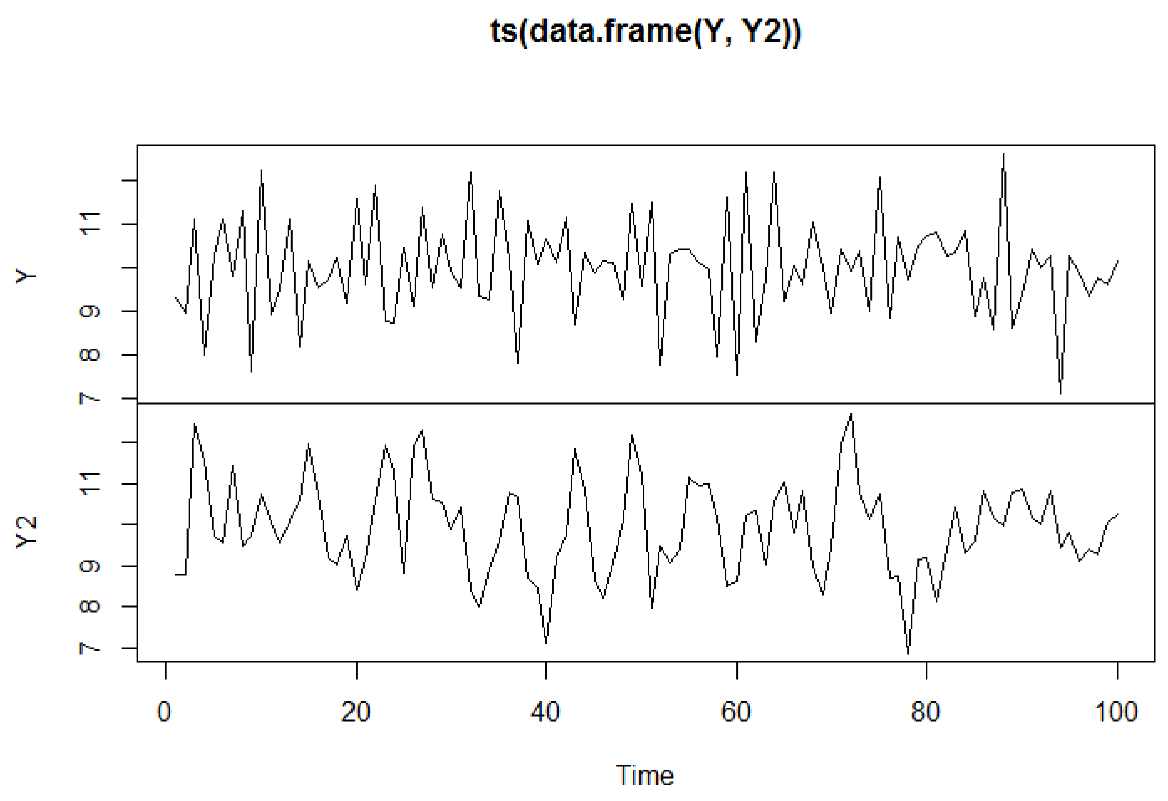
    - So accommodating a negative lag 1 correlation actually improves our inference for the mean!

- Suppose that it was +0.5* previous innovation….
    - And we ignored that component of the process and used our standard procedure?

So let's try this out for a real process using a mixed model (generalized least squares) approach (later we will revisit this using an ARIMA model)…

Using an AR(1) error structure… is this correct?

```
> require(nlme)
> require(TSA)
> set.seed(97531)
> mu<-10; e<-rnorm(101)
> ek<-e-0.5*zlag(e)
> Y<-mu+ek[-1]
> summary(lm(Y~1))
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   9.9910     0.1152   86.69   <2e-16 ***

Residual standard error: 1.152 on 99 degrees of freedom

> summary(gls(Y~1,correlation=corAR1(),method="ML"))
Generalized least squares fit by maximum likelihood
       AIC      BIC    logLik
  284.3187 292.1342 -139.1594

Correlation Structure: AR(1)
 Formula: ~1
 Parameter estimate(s):
      Phi
-0.527181

Coefficients:
               Value  Std.Error  t-value p-value
(Intercept) 9.992899 0.06415248 155.7679       0

> e2<-rnorm(101)
> ek2<-e2+0.5*zlag(e2)
> Y2<-mu+ek2[-1]
> summary(lm(Y2~1))
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   9.9232     0.1173    84.6   <2e-16 ***

Residual standard error: 1.173 on 99 degrees of freedom

> summary(gls(Y2~1,correlation=corAR1(),method="ML"))
Generalized least squares fit by maximum likelihood
  Model: Y2 ~ 1
  Data: NULL
       AIC      BIC    logLik
  304.2658 312.0813 -149.1329

Correlation Structure: AR(1)
 Formula: ~1
 Parameter estimate(s):
      Phi
0.3890659

Coefficients:
               Value Std.Error  t-value p-value
(Intercept) 9.918282 0.1755991 56.48255       0

Residual standard error: 1.166068
Degrees of freedom: 100 total; 99 residual
> plot.ts(ts(data.frame(Y,Y2)))
```



AR(1) process that was used in the previous model (ignoring the mean being non-zero to start):

Back to CC page 29…

- Equation 3.2.5 provides a useful approximation and then uses it with a geometric series to consider the results for a stationary AR(1) process where $\rho_k = \phi^{|k|}$ with ϕ between -1 and 1 (producing equation 3.2.6):

- And finally, CC show similar results for a random walk to illustrate what happens when nonstationarity is encountered (error in equation at top of CC p. 30)

3.3: Regression Methods (CM Ch. 5):

Linear Trends:

Model specification

Estimation

Testing

Assumptions?

Nonlinear Time Trends:

- Polynomial trends using linear models

  - Issue with polynomial estimation and testing with raw time?

- Nonparametric regression techniques provide polynomial-like trends that are more flexible and sometimes more efficient with using up degrees of freedom

  - Kernel regression, **Regression Splines**, Local Polynomial Regression are all possibilities
    - some are implemented with Generalized Additive Models (GAMs, good references: Wood 2006; Zuur et al. 2009)

- Smoothing splines are the easiest quickly accessed method (especially for interpolating missing observations)
  - `Model1<-smooth.spline(time,y)`
    - Defaults to choosing the smoothness of the estimated function using generalized cross-validation
  - `predict(Model1,x)`
    - where x could be or include the times that have missing values within the range of the time series
    - Can be useful for interpolating missing observations to access typical time series methods that aren't very forgiving regarding missing observations
    - Also could be used to change the sampling rate or to go from unequally spaced observations to equally spaced (in time) observations
    - Can also access estimated derivative with respect to x via the predict function

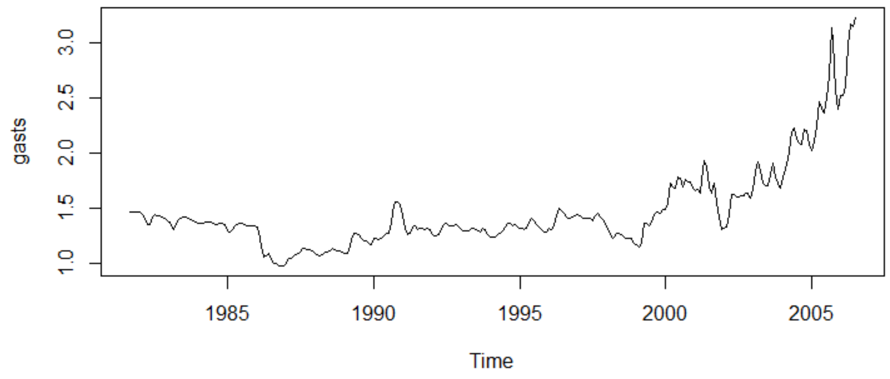
- Nonlinear regression models (y_t=f(time)+e_t) are also possible but rarely used

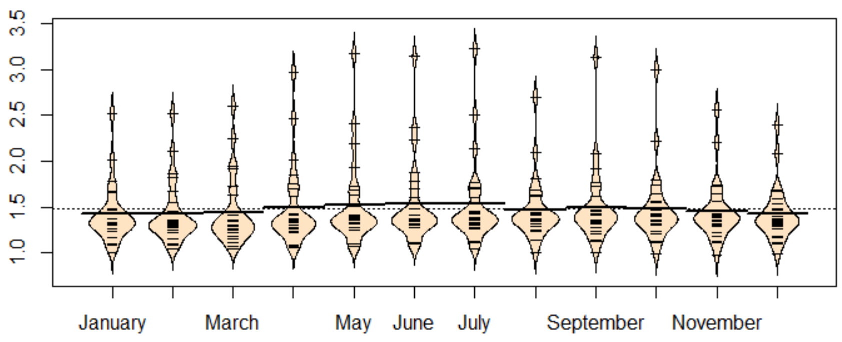
- Transformations can linearize a nonlinear time trend:
  - log(y_t)=…+e_t

  - Implications for models on original scale is not always considered (or sometimes this transformation is used because of what it means on the original scale)

Cyclical or Seasonal "Trends":

- There are many ways to deal with seasonality in Time Series models
  - Remember that seasonality is a consistent difference in the means that corresponds with the natural period of the series
  - This can be in the presence of a stochastic or deterministic long-term trend.
- The most "flexible" deterministic approach doesn't assume that there is any (smooth) pattern to the seasonality
  - Each step in the cycle is fit with its own unique value (called the "seasonal means model" using additive seasonal indicators (CM p.99)):
  - The `cycle` and `season` functions can extract the seasonal numbers or names from a `ts` object:
    - `month<-cycle(gasts)`
  - In R, if we have the month coded numerically, then `lm(y~factor(month)-1)` will fit the "cell means" version of a seasonal means model, ignoring the long term trend for the moment.





```
> gas<-read.table("https://dl.dropboxusercontent.com/u/77307195/usa.txt")
> colnames(gas)<-c("Date","Price")
> gasts<-ts(gas$Price,frequency=12,start=c(1981,9))
> par(mfrow=c(2,1))
> ts.plot(gasts)
> require(TSA)
> require(beanplot)
> beanplot(gasts~season(gasts),col="bisque",method="jitter",log="")

> seas1<-lm(gasts~season(gasts))
> anova(seas1)
Analysis of Variance Table
               Df Sum Sq  Mean Sq F value Pr(>F)
season(gasts)  11  0.492 0.044685   0.287  0.988
Residuals     287 44.687 0.155703

> seas_cm<-lm(gasts~season(gasts)-1)
> summary(seas_cm)
                       Estimate Std. Error t value Pr(>|t|)
season(gasts)January    1.42184    0.07892   18.02   <2e-16 ***
season(gasts)February   1.42556    0.07892   18.06   <2e-16 ***
season(gasts)March      1.44320    0.07892   18.29   <2e-16 ***
season(gasts)April      1.49396    0.07892   18.93   <2e-16 ***
season(gasts)May        1.52760    0.07892   19.36   <2e-16 ***
season(gasts)June       1.53504    0.07892   19.45   <2e-16 ***
season(gasts)July       1.52936    0.07892   19.38   <2e-16 ***
season(gasts)August     1.46721    0.08055   18.22   <2e-16 ***
season(gasts)September  1.49712    0.07892   18.97   <2e-16 ***
season(gasts)October    1.48444    0.07892   18.81   <2e-16 ***
season(gasts)November   1.45420    0.07892   18.43   <2e-16 ***
season(gasts)December   1.42376    0.07892   18.04   <2e-16 ***

Residual standard error: 0.3946 on 287 degrees of freedom
Multiple R-squared:  0.9358,   Adjusted R-squared:  0.9331
F-statistic: 348.6 on 12 and 287 DF,  p-value: < 2.2e-16

> sqrt(.1557)
[1] 0.3945884

> sqrt(.1557)/sqrt(25)
[1] 0.07891768

> sqrt(.1557)/sqrt(24)
[1] 0.08054502

> confint(seas_cm)
                          2.5 %   97.5 %
season(gasts)January   1.266508 1.577172
season(gasts)February  1.270228 1.580892
season(gasts)March     1.287868 1.598532
season(gasts)April     1.338628 1.649292
season(gasts)May       1.372268 1.682932
season(gasts)June      1.379708 1.690372
season(gasts)July      1.374028 1.684692
season(gasts)August    1.308673 1.625744
season(gasts)September 1.341788 1.652452
season(gasts)October   1.329108 1.639772
season(gasts)November  1.298868 1.609532
season(gasts)December  1.268428 1.579092
```

But is there any evidence to support this model over a common mean for all months?

```
> anova(seas_cm)
Analysis of Variance Table

Response: gasts
               Df Sum Sq Mean Sq F value    Pr(>F)    
season(gasts)  12 651.27  54.272  348.56 < 2.2e-16 ***
Residuals     287  44.69   0.156                      
```

```
> reducedmodel<-lm(gasts~1) #mean-only model
> anova(reducedmodel,seas_cm)
Analysis of Variance Table

Model 1: gasts ~ 1
Model 2: gasts ~ season(gasts) - 1
  Res.Df    RSS Df Sum of Sq     F Pr(>F)
1    298 45.178                          
2    287 44.687 11   0.49153 0.287  0.988
```

```
> AIC(reducedmodel,seas_cm)
             df      AIC
reducedmodel  2 287.4663
seas_cm      13 306.1954
```

- What are the assumptions of this model?