A brief introduction to time-series analysis

Carsten F. Dormann
Biometry & Environmental System Analysis
University of Freiburg, Germany

November 19, 2019

Contents

Abstract

This introductory document aims to raise a few points that typically show up in first steps in time-series analysis.

1 Introduction

There is a sheer endless number of time-series resources out there![1] Have a look at the open book of Hyndman & Athanasopoulos, called "Forecasting: Principles and Practice".[2]

Time series refers to data sets of a response that is recorded over longer periods of time. The response can be univariate (e.g. the famous Keeling CO_2-data we'll look at below), or multidimensional (e.g. climate data from 121 weather stations, or share values for 4201 companies in Japan).

There are, fundamentally, two different types of questions, and hence approaches, to time series analysis: (a) understanding whether something has an effect on a response that just happens to be temporal; (b) predicting a time series into the future ("forecasting"). In other words, (a) looks at *significant* effects of some predictor on y, just like in a regression model; (b) makes extrapolations of y.

Let's compare two rather different univariate time series examples.[3]

[1] `https://www.statmethods.net/advstats/timeseries.html`

[2] `https://otexts.org/fpp2/`

[3] Get the extended version of the `co2`-data for validation of forecast e.g. here: `https://datahub.io/core/co2-ppm`.

```
par(mfrow=c(1,2))
library(ggplot2)
library(gridExtra)
library(fpp2) # for the book of Hyndman & Athanasopoulos; calls ggplot2
p1 <- autoplot(co2)
p2 <- autoplot(melsyd[,"Economy.Class"]) +
  ggtitle("Economy class passengers: Melbourne-Sydney")
grid.arrange(p1, p2, ncol=2)
```

One shows a very clear seasonal pattern, the other not; one shows a clear trend, the other not. For neither we have explanatory variables. Both are **time-series objects** in *R*!

```
str(co2)

##  Time-Series [1:468] from 1959 to 1998: 315 316 316 318 318 ...

str(melsyd)

##  Time-Series [1:283, 1:3] from 1987 to 1993: 1.91 1.85 1.86 2.14 2.12 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:3] "First.Class" "Business.Class" "Economy.Class"
```

The nice thing about having data as time-series object is that they work well for plotting. It is somewhat tricky to get them into this shape, though. There is no time and place here to go into that, but you may want to consult the help pages of `ts` and `POSIXct` to generate and format temporal data.[4] Typically, time series require NA-free data!

Since melsyd's Economy.Class has one NA, we replace it by the mean of the data points around it (no comment):

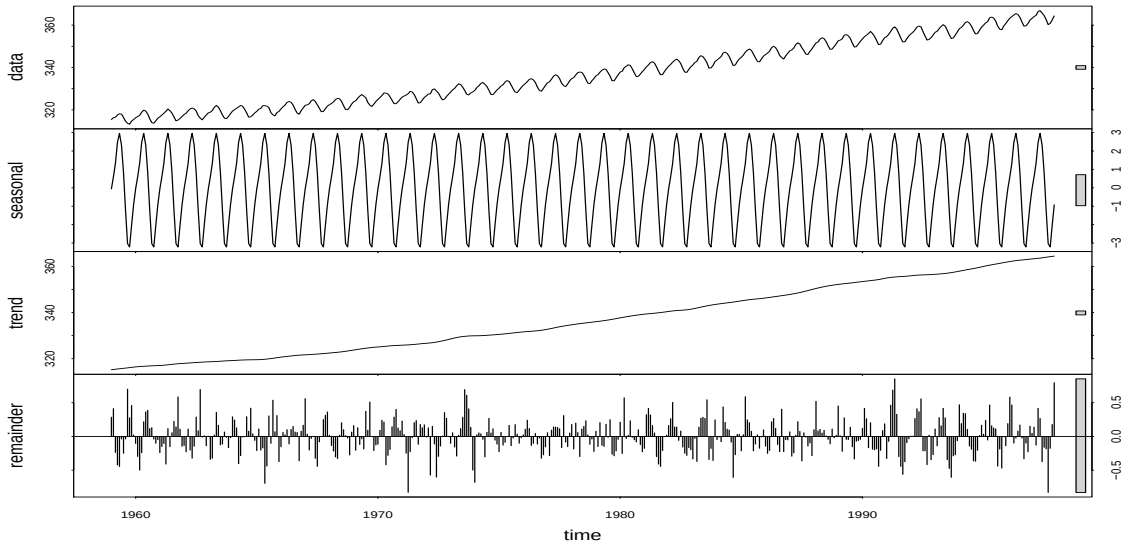
```
(missing <- which(is.na(melsyd[, "Economy.Class"])))

## [1] 13

melsyd[missing, "Economy.Class"] <- mean(melsyd[c(missing-1, missing+1), "Economy.Class"], na.rm=T
MSE <- melsyd[, "Economy.Class"]
```

We shall now go through a few typical exploratory steps, before turning to actual temporal models.

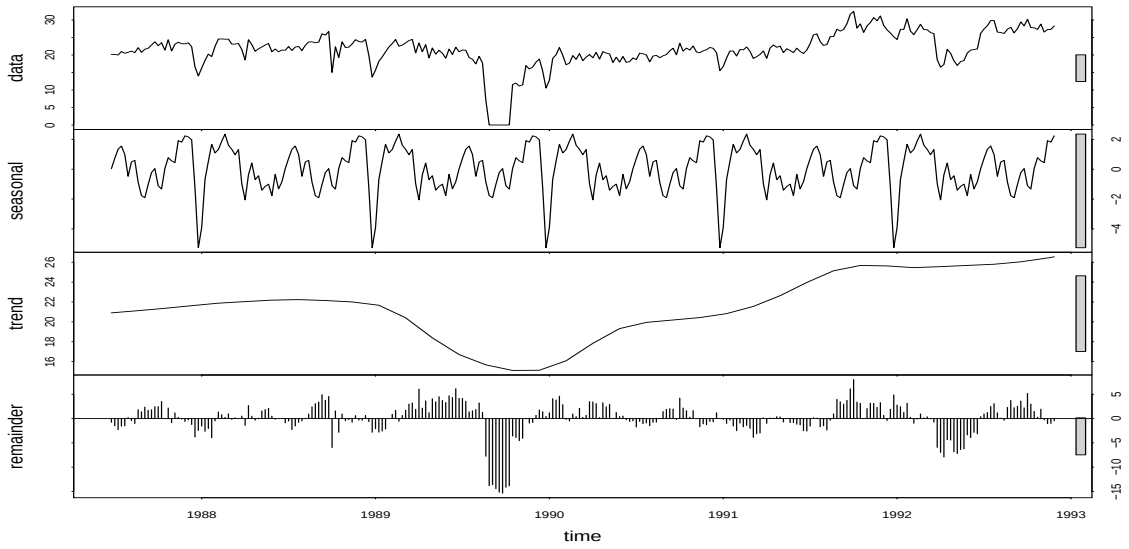
2 *Decomposition into trend, seasonality and noise using loess*

[4] In the "tidyverse", the **lubridate** is your friend!

```
plot(stl(co2, s.window="periodic"))
```




```
# s.window: period in lags for seasonal extraction, minimally 7; or "periodic"
plot(stl(MSE, s.window="periodic"))
```



The two decompositions couldn't be more different. This "stl" correctly identifies the annual cycle in the CO_2-concentration, and it also finds a periodicity in the air travels (with a noticeable dip around Christmas). But, as the grey boxes on the right-hand side of each panel show, the remainder (aka "noise") is a tiny fraction of the trend in the CO_2 data, but a substantial part in the flights (note that this is a scaling bar, i.e. it covers the same data range in each panel; roughly 1.8 for CO_2 and 6 for the flights).

3 *Detrending*

Many analyses start with detrending, i.e. removing a trend that exists in the data. That would be the "trend" component in the above plots. "Why?", you may ask. Well, for example if you are only interested in the periodic element, but not in the trend.

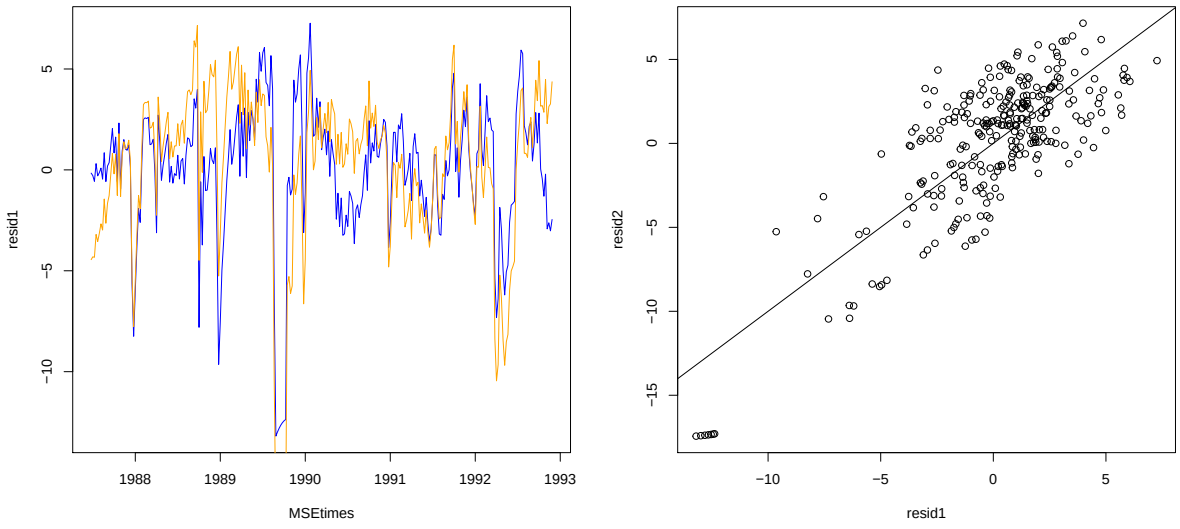
Think of tree ring widths. As the tree becomes older, tree rings become wider (because only the outer few cm provide all the water flux for the whole tree), and then smaller

again (as the old tree growths slower). If you now want to compare tree-ring widths across many trees, you need to remove the age-trend in these data, because these are irrelevant for, say, the climate signal in the data.

Detrending is done in slightly different ways in different disciplines. Two things seems to be clear: the choice of detrending affects the results, and **linear detrending is probably universally wrong** (and hence will not be shown here).

Let's have a look at two ways of detrending, applied to the flights data. The first is detrending using a spine, the second uses "empirical mode decomposition" (aka Hilbert-Huang transformation[5]), which seems to be a favourite with physicists (so it must be great):

```
library(mgcv)
MSEtimes <- as.numeric(time(MSE))
library(zoo)
head(as.yearmon(MSEtimes)) # just show that these are monthly values

## [1] "Jun 1987" "Jul 1987" "Jul 1987" "Jul 1987" "Jul 1987" "Jul 1987"

detrended1 <- gam(MSE ~ s(MSEtimes))
#plot(detrended1, residuals=T, pch="+") # for plotting the fitted trend
resid1 <- residuals(detrended1)
# now for EMD.
library(EMD)
detrended2 <- emd(xt=MSE, tt=MSEtimes)
# plot(MSEtimes, detrended2$residue) # the trend extracted
resid2 <- MSE - detrended2$residue
par(mfrow=c(1,2))
plot(MSEtimes, resid1, col="blue", type="l")
lines(MSEtimes, resid2, col="orange")
plot(resid1, resid2)
abline(0,1)
```



So clearly there is some similarity, and some difference between the methods.

4 *(Fourir- and wavelet-transformations)*

(Not covered here, for lack of time.)

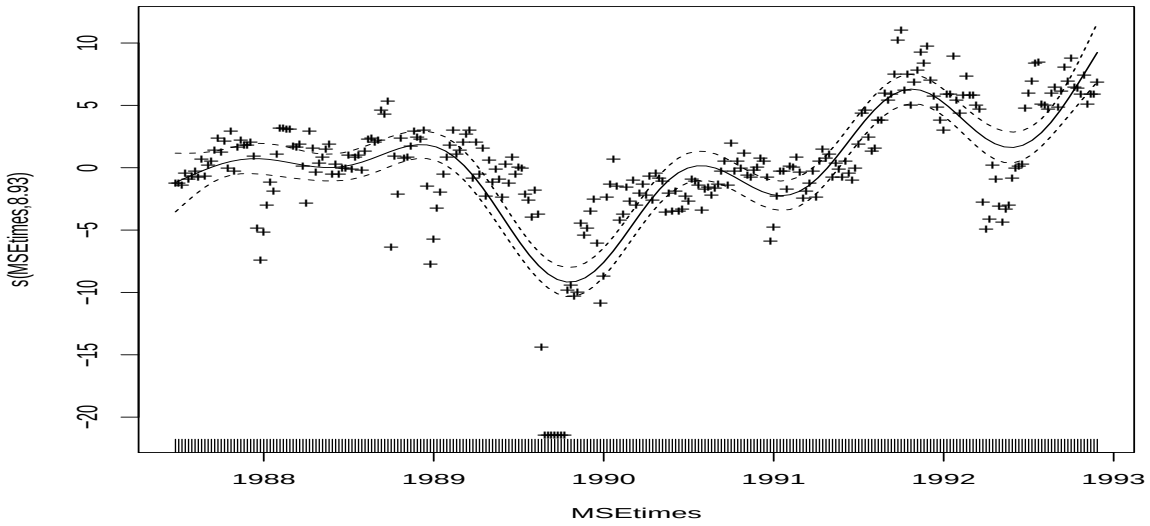
[5] https://en.wikipedia.org/wiki/Hilbert-Huang_transform

5 *Testing for a linear trend*

Why would you want to test for a *linear* trend? There are infinitely many shapes a trend can have, why single out the linear? Just like with detrending, very often people analyse for linear trends because that is what they learned, not what they need. Do you really have an hypothesis about linearity *before* you look at the data?[6]

Instead, we can analyse for a trend using the above GAM and get a significance of that trend.

```
summary(detrended1)

##
## Family: gaussian
## Link function: identity
##
## Formula:
## MSE ~ s(MSEtimes)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  21.5132     0.2037   105.6   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##              edf Ref.df     F p-value
## s(MSEtimes) 8.93  8.999 40.92  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.563   Deviance explained = 57.7%
## GCV = 12.166  Scale est. = 11.739    n = 283

plot(detrended1, residuals=T, pch="+")
```

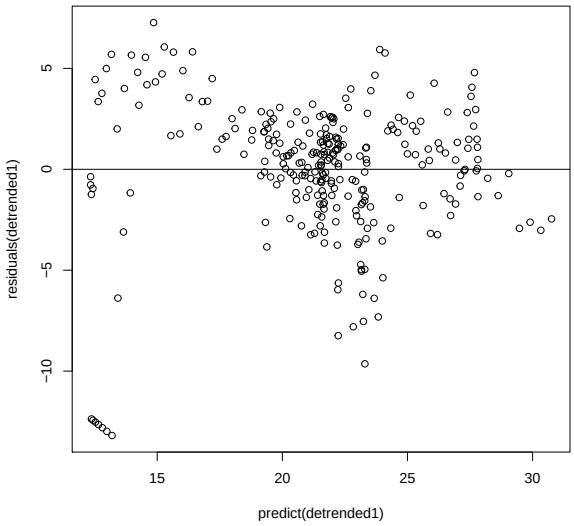


Isn't that great? So there is a non-linear trend (because the edf is > 1), which is highly significant and explains 58/

WAIT! Just like any other regression model, also this model has assumptions that need to be checked. Let's have a look at the residuals:

[6] https://stats.stackexchange.com/questions/225003/test-for-trend-and-seasonality-in-time-series

```
plot(predict(detrended1), residuals(detrended1))
abline(h=0)
```

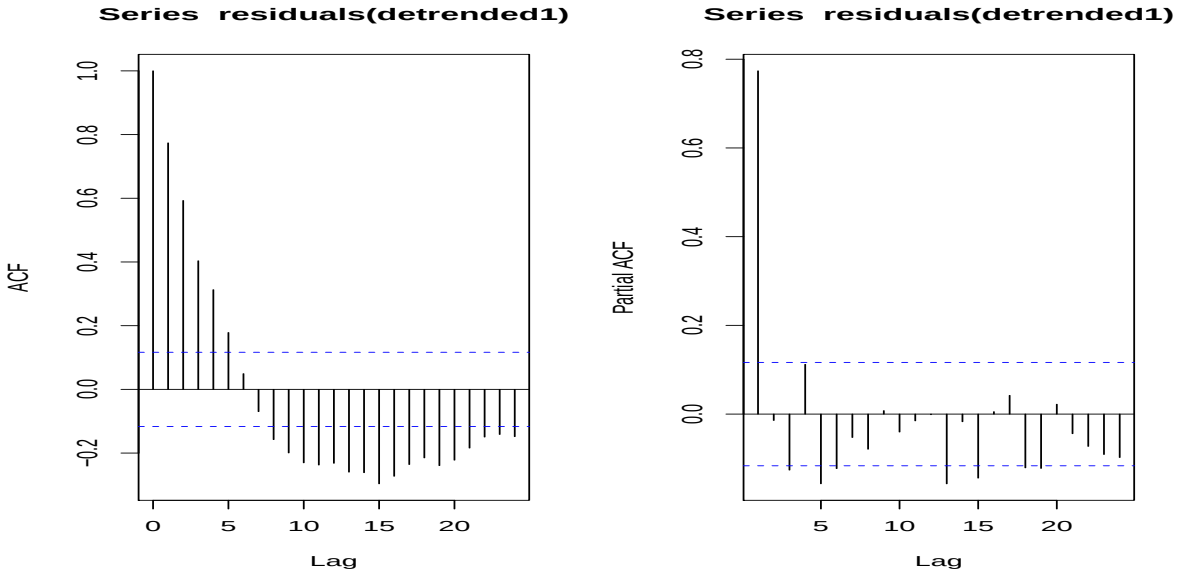


Hm, there is the cluster of 0s at the bottom left, which causes some heterogeneity, but otherwise there is no obvious pattern in the data.

Given that the data are collected over some time *at the same place*, we may expect that they are not independent. If I know how many people flew in January, I can guess fairly accurately how many will fly in February. Thus, there may well be **temporal autocorrelation** in the data, which we should investigate!

The most common way to do this is by means of the (partial) autocorrelation function-plot:

```
par(mfrow=c(1,2))
acf(residuals(detrended1))
pacf(residuals(detrended1))
```



What the ACF-plot shows is the correlation (on the y-axis) of the residuals with themselves, shifted by 1, 2, 3, ... positions ("lag"). Clearly, residuals are *not* independent, but carry some temporal autocorrelation for up to 5 units (months, in this case).

The partial ACF-plot is more interesting and revealing. It is computed by subtracting

from each correlation the expected correlation from all previous lags.[7] As a result, we see that really there is "only" a strong 1-lag autocorrelation, which carries over to the following lags.

Thus, *our data are not independent and hence violate the assumptions of maximum likelihood estimation* (and thereby those of OLS and GAM etc.). We have to find a way to accommodate the temporal autocorrelation! We do that in the next section.

6 "Time-aware" (generalised) linear models

OLS, GLM and alike assume independence of data points. This is explicit in the fact that we compute maximum likelihood as the sum of log-transformed probability densities of the data points. This sum is valid only if data are independent (since $P(A, B) = P(A)P(B)$ iff[8] A and B are independent).

We can write a linear (additive) model in this form:

$$\mathbf{y} = f(\mathbf{X}) + \varepsilon, \quad \text{with} \quad \varepsilon \sim \mathcal{N}(\mu = 0, \sigma = c),$$

where f is some function of the various predictors $\mathbf{X}$ in the model, and there is a normally distributed additive error ε (with mean 0).

When data are non-independent, ε comes from a different distribution, namely one where the value of ε_1 is correlated with that of ε_2, and so forth. This dependence has to describe the PACF we visualised in the previous section. Mathematically, we write

$$\varepsilon \sim \mathcal{N}(0, \boldsymbol{\Sigma}),$$

where $\boldsymbol{\Sigma}$ is an $n \times n$ matrix, with entries s_{ij}^2, the expected co-variances between data points. $\boldsymbol{\Sigma}$ is called a "variance-covariance" matrix.

So, how do we estimate s_{ij}^2? Well, we assume that the correlation of two data points is a function of their distance in time, ΔT, e.g. decreasing exponentially:

$$s_{ij}^2 = ae^{-b\Delta T} = ae^{-b|T_i - T_j|},$$

with a and b being positive parameters to be estimated from the data.

Or we can make the specific assumption that only a lag of one is present, and hence only the previous data point is correlated with the focal point:

$$s_{ij}^2 = \begin{cases} c & \text{if } |i - j| = 1 \\ 0 & \text{else} \end{cases}$$

In this case, $\boldsymbol{\Sigma}$ will be a matrix with entries only on the diagonal and the first off-diagonals (if the data are arranged by time). The entries on the diagonal are the variances of each data point, which we typically assume to be identical, and the constant off-diagonal entries c are the co-variances.

This model is known as the autoregressive model with lag one, AR1. It estimates essentially two parameters, apart from those in the function f, namely c and s_{ii}^2. This latter variance is now independently normal distributed ("white noise").

Estimation of the time-dependent covariances is achieved using "Generalised Least Squares" (GLS), which actually is relatively simple in the case of an AR1-model. So let us amend our previous flight-analysis by an AR1 structure, first in a GLS before going to its implementation in the GAM:

[7] https://en.wikipedia.org/wiki/Partial_autocorrelation_function

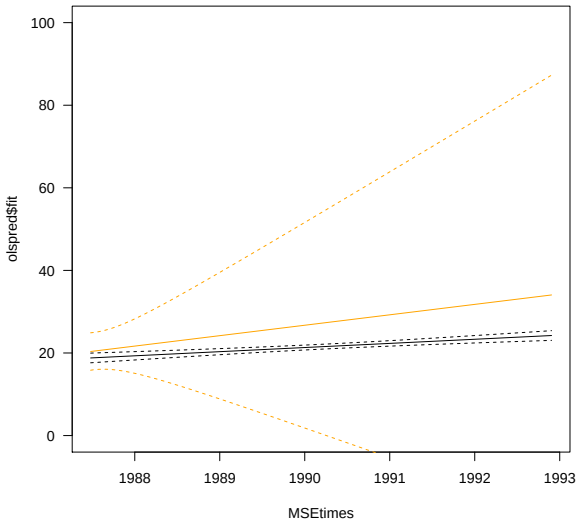
[8] No, this is not a typo! "Iff" is the mathematical shorthand for "if and only if".

```r
#summary(OLS1 <- gls(MSE ~ MSEtimes))
summary(OLS1 <- lm(MSE ~ MSEtimes)) # identical!

##
## Call:
## lm(formula = MSE ~ MSEtimes)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -21.0895  -2.0586   0.7456   3.3912   9.3949
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1971.5133   372.6978  -5.290 2.47e-07 ***
## MSEtimes        1.0014     0.1873   5.348 1.85e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.949 on 281 degrees of freedom
## Multiple R-squared:  0.09237,Adjusted R-squared:  0.08914
## F-statistic:  28.6 on 1 and 281 DF,  p-value: 1.849e-07

GLS1.AR1 <- gls(MSE ~ MSEtimes, correlation=corAR1(form= ~MSEtimes))
summary(GLS1.AR1)

## Generalized least squares fit by REML
##   Model: MSE ~ MSEtimes
##   Data: NULL
##        AIC      BIC    logLik
##   1264.919 1279.472 -628.4593
##
## Correlation Structure: ARMA(1,0)
##  Formula: ~MSEtimes
##  Parameter estimate(s):
##      Phi1
## 0.9538845
##
## Coefficients:
##                 Value Std.Error    t-value p-value
## (Intercept) -4998.352  9775.426 -0.5113181  0.6095
## MSEtimes        2.525     4.918  0.5134086  0.6081
##
##  Correlation:
##          (Intr)
## MSEtimes -1
##
## Standardized residuals:
##         Min          Q1         Med          Q3         Max
## -11.5079978  -3.6676393  -2.5622524  -0.6448213   1.4113635
##
## Residual standard error: 2.271187
## Degrees of freedom: 283 total; 281 residual
```

This is to show that the temporal autocorrelation can severely affect your model fit and inference!

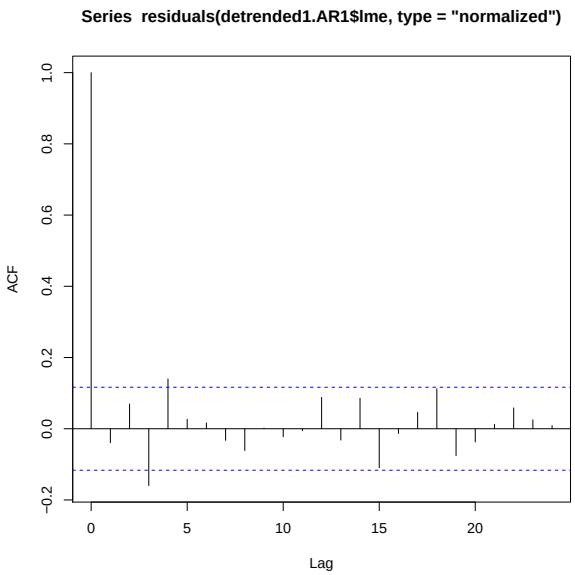
The ordinary linear regression is highly significant, suggesting an increase by one passenger per month, while the GLS finds nothing of that sort, due to dramatically larger standard error:

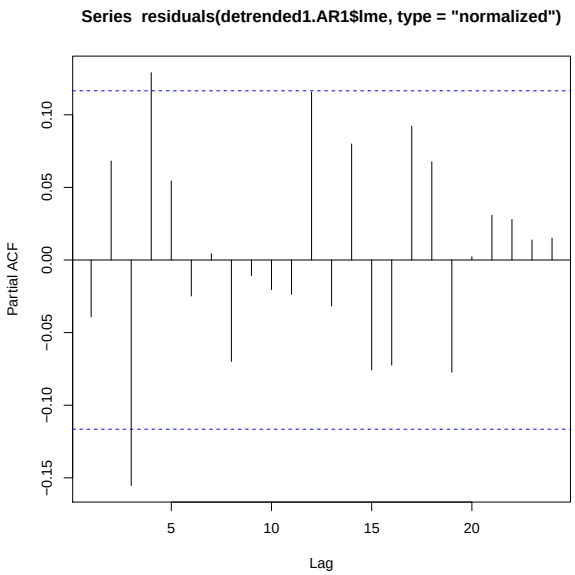
```r
olspred <- predict(OLS1, se.fit=T)
library(AICcmodavg)
glspred <- predictSE(GLS1.AR1, newdata=data.frame("MSEtimes"=MSEtimes), se.fit=T, ylab="passenger
plot(MSEtimes, olspred$fit, type="l", las=1, ylim=c(0, 100))
lines(MSEtimes, olspred$fit + 2*olspred$se.fit, lty=2)
lines(MSEtimes, olspred$fit - 2*olspred$se.fit, lty=2)
lines(MSEtimes, glspred$fit, col="orange", lty=1)
lines(MSEtimes, glspred$fit + 2*glspred$se.fit, col="orange", lty=2)
lines(MSEtimes, glspred$fit - 2*glspred$se.fit, col="orange", lty=2)
```



Now we try the same with a more sophisticated wiggly line for the trend, only to immediately encounter a problem:

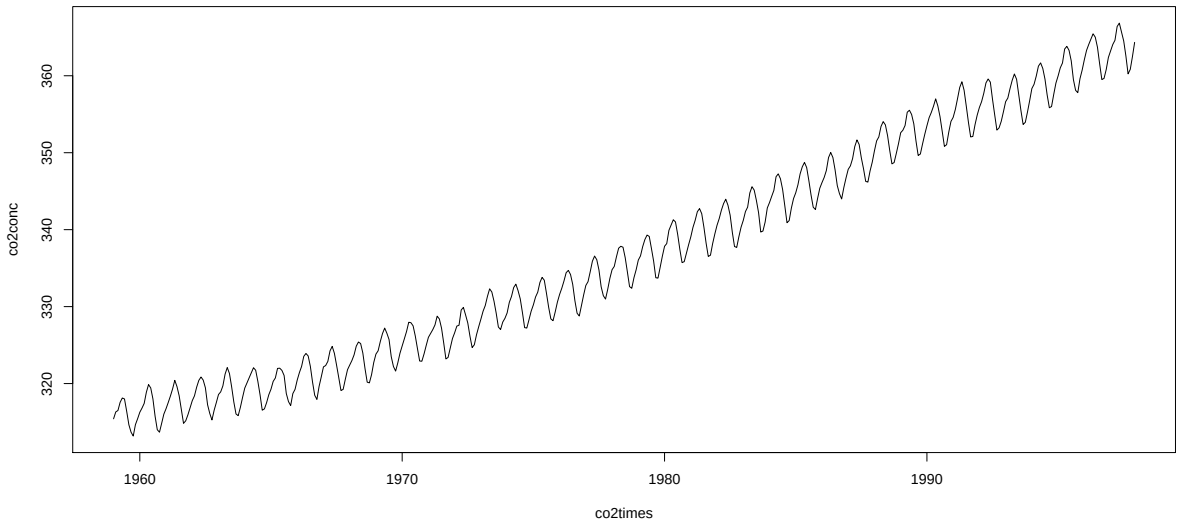
```r
detrended1.AR1 <- gamm(MSE ~ s(MSEtimes, bs="cs"), correlation=corAR1(form= ~MSEtimes))

## Error in chol.default(V$V): the leading minor of order 2 is not positive definite

# estimate phi independently:
res <- residuals(detrended1)
gls(res ~ 1, correlation=corAR1(form= ~MSEtimes)) # 0.7665

## Generalized least squares fit by REML
##   Model: res ~ 1
##   Data: NULL
##   Log-restricted-likelihood: -629.3797
##
## Coefficients:
## (Intercept)
##  0.01227999
##
## Correlation Structure: ARMA(1,0)
##  Formula: ~MSEtimes
##  Parameter estimate(s):
##      Phi1
## 0.7665213
## Degrees of freedom: 283 total; 282 residual
## Residual standard error: 2.253483

detrended1.AR1 <- gamm(MSE ~ s(MSEtimes, bs="cs"), correlation=corAR1(0.77)) # re-estimates phi
par(mfrow=c(1,2))
acf(residuals(detrended1.AR1$lme, type="normalized")) # gone! (more or less)
pacf(residuals(detrended1.AR1$lme, type="normalized"))
```



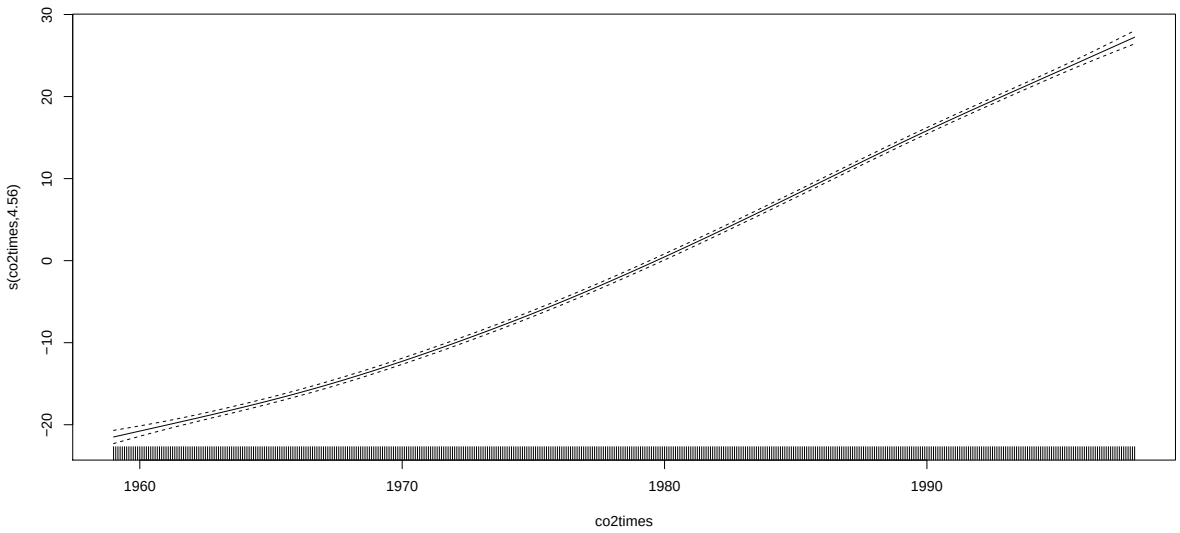


Time for an analysis of the CO_2-data!

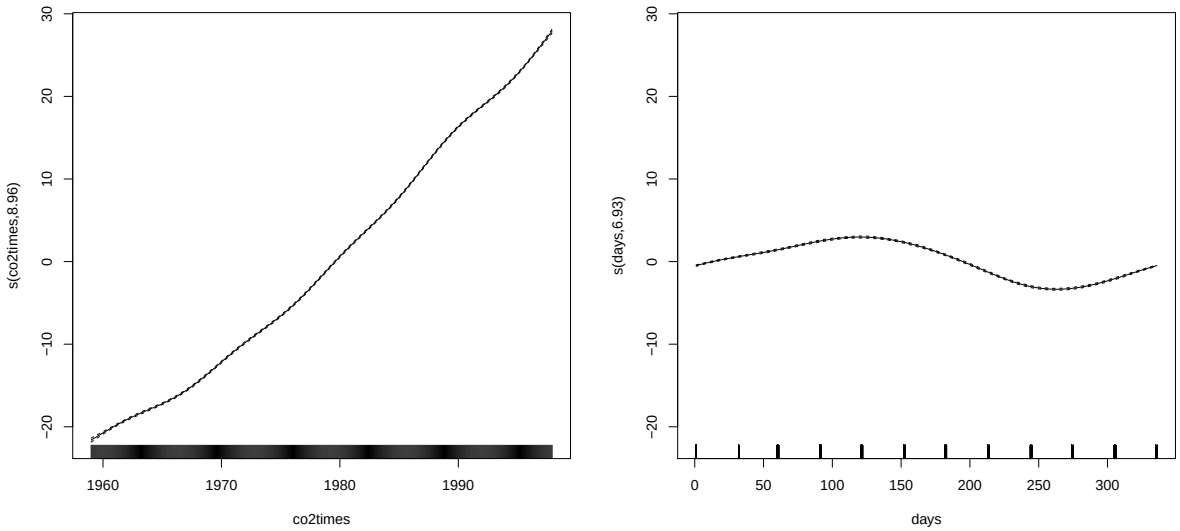
```
co2times <- as.numeric(time(co2))
co2conc <- as.numeric(co2)
plot(co2times, co2conc, type="l")
```



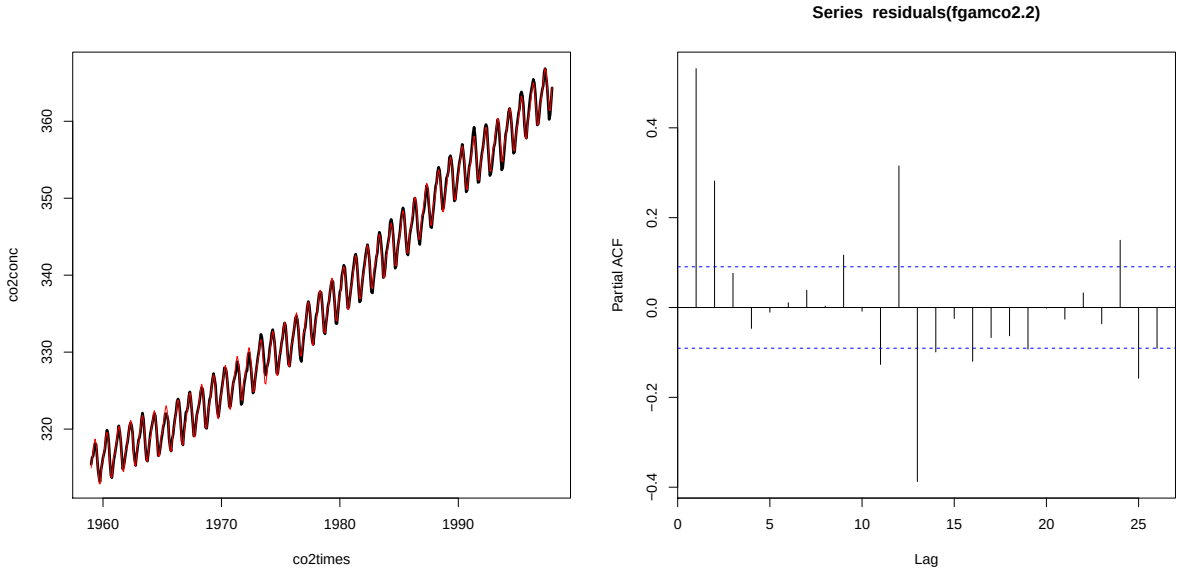

```
fgamco2.1 <- gam(co2conc ~ s(co2times))
plot(fgamco2.1) # only the trend, no season predictions
```



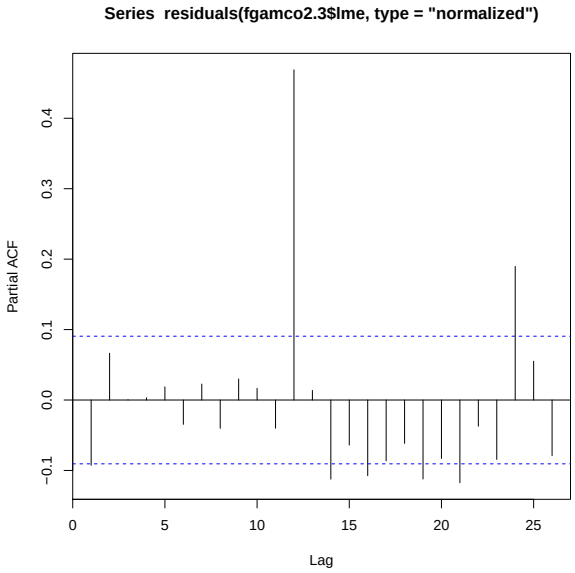

```r
days <- as.numeric(format(as.Date(time(co2)), "%j")) # ugly
fgamco2.2 <- gam(co2conc ~ s(co2times) + s(days, bs="cc")) # adds season
# plot effects
par(mfrow=c(1,2))
plot(fgamco2.2)
```




```r
# plot fit
plot(co2times, co2conc, type="l", lwd=3)
lines(co2times, predict(fgamco2.2), col="red") # alright, I guess
pacf(residuals(fgamco2.2)) # more than lag 1 !
```




```
fgamco2.3 <- gamm(co2conc ~ s(co2times) + s(days, bs="cc"), correlation=corAR1(0.6))
pacf(residuals(fgamco2.3$lme, type="normalized")) # fine, but spike at 12!
#period12 <- as.numeric(format(as.Date(time(co2)), "%Y")) %% 12
#fgamco2.4 <- gam(co2conc ~ s(co2times) + s(days, bs="cc") + s(period12, bs="cc"))
#pacf(residuals(fgamco2.4))
```

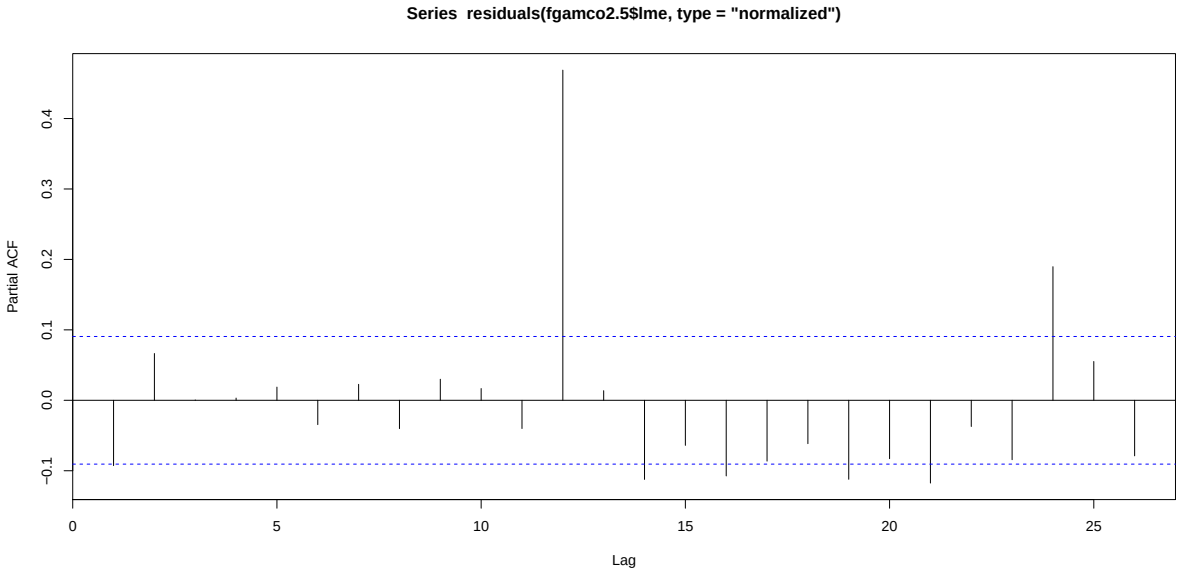


We can also fit a more general, more flexible model structure, e.g. a decay of covariance with time:

```
fgamco2.5 <- gamm(co2conc ~ s(co2times) + s(days, bs="cc"), correlation=corExp(form=~co2times))
fgamco2.5$lme # range is important!

## Linear mixed-effects model fit by maximum likelihood
##   Data: strip.offset(mf)
##   Log-likelihood: -228.9903
##   Fixed: y ~ X - 1
##    X(Intercept) Xs(co2times)Fx1
##        337.0556         13.5952
##
## Random effects:
##  Formula: ~Xr - 1 | g
```

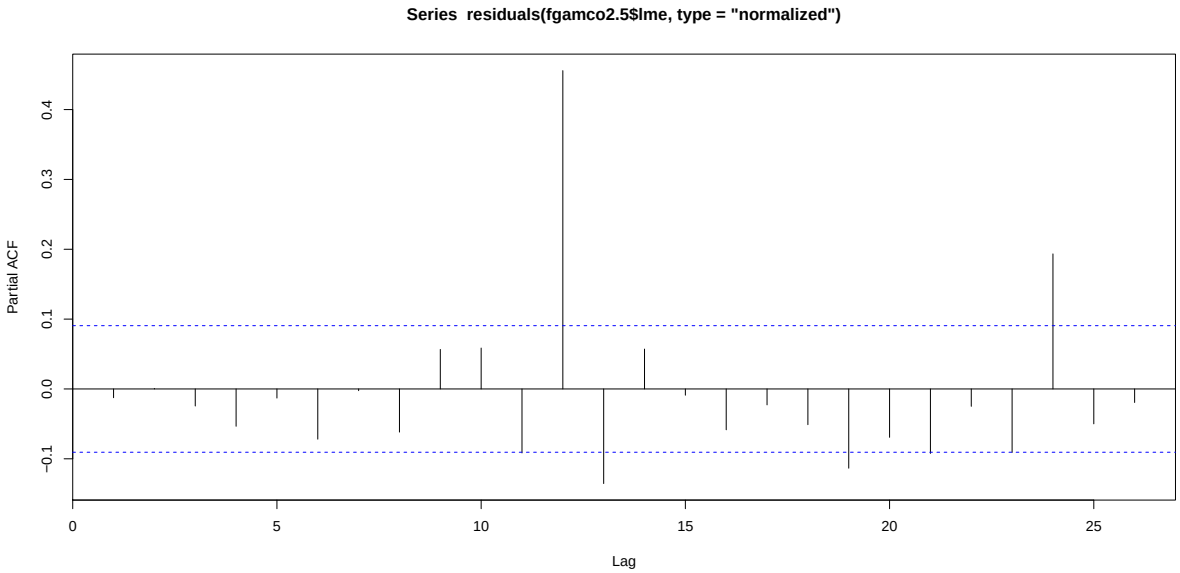
```
##  Structure: pdIdnot
##              Xr1      Xr2      Xr3      Xr4      Xr5      Xr6      Xr7
## StdDev: 8.723869 8.723869 8.723869 8.723869 8.723869 8.723869 8.723869
##              Xr8
## StdDev: 8.723869
##
##  Formula: ~Xr.0 - 1 | g.0 %in% g
##  Structure: pdIdnot
##             Xr.01     Xr.02     Xr.03     Xr.04     Xr.05     Xr.06
## StdDev: 0.3829565 0.3829565 0.3829565 0.3829565 0.3829565 0.3829565
##             Xr.07     Xr.08  Residual
## StdDev: 0.3829565 0.3829565 0.4981571
##
## Correlation Structure: Exponential spatial correlation
##  Formula: ~co2times | g/g.0
##  Parameter estimate(s):
##     range
## 0.2135817
## Number of Observations: 468
## Number of Groups:
##          g g.0 %in% g
##          1          1
```

```
pacf(residuals(fgamco2.5$lme, type="normalized"))
```



Now this is overall very similar to the AR1.

```
fgamco2.5 <- gamm(co2conc ~ s(co2times) + s(days, bs="cc"), correlation=corARMA(value=c(.5, 0.4),
pacf(residuals(fgamco2.5$lme, type="normalized"))
```



7 *Conclusions*

Time series analysis is actually quite a diverse set of statistical acitivities, ranging from cool descriptives to nasty modelling. Make sure you know what you are doing! Feedback, improvements, corrections and additions are welcomed!