Stat 565

Fitting Arma Models

Jan 28 2016

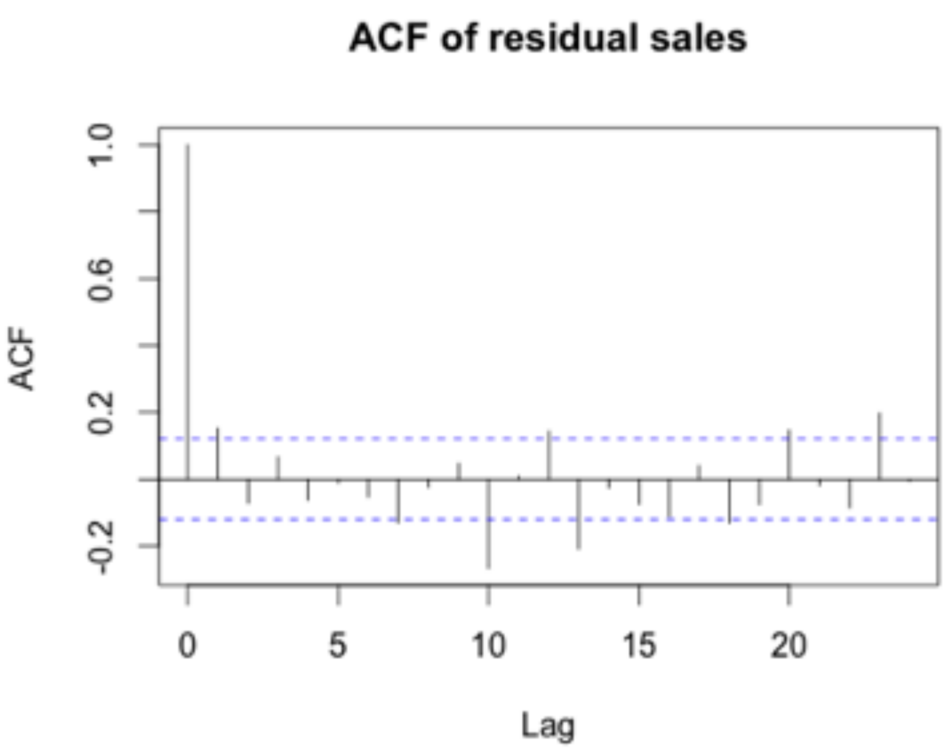
Charlotte Wickham

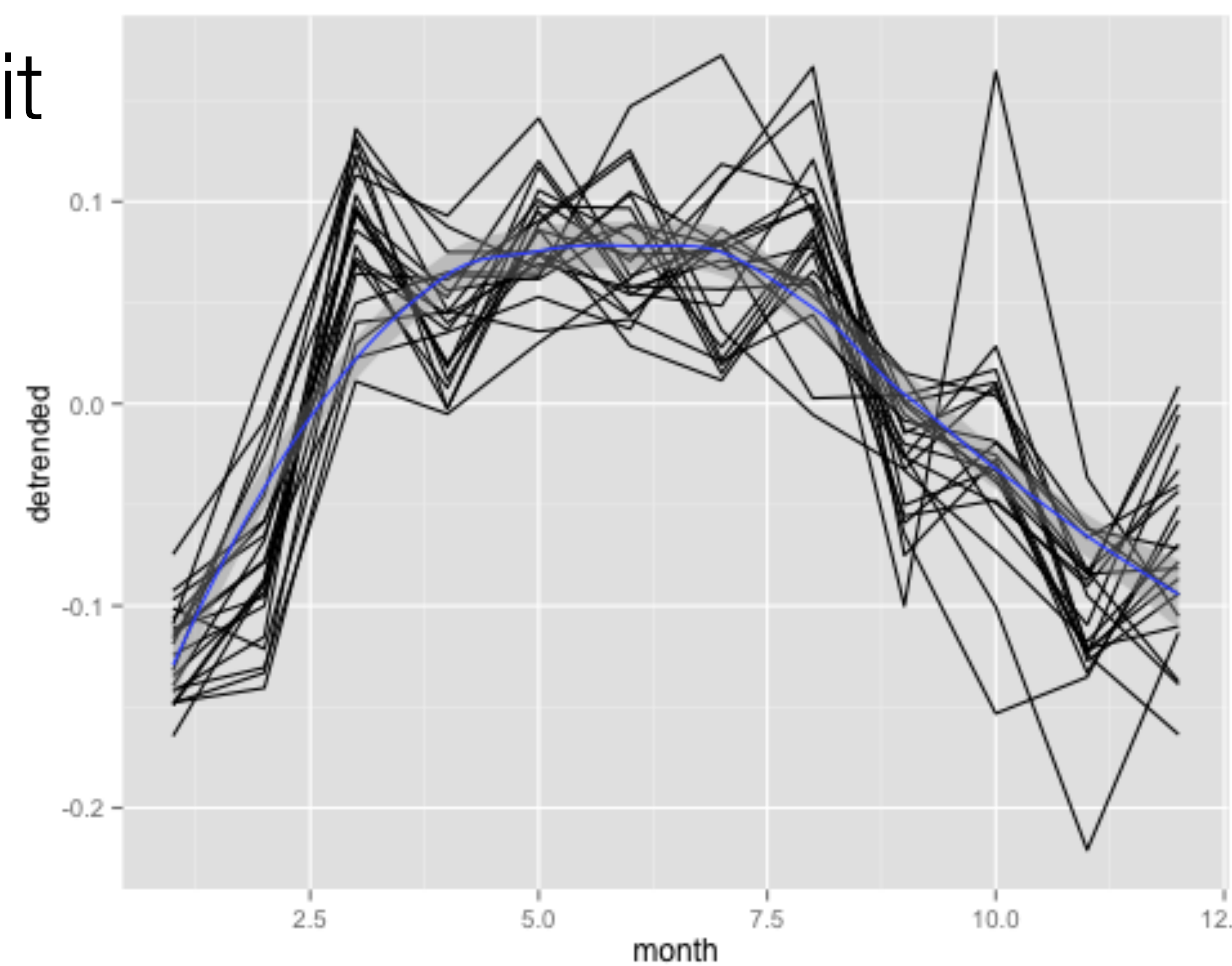
stat565.cwick.co.nz

HW#2

Be careful smoothing things that aren't smooth

Log transform helped a bit





A sudden break at

Procedure for ARMA modeling

We'll assume the primary goal is getting a forecast.

1. Plot the data. Transform? Outliers?

2. Examine acf and pacf and pick p and q. Or try a few. `acf` and `pacf`

3. Fit ARMA(p, 0, q) model to original data. Or fit a few. AIC might be useful. `arima`

4. Check model diagnostics

5. Forecast (back transform?)

Diagnostics

If the correct model is fit, then the residuals (our estimates of the white noise process), should be roughly i.i.d Normal variates.

Check:

Time plot of residuals

Autocorrelation (and partial) plot of residuals

Normal qq plot

Pair data analysis

Each pair gets a time series

Try to identify and estimate the model.

Once you are satisfied, find the other pair(s), with the same series, and see if you agree.

Explain your series to a pair with a different one.

Time series

```
load(url('http://stat565.cwick.co.nz/data/series-1.rda'))
load(url('http://stat565.cwick.co.nz/data/series-2.rda'))
load(url('http://stat565.cwick.co.nz/data/series-3.rda'))
load(url('http://stat565.cwick.co.nz/data/series-4.rda'))
load(url('http://stat565.cwick.co.nz/data/series-5.rda'))
load(url('http://stat565.cwick.co.nz/data/series-6.rda'))
load(url('http://stat565.cwick.co.nz/data/series-7.rda'))
load(url('http://stat565.cwick.co.nz/data/series-8.rda'))
```

Rough code outline

1. Plot, already done:

```
qplot(1:length(x), x, geom = "line")
```

2. ACF & PACF

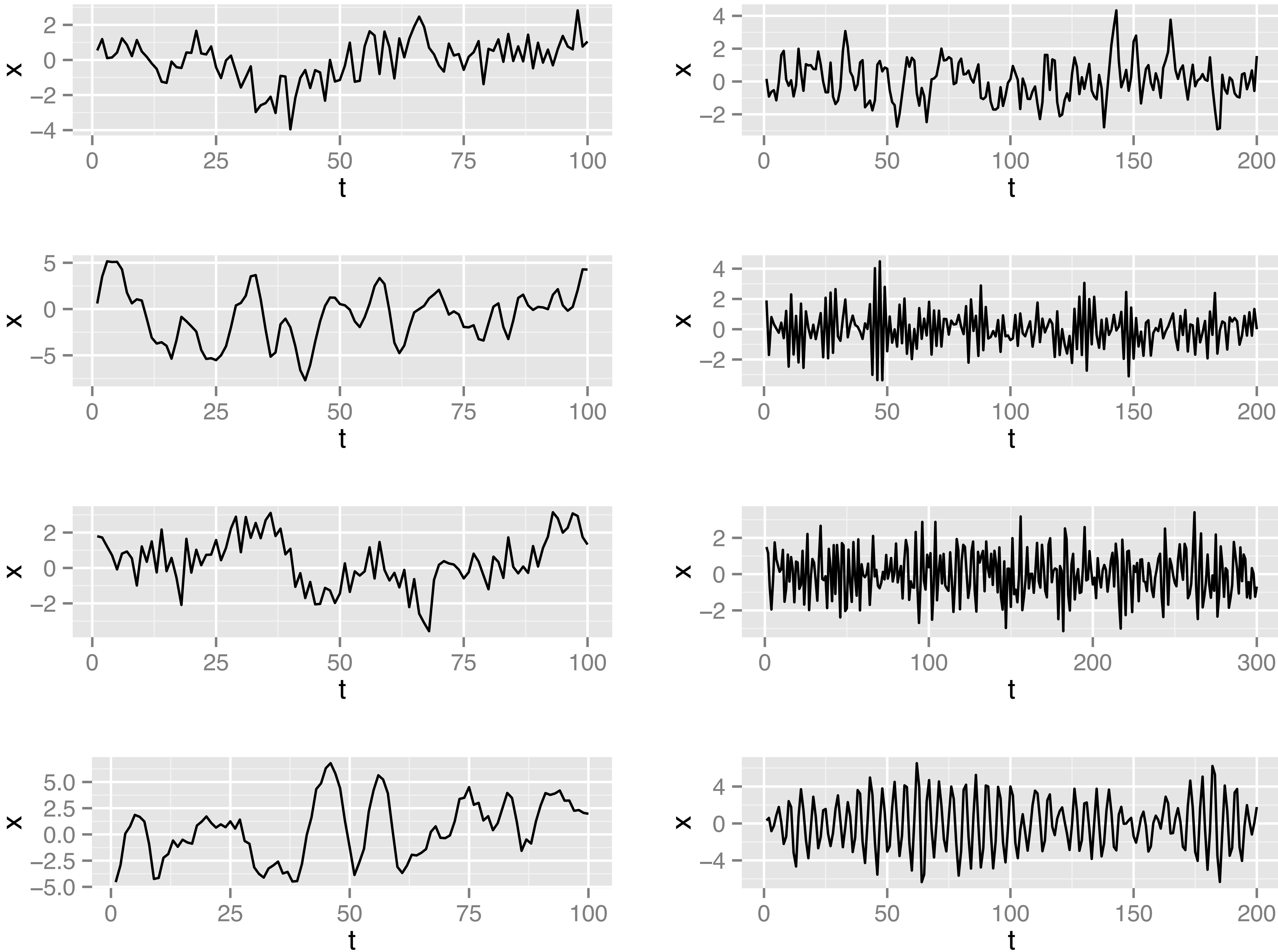
```
acf(x)
pacf(x)
```

3. Fit tentative model

```
fit <- arima(x, c(p, 0, q))
```

4. Diagnostics

```
res <- residuals(fit)
acf(res)
pacf(res)
qplot(sample = res)
```


x
t
2
0
−2
−4
0
25
50
75
100
4
2
0
−2
0
50
100
150
200
5
0
−5
0
25
50
75
100
4
2
0
−2
0
50
100
150
200
2
0
−2
0
25
50
75
100
2
0
−2
0
100
200
300
5.0
2.5
0.0
−2.5
−5.0
0
25
50
75
100
4
0
−4
0
50
100
150
200

Stationarity by differencing

HW#2.2 you saw once differencing a process with a linear trend results in a stationary process.

In general "d" times differencing a process with a polynomial trend of order d results in a stationary process.

Suggests the approach:

1. Difference the data until it is stationary
2. Fit a stationary ARMA model to capture the correlation
3. Work backwards (by summing) to get original series

The alternative is to explicitly model the trend and seasonality...more next week

HW #2 example

$$x_t = \beta_0 + \beta_1 t + w_t$$

a linear trend

$$\nabla x_t = x_t - x_{t-1} = \beta_1 + w_t - w_{t-1}$$

an MA(1) process with

constant mean β_1

x_t is called ARIMA(0, 1, 1)

ARIMA(p, d, q)

Autoregressive Integrated Moving Average

A process x_t is ARIMA(p, d, q) if x_t differenced d times ($\nabla^d x_t$) is an ARMA(p, q) process.

I.e. x_t is defined by

$$\phi(B)\, \nabla^d x_t = \theta(B)\, w_t$$

$$\phi(B)\, (1 - B)^d x_t = \theta(B)\, w_t$$

forces constant in 1st differenced series

```
arima(x, order = c(p, 1, q), xreg = 1:length(x))
```

Procedure for ARIMA modeling

We'll assume the primary goal is getting a forecast.

`diff`

1. Plot the data. Transform? Outliers? Differencing?
2. Difference until series is stationary, i.e. find d.
3. Examine differenced series and pick p and q.
4. Fit ARIMA(p, d, q) model to original data.
5. Check model diagnostics
6. Forecast (back transform?)

Pick one:

Oil prices

```
install.packages('TSA')
data(oil.price, package = 'TSA')
```

Global temperature

```
load(url("http://www.stat.pitt.edu/stoffer/tsa3/tsa3.rda"))
gtemp
```

US GNP

```
load(url("http://www.stat.pitt.edu/stoffer/tsa3/tsa3.rda"))
gnp
```

Sulphur Dioxide (LA county)

```
load(url("http://www.stat.pitt.edu/stoffer/tsa3/tsa3.rda"))
so2
```

My code to turn a ts into a data.frame

```
library(ggplot2)

source(url("http://stat565.cwick.co.nz/code/fortify-ts.r"))

fortify(gtemp)
```

SARIMA models

I haven't shown you any data with seasonality.

The idea is very similar, if one seasonal cycle lasts for s measurements, then if we difference at lag s,

$y_t = \nabla_s x_t = x_t - x_{t-s} = (1 - B^s)x_t,$

we will remove the seasonality.

Differencing seasonally D times is denoted,

$\nabla^D{}_s x_t = (1 - B^s)^D x_t,$

SARIMA

A multiplicative seasonal autoregressive integrated moving average model,

$\text{SARIMA}(p, d, q) \times (P, D, Q)_s$

is given by

$$\Phi(B^s)\phi(B)\,\nabla^D_s\nabla^d x_t = \Theta(B^s)\theta(B)w_t$$

$\nabla^D_s\nabla^d x_t$ is just an ARMA model with lots of coefficients set to zero.

Have to specify s, then choose p, d, q, P, D and Q

Procedure for **S**ARIMA modeling

We'll assume the primary goal is getting a forecast.

1. Plot the data. Transform? Outliers? Differencing?

2. Difference to remove trend, find d. Then difference to remove seasonality, find D.

3. Examine acf and pacf of differenced series. Find P and Q first, by examining just at lags s, 2s, 3s, etc. Find p and q by examining between seasonal lags.

4. Fit SARIMA$(p, d, q)x(P, D, Q)_s$ model to original data.

5. Check model diagnostics

6. Forecast (back transform?)

library(forecast)

Has some "automatic" ways to select arima models (and seasonal ARIMA models).

```
auto.arima(log(oil$price))
```

Finds the model (up to a certain order) with the lowest AIC. You should still check if a simpler model has almost the same AIC.