BUS41100 Applied Regression Analysis

Week 8: An Introduction to Time Series

Dependent data, autocorrelation,
AR and periodic regression models

Max H. Farrell
The University of Chicago Booth School of Business

Time series data and dependence

Time-series data are simply a collection of observations gathered over time. For example, suppose $y_1, \ldots, y_T$ are

- annual GDP,
- quarterly production levels,
- weekly sales,
- daily temperature,
- 5-minutely stock returns.

In each case, we might expect what happens at time t to be correlated with time $t-1$.

Suppose we measure temperatures, daily, for several years.

Which would work better as an estimate for today's temp:

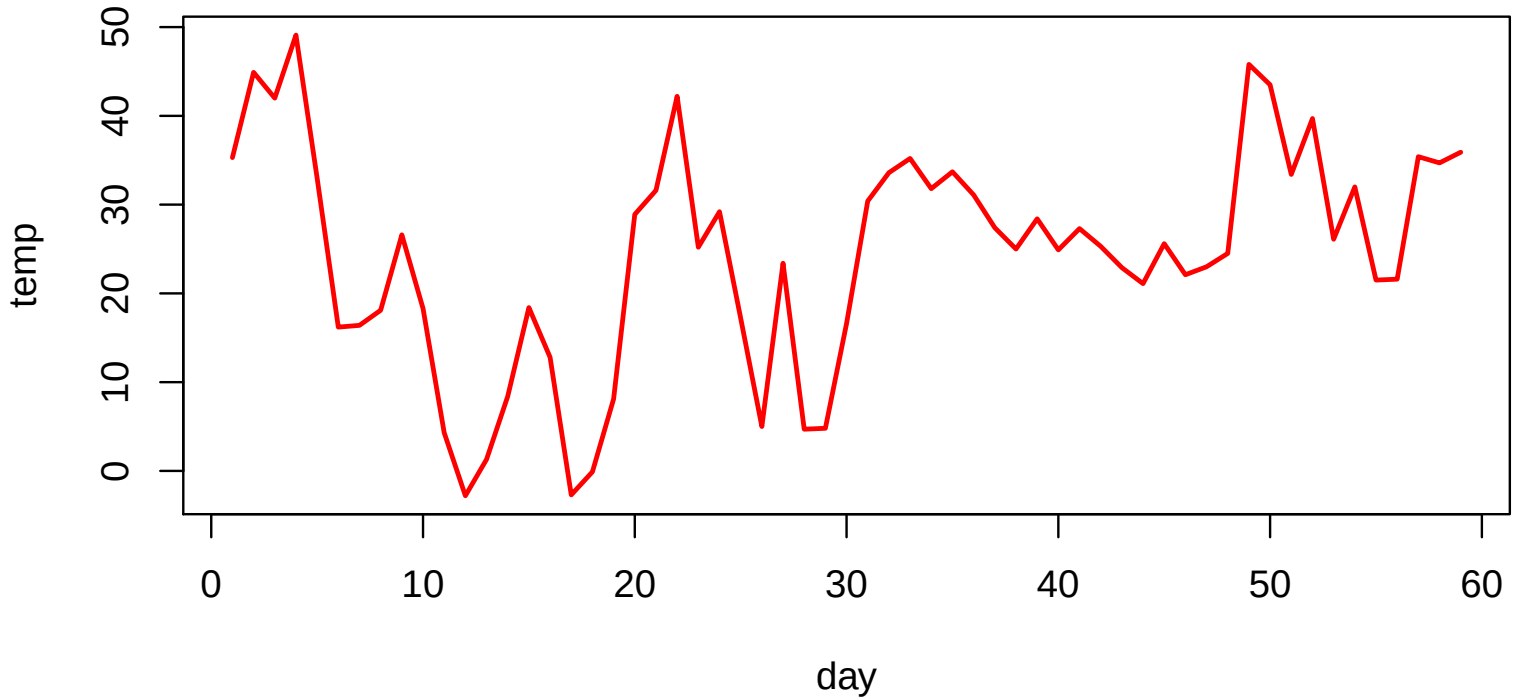
- The average of the temperatures from the previous year?
- The temperature on the previous day?

How would this change if the readings were iid $\mathcal{N}(\mu, \sigma^2)$?

Correlated errors require fundamentally different techniques.

Example: Y_t = average daily temp. at O'Hare, Jan-Feb 1997.

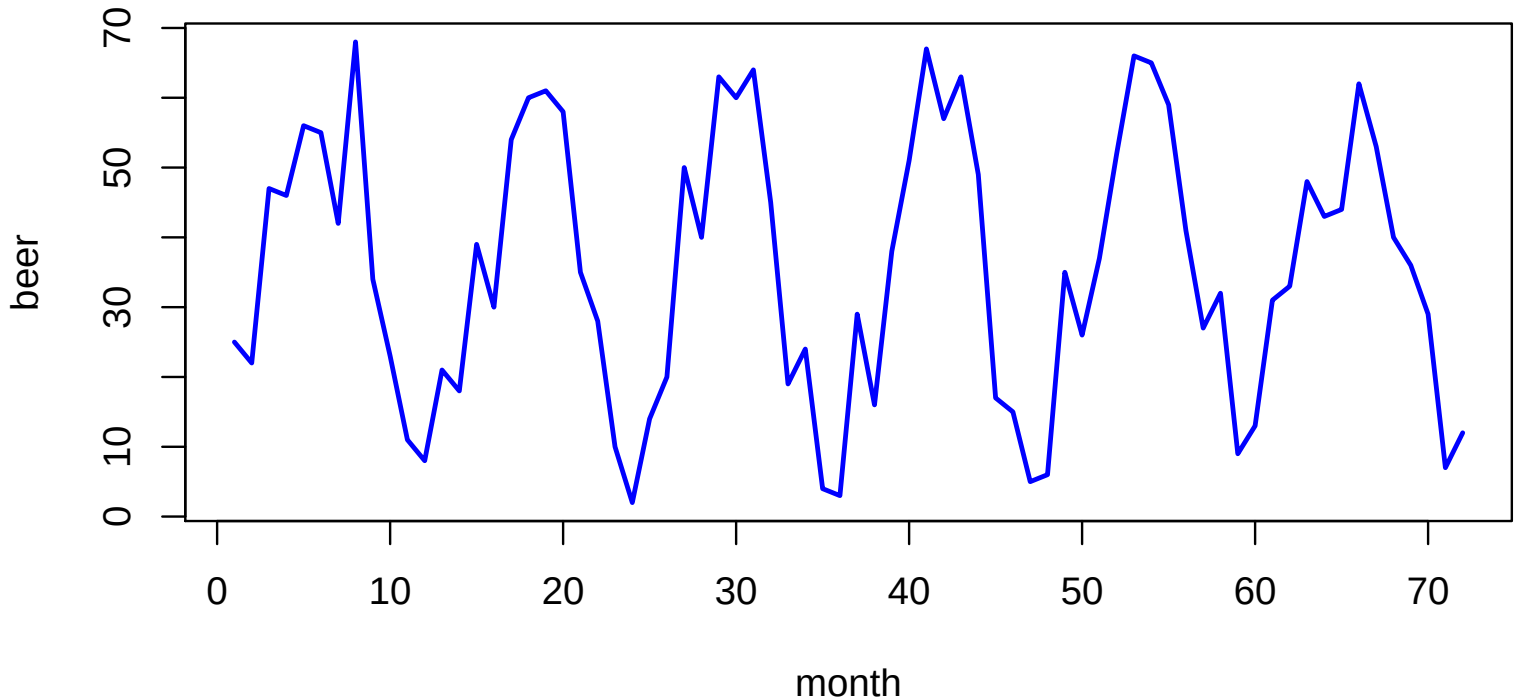
```
> weather <- read.csv("weather.csv")
> plot(weather$temp, xlab="day", ylab="temp", type="l",
+   col=2, lwd=2)
```



- "sticky" sequence: today tends to be close to yesterday.

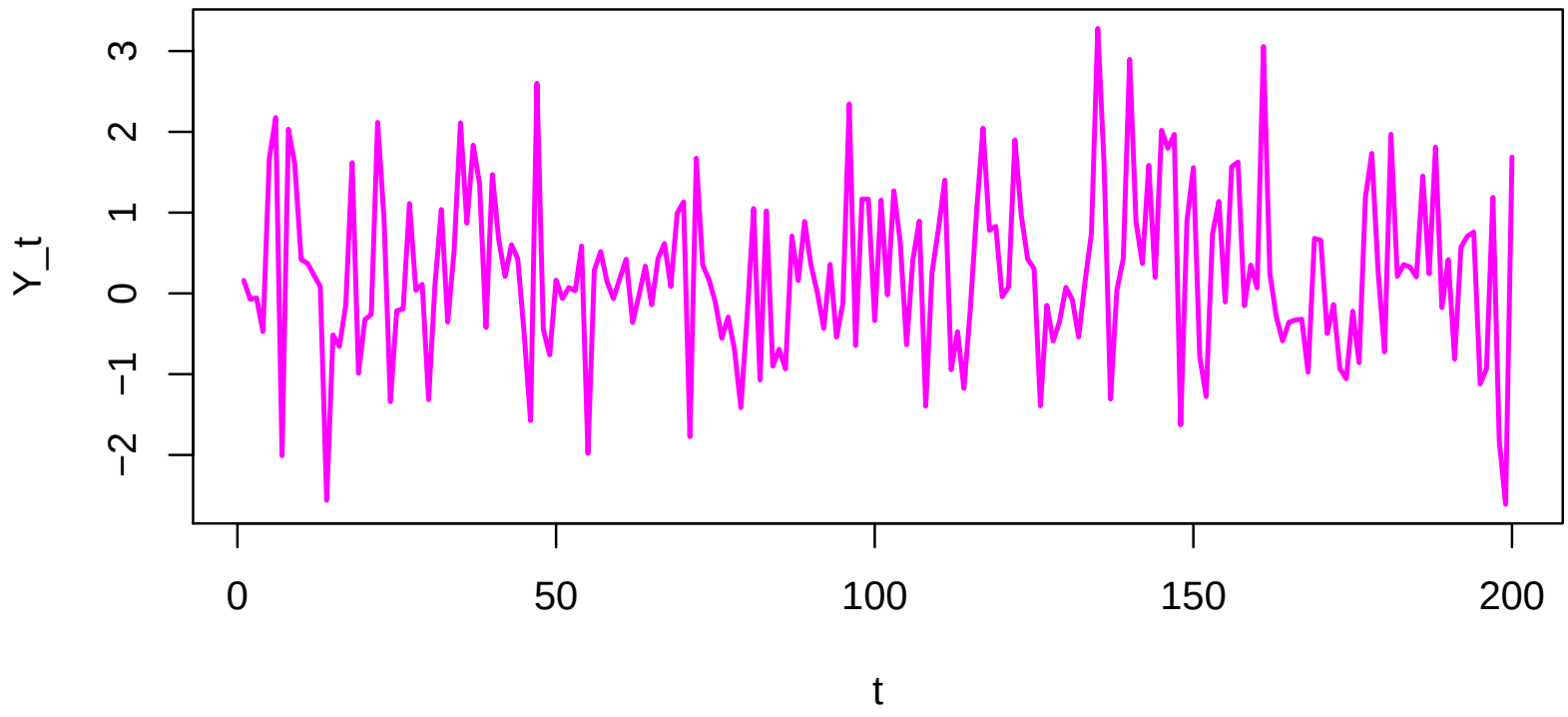
Example: $Y_t =$ monthly U.S. beer production (Mi/barrels).

```
> beer <- read.csv("beer.csv")
> plot(beer$prod, xlab="month", ylab="beer", type="l",
+   col=4, lwd=2)
```



- The same pattern repeats itself year after year.

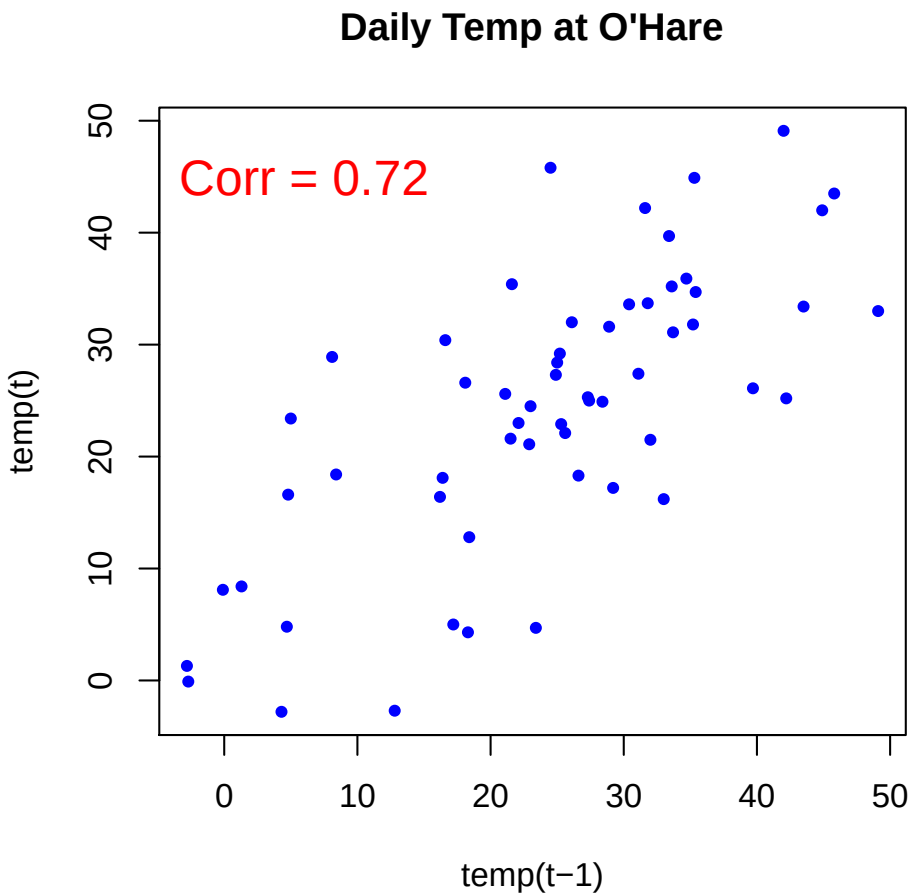
```
> plot(rnorm(200), xlab="t", ylab="Y_t", type="l",
+   col=6, lwd=2)
```



- It is tempting to see patterns even where they don't exist.

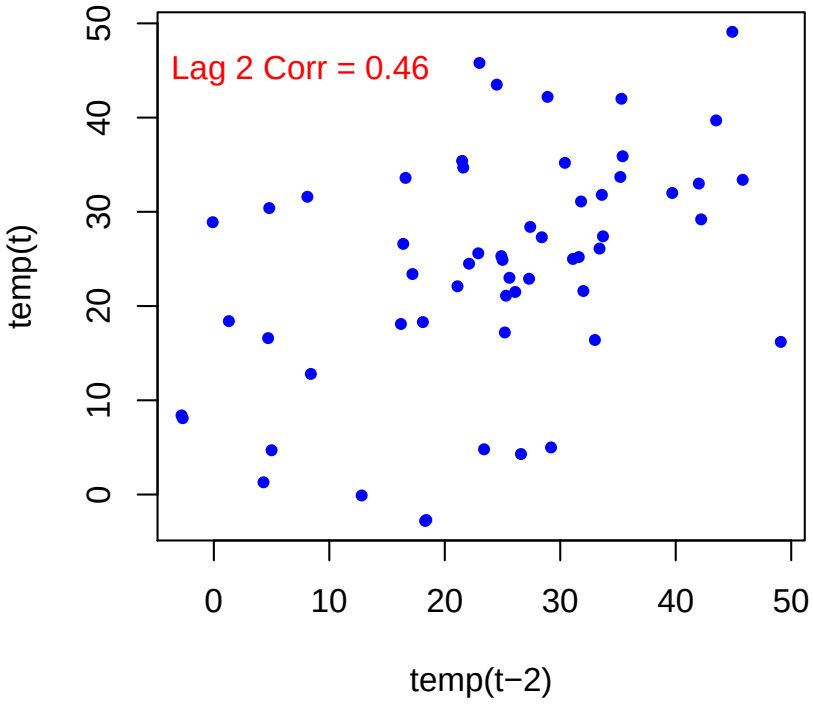
Checking for dependence

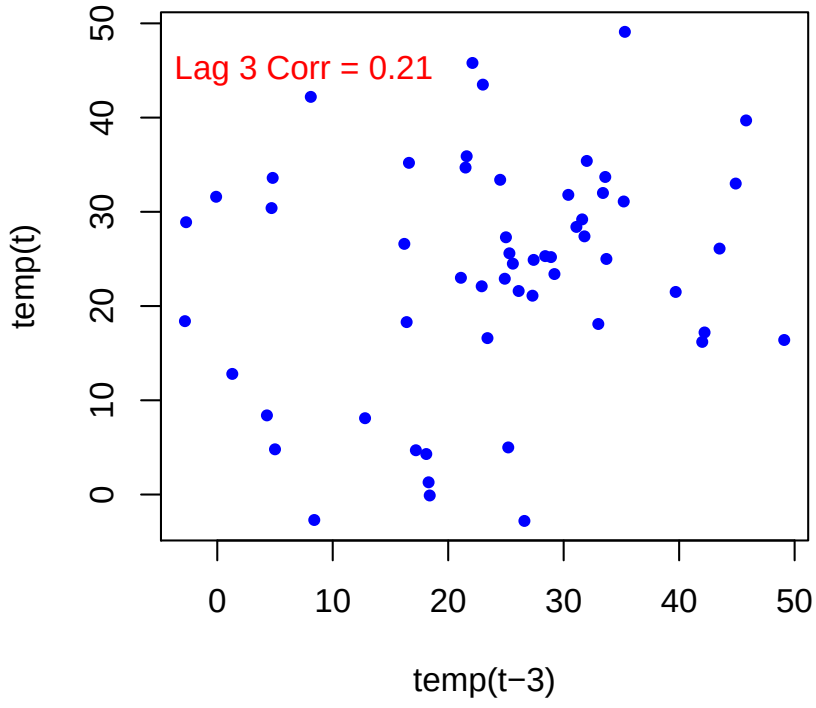
To see if Y_{t-1} would be useful for predicting Y_t, just plot them together and see if there is a relationship.



- Correlation between Y_t and Y_{t-1} is called autocorrelation.

We can plot Y_t against $Y_{t-\ell}$ to see ℓ-period lagged relationships.





- It appears that the correlation is getting weaker with increasing ℓ.

Autocorrelation

To summarize the time-varying dependence, compute lag-ℓ correlations for $\ell = 1, 2, 3, \ldots$

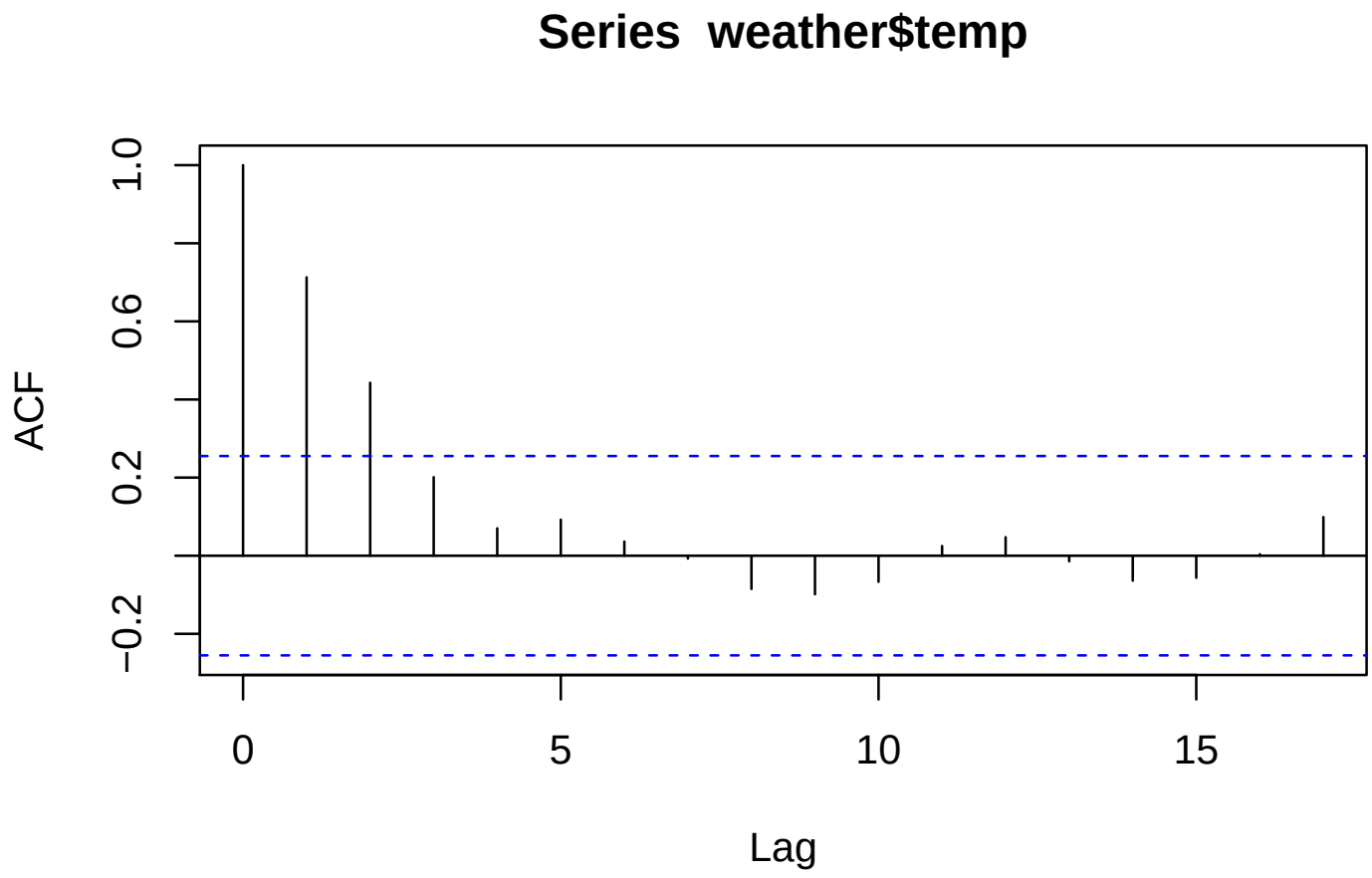
In general, the autocorrelation function (ACF) for Y is

$$r(\ell) = \text{cor}(Y_t, Y_{t-\ell})$$

For our O'Hare temperature data:

```
> print(acf(weather$temp))
    0     1     2     3     4     5     6     7     8
 1.00  0.71  0.44  0.20  0.07  0.09  0.04 -0.01 -0.09
    9    10    11    12    13    14    15    16    17
-0.10 -0.07  0.03  0.05 -0.01 -0.06 -0.06  0.00  0.10
```

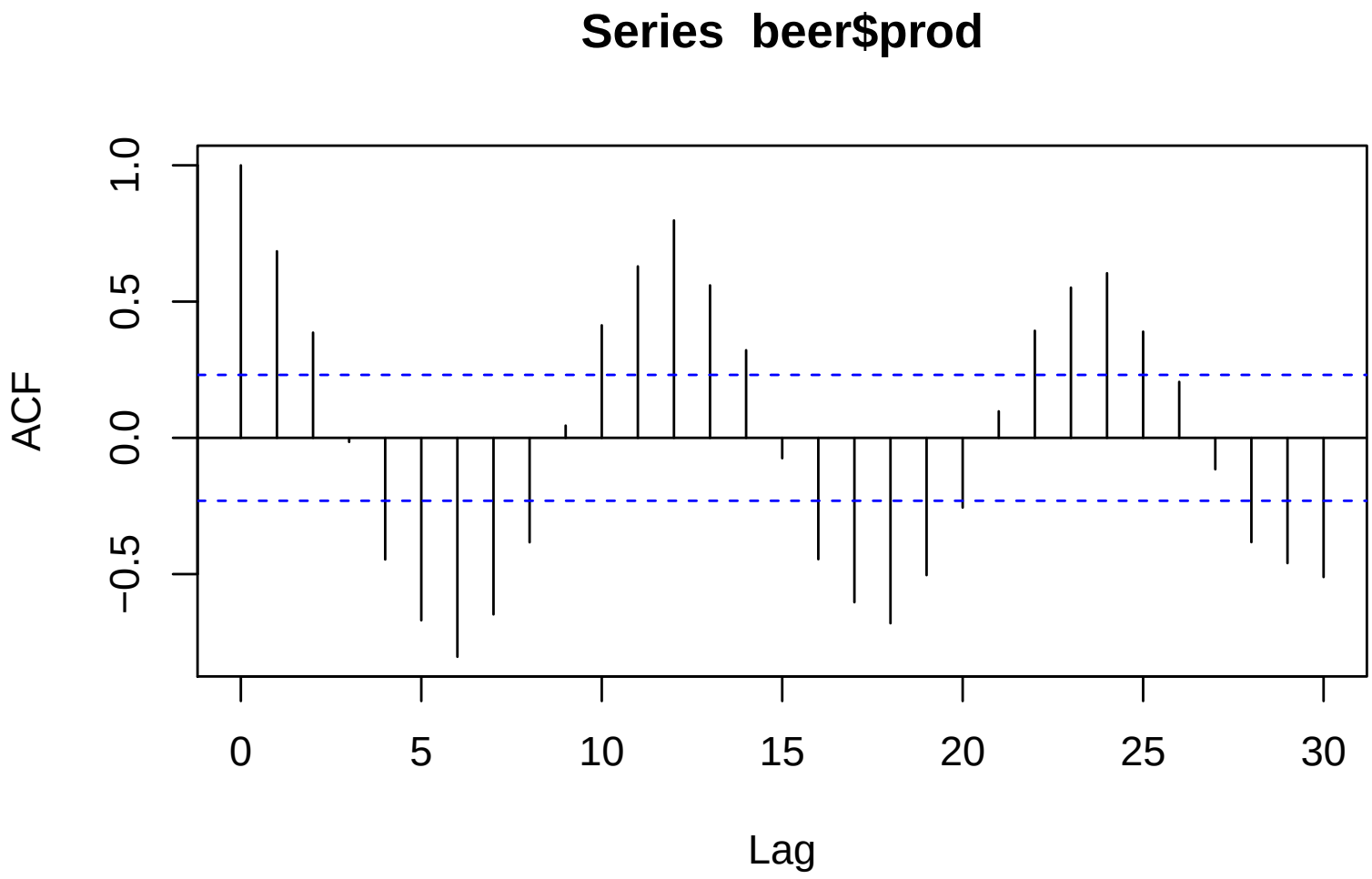
R's `acf` function shows the ACF visually.



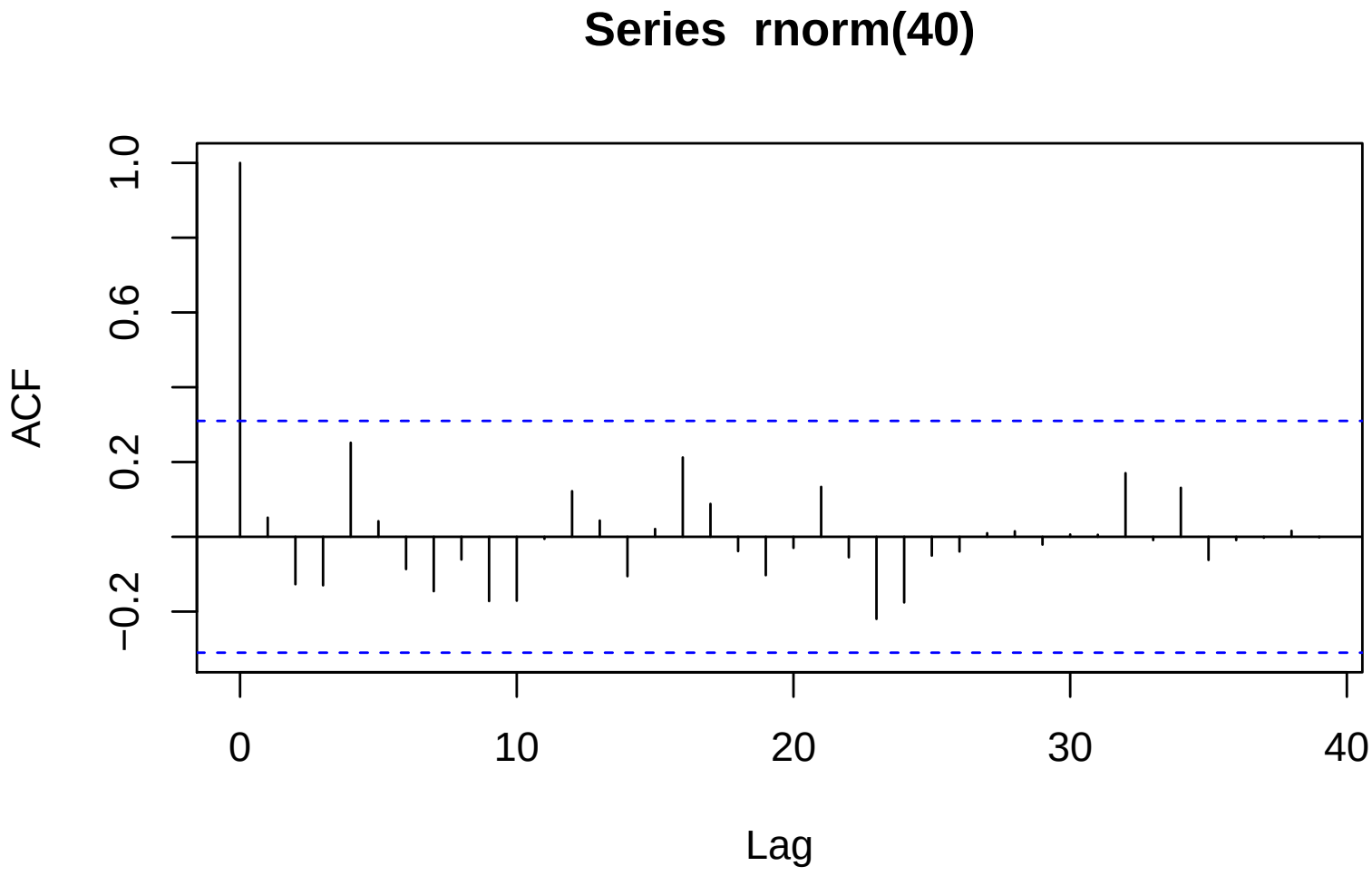
It provides a visual summary of our data dependence.

(Blue lines mark "statistical significance" for the `acf` values.)

The beer data shows an alternating dependence structure which causes time series oscillations.



An acf plot for iid normal data shows no significant correlation.



. . . but what about next time?

Autoregression

The autoregressive model of order one holds that

$$AR(1): Y_t = \beta_0 + \beta_1 Y_{t-1} + \varepsilon_t, \quad \varepsilon_t \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2).$$

This is just a SLR model of Y_t regressed onto lagged Y_{t-1}.

- Y_t depends on errors going **all the way back** to the beginning, but the whole past is captured by **only** Y_{t-1}

It assumes all of our standard regression model conditions.

- The residuals should look iid and be uncorrelated with $\hat{Y}_t$.
- All of our previous diagnostics and transforms still apply.

$$AR(1): Y_t = \beta_0 + \beta_1 Y_{t-1} + \varepsilon_t$$

Again, Y_t depends on the past only through Y_{t-1}.

- Previous lag values ($Y_{t-2}, Y_{t-3}, \ldots$) do not help predict Y_t if you already know Y_{t-1}.

Think about daily temperatures:

- If I want to guess tomorrow's temperature (without the help of a meterologist!), it is sensible to base my prediction on today's temperature, ignoring yesterday's.

Other examples: Consumption, stock prices, ...

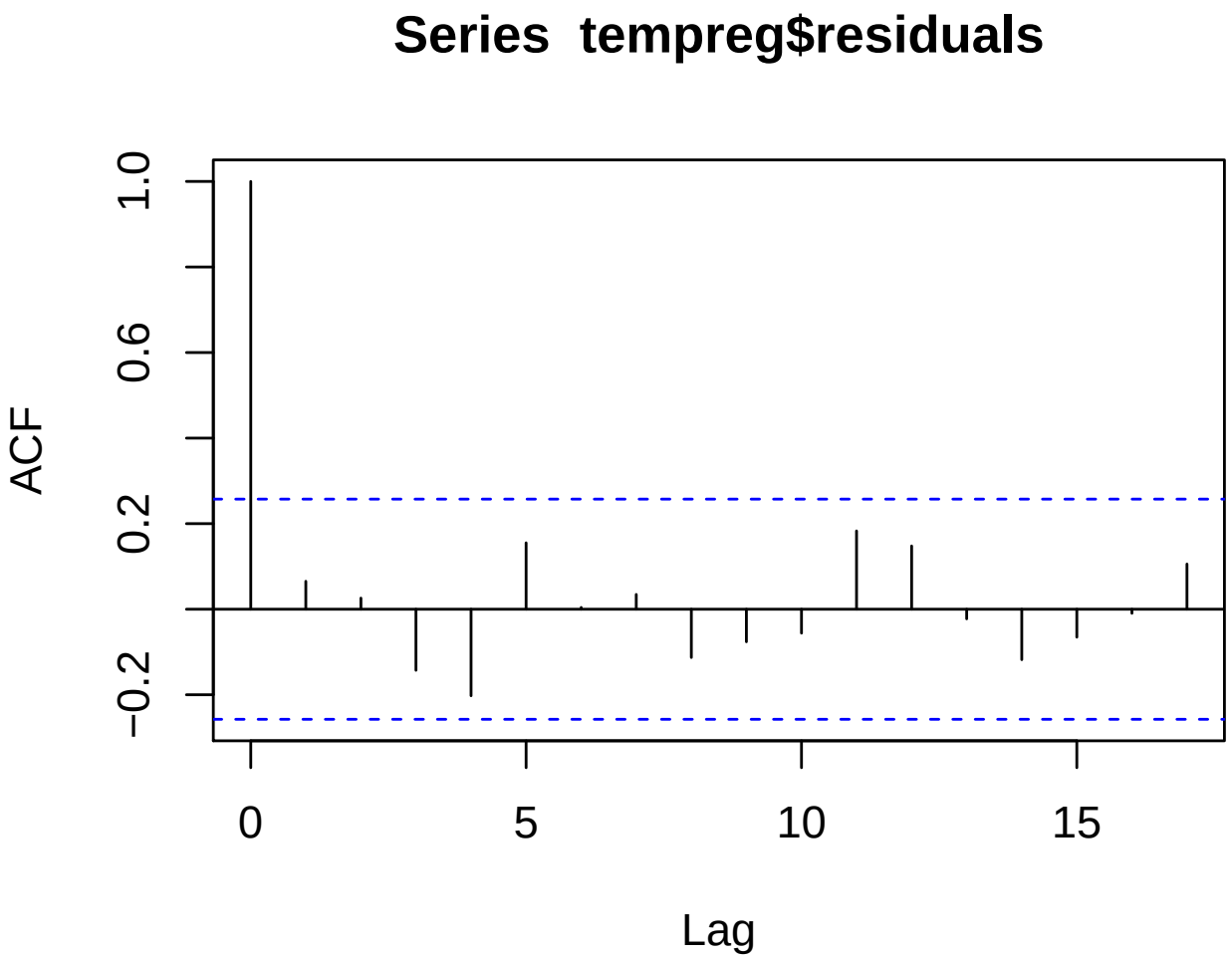
For the O'Hare temperatures, there is a clear autocorrelation.

```
> tempreg <- lm(weather$temp[2:59] ~ weather$temp[1:58])
> summary(tempreg)  ## abbreviated output

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)         6.70580    2.51661   2.665   0.0101 *
weather$temp[1:58]  0.72329    0.09242   7.826  1.5e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 8.79 on 56 degrees of freedom
Multiple R-squared:  0.5224,  Adjusted R-squared:  0.5138
F-statistic: 61.24 on 1 and 56 DF,  p-value: 1.497e-10
```

- The autoregressive term ($b_1 \approx 0.7$) is highly significant!

We can check residuals for any "left-over" correlation.

```
> acf(tempreg$residuals)
```



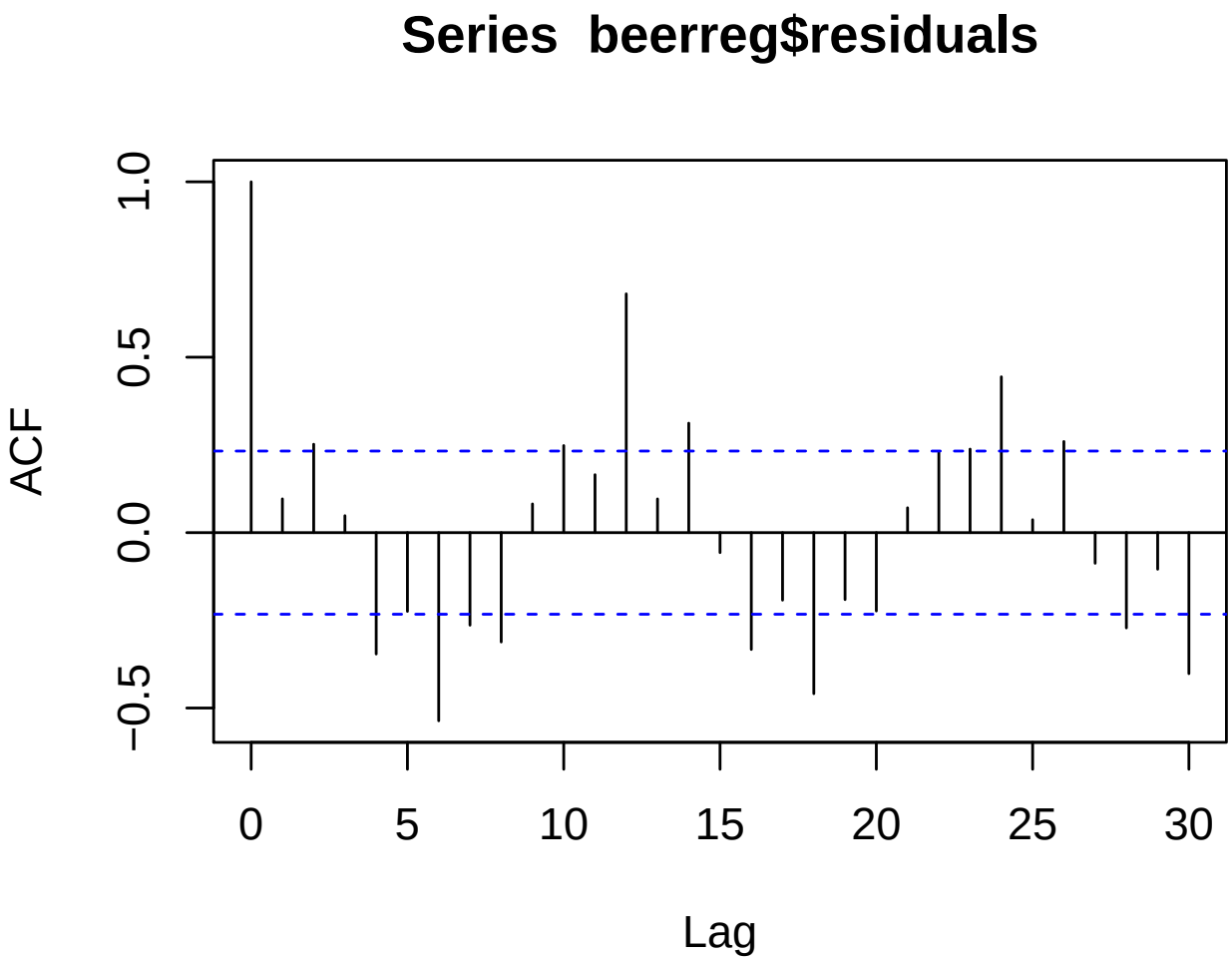
- Looks like we've got a good fit.

For the beer data, the autoregressive term is also highly significant.

```
> beerreg <- lm(beer$prod[2:72] ~  beer$prod[1:71])
> summary(beerreg) ## abbreviated output

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     10.64818    3.56983   2.983  0.00395 **
beer$prod[1:71]  0.69960    0.08748   7.997 2.02e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 14.08 on 69 degrees of freedom
Multiple R-squared:  0.481, Adjusted R-squared:  0.4735
F-statistic: 63.95 on 1 and 69 DF,  p-value: 2.025e-11
```

But residuals show a clear pattern of left-over autocorrelation.

```
> acf(beerreg$residuals)
```



- We'll talk later about how to model this type of pattern ...

Many different types of series may be written as an AR(1).

$$AR(1) : Y_t = \beta_0 + \beta_1 Y_{t-1} + \varepsilon_t$$

The value of β_1 is key!

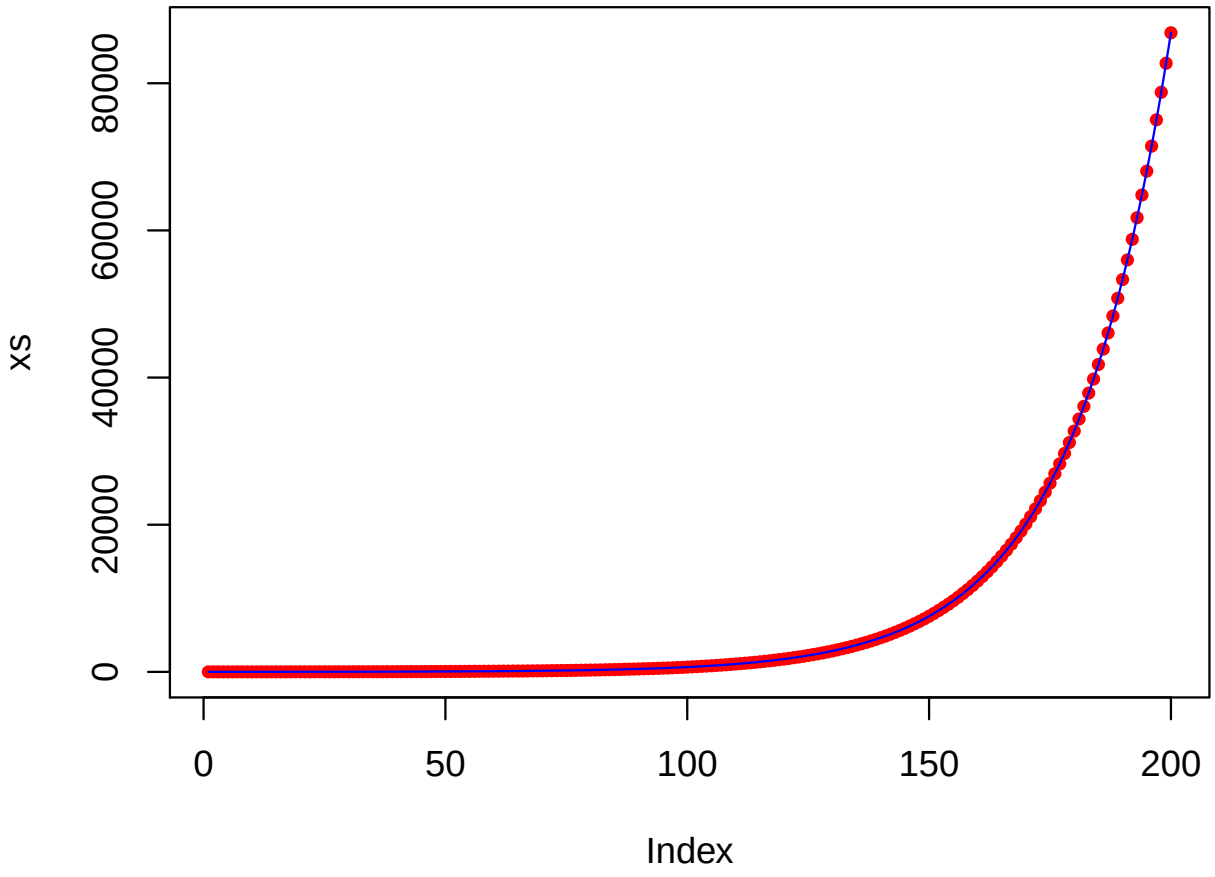
- If $|\beta_1| > 1$, the series *explodes*.
- If $|\beta_1| = 1$, we have a *random walk*.
- If $|\beta_1| < 1$, the values are *mean reverting*.

Not only does the behavior of the series depend on β_1, but so does the sampling distribution of b_1!

Exploding series

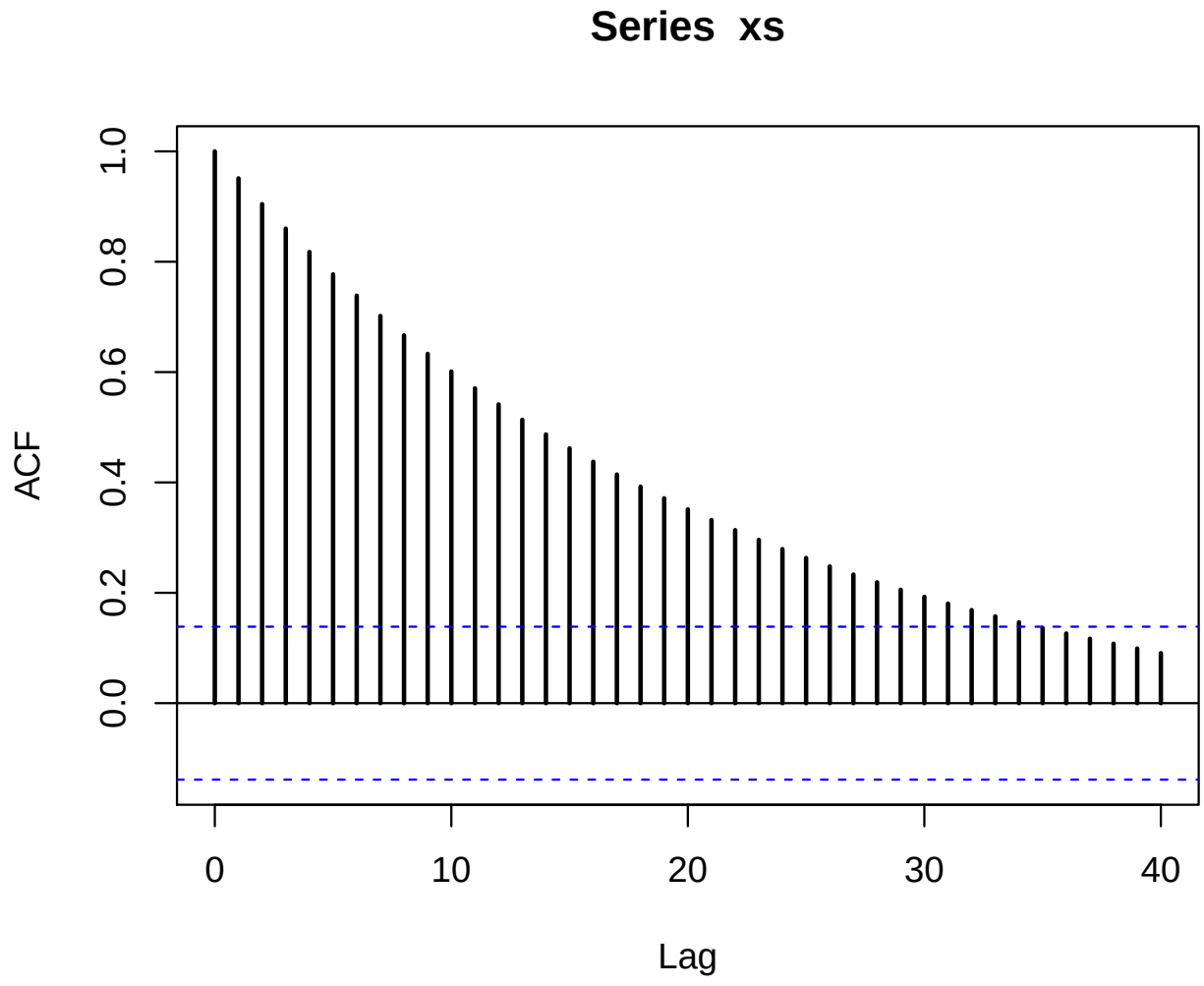
For AR term > 1, the Y_t's move exponentially far from Y_1.

$$\beta_1 = 1.05$$



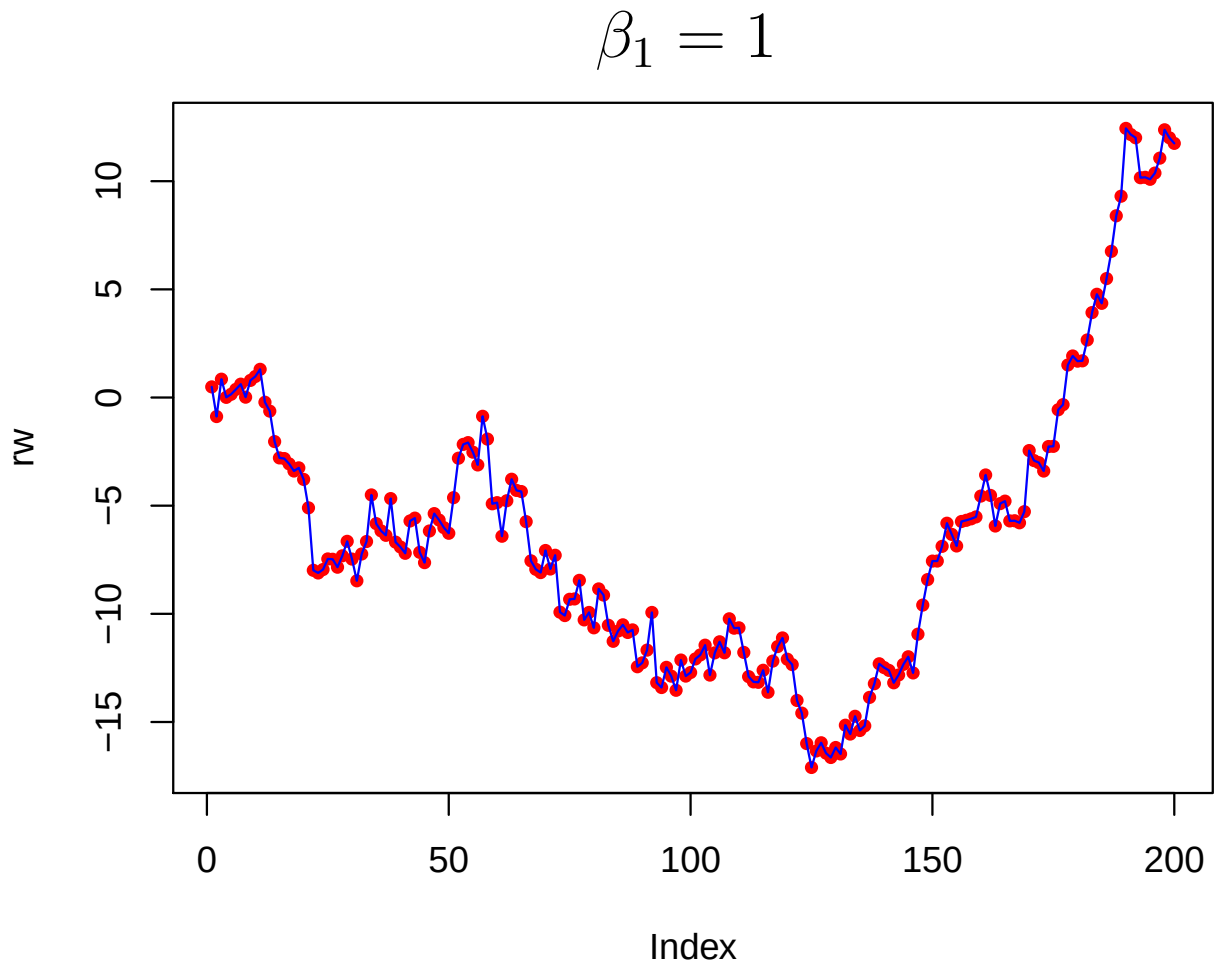
- What does prediction mean here?

Autocorrelation of an exploding series is high for a long time.

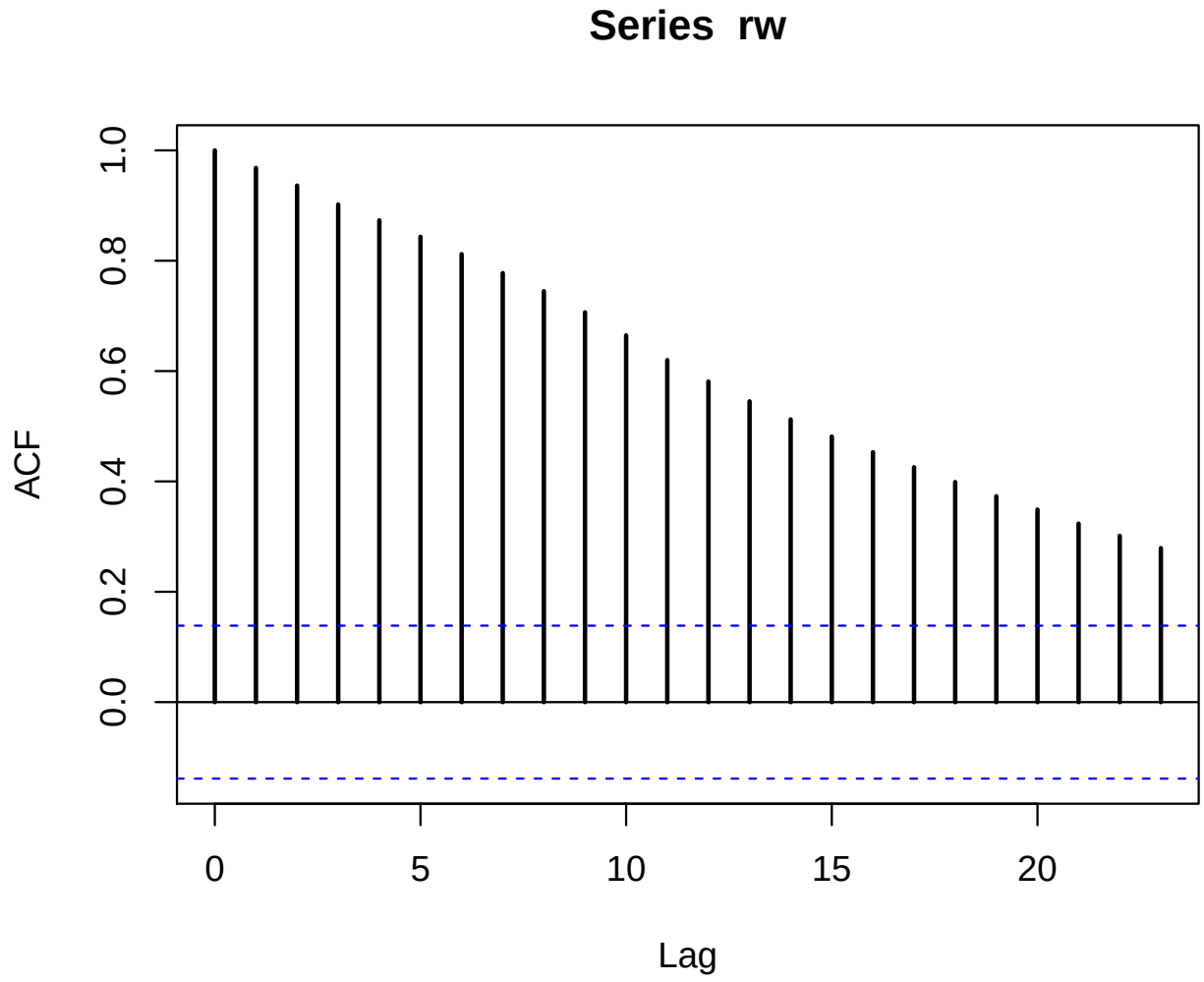


Random walk

In a random walk, the series just wanders around.



Autocorrelation of a random walk is high for a long time.



The random walk has some special properties ...

$Y_t - Y_{t-1} = \beta_0 + \varepsilon_t$, and β_0 is called the "drift parameter".

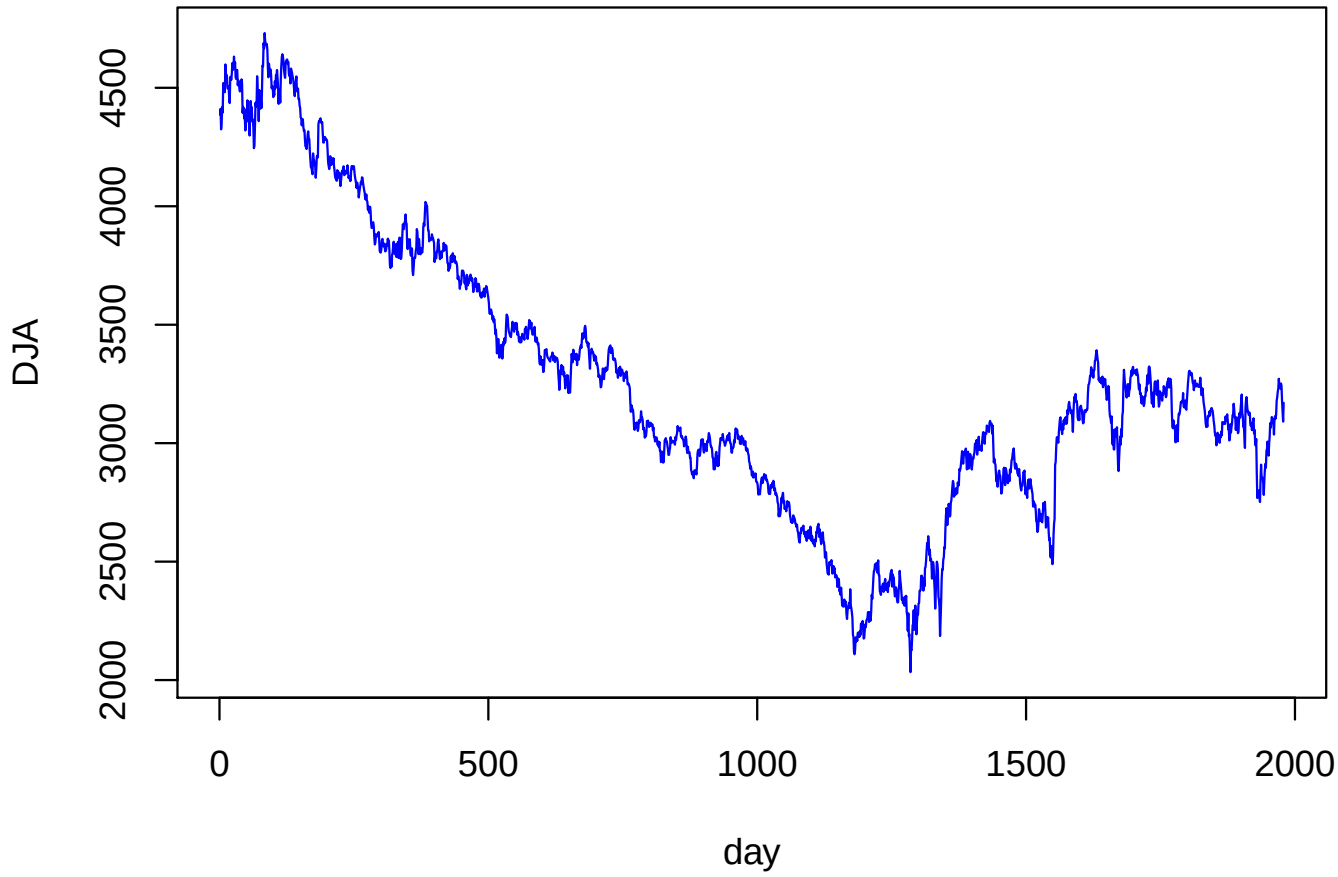
The series is nonstationary:

- it has no average level that it wants to be near, but rather just wanders off into space.

The random walk without drift ($\beta_0 = 0$) is a common model for simple processes

- $Y_1 = \varepsilon_1$, $Y_2 = \varepsilon_1 + \varepsilon_2$, $Y_3 = \varepsilon_1 + \varepsilon_2 + \varepsilon_3$, etc.
- the expectation of what will happen is always what happened most recently. $\mathbb{E}[Y_t] = Y_{t-1}$

Example: monthly Dow Jones composite index, 2000–2007.



- Appears as though it is just wandering around.

Sure enough, our regression indicates a random walk ($b_1 \approx 1$):

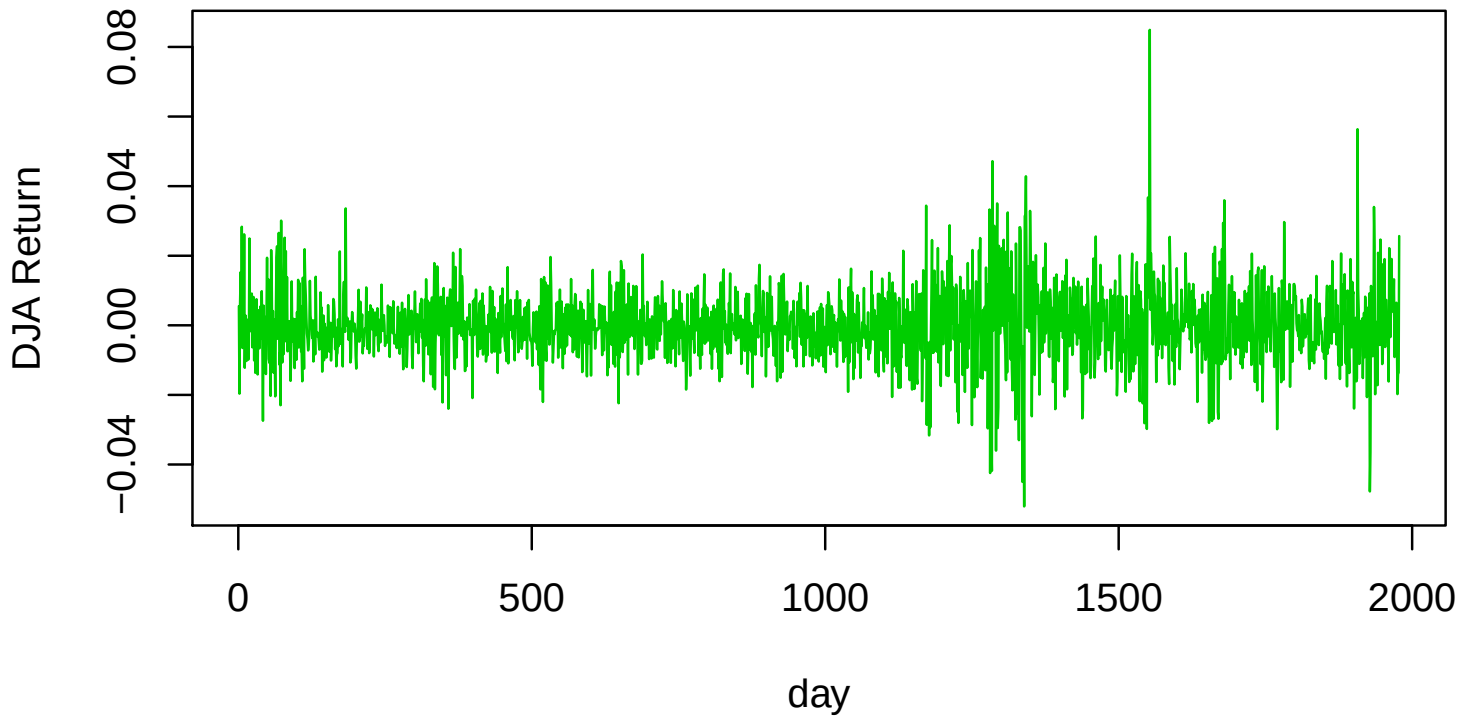
```
> n <- length(dja)
> ARdj <- lm(dja[2:n] ~ dja[1:(n-1)])
> summary(ARdj)  ## abbreviated output

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    7.05419    4.00385   1.762   0.0782 .
dja[1:(n - 1)] 0.99764    0.00121 824.298   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- $b_0 > 0$ and $b_1 \approx 1$, but that's all we can learn
- Sampling distributions change with β_1!!

  $\hookrightarrow$ see `week8-ARMonteCarlo.R`.

When you switch to returns, however, it's just white noise.

```
> returns <- (dja[2:n]-dja[1:(n-1)])/dja[1:(n-1)]
> plot(returns, type="l", col=3, xlab="day", ylab="DJA Return")
```



- $(Y_t - Y_{t-1})/Y_{t-1}$ appears to remove the dependence.

And now the regression model finds nothing significant.

```
> ret <- lm(returns[2:n] ~ returns[1:(n-1)])
> summary(ret) ## abbreviated output

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)       -0.0001138  0.0002363  -0.482    0.630
returns[1:(n - 1)] -0.0144411  0.0225321  -0.641    0.522

Residual standard error: 0.01051 on 1975 degrees of freedom
  (1 observation deleted due to missingness)
Multiple R-squared: 0.000208, Adjusted R-squared: -0.000298
F-statistic: 0.4108 on 1 and 1975 DF,  p-value: 0.5217
```

This is common with random walks: $Y_t - Y_{t-1}$ is iid.

Unit Root Testing

(Augmented) Dickey Fuller Test

$$H_0 : \beta_1 = 1$$

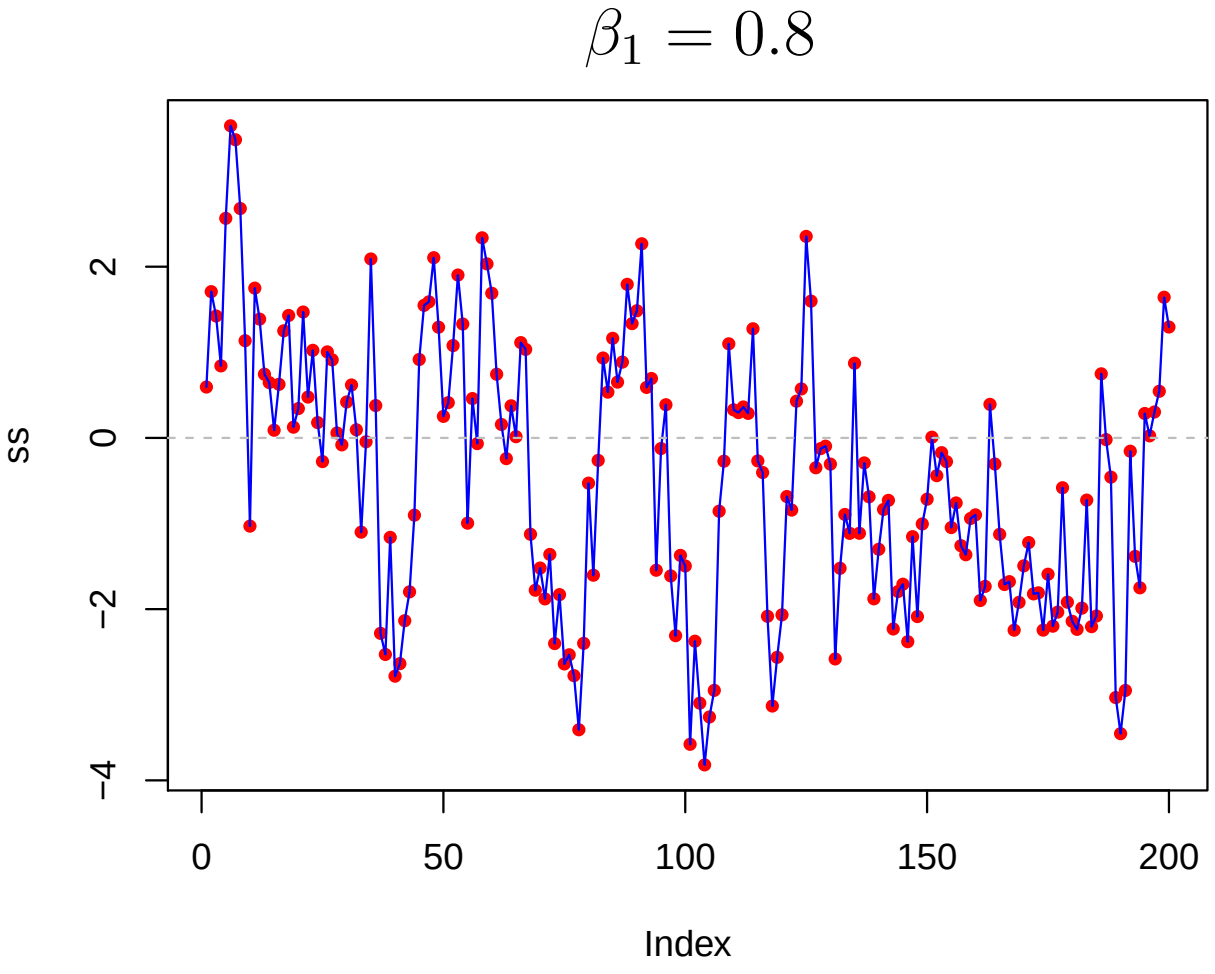
```
> library("urca")  ## lots of other packages
> summary(ur.df(dja))    ## output abbreviated

Value of test-statistic is: -1.2286

Critical values for test statistics:
      1pct  5pct 10pct
tau1 -2.58 -1.95 -1.62
```

Augment with a drift, trend, more lags, ...

Unit root testing is a huge field in financial/macro econometrics, including recent research. This is only one test of many.

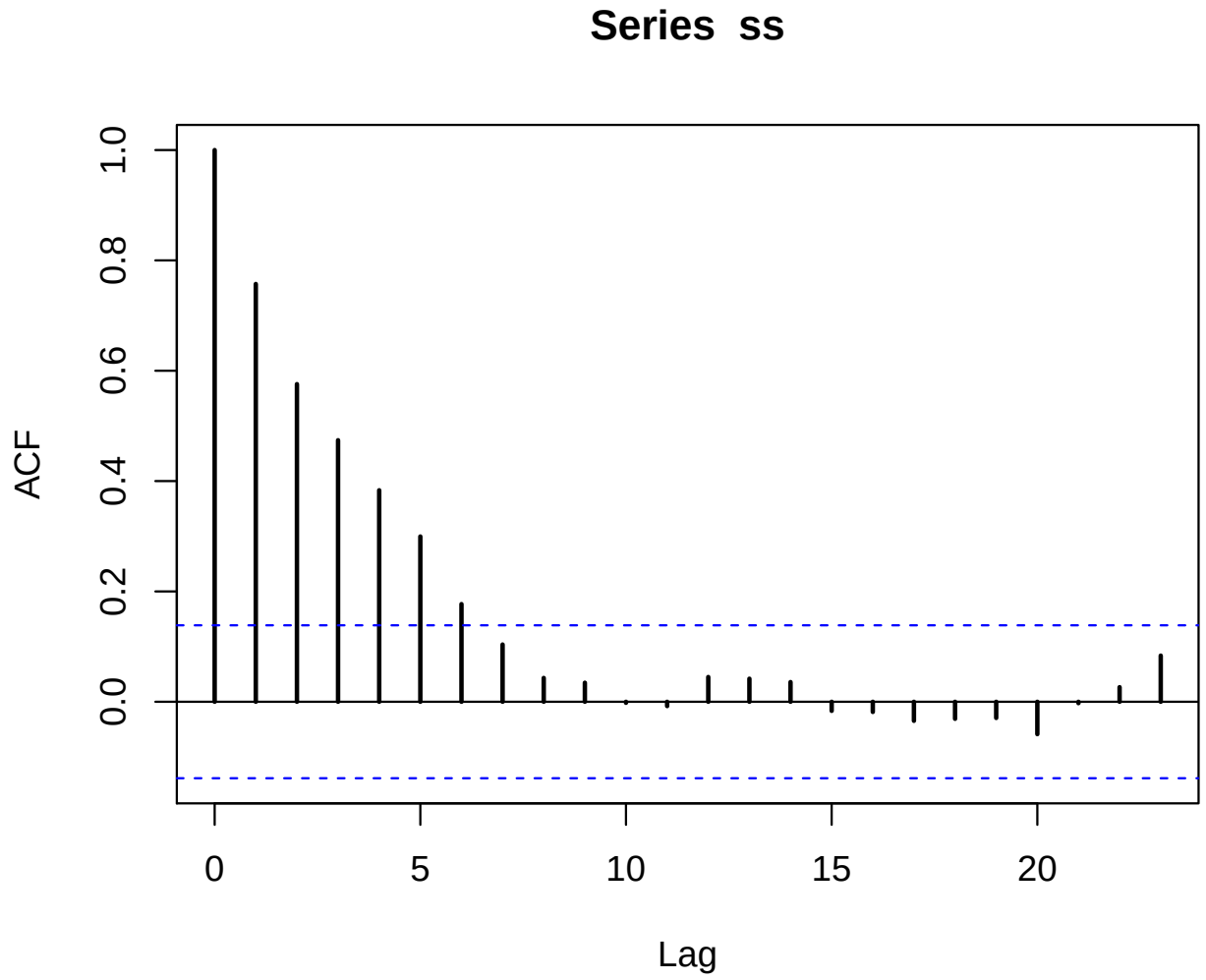
Stationary series

For AR term < 1, Y_t is always pulled back towards the mean.



- These will be our focus for the rest of today.

Autocorrelation for the stationary series drops off right away.



- The past matters, but with limited horizon.

Mean reversion

An important property of stationary series is mean reversion.

Think about shifting both Y_t and Y_{t-1} by their mean μ.

$$Y_t - \mu = \beta_1(Y_{t-1} - \mu) + \varepsilon_t$$

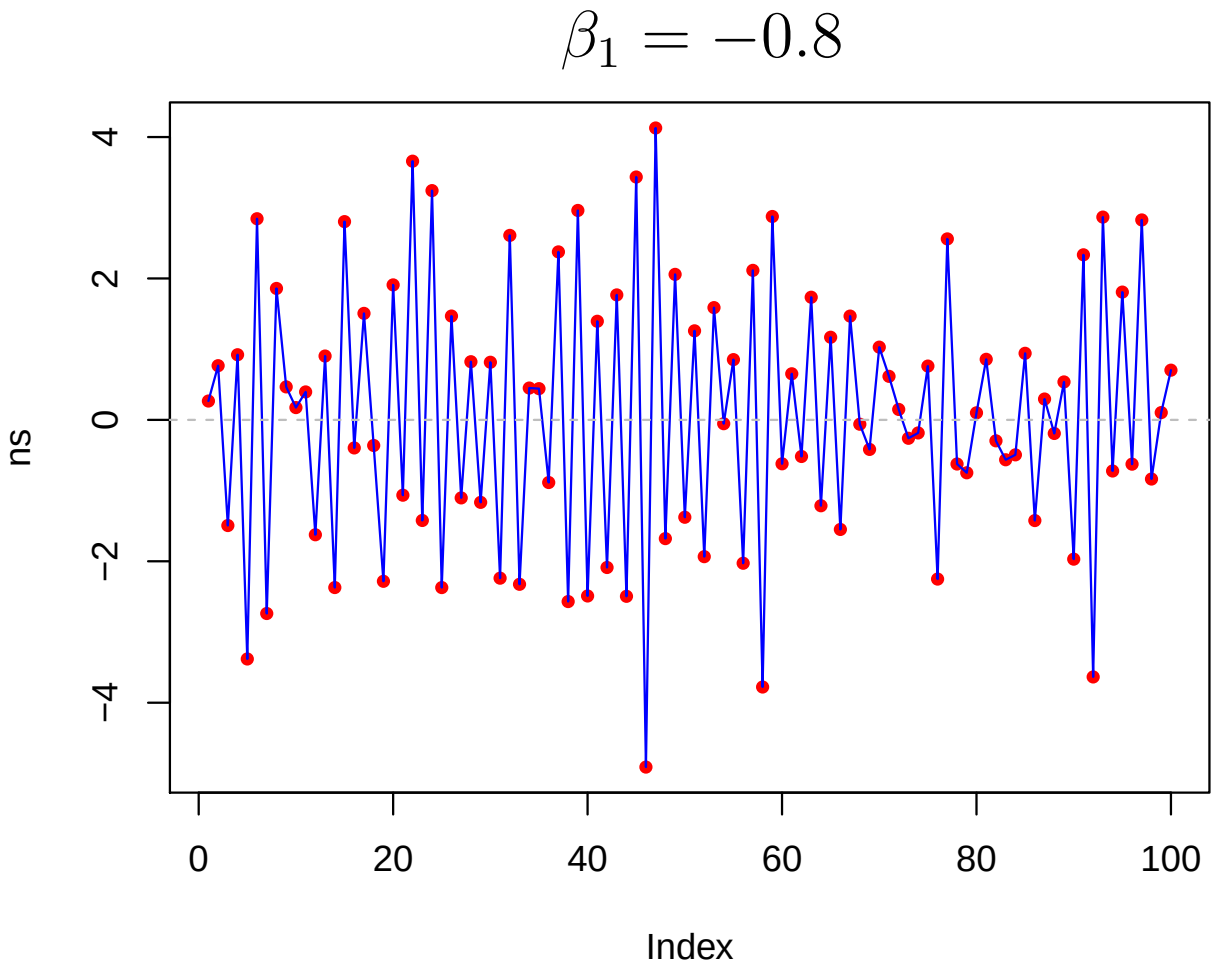
Since $|\beta_1| < 1$, Y_t is expected to be closer to μ than Y_{t-1}.

Mean reversion is all over, and helps predict future behaviour:

- "alpha" in repeated CAPM models,
- weekly sales numbers,
- daily temperature.

Negative correlation

It is also possible to have negatively correlated AR(1) series.



- But you see these far less often in practice.

Summary of AR(1) behavior

$|\beta_1| > 1$: The series explodes, is nonstationary, and might reflect something else going on.

$|\beta_1| < 1$: The series has a mean level to which it reverts. For positive β_1, the series tends to wander above or below the mean level for a while. For negative β_1, the series tends to flip back and forth around the mean. The series is stationary, meaning that the mean level does not change over time.

$|\beta_1| = 1$: A random walk series. The series has no mean level and, thus, is called nonstationary. The drift parameter β_0 is the direction in which the series wanders.

AR(p) models

It is possible to expand the AR idea to higher lags

$$AR(p) : Y_t = \beta_0 + \beta_1 Y_{t-1} + \cdots + \beta_p Y_{t-p} + \varepsilon.$$

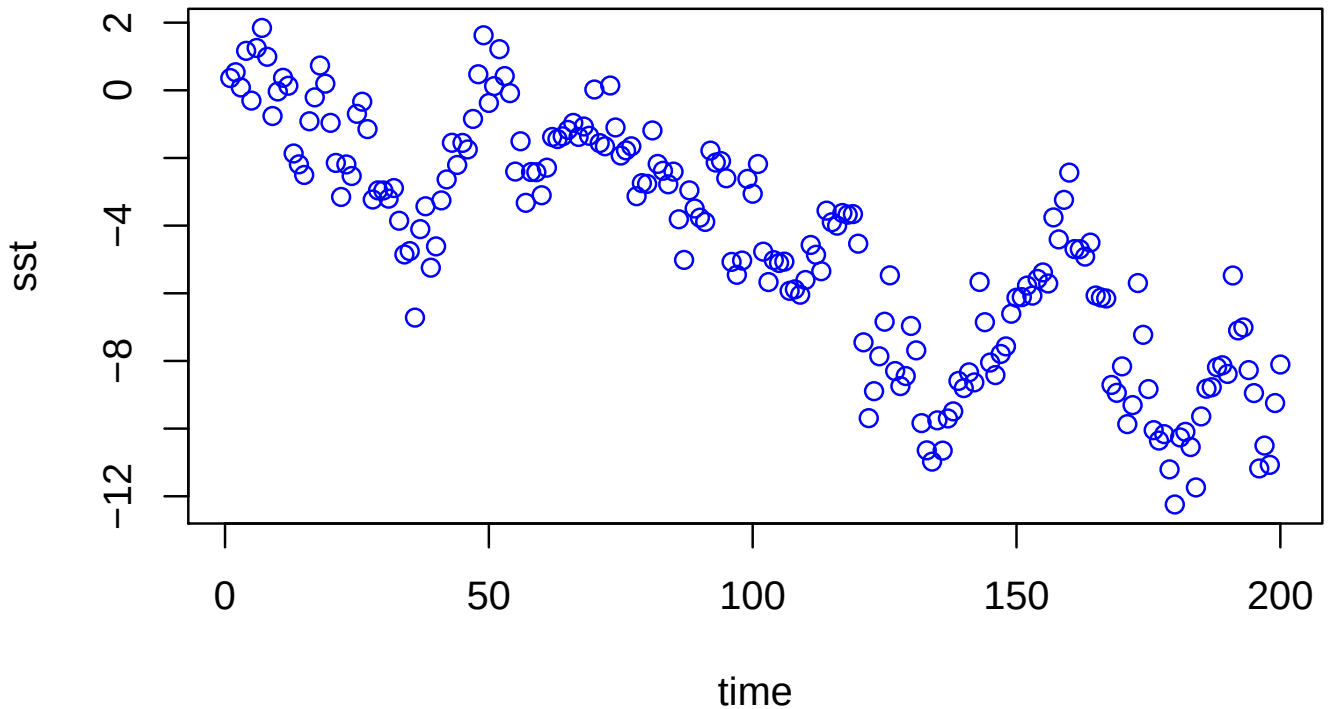
However, it is seldom necessary to fit AR lags for $p > 1$.

- Like having polynomial terms higher than 2, this just isn't usually required in practice.
- stationary vs. nonstationary less intuitive, still an issue.
- Often, the need for higher lags is symptomatic of (missing) a more persistent trend or periodicity in the data ...

Trending series

Often, you'll have a linear trend in your time series.

$\Rightarrow$ AR structure, sloping up or down in time.



This is easy to deal with: just put "time" in the model.

AR with linear trend:

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 t + \varepsilon_t$$

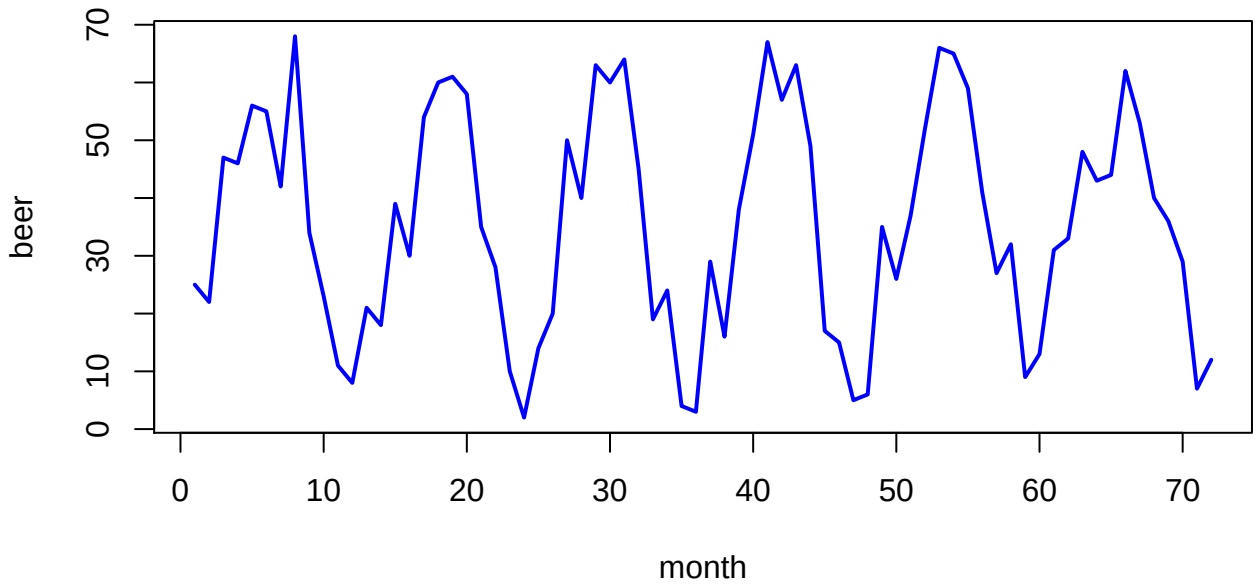
```
> t <- 1:199
> sst.fit <- lm(sst[2:200] ~ sst[1:199] + t)
> summary(sst.fit)  ## abbreviated output

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.571525   0.178110  -3.209  0.00156 **
sst[1:199]   0.735840   0.048062  15.310  < 2e-16 ***
t           -0.009179   0.002160  -4.249 3.32e-05 ***
```

Periodic models

It is very common to see seasonality or periodicity in series.

- Temperature goes up in Summer and down in Winter.
- Gas consumption in Chicago would do the opposite.

Recall the monthly beer production data:



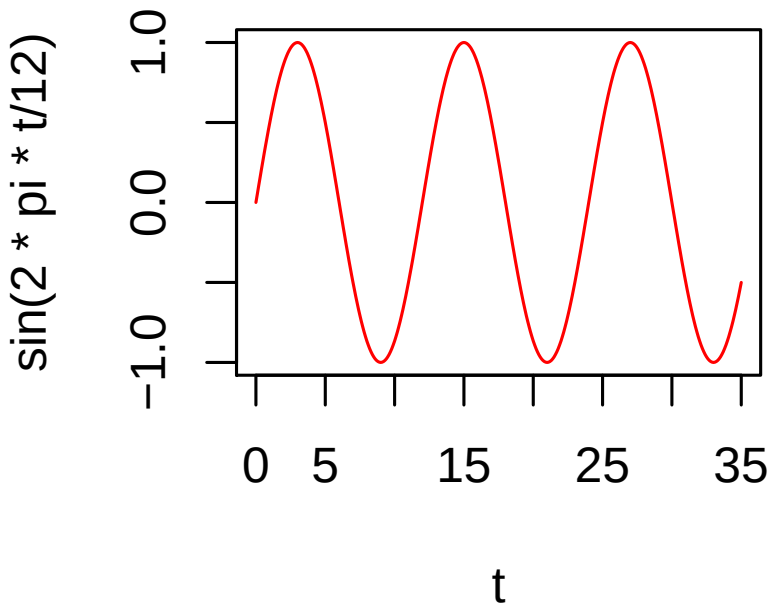
- Appears to oscillate on a 12-month cycle.

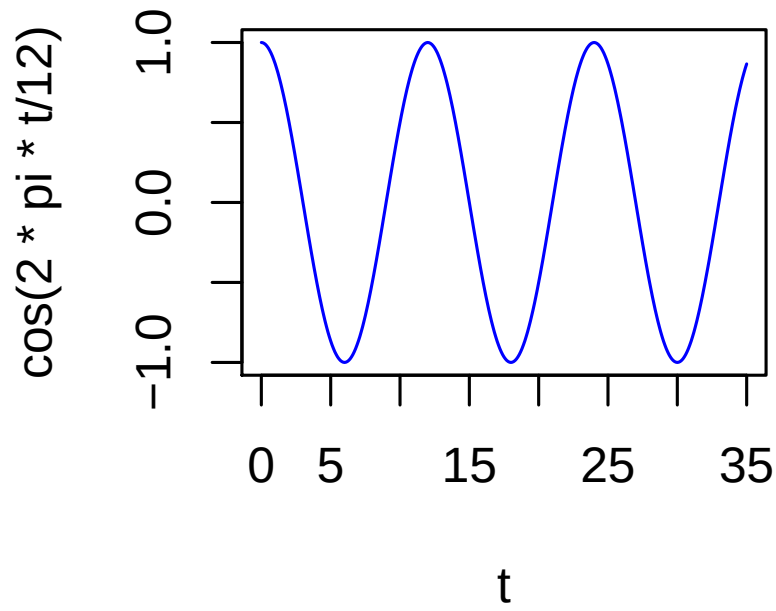
The straightforward solution: Add periodic predictors.

$\text{Period}-k$ model:

$$Y_t = \beta_0 + \beta_1 \sin(2\pi t/k) + \beta_2 \cos(2\pi t/k) + \varepsilon_t$$

Remember your sine and cosine!





- Repeating themselves every 2π.

Period$-k$ model:

$$Y_t = \beta_0 + \beta_1 \sin(2\pi t/k) + \beta_2 \cos(2\pi t/k) + \varepsilon_t$$

It turns out that you can represent any smooth periodic function as a sum of sines and cosines.

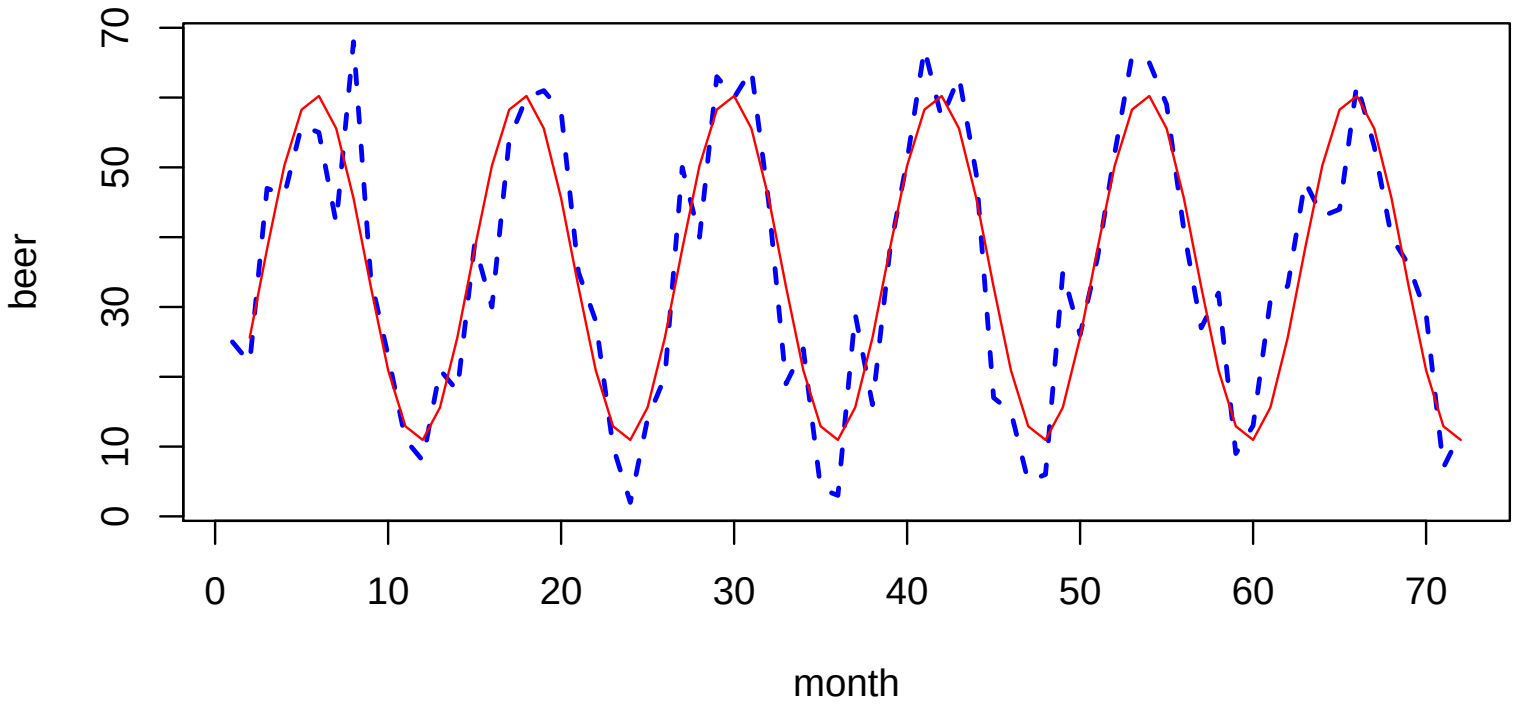
You choose k to be the number of "times" in a single period.

- For monthly data, $k = 12$ implies an annual cycle.
- For quarterly data, usually $k = 4$.
- For hourly data, $k = 24$ gives you a daily cycle.

On the beer data ...

```
> t <- 2:72
> sin12 <- sin(2*pi*t/12)
> cos12 <- cos(2*pi*t/12)
> beerreg3 <- lm(beer$prod[2:72] ~ sin12 + cos12)
> summary(beerreg3) ## abbreviated output

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  35.5918     0.9836  36.185   <2e-16 ***
sin12         2.6992     1.3861   1.947   0.0556 .
cos12       -24.6348     1.3960 -17.647   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
> plot(beer$prod, xlab="month", ylab="beer", type="l",
+   col=4, lwd=2, lty=2)
> lines(t, beerreg3$fitted, col=2)
```



- Good, but maybe not perfect?

```
> acf(beerreg3$resid)
```

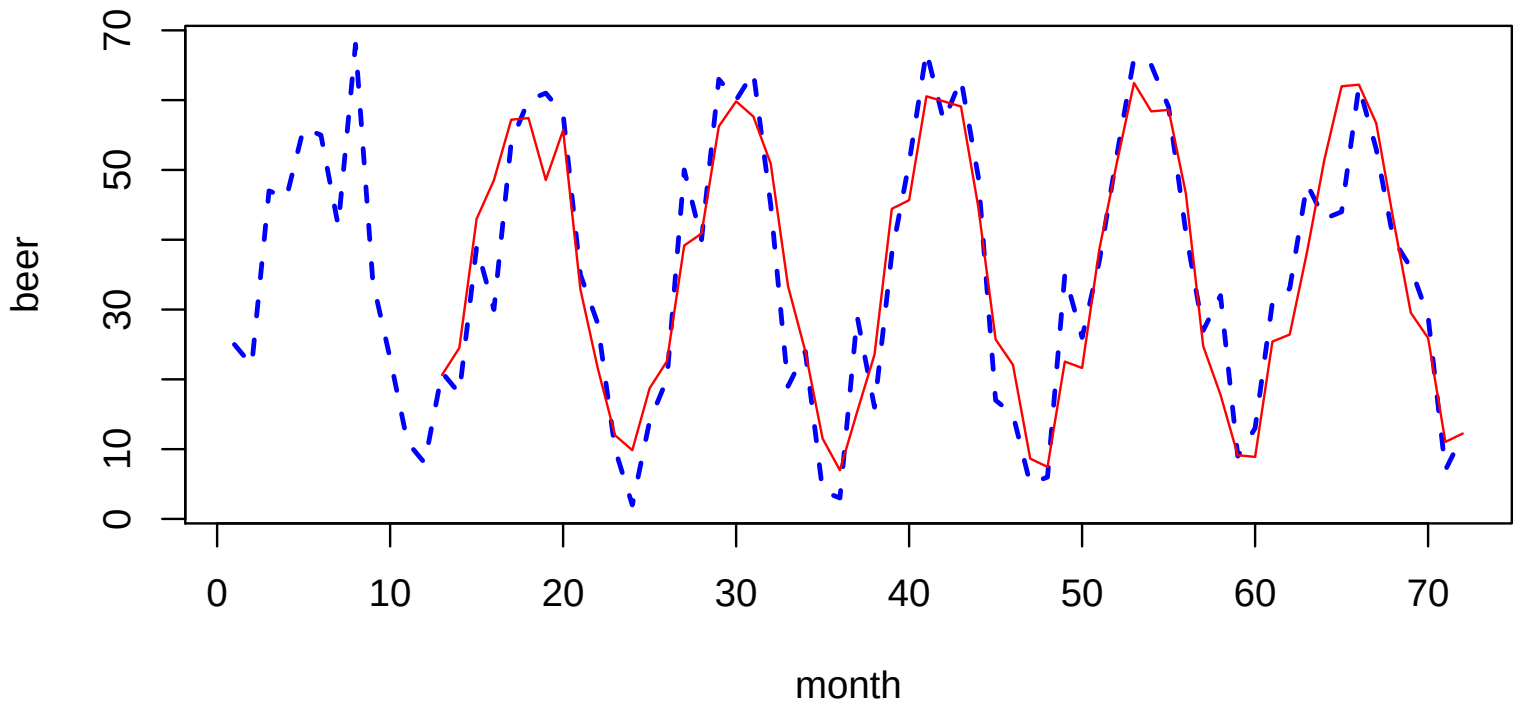
- We may not be getting the entire yearly pattern ...

A two-pronged approach:

```
> t <- 13:72
> sin12 <- sin(2*pi*t/12); cos12 <- cos(2*pi*t/12)
> beerreg4 <- lm(beer$prod[t]~sin12+cos12+beer$prod[t-12])
```

- Boom.

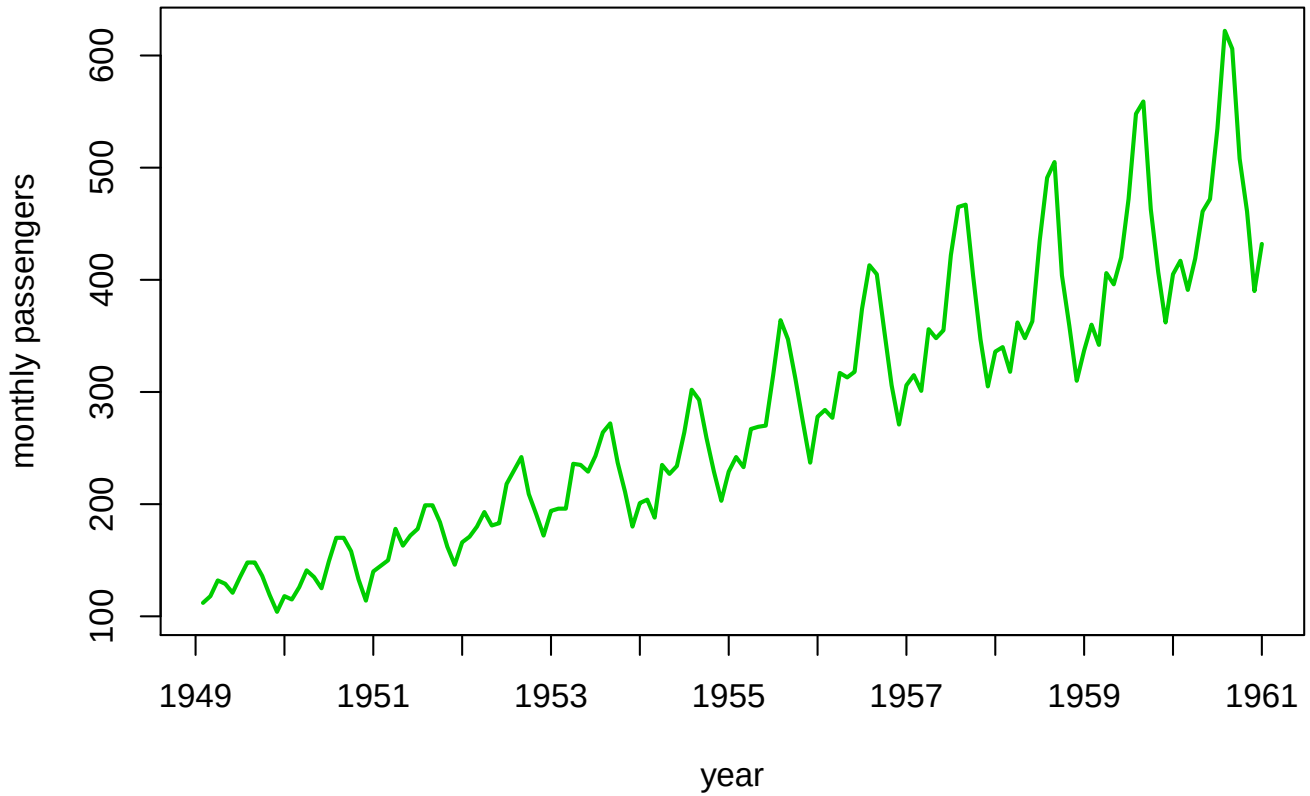
```
> plot(beer$prod, xlab="month", ylab="beer", type="l",
+   col=4, lwd=2, lty=2)
> lines(t, beerreg4$fitted, col=2)
```



- A bit better.

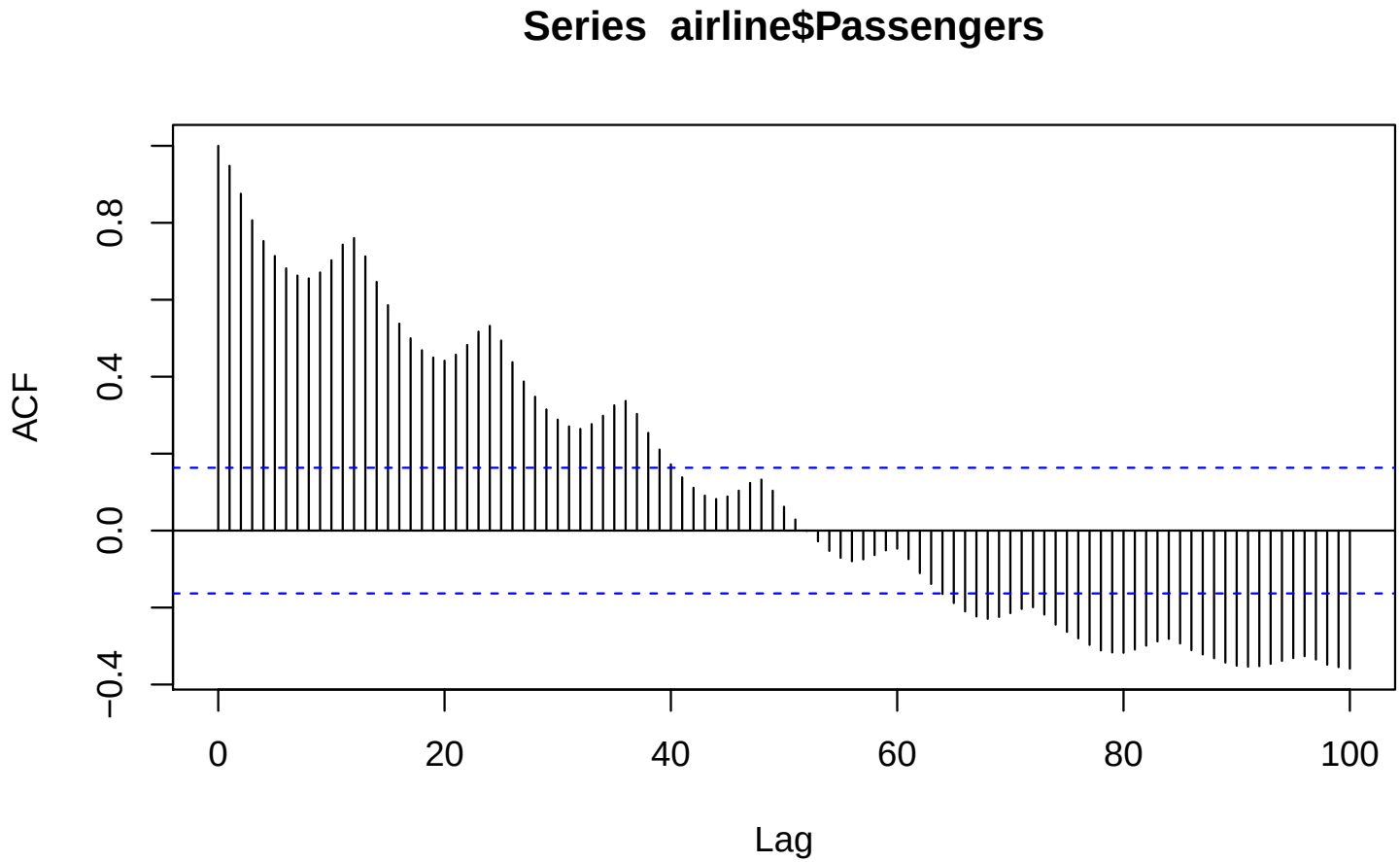
Putting it all together: Airline data

- $Y_t =$ monthly total international passengers, 1949-1960.



- Increasing annual oscillation and positive linear trend.

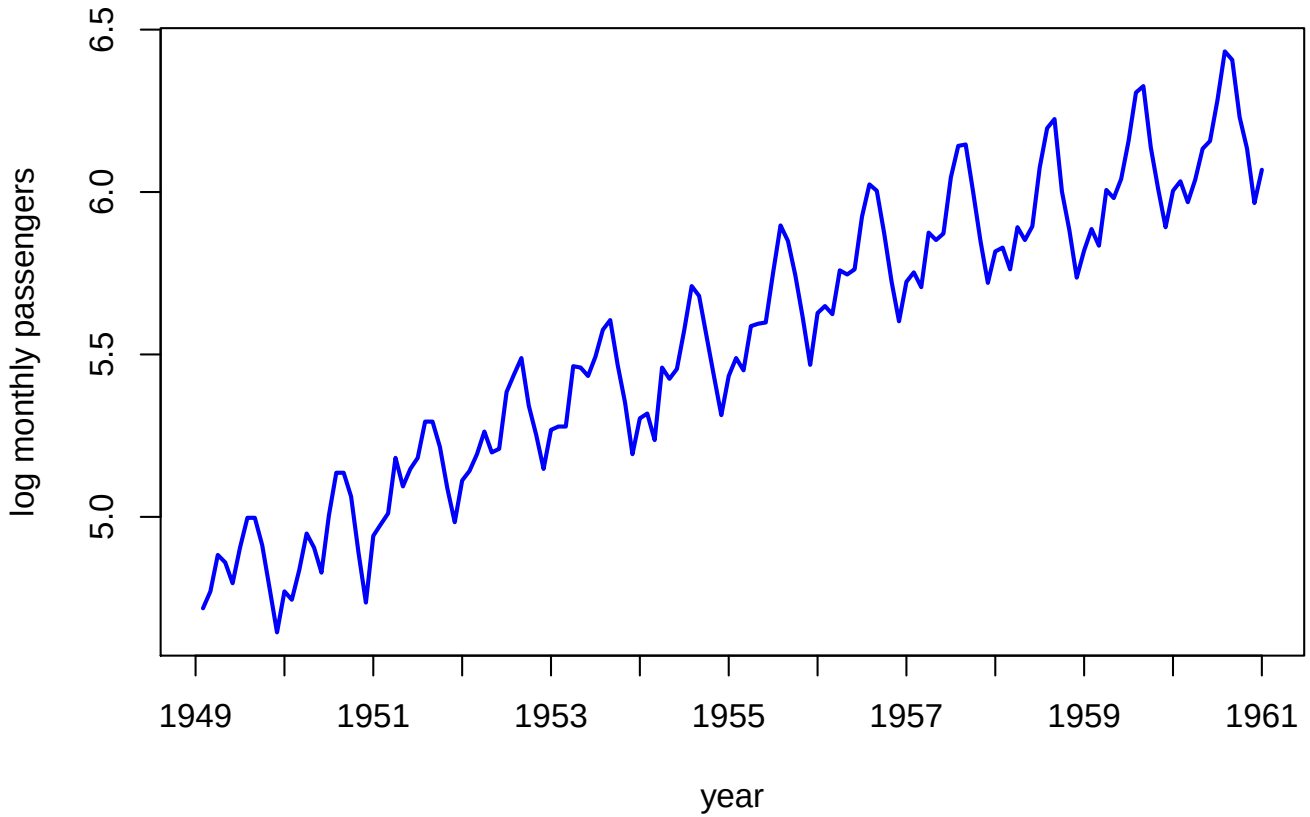
The data shows a strong persistence in correlation.



Annual (12 month) periodicity shows up here as well.

Fitting the model: first, don't forget your fundamentals!

- The series variance is increasing in time.
- Passenger numbers are like sales volume.
- We should be working on log scale!



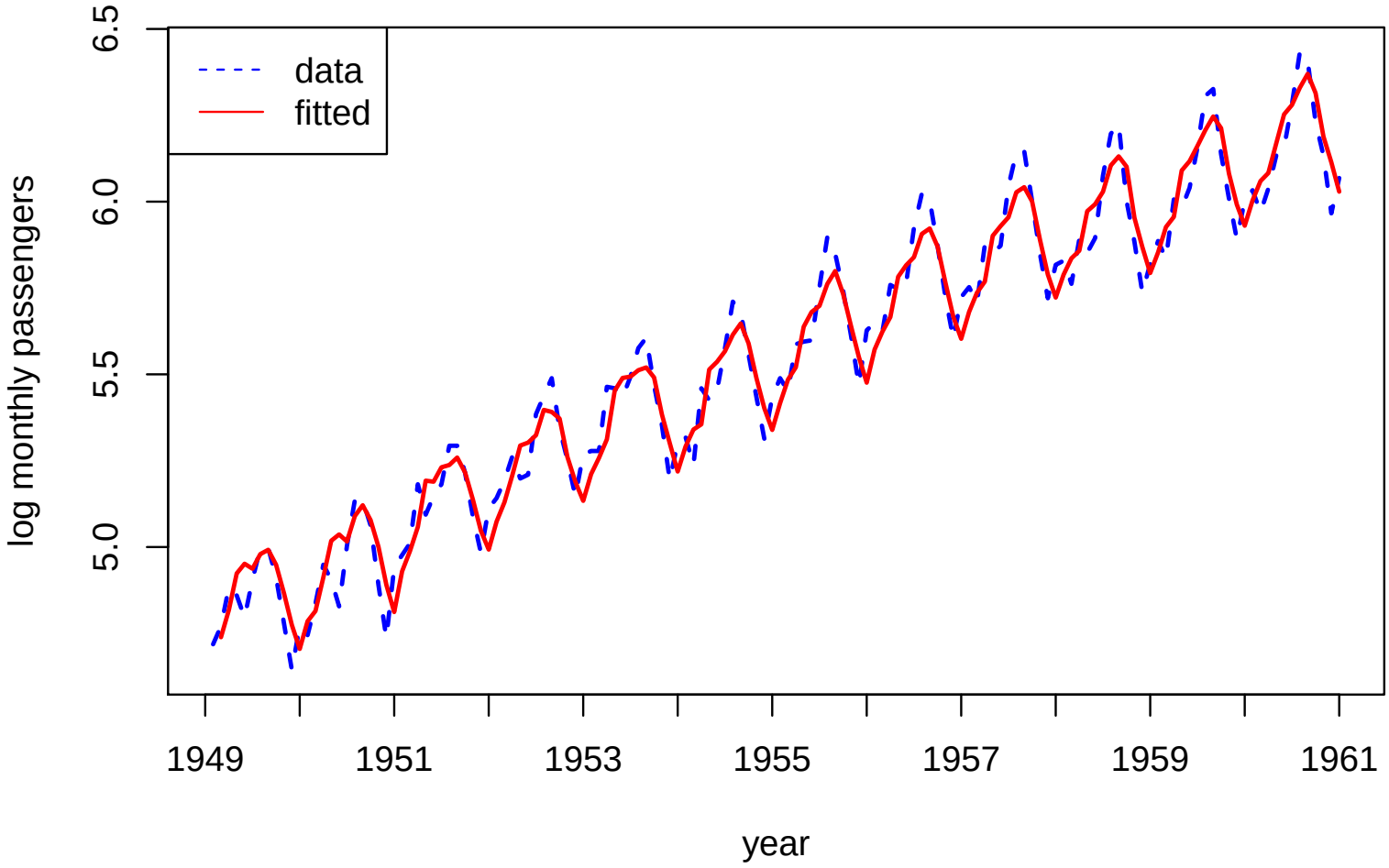
The series shows a linear trend, an oscillation of period 12, and we expect to find autoregressive errors.

$$\log(Y_t) = \beta_0 + \beta_1 \log(Y_{t-1}) + \beta_2 t + \beta_3 \sin\left(\frac{2\pi t}{12}\right) + \beta_4 \cos\left(\frac{2\pi t}{12}\right) + \varepsilon_t$$

```
> t <- 2:nrow(airline)
> YX <- data.frame(logY=log(airline$Passengers[2:144]),
+   logYpast=log(airline$Passengers[1:143]), t=t,
+   sin12=sin(2*pi*t/12), cos12=cos(2*pi*t/12))
> airlm <- lm(logY ~ logYpast + t + sin12 + cos12, data=YX)
```

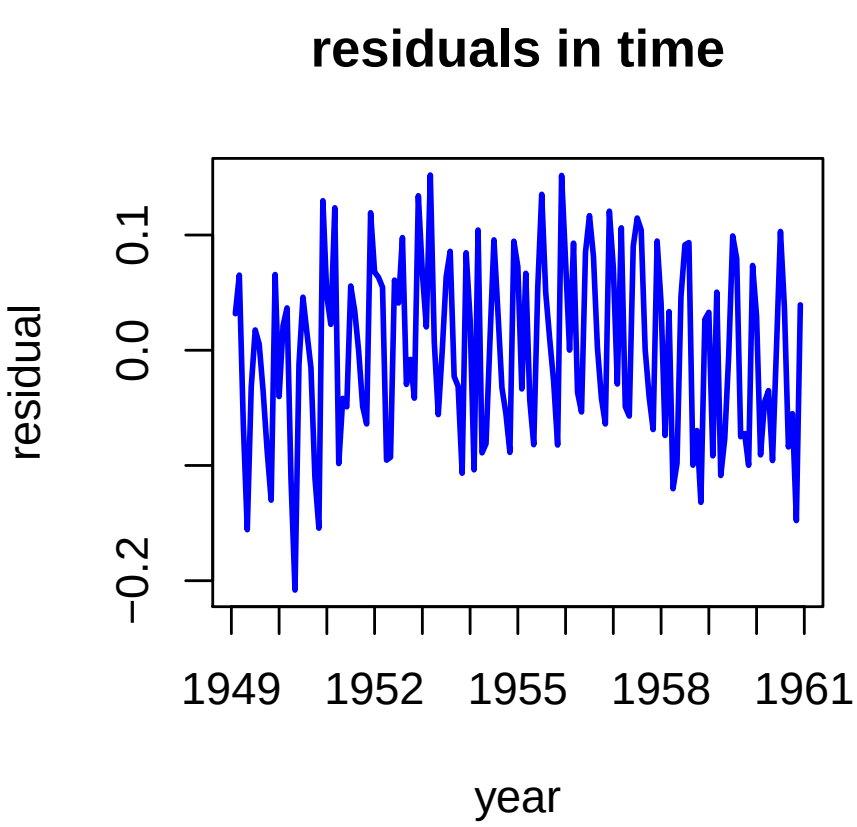
```
> summary(airlm) ## abbreviated output

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.5323909  0.3603010   7.029 8.77e-11 ***
logYpast     0.4748286  0.0749506   6.335 3.12e-09 ***
t            0.0052759  0.0007703   6.849 2.25e-10 ***
sin12        0.0040818  0.0126512   0.323    0.747
cos12       -0.0960295  0.0119032  -8.068 3.12e-13 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.07929 on 138 degrees of freedom
Multiple R-squared:  0.9681,  Adjusted R-squared:  0.9672
F-statistic:  1047 on 4 and 138 DF,  p-value: < 2.2e-16
```

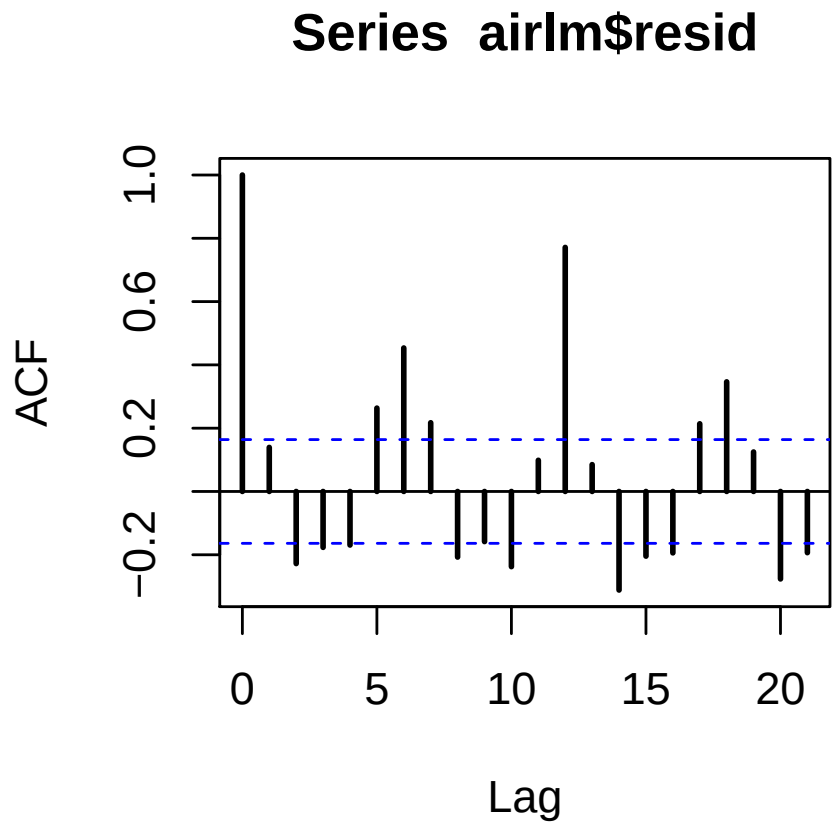
The model predictions look pretty good!



- Sine and cosine trends seem to capture the periodicity.

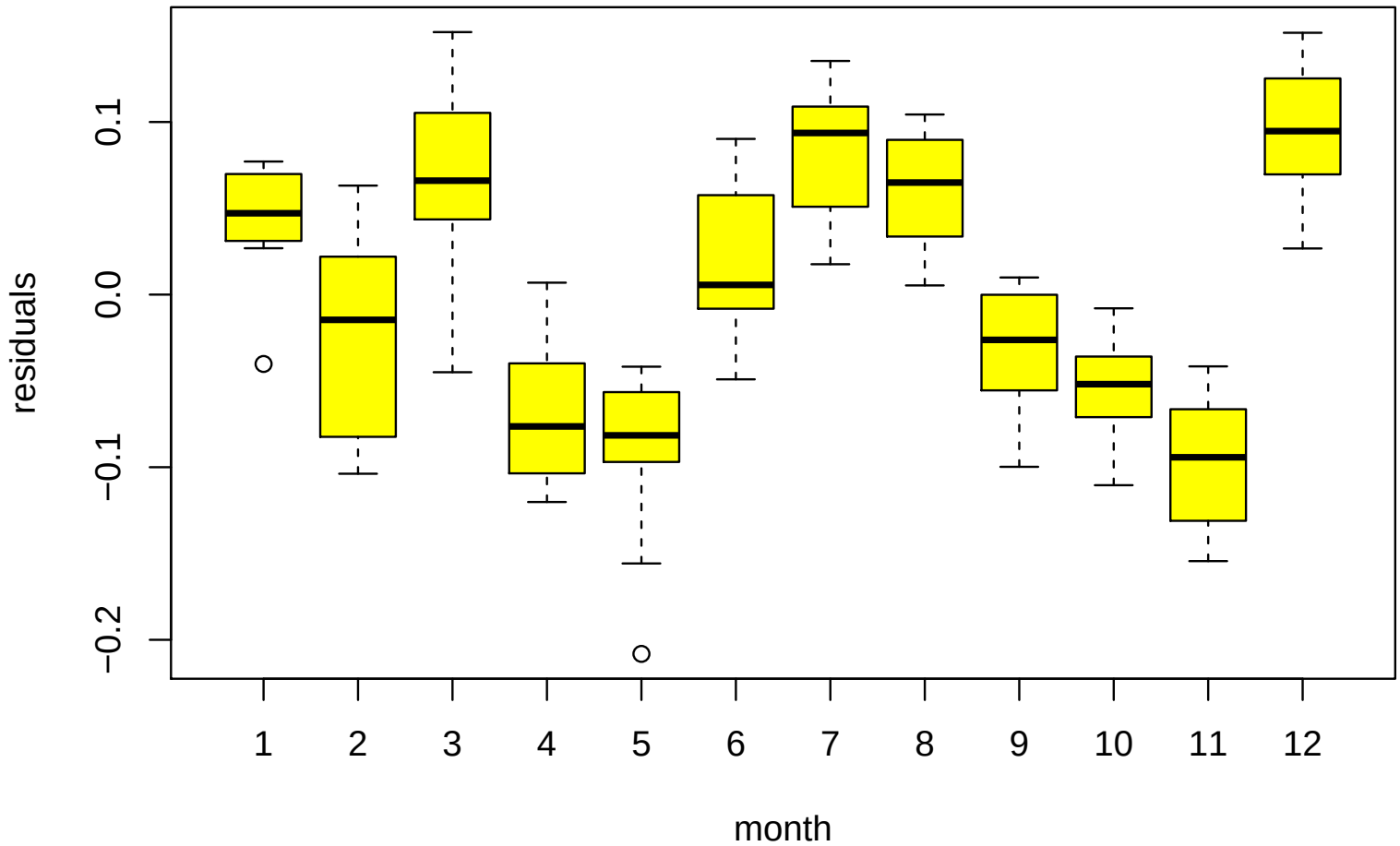
However, a closer look exposes residual autocorrellation.





- How can we fix this?

You can see the relationship show up in monthly residuals.



- This is probably due to holiday/shoulder season effects.

We create some useful dummy variables:

```
> YX$holidays <- airline$Month[t] %in% c(3,6,7,8,12)
> YX$jan <- airline$Month[t]==1
> YX$nov <- airline$Month[t]==11
> YX$jul <- airline$Month[t]==7
```

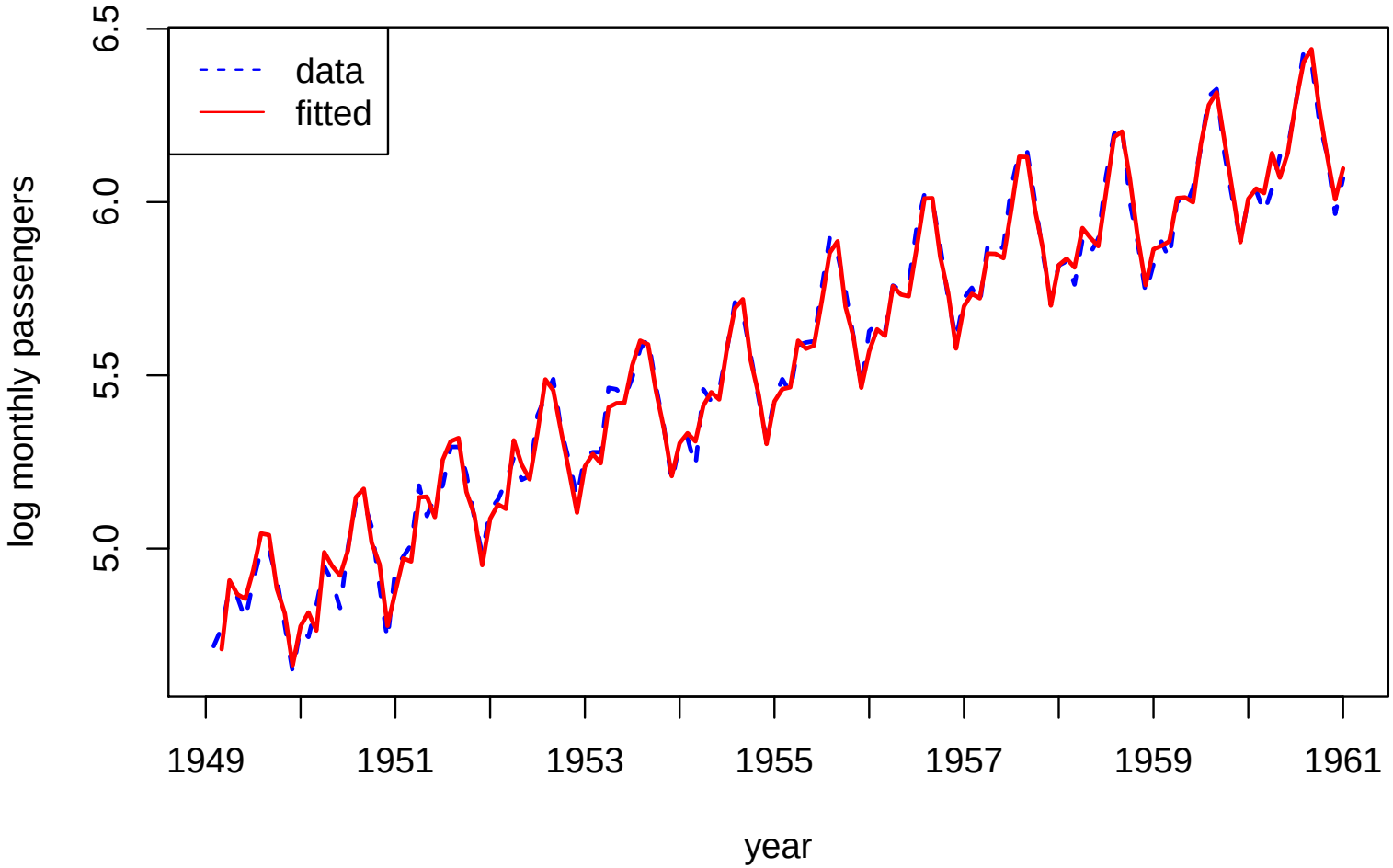
Then re-fit the model with `holidays`, `nov`, `jan`, and `jul`.

- Months with `holidays` have an obvious effect.
- `nov` and `jan` have fewer vacation days.
- `jul` is unique as the entire month is school holiday.

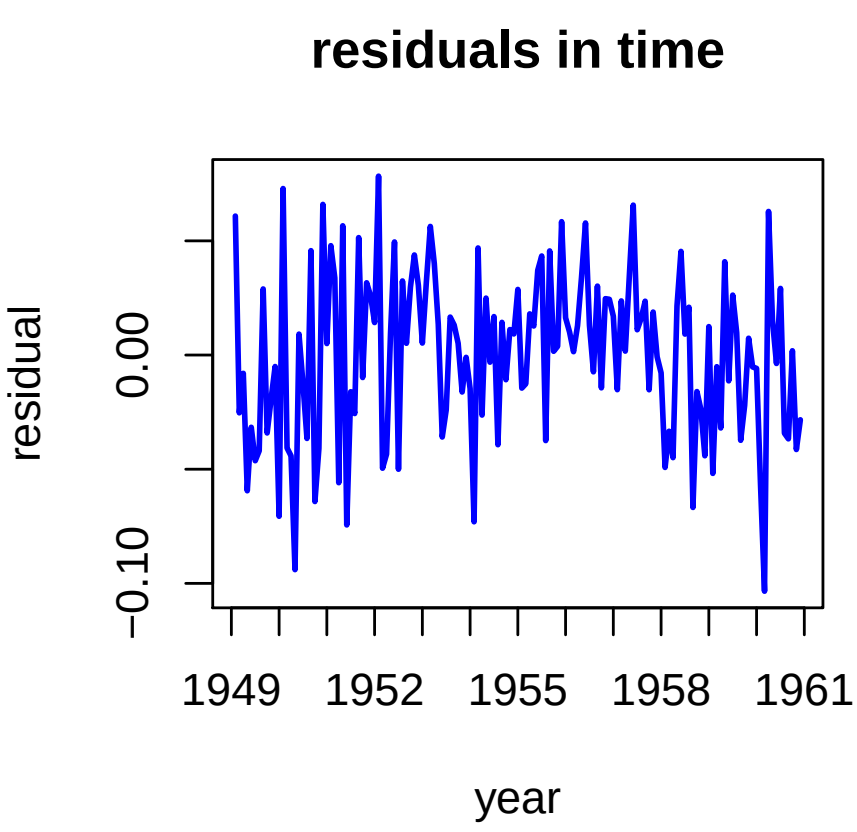
Everything shows up as being very significant.

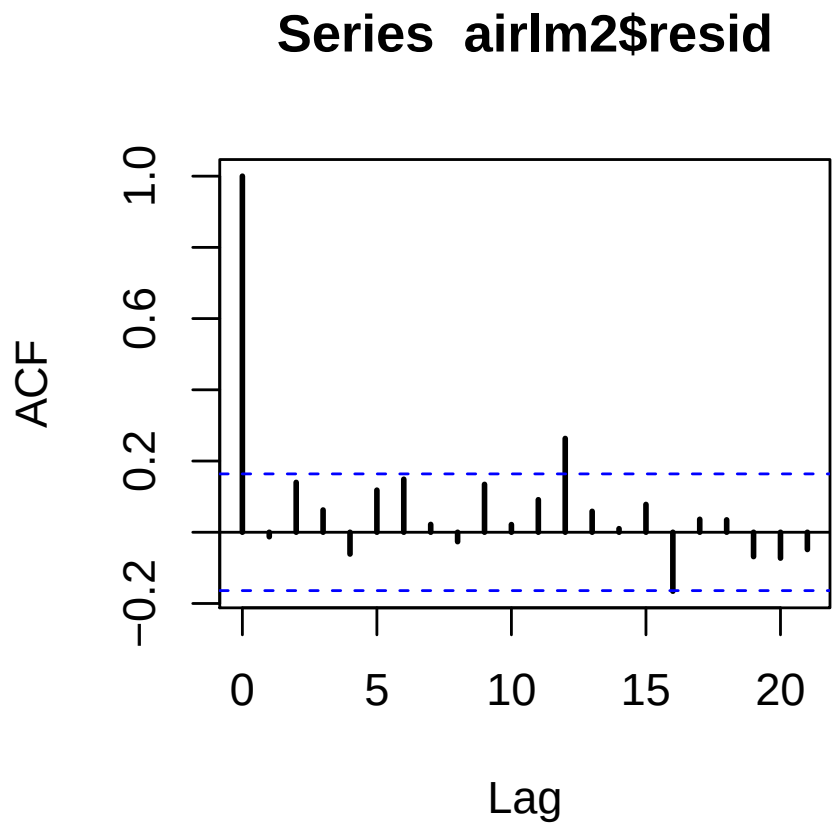
```
> airlm2 <- lm(logY ~ logYpast + t + sin12 + cos12
+   + holidays + nov + jan + jul, data=YX)
> summary(airlm2)

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.3427507  0.1945587   6.902 1.86e-10 ***
logYpast     0.7100231  0.0401417  17.688  < 2e-16 ***
t            0.0028983  0.0004111   7.050 8.57e-11 ***
sin12        0.0332607  0.0069795   4.765 4.84e-06 ***
cos12       -0.0355395  0.0070772  -5.022 1.60e-06 ***
holidaysTRUE 0.1361014  0.0079670  17.083  < 2e-16 ***
novTRUE     -0.0571301  0.0136937  -4.172 5.39e-05 ***
janTRUE      0.0619620  0.0136601   4.536 1.26e-05 ***
julTRUE      0.0473444  0.0131525   3.600 0.000447 ***
```

The one-step-ahead model predictions look even better.



- We're now really able to capture the annual dynamics.





- There is a bit of left-over 12 month autocorrelation, but nothing to get overly worried about.

Alternative Periodicity

An alternative way to add periodicity would be to simply add a dummy variable for each month (`feb`, `mar`, `apr`, `...`).

- This achieves basically the same fit as above, without requiring you to add sine or cosine.
- However, this takes 11 periodic parameters while we use only 6 (sine and cosine + `holidays`, `nov`, `jan`, and `jul`).

I like to think of the periodicity as a smooth oscillation, with sharp day/month effects added for special circumstances.

- Requires more thought, but leads to better models.
- The $\sin + \cos$ technique works regardless of the number of increments in a period (e.g. 365 days).

The exception:

- Since quarterly data has a period of only 4, it is often fine to just add "quarter" effects.

Time series – wrapping up

The tools here are good, but not the best:

- In many situations you want to allow for β or σ parameters that can change in time.
- This can leave us with some left-over autocorrelation.
- Jeff Russell (41202) and Ruey Tsay (41203) teach advanced time series classes.