VectorBiTE Methods Training
Introduction to Auto-correlated Data and Time Series

The VectorBiTE Team
(Leah R. Johnson, Virginia Tech)

Summer 2021

Learning Objectives

In this module you will:

1. Learn what makes time series (TS) data different from other regression data.
2. See how to visualize and learn about autocorrelation in a TS.
3. Learn how to use basic regression techniques to fit models that include AR, trending, and periodic components.
4. Learn what diagnostics to examine to determine if your models are fitting well.

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

- daily temperature,
- solar activity,
- CO_2 levels,
- GDP,
- yearly population size.

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?

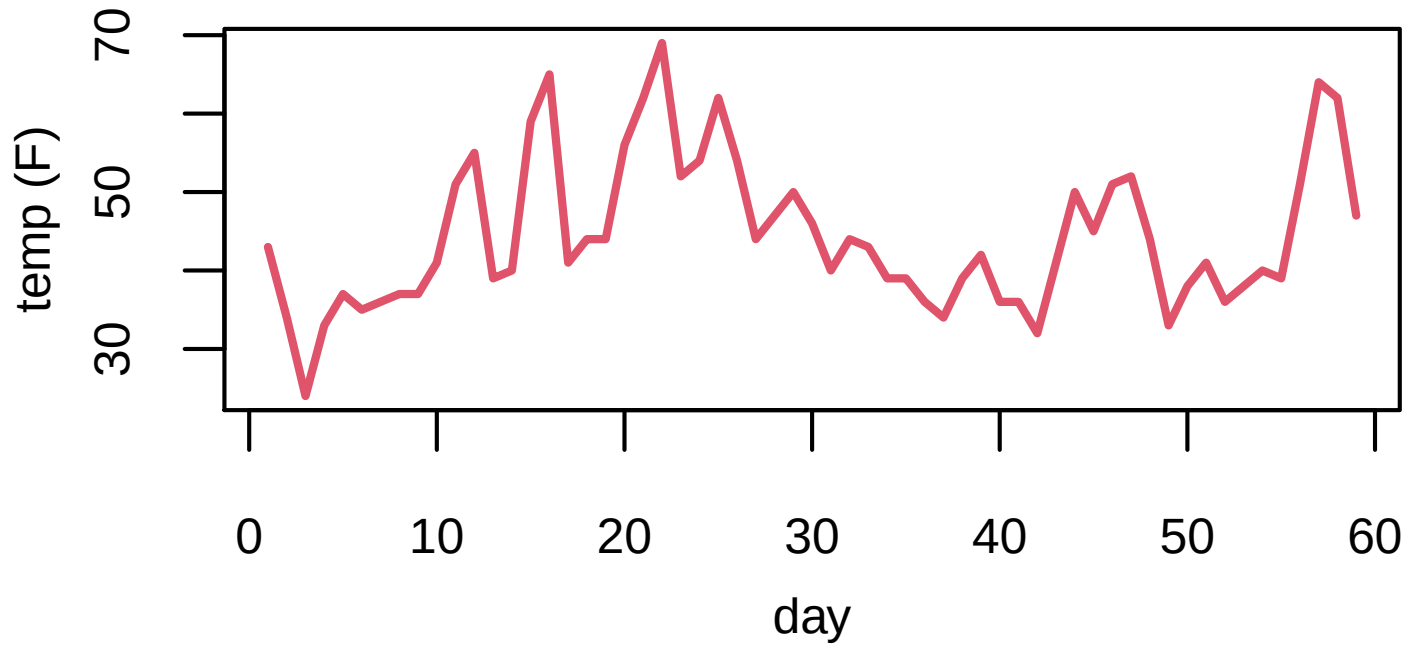
insert_slide_break()

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 ROA, Feb-Mar 2018.

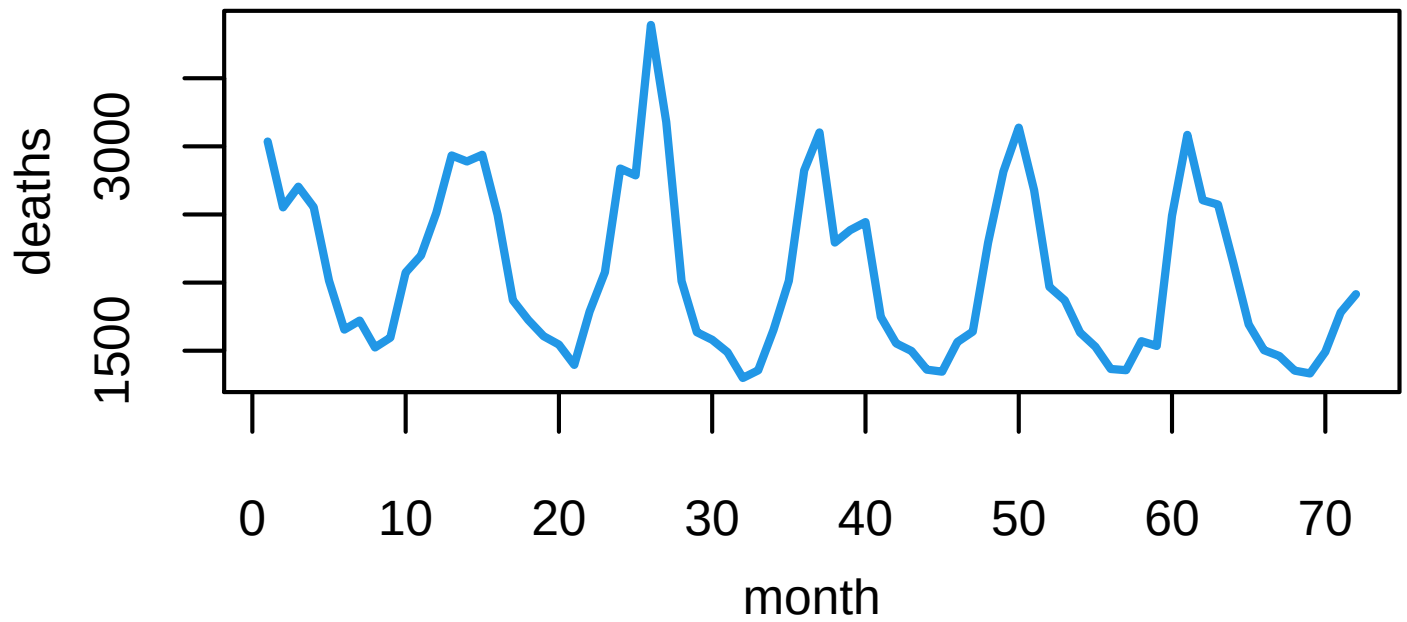
```
weather <- read.csv("../data/Roanoke_weather2018.csv")
days<-32:90 ## This picks out February and March
weather<-weather[days,c(1,3,7)]; names(weather)[3]<-"temp"
plot(weather$temp, xlab="day", ylab="temp (F)", type="l",
     col=2, lwd=2, cex.lab=0.75, cex.axis=0.75, mgp=c(2,1,0))
```



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

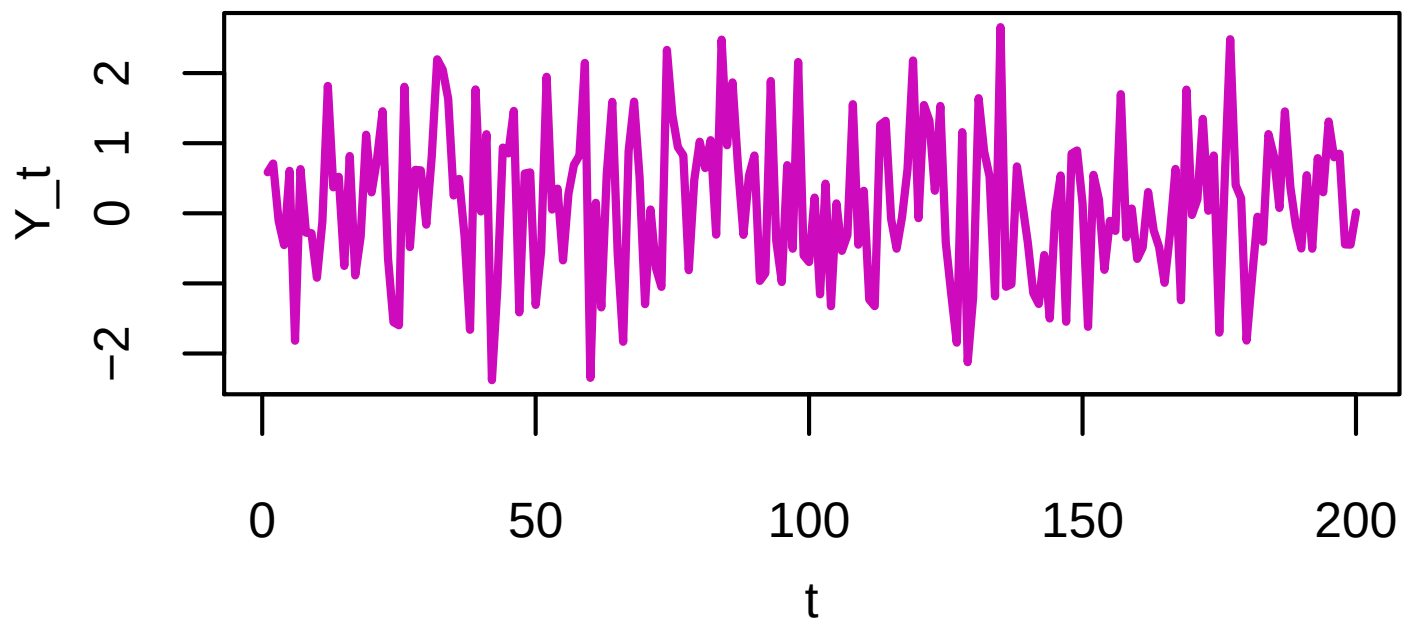
Example: Y_t = monthly UK deaths due to lung infections

```
ld<-as.vector(ldeaths)
plot(ld, xlab="month", ylab="deaths", type="l",
     col=4, lwd=2, cex.lab=0.75, cex.axis=0.75, mgp=c(2,1,0))
```



- The same pattern repeats itself year after year.

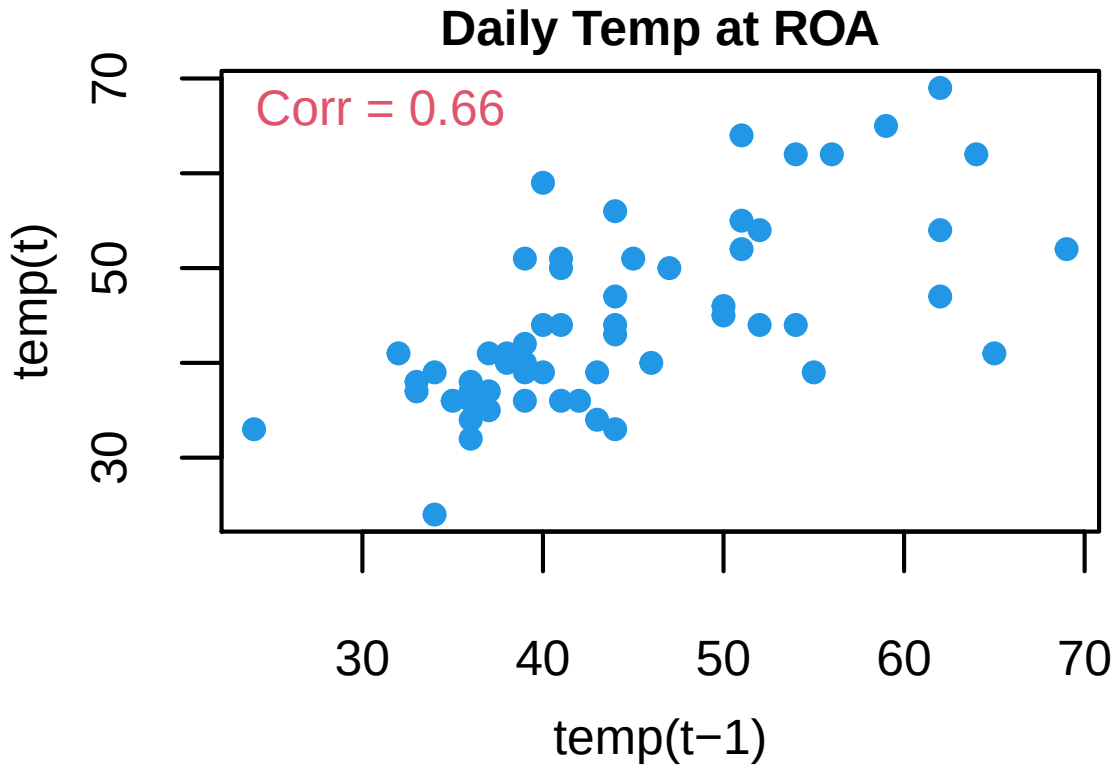
```
rand<-rnorm(200)
plot(rand, xlab="t", ylab="Y_t", type="l",
     col=6, lwd=2, cex.lab=0.75, cex.axis=0.75, mgp=c(2,1,0))
```



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

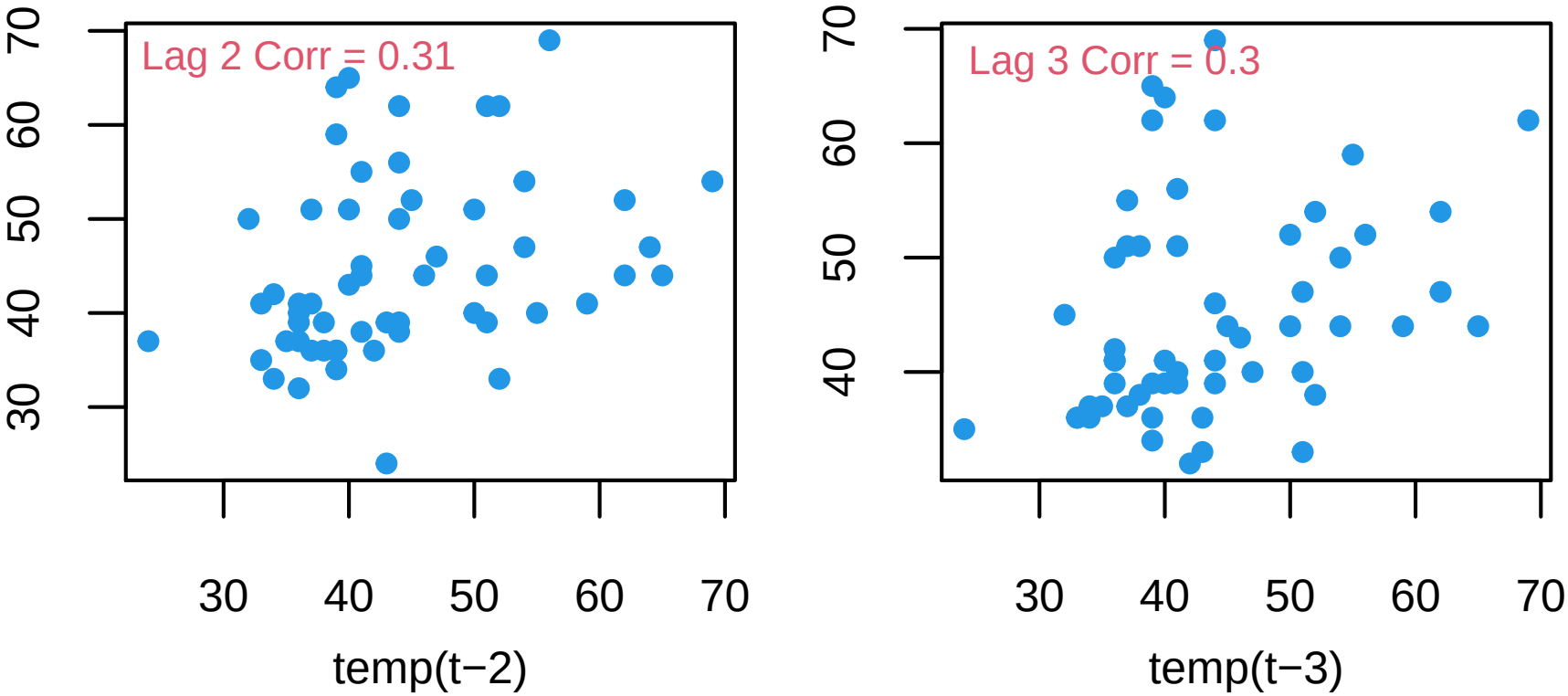
Checking for dependence

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.



- Correlation appears to be 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

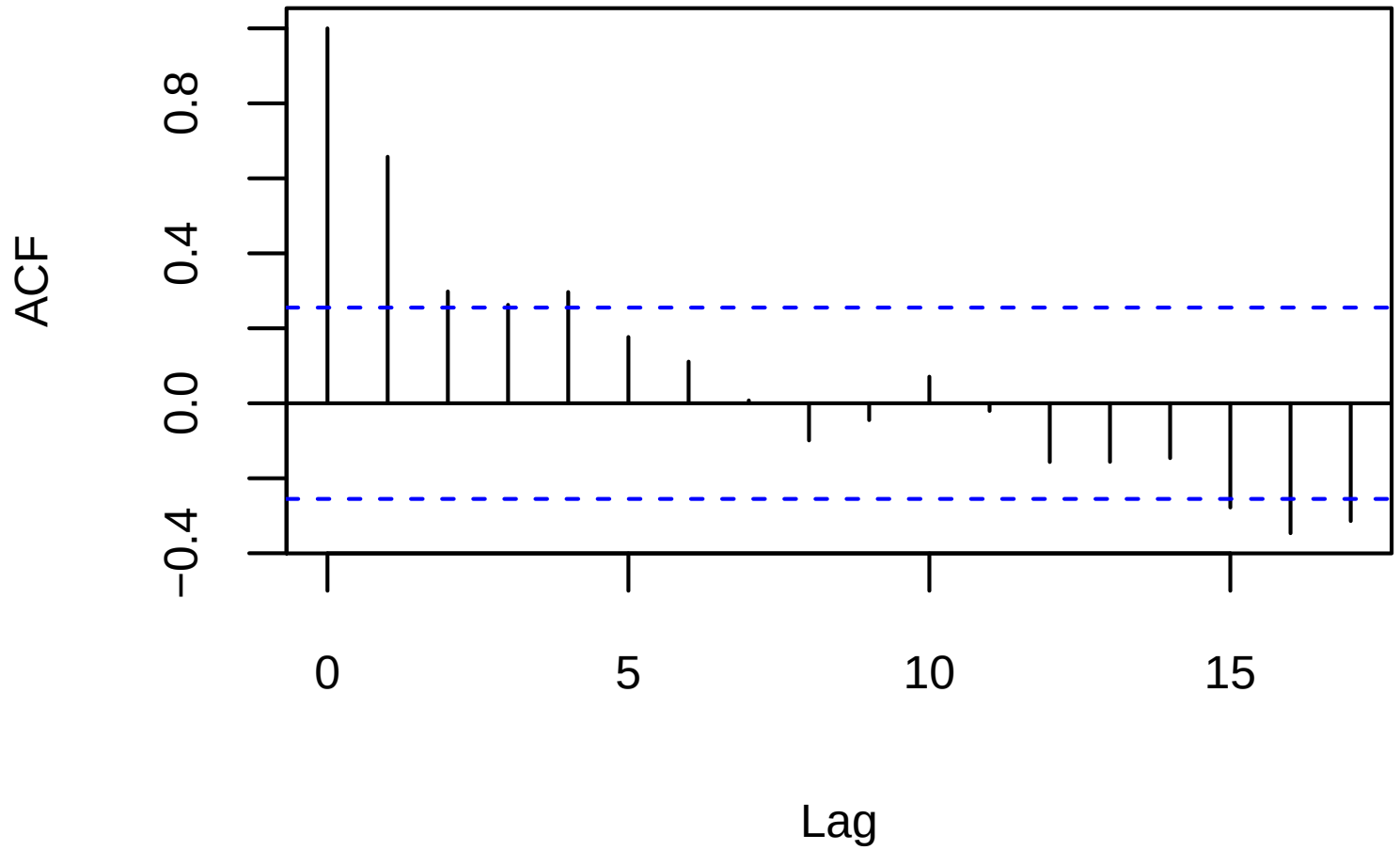
$$r(\ell) = \mathrm{cor}(Y_t, Y_{t-\ell})$$

For our Roanoke temperature data:

```
print(acf(weather$temp, plot=FALSE))
>
> Autocorrelations of series 'weather$temp', by lag
>
>      0      1      2      3      4      5      6      7      8      9
>  1.000  0.658  0.298  0.263  0.297  0.177  0.111  0.008 -0.099 -0.045
>     11     12     13     14     15     16     17
> -0.020 -0.157 -0.156 -0.146 -0.278 -0.346 -0.314
```

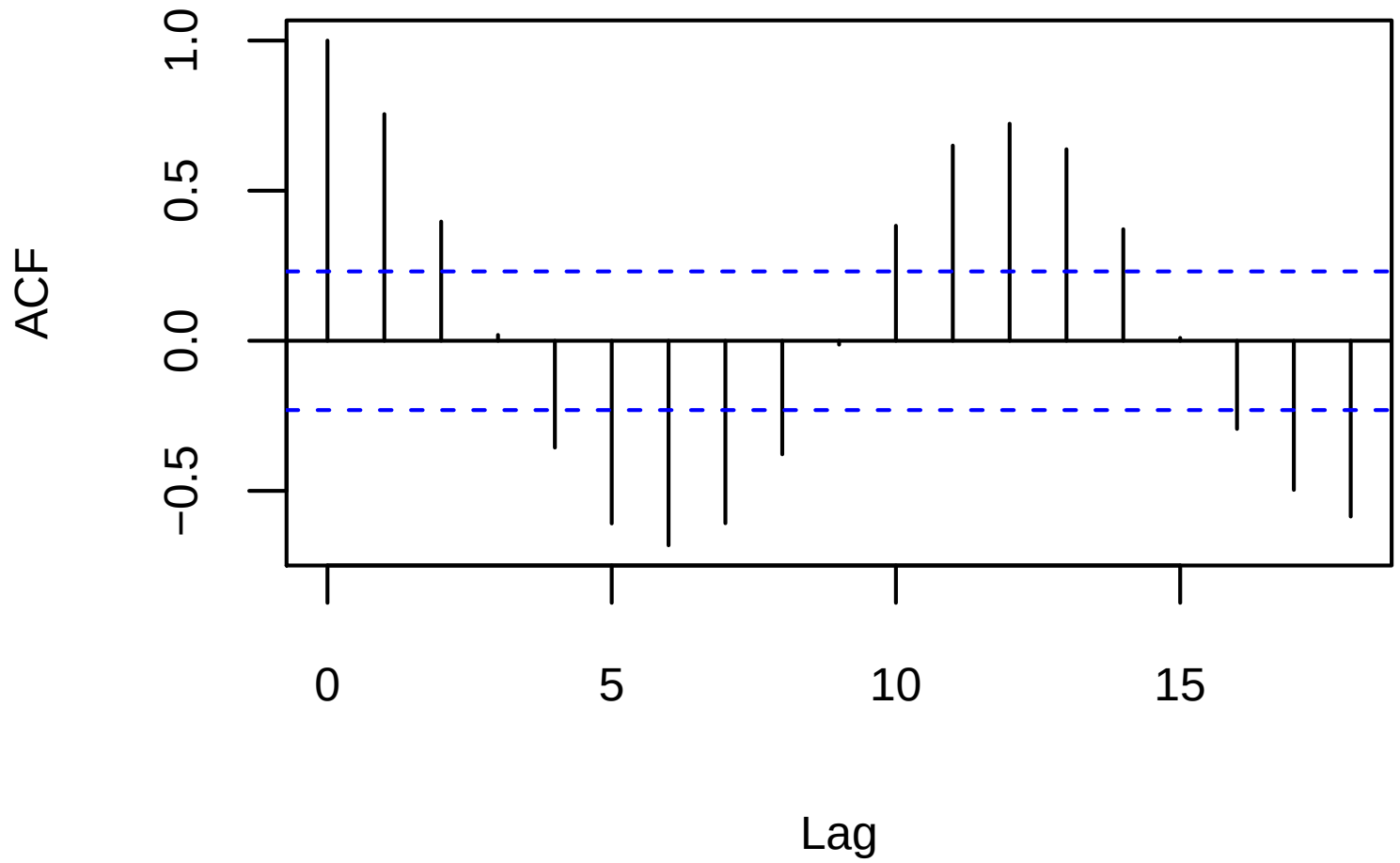
R's `acf` function provides a visual summary of our data dependence:

```
acf(weather$temp, cex.lab=0.75, cex.axis=0.75, main="")
```



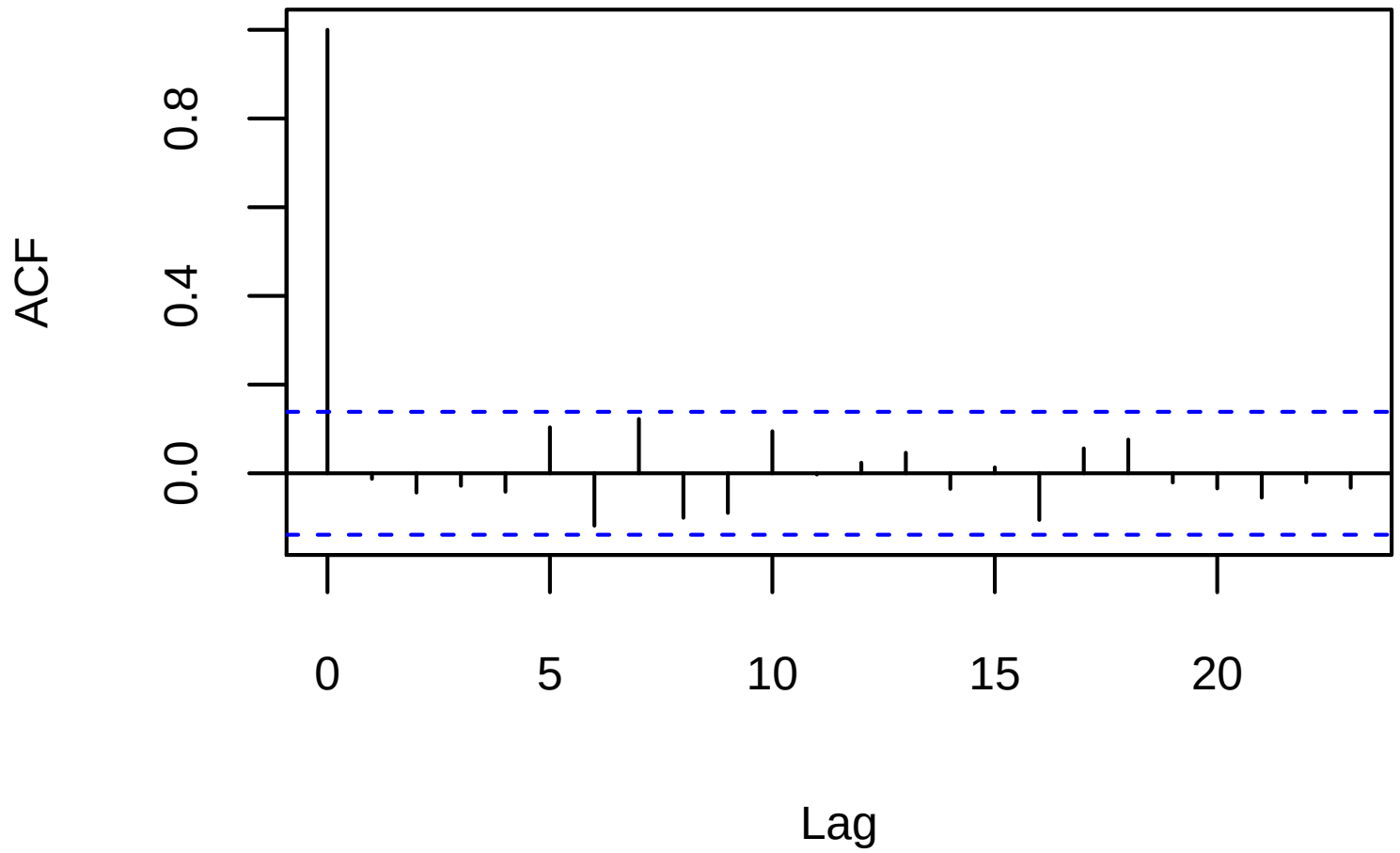
The lung infection data shows an alternating dependence structure which causes time series oscillations.

```
acf(ld, cex.lab=0.75, cex.axis=0.75, main="")
```



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

```
acf(rand, cex.lab=0.75, cex.axis=0.75, main="")
```



Autoregression

How do we model data that exhibits autocorrelation?

Suppose $Y_1 = \varepsilon_1$, $Y_2 = \varepsilon_1 + \varepsilon_2$, $Y_3 = \varepsilon_1 + \varepsilon_2 + \varepsilon_3$, etc.
Then $Y_t = \sum_{i=1}^{t} \varepsilon_i = Y_{t-1} + \varepsilon_t$ and $\mathbb{E}[Y_t] = Y_{t-1}$.

This is called a random walk model for Y_t:

- the expectation of what will happen is always what happened most recently.

Even though Y_t is a function of errors going all the way back to the beginning, you can write it as depending only on Y_{t-1}.

Random walks are just a version of a more general model ...

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}.

It assumes all of our standard regression model conditions.

- The residuals should look *iid* and be uncorrelated with $\hat{Y}_t$.
- All of our standard 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.

For the Roanoke temperatures, there was clear autocorrelation when we plotted above.

Let's fit a linear model between temperature and its one step lag value:

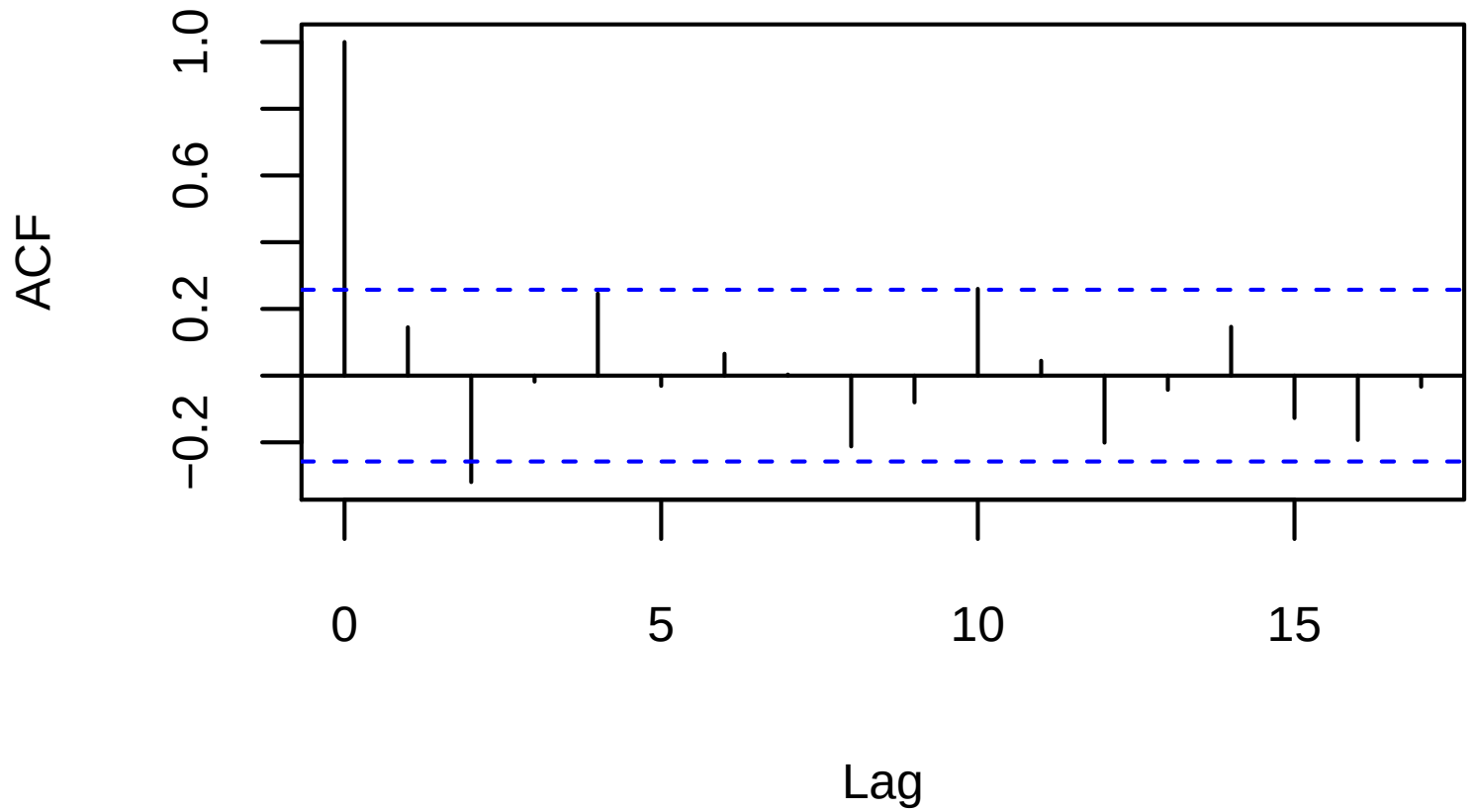
```
tempreg <- lm(weather$temp[2:59] ~ weather$temp[1:58])
```

Then look at the summary (see next page) . . . .The autoregressive term ($b_1 \approx 0.66$) is highly significant!

```
summary(tempreg)
>
> Call:
> lm(formula = weather$temp[2:59] ~ weather$temp[1:58])
>
> Residuals:
>      Min       1Q   Median       3Q      Max
> -17.0361  -4.8393  -0.5893   4.3457  17.4232
>
> Coefficients:
>                    Estimate Std. Error t value Pr(>|t|)
> (Intercept)         15.2420     4.5729   3.333  0.00153 **
> weather$temp[1:58]   0.6584     0.1007   6.541 1.98e-08 ***
> ---
> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> Residual standard error: 7.328 on 56 degrees of freedom
> Multiple R-squared:  0.4331,  Adjusted R-squared:  0.423
> F-statistic: 42.78 on 1 and 56 DF,  p-value: 1.98e-08
```

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

```
acf(tempreg$residuals, cex.lab=0.75, cex.axis=0.75, main="")
```



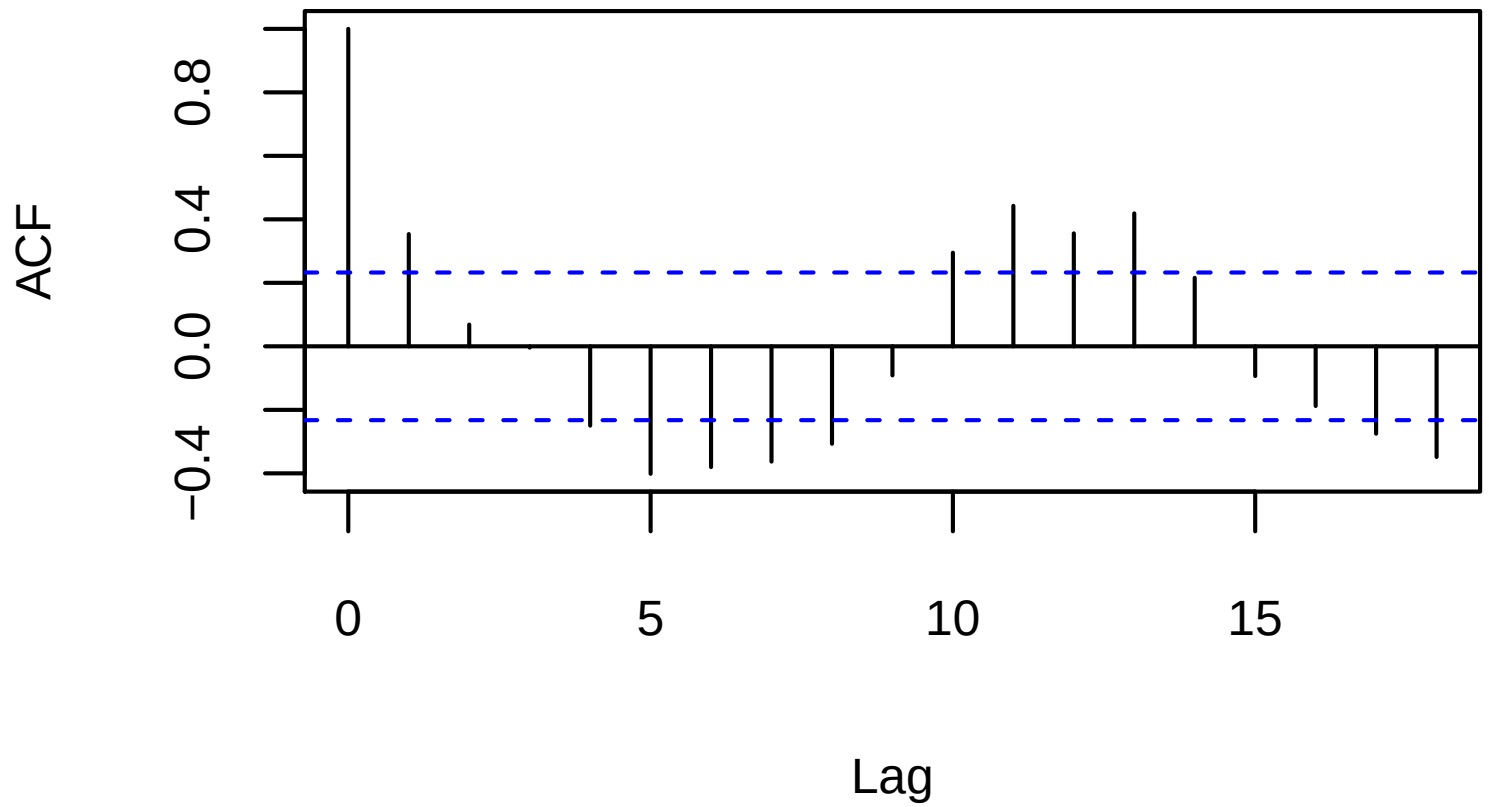
- Looks like we've got a good fit!

For the lung infection data, the AR term is also highly significant:

```
lungreg <- lm(ld[2:72] ~  ld[1:71]); summary(lungreg)
>
> Call:
> lm(formula = ld[2:72] ~ ld[1:71])
>
> Residuals:
>     Min      1Q  Median      3Q     Max
> -878.5 -251.3 -138.6  175.9 1297.7
>
> Coefficients:
>              Estimate Std. Error t value Pr(>|t|)
> (Intercept) 487.13140  162.00888   3.007  0.00368 **
> ld[1:71]      0.75571    0.07546  10.015 4.39e-15 ***
> ---
> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> Residual standard error: 387.6 on 69 degrees of freedom
> Multiple R-squared:  0.5924,  Adjusted R-squared:  0.5865
> F-statistic: 100.3 on 1 and 69 DF,  p-value: 4.39e-15
```

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

```
acf(lungreg$residuals, cex.lab=0.75, cex.axis=0.75, main="")
```



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

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

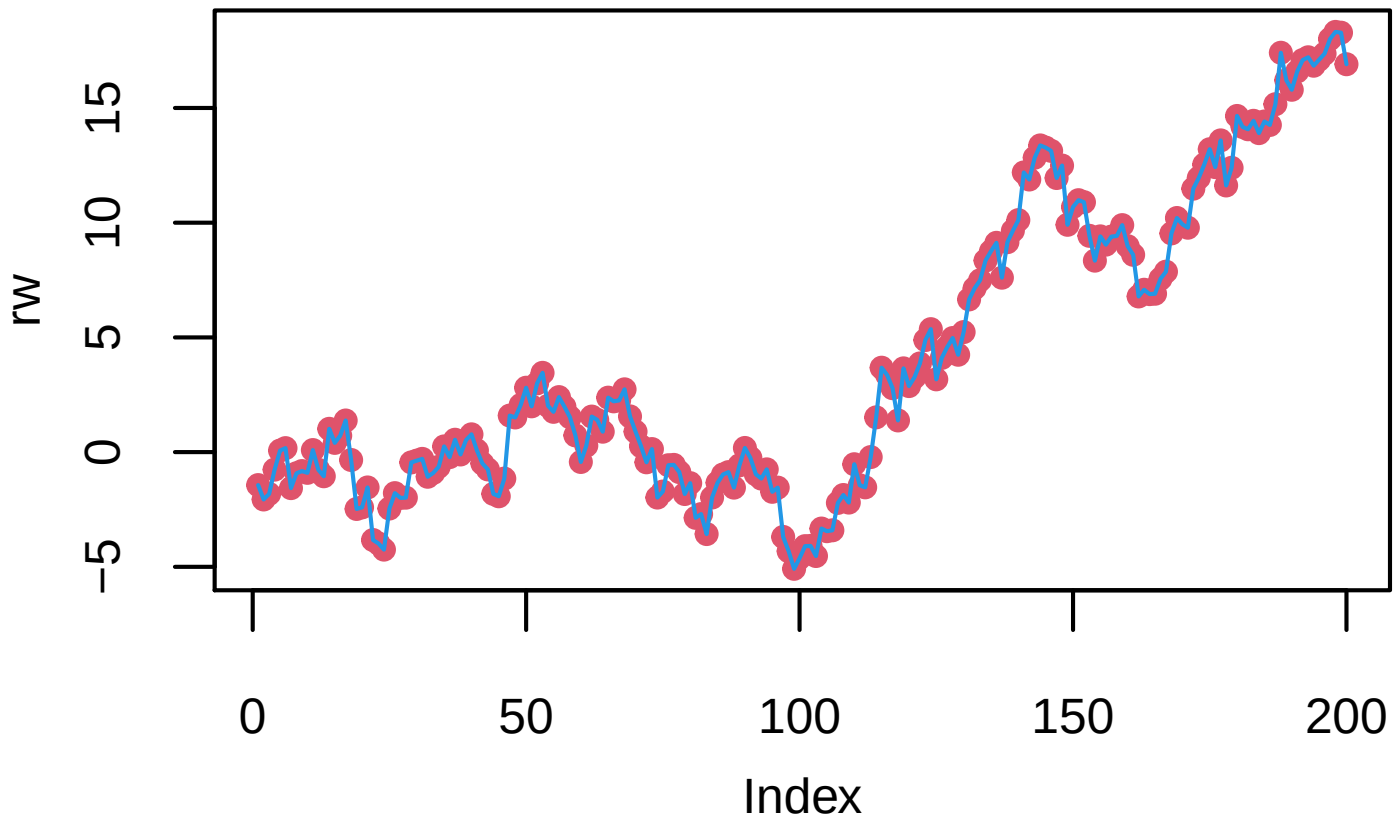
$$AR(1): Y_t = \beta_0 + \beta_1 Y_{t-1} + \varepsilon_t$$

The value of β_1 is key!

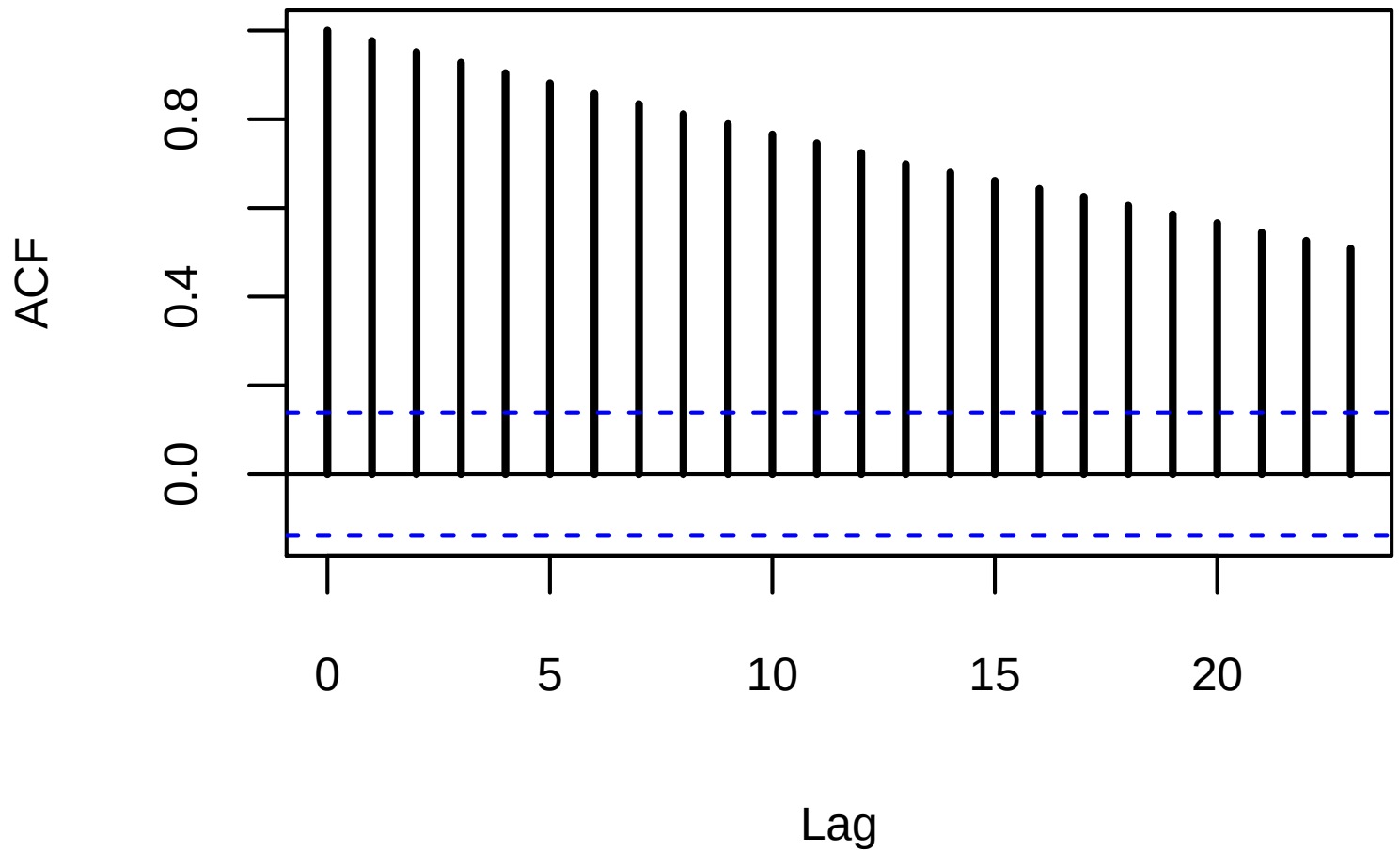
- If $|\beta_1| = 1$, we have a **random walk**.
- If $|\beta_1| > 1$, the series **explodes**.
- If $|\beta_1| < 1$, the values are **mean reverting**.

Random walk: $\beta_1 = 1$

In a random walk, the series just wanders around.



Autocorrelation of a random walk stays 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:

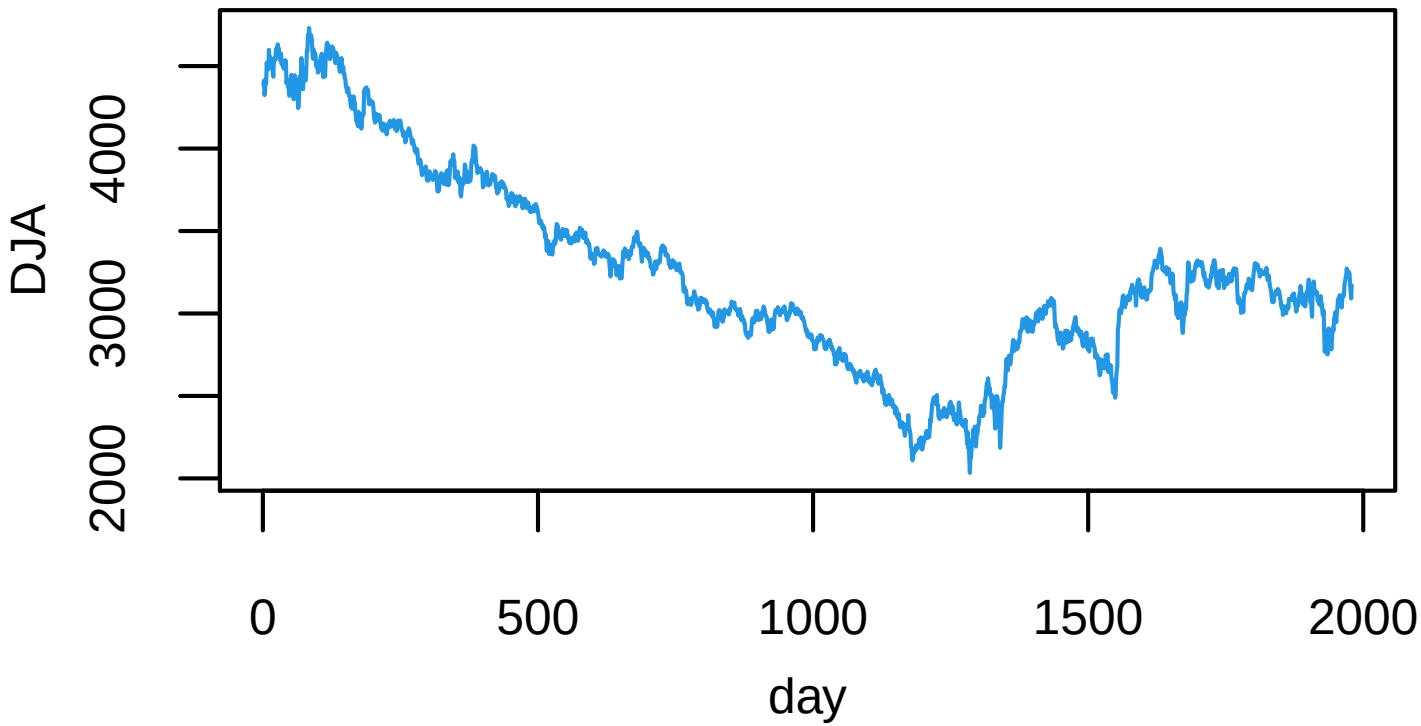
- it has no average level that it wants to be near, but rather just wanders off into space.

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

- since $\mathbb{E}[Y_t] = \mathbb{E}[Y_{t-1}]\}$, e.g., tomorrow $\approx$ today

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

```
dja <- read.csv("../data/dja.csv")$DJ
```



- Appears as though it is just wandering around.

Let's do our regression and check the output:

```
n <- length(dja)
ARdj <- lm(dja[2:n] ~ dja[1:(n-1)])
```

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

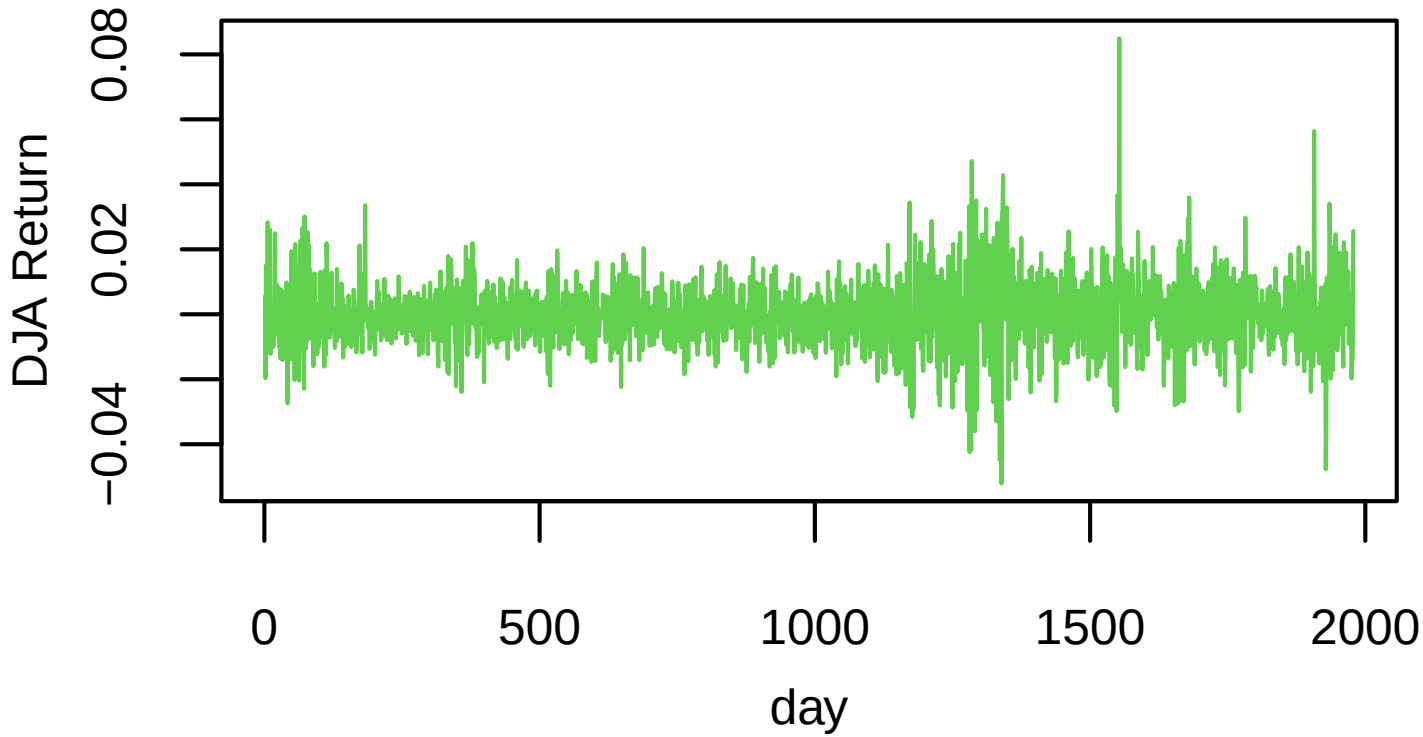
- ($b_0 > 0$, but not enough data to be sure of positive drift.)

(see summary, next page)

```
summary(ARdj)
>
> Call:
> lm(formula = dja[2:n] ~ dja[1:(n - 1)])
>
> Residuals:
>     Min      1Q  Median      3Q     Max
> -144.00  -18.55   -1.02   19.02  226.86
>
> 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
>
> Residual standard error: 32.13 on 1976 degrees of freedom
> Multiple R-squared:  0.9971,  Adjusted R-squared:  0.9971
> F-statistic: 6.795e+05 on 1 and 1976 DF,  p-value: < 2.2e-16
```

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

```
returns <- (dja[2:n]-dja[1:(n-1)])/dja[1:(n-1)]
```

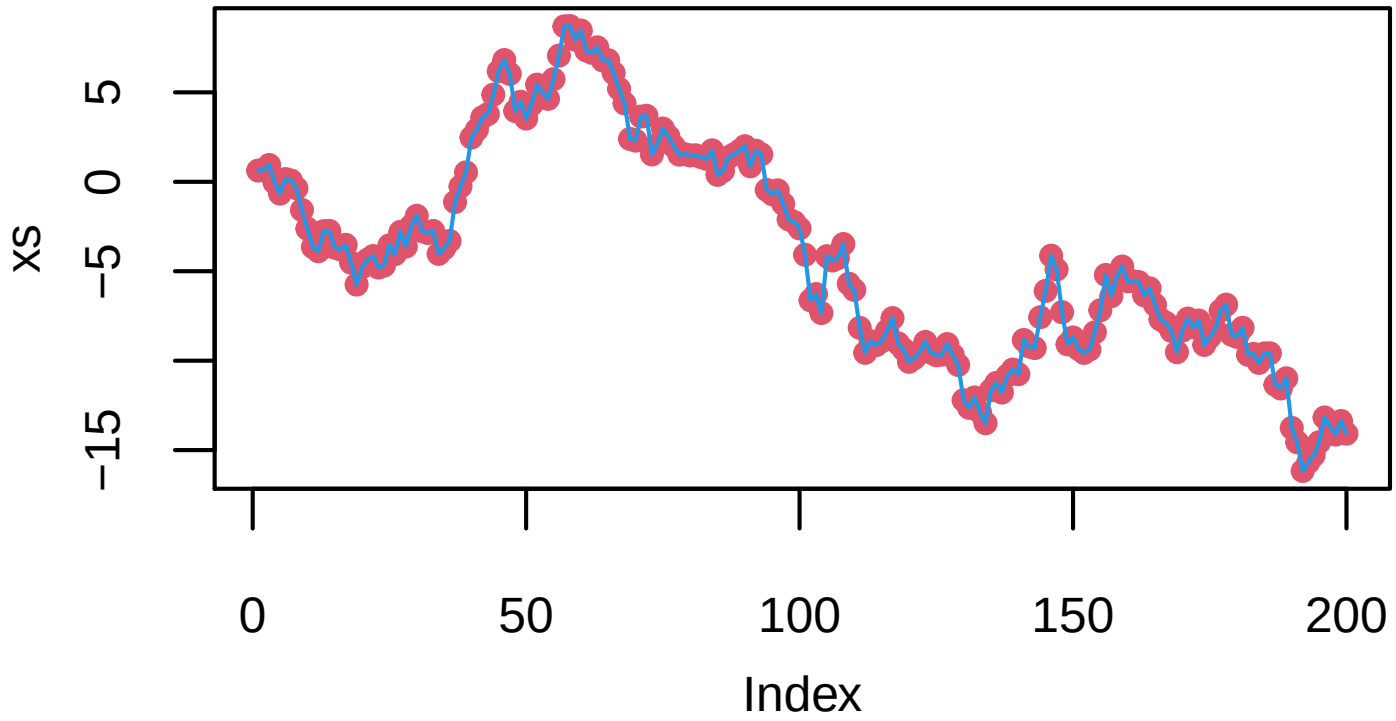


$(Y_t - Y_{t-1})/Y_{t-1}$ appears to remove the dependence, and now the regression model finds nothing significant. This is common with random walks $\Rightarrow$ $Y_t - Y_{t-1}$ is iid.

```
ret <- lm(returns[2:n] ~ returns[1:(n-1)]); summary(ret)
>
> Call:
> lm(formula = returns[2:n] ~ returns[1:(n - 1)])
>
> Residuals:
>       Min         1Q     Median         3Q        Max
> -0.052138 -0.005867 -0.000388  0.005707  0.085140
>
> 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:  0.0002079,   Adjusted R-squared:  -0.0002983
> F-statistic: 0.4108 on 1 and 1975 DF,  p-value: 0.5217
```

Exploding series $\beta_1 = 1.02$

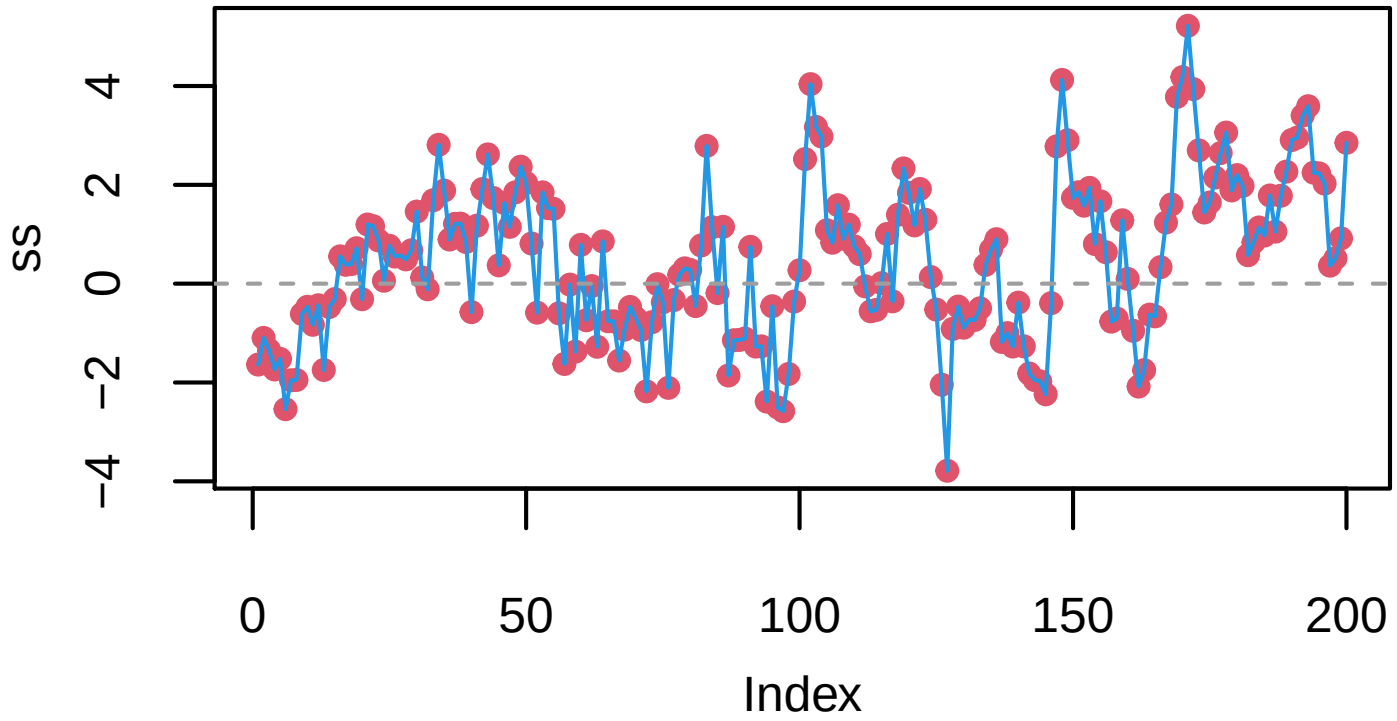
For AR term > 1, the Y_t's move exponentially far from Y_1.



- Useless for modeling and prediction.

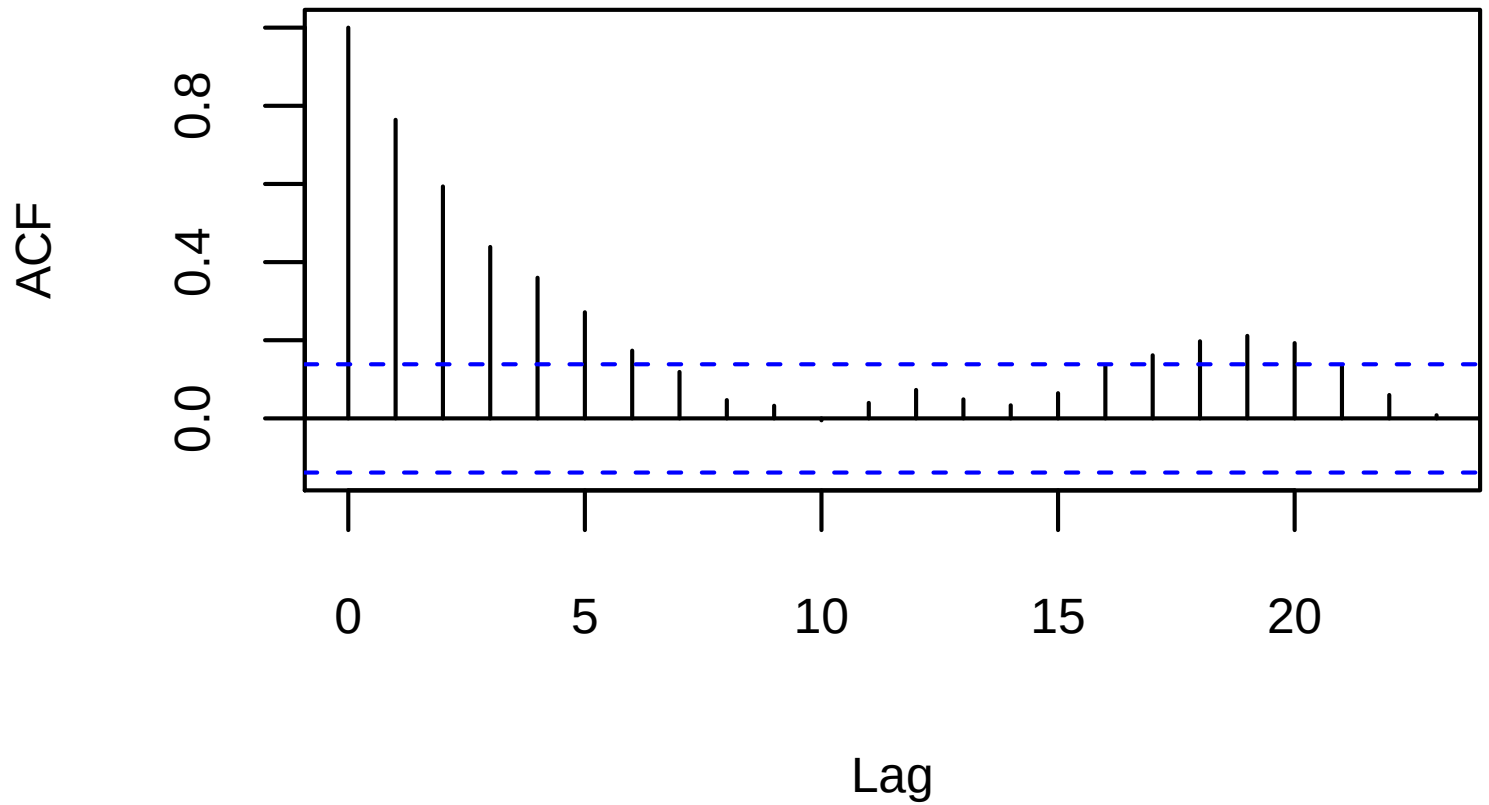
Stationary series: $\beta_1 = 0.8$

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



- These are the most common, and most useful, type of AR series.

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.

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

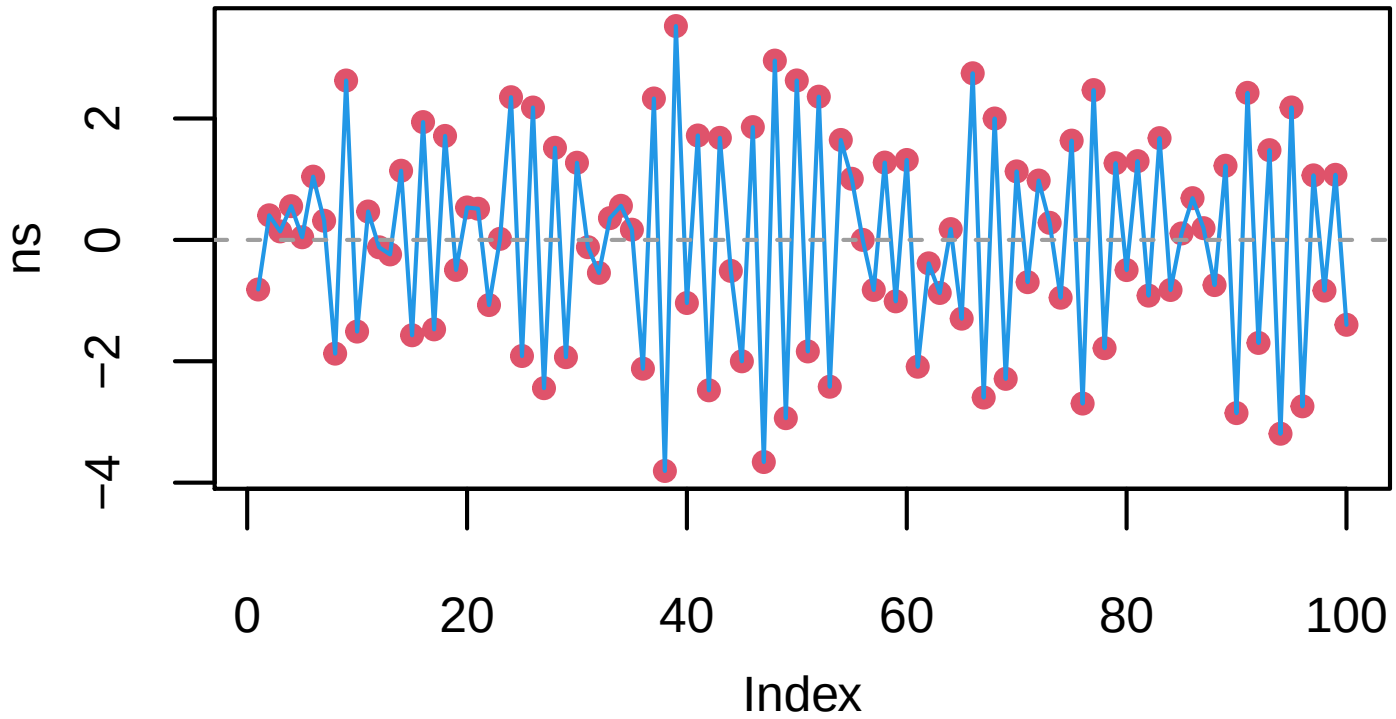
Since $|\beta_1| < 1$, Y_t is expected to be closer to μ than Y_{t-1}.

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

- weekly sales numbers,
- daily temperature.

Negative correlation: $\beta_1 = -0.8$

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 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.
- $|\beta_1| > 1$: The series explodes, is nonstationary, and pretty much useless.

AR(p) models

It is possible to expand the AR idea to higher lags

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

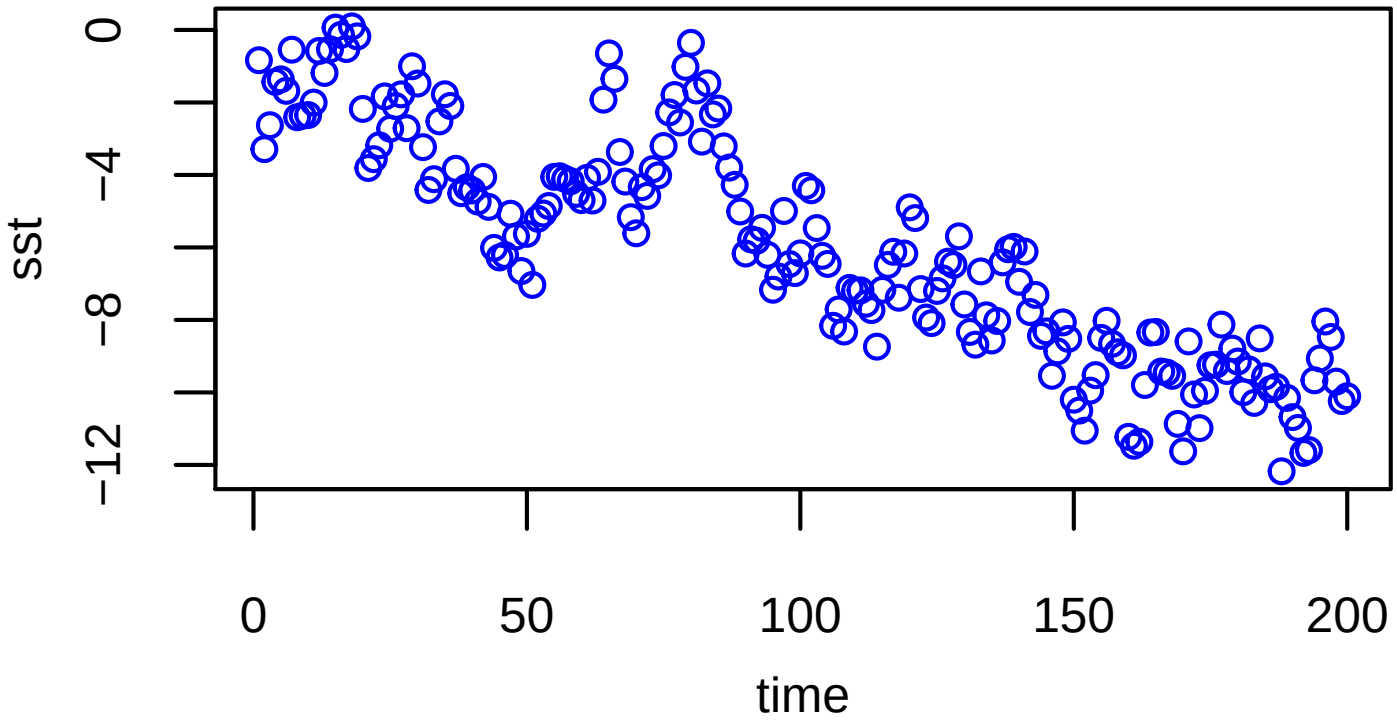
However, it is seldom necessary to fit AR lags for $p > 1$.

- Like having polynomial terms higher than 2, this just isn't usually required in practice.
- You lose all of the stationary/non-stationary intuition.
- 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)
```

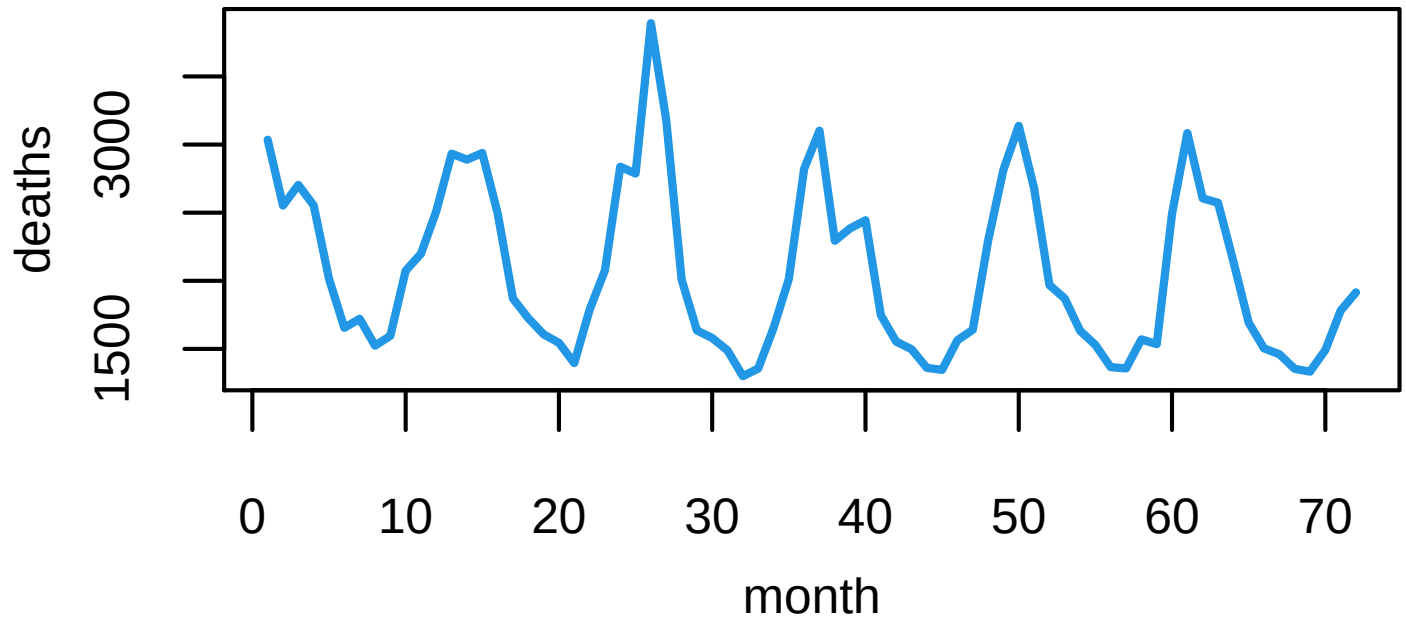
```
summary(sst.fit)  ## abbreviated output
>
> Call:
> lm(formula = sst[2:200] ~ sst[1:199] + t)
>
> Residuals:
>      Min       1Q   Median       3Q      Max
> -2.29473 -0.66175 -0.05942  0.61585  2.50262
>
> Coefficients:
>              Estimate Std. Error t value Pr(>|t|)
> (Intercept) -0.349424   0.145975  -2.394   0.0176 *
> sst[1:199]   0.757453   0.046663  16.232  < 2e-16 ***
> t           -0.011416   0.002515  -4.538 9.87e-06 ***
> ---
> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> Residual standard error: 0.9493 on 196 degrees of freedom
> Multiple R-squared:  0.907,   Adjusted R-squared:  0.906
> F-statistic: 955.6 on 2 and 196 DF,  p-value: < 2.2e-16
```

Periodic models

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

- Temperature goes up in Summer and down in Winter.
- Gas consumption (used for heating) would do the opposite.

Recall the monthly lung infection data:



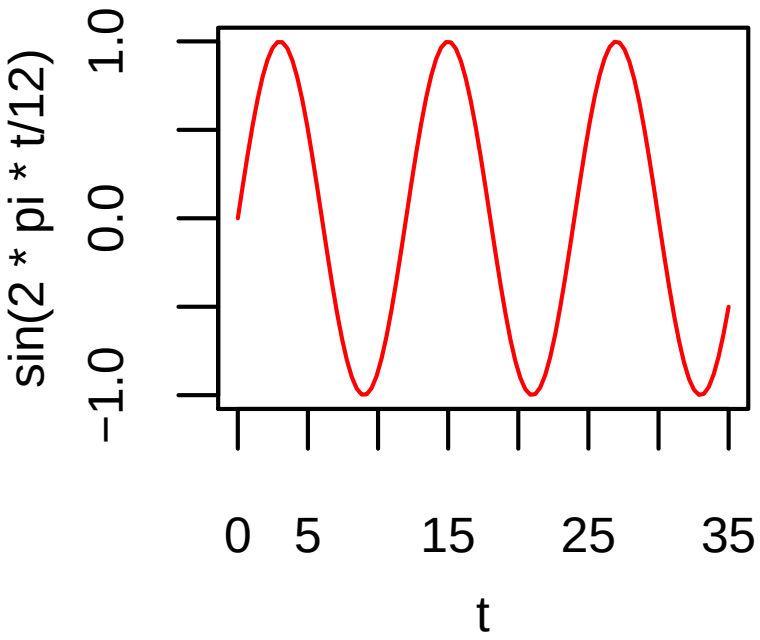
- Appears to oscillate on a 12-month cycle.

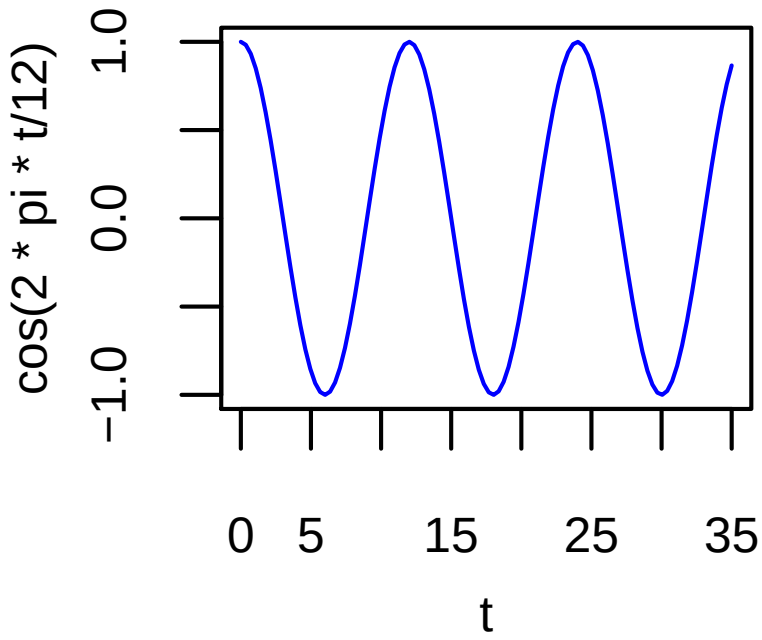
The straightforward solution: Add periodic predictors.

$\text{Period}-k$ 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 functionas 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.

Let's fit an AR with sine/cosine predictors:

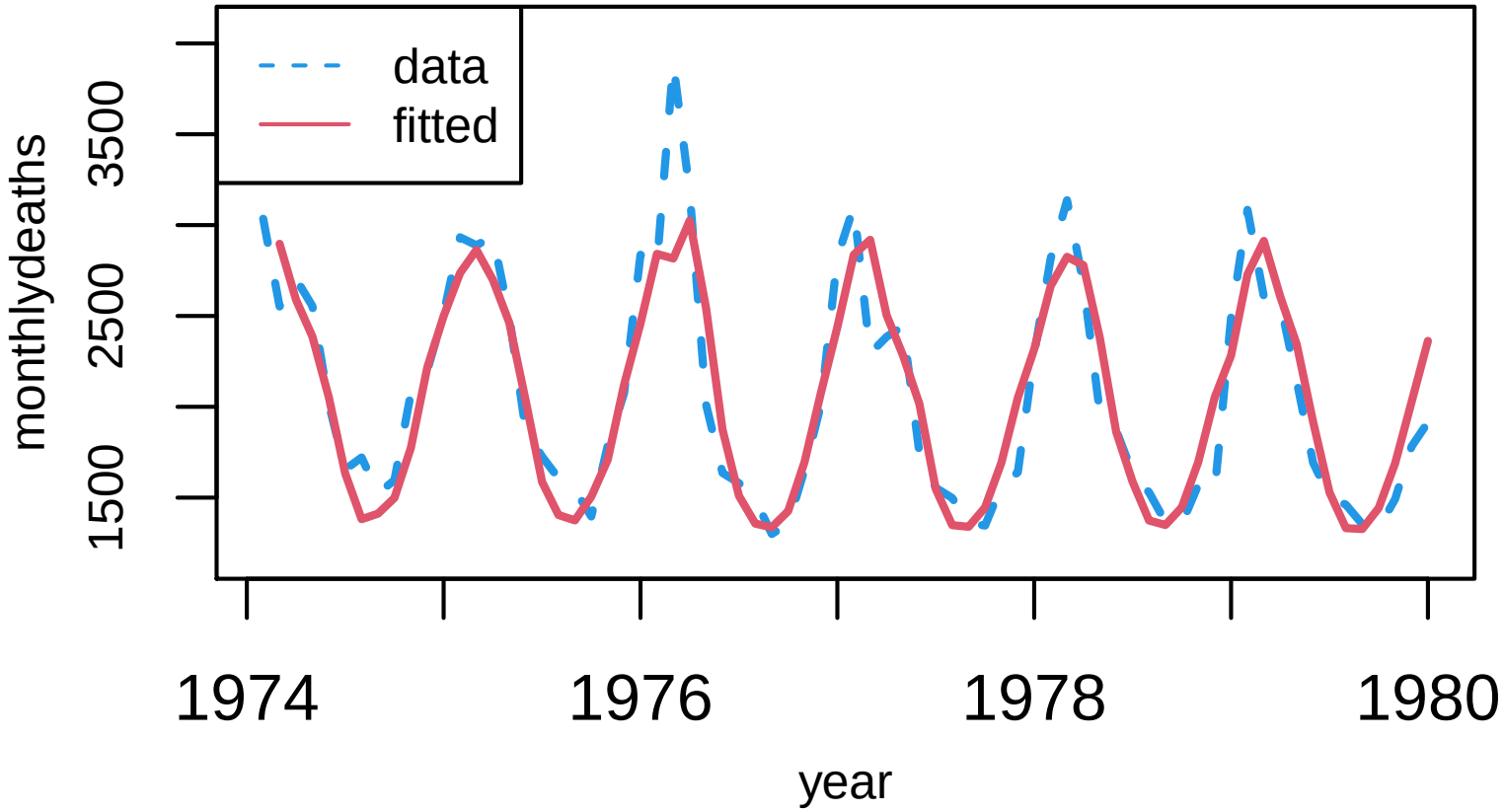
$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 \sin(2\pi t/k) + \beta_3 \cos(2\pi t/k) + \varepsilon_t$$

We want to make new predictors/dataframe, much like when we add polynomial terms, and then fit.

```
tmax<-length(ld)
t <- 2:tmax
YX <- data.frame(ld=ld[2:tmax], ldpast=ld[1:(tmax-1)], t=t,
                 sin12=sin(2*pi*t/12), cos12=cos(2*pi*t/12))
lunglm <- lm(ld ~ ldpast + sin12 + cos12, data=YX)
```

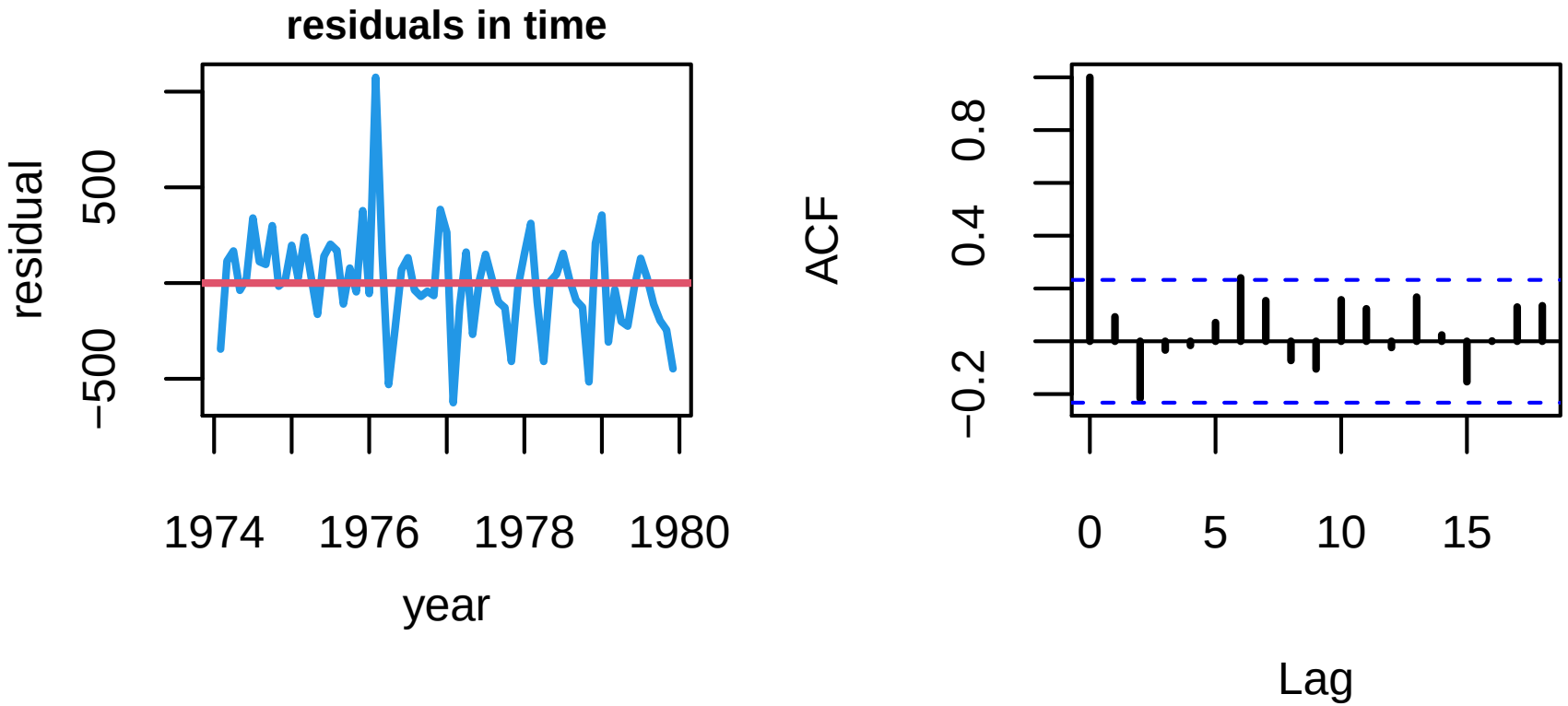
```
summary(lunglm) ## abbreviated output
>
> Call:
> lm(formula = ld ~ ldpast + sin12 + cos12, data = YX)
>
> Residuals:
>     Min      1Q  Median      3Q     Max
> -624.89 -114.94    8.84  150.13 1074.79
>
> Coefficients:
>             Estimate Std. Error t value Pr(>|t|)
> (Intercept) 1379.3598   247.5536   5.572 4.84e-07 ***
> ldpast         0.3260     0.1189   2.741 0.007853 **
> sin12        377.5492   101.9206   3.704 0.000431 ***
> cos12        402.7845    44.8015   8.990 3.98e-13 ***
> ---
> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> Residual standard error: 260.9 on 67 degrees of freedom
> Multiple R-squared:  0.8207,  Adjusted R-squared:  0.8127
> F-statistic: 102.2 on 3 and 67 DF,  p-value: < 2.2e-16
```

The model predictions look pretty good!



- Sine and cosine trends seem to capture the periodicity.

The residuals look pretty good.



- Maybe a bit of downward linear trend?

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 2.

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 Sum-Up

As with other regression approaches, there are many possible models; you can use BIC with $\texttt{extractAIC}(\texttt{reg}, \texttt{k} = \texttt{log}(\texttt{n}))$ to compare and choose.

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.
- These approaches only work for data that are evenly spaced with no gaps

Practice: In the Roanoke Airport weather dataset, I used the average temperature (`TAVG`), but there are additional columns.

For one of either `TMIN` or `TMAX`

- Plot the data from Feb and March.
- Plot the correlation between temperature at time t and $t-1$ and calculate the correlation.
- Plot the ACF of the time series.
- Fit a simple autoregressive model to the TS data. What is your coefficient and what do you conclude about the kind of behavior that your TS exhibits?

Practice: Monthly Sunspot Data.

Data on the mean number of sunspots in each month from 1749 to 2013 are available in R as `sunspot.month`.

- Plot the sunspot data. What do you observe? Do you think you would want to take any transformations of the data?
- Plot the correlation between the number of sunspots at time t and $t-1$ and calculate the correlation.
- Plot the ACF of the time series (note – you'll need to make the max lag very large).
- Based on the previous parts, propose a simple AR model plus a periodic and trend component. Check your residuals and your predictions. How well did you do? If you have trouble fitting, try slightly changing the frequency of your periodic component.

Next Steps

In the practical for this component we'll step you through fitting a variety of models with trending, seasonal, and AR components for a classic time series data set. You'll then get to try it yourself on dengue case data.