MONASH University

MONASH BUSINESS SCHOOL

ETC3550: Applied forecasting for business and economics

Ch2. Time series graphics

OTexts.org/fpp2/

Outline

`ts` objects and `ts` function

A time series is stored in a `ts` object in R:

- a list of numbers
- information about times those numbers were recorded.

Example

Year	Observation
2012	123
2013	39
2014	78
2015	52
2016	110

```
y <- ts(c(123,39,78,52,110), start=2012)
```

`ts` objects and `ts` function

For observations that are more frequent than once per year, add a `frequency` argument.
E.g., monthly data stored as a numerical vector `z`:

```
y <- ts(z, frequency=12, start=c(2003, 1))
```

`ts` objects and `ts` function

`ts(data, frequency, start)`

Type of data	frequency	start example
Annual		
Quarterly		
Monthly		
Daily		
Weekly		
Hourly		
Half-hourly		

`ts` objects and `ts` function

`ts(data, frequency, start)`

Type of data	frequency	start example
Annual	1	
Quarterly		
Monthly		
Daily		
Weekly		
Hourly		
Half-hourly		

`ts` objects and `ts` function

`ts(data, frequency, start)`

Type of data	frequency	start example
Annual	1	1995
Quarterly		
Monthly		
Daily		
Weekly		
Hourly		
Half-hourly		

`ts` objects and `ts` function

`ts(data, frequency, start)`

Type of data	frequency	start example
Annual	1	1995
Quarterly	4	
Monthly		
Daily		
Weekly		
Hourly		
Half-hourly		

`ts` objects and `ts` function

`ts(data, frequency, start)`

Type of data	frequency	start example
Annual	1	1995
Quarterly	4	c(1995,2)
Monthly		
Daily		
Weekly		
Hourly		
Half-hourly		

`ts` objects and `ts` function

`ts(data, frequency, start)`

Type of data	frequency	start example
Annual	1	1995
Quarterly	4	c(1995,2)
Monthly	12	
Daily		
Weekly		
Hourly		
Half-hourly		

`ts` objects and `ts` function

`ts(data, frequency, start)`

Type of data	frequency	start example
Annual	1	1995
Quarterly	4	c(1995,2)
Monthly	12	c(1995,9)
Daily		
Weekly		
Hourly		
Half-hourly		

`ts` objects and `ts` function

`ts(data, frequency, start)`

Type of data	frequency	start example
Annual	1	1995
Quarterly	4	c(1995,2)
Monthly	12	c(1995,9)
Daily	*7 or 365.25*	
Weekly		
Hourly		
Half-hourly		

`ts` objects and `ts` function

`ts(data, frequency, start)`

Type of data	frequency	start example
Annual	1	1995
Quarterly	4	c(1995,2)
Monthly	12	c(1995,9)
Daily	7 *or* 365.25	1 *or* c(1995,234)
Weekly		
Hourly		
Half-hourly		

`ts` objects and `ts` function

`ts(data, frequency, start)`

Type of data	frequency	start example
Annual	1	1995
Quarterly	4	c(1995,2)
Monthly	12	c(1995,9)
Daily	7 *or* 365.25	1 *or* c(1995,234)
Weekly	52.18	
Hourly		
Half-hourly		

`ts` objects and `ts` function

```
ts(data, frequency, start)
```

Type of data	frequency	start example
Annual	1	1995
Quarterly	4	c(1995,2)
Monthly	12	c(1995,9)
Daily	7 *or* 365.25	1 *or* c(1995,234)
Weekly	52.18	c(1995,23)
Hourly		
Half-hourly		

`ts` objects and `ts` function

`ts(data, frequency, start)`

Type of data	frequency	start example
Annual	1	1995
Quarterly	4	c(1995,2)
Monthly	12	c(1995,9)
Daily	7 *or* 365.25	1 *or* c(1995,234)
Weekly	52.18	c(1995,23)
Hourly	24 *or* 168 *or* 8,766	
Half-hourly		

`ts` objects and `ts` function

`ts(data, frequency, start)`

Type of data	frequency	start example
Annual	1	1995
Quarterly	4	c(1995,2)
Monthly	12	c(1995,9)
Daily	7 *or* 365.25	1 *or* c(1995,234)
Weekly	52.18	c(1995,23)
Hourly	24 *or* 168 *or* 8,766	1
Half-hourly		

`ts` objects and `ts` function

`ts(data, frequency, start)`

Type of data	frequency	start example
Annual	1	1995
Quarterly	4	c(1995,2)
Monthly	12	c(1995,9)
Daily	7 *or* 365.25	1 *or* c(1995,234)
Weekly	52.18	c(1995,23)
Hourly	24 *or* 168 *or* 8,766	1
Half-hourly	48 *or* 336 *or* 17,532	

`ts` objects and `ts` function

`ts(data, frequency, start)`

Type of data	frequency	start example
Annual	1	1995
Quarterly	4	c(1995,2)
Monthly	12	c(1995,9)
Daily	7 *or* 365.25	1 *or* c(1995,234)
Weekly	52.18	c(1995,23)
Hourly	24 *or* 168 *or* 8,766	1
Half-hourly	48 *or* 336 *or* 17,532	1

Australian GDP

```
ausgdp <- ts(x, frequency=4, start=c(1971,3))
```

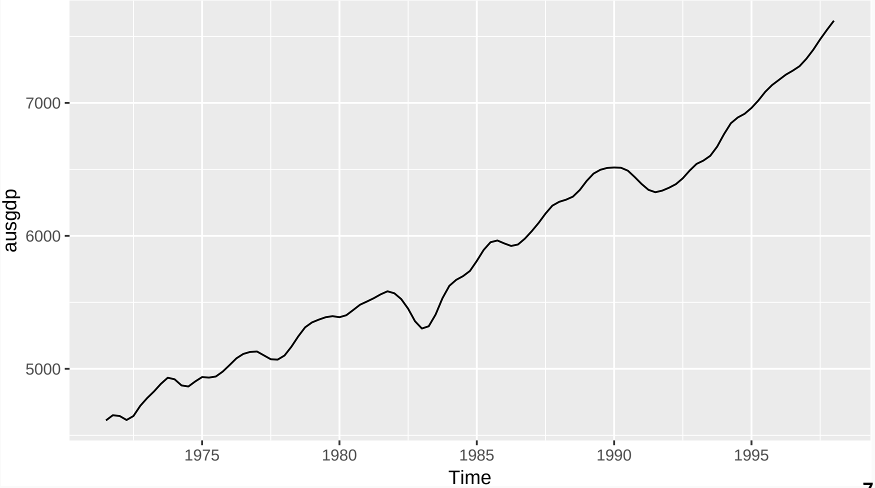
- Class: "ts"
- Print and plotting methods available.

```
ausgdp
```

```
##      Qtr1 Qtr2 Qtr3 Qtr4
## 1971           4612 4651
## 1972 4645 4615 4645 4722
## 1973 4780 4830 4887 4933
## 1974 4921 4875 4867 4905
## 1975 4938 4934 4942 4979
## 1976 5028 5079 5112 5127
## 1977 5130 5101 5072 5069
## 1978 5100 5166 5244 5312
## 1979 5349 5370 5388 5396
```

Australian GDP

```r
autoplot(ausgdp)
```



Residential electricity sales

```
elecsales
```

```
## Time Series:
## Start = 1989
## End = 2008
## Frequency = 1
##  [1] 2354.34 2379.71 2318.52 2468.99 2386.09
##  [6] 2569.47 2575.72 2762.72 2844.50 3000.70
## [11] 3108.10 3357.50 3075.70 3180.60 3221.60
## [16] 3176.20 3430.60 3527.48 3637.89 3655.00
```

Class package

```
> library(fpp2)
```

Class package

```
> library(fpp2)
```

This loads:

- some data for use in examples and exercises

Class package

```
> library(fpp2)
```

This loads:

- some data for use in examples and exercises
- **forecast** package (for forecasting functions)
- **ggplot2** package (for graphics functions)
- **fma** package (for lots of time series data)
- **expsmooth** package (for more time series data)

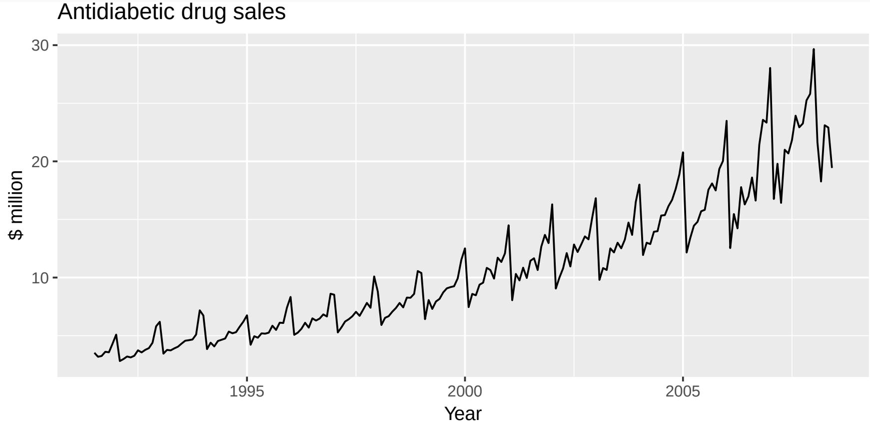
Outline

Time plots

```r
autoplot(melsyd[,"Economy.Class"])
```

Time plots

```
autoplot(a10) + ylab("$ million") + xlab("Year") +
        ggtitle("Antidiabetic drug sales")
```

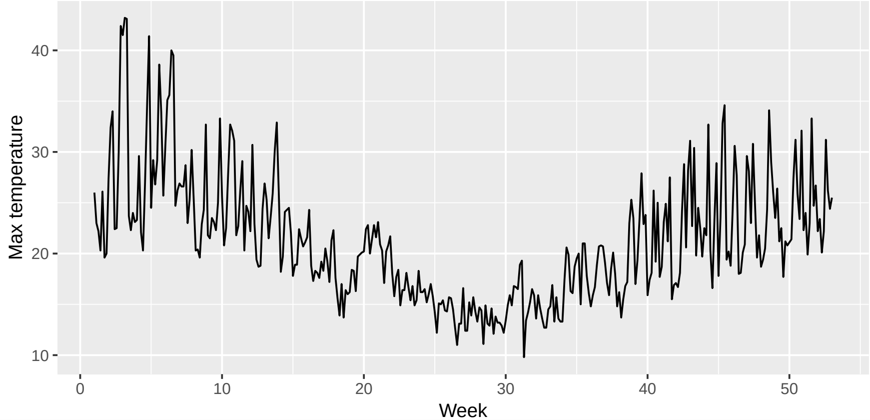


Your turn

- Create plots of the following time series: `dole`, `bricksq`, `lynx`, `goog`
- Use `help()` to find out about the data in each series.
- For the last plot, modify the axis labels and title.

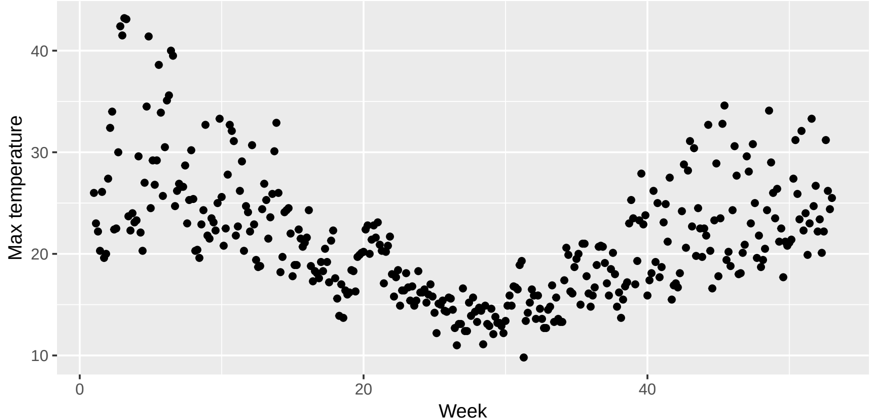
Are time plots best?

```r
autoplot(elecdaily[,"Temperature"]) +
  xlab("Week") + ylab("Max temperature")
```

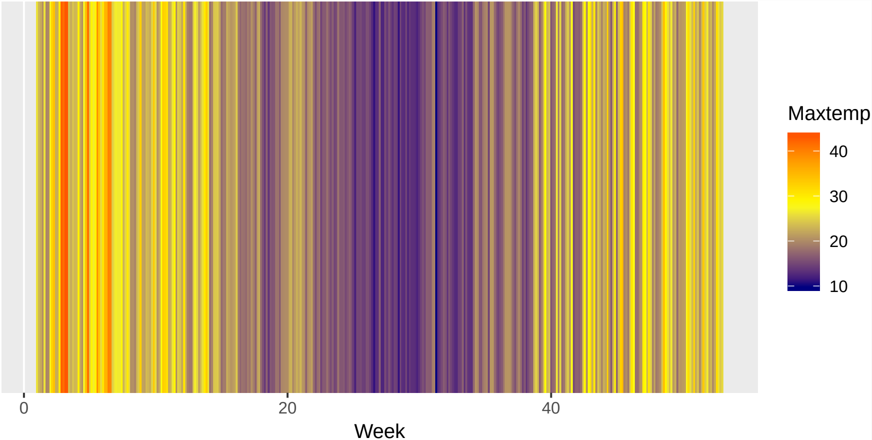


Are time plots best?

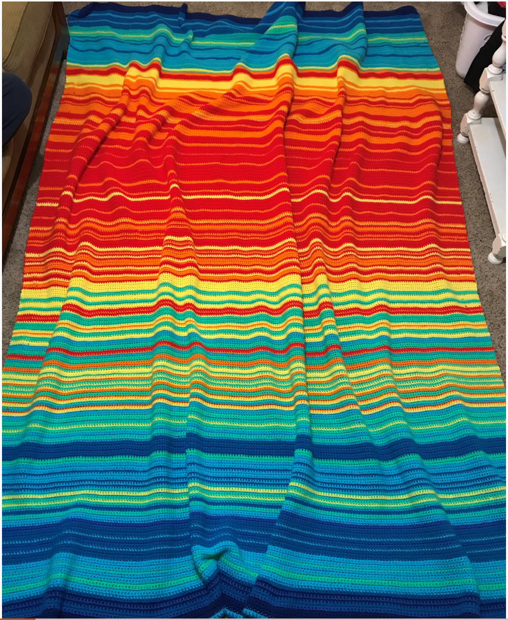
```
qplot(time(elecdaily), elecdaily[,"Temperature"]) +
  xlab("Week") + ylab("Max temperature")
```



Are time plots best?



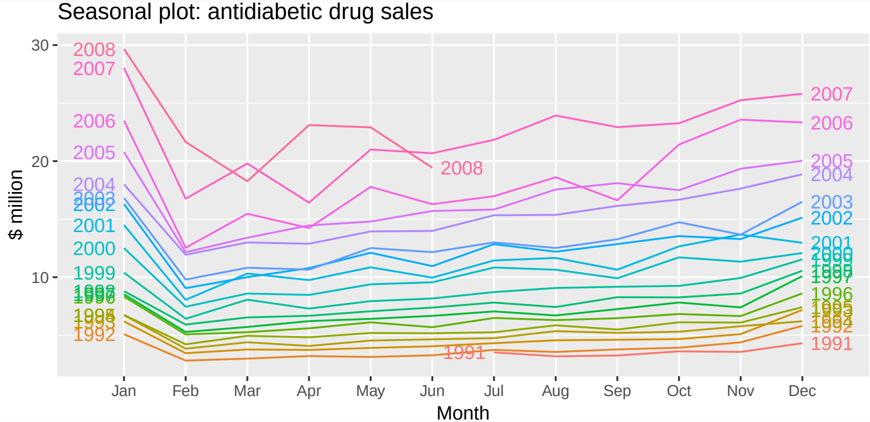
Are time plots best?



Outline

Seasonal plots

```
ggseasonplot(a10, year.labels=TRUE, year.labels.left=TRUE) +
  ylab("$ million") +
  ggtitle("Seasonal plot: antidiabetic drug sales")
```

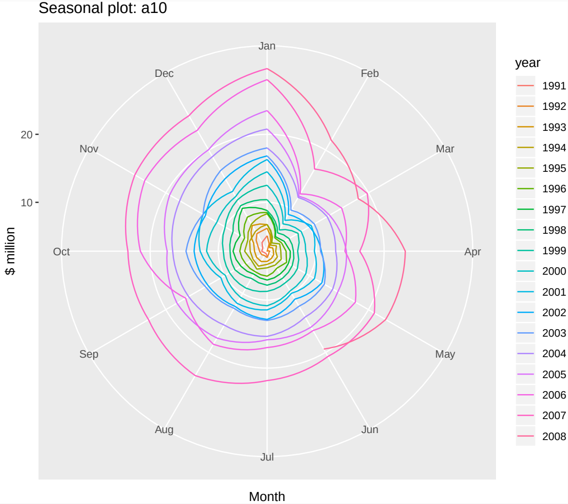


Seasonal plots

- Data plotted against the individual “seasons” in which the data were observed. (In this case a “season” is a month.)
- Something like a time plot except that the data from each season are overlapped.
- Enables the underlying seasonal pattern to be seen more clearly, and also allows any substantial departures from the seasonal pattern to be easily identified.
- In R: `ggseasonplot()`

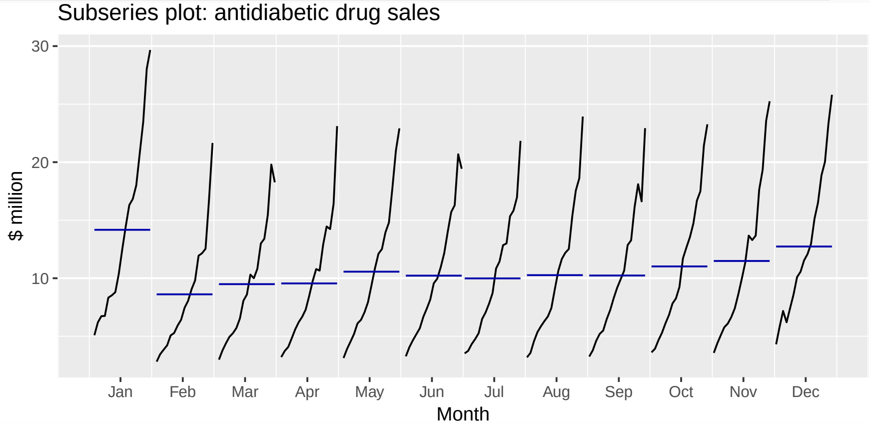
Seasonal polar plots

```
ggseasonplot(a10, polar=TRUE) + ylab("$ million")
```



Seasonal subseries plots

```
ggsubseriesplot(a10) + ylab("$ million") +
  ggtitle("Subseries plot: antidiabetic drug sales")
```

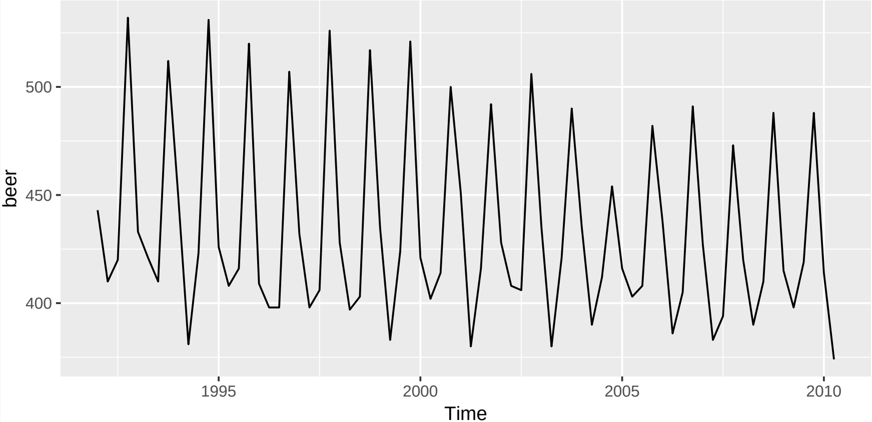


Seasonal subseries plots

- Data for each season collected together in time plot as separate time series.
- Enables the underlying seasonal pattern to be seen clearly, and changes in seasonality over time to be visualized.
- In R: `ggsubseriesplot()`

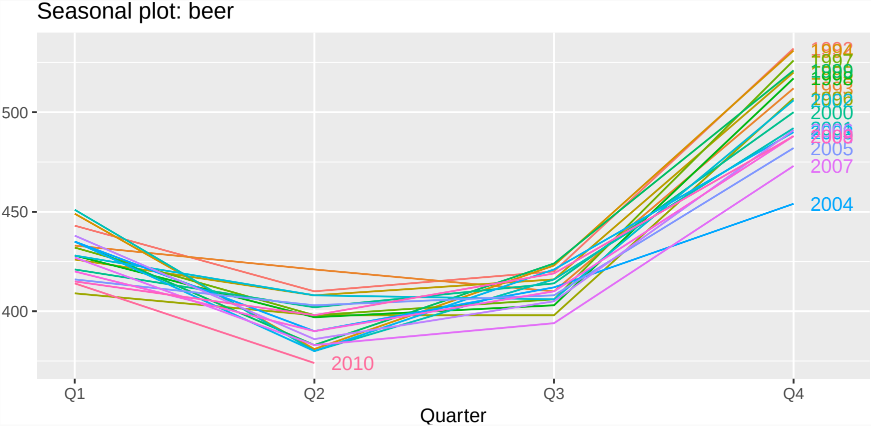
Quarterly Australian Beer Production

```r
beer <- window(ausbeer,start=1992)
autoplot(beer)
```



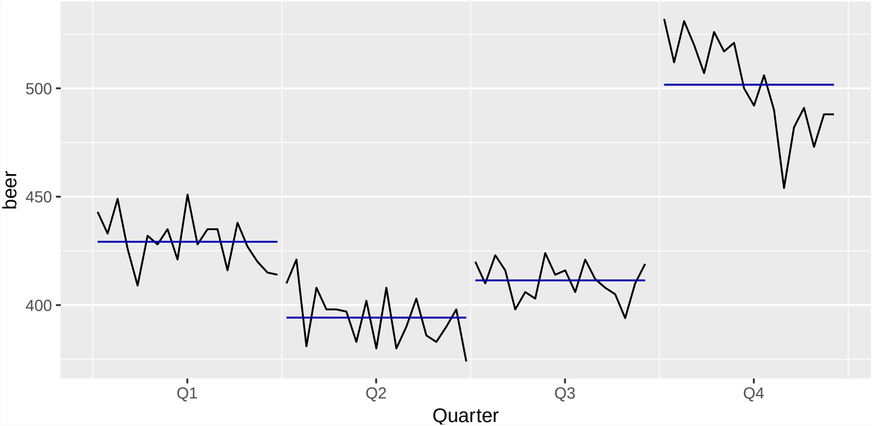
Quarterly Australian Beer Production

```r
ggseasonplot(beer,year.labels=TRUE)
```



Quarterly Australian Beer Production

```
ggsubseriesplot(beer)
```



Your turn

The `arrivals` data set comprises quarterly international arrivals (in thousands) to Australia from Japan, New Zealand, UK and the US.

- Use `autoplot()` and `ggseasonplot()` to compare the differences between the arrivals from these four countries.
- Can you identify any unusual observations?

Outline

1. Time series in R
2. Time plots
3. Seasonal plots
4. **Seasonal or cyclic?**
5. Lag plots and autocorrelation
6. White noise

Time series patterns

Trend pattern exists when there is a long-term increase or decrease in the data.

Seasonal pattern exists when a series is influenced by seasonal factors (e.g., the quarter of the year, the month, or day of the week).

Cyclic pattern exists when data exhibit rises and falls that are *not of fixed period* (duration usually of at least 2 years).

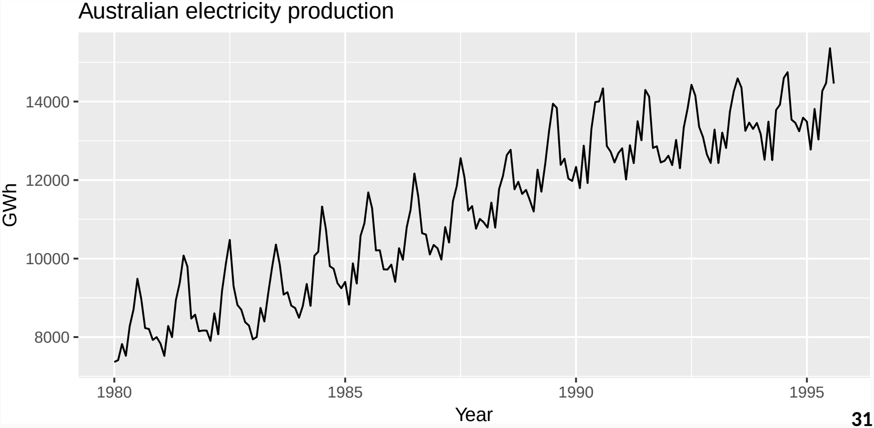
Time series components

Differences between seasonal and cyclic patterns:

- seasonal pattern constant length; cyclic pattern variable length
- average length of cycle longer than length of seasonal pattern
- magnitude of cycle more variable than magnitude of seasonal pattern

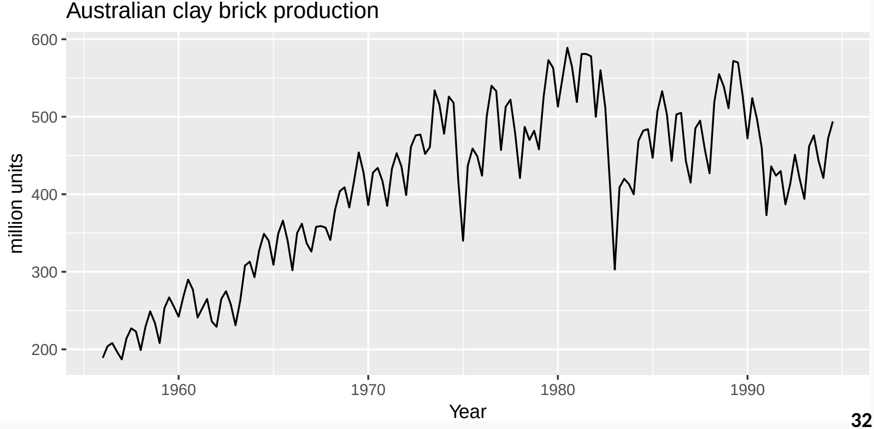
Time series patterns

```r
autoplot(window(elec, start=1980)) +
  ggtitle("Australian electricity production") +
  xlab("Year") + ylab("GWh")
```



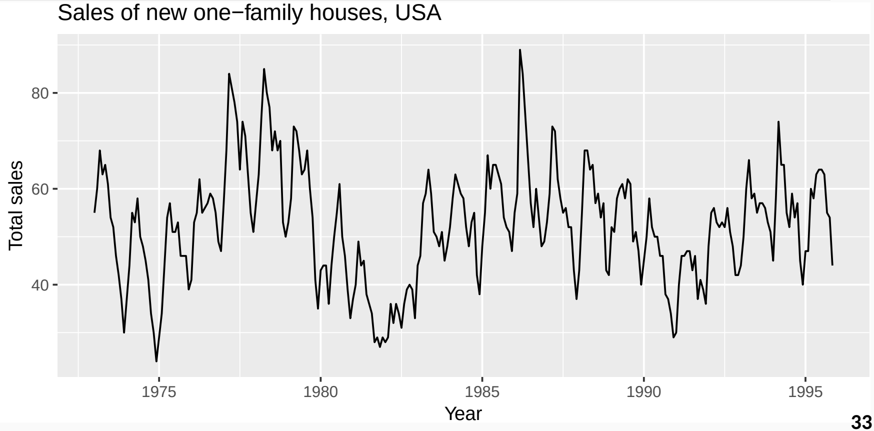
Time series patterns

```r
autoplot(bricksq) +
  ggtitle("Australian clay brick production") +
  xlab("Year") + ylab("million units")
```



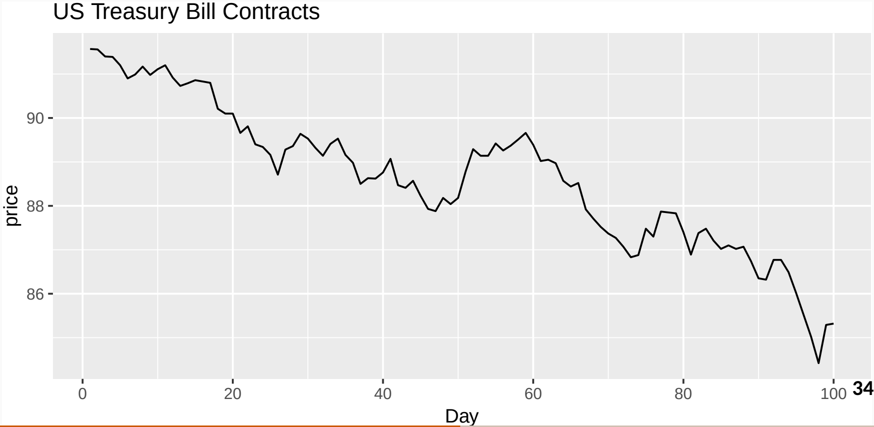
Time series patterns

```r
autoplot(hsales) +
  ggtitle("Sales of new one-family houses, USA") +
  xlab("Year") + ylab("Total sales")
```



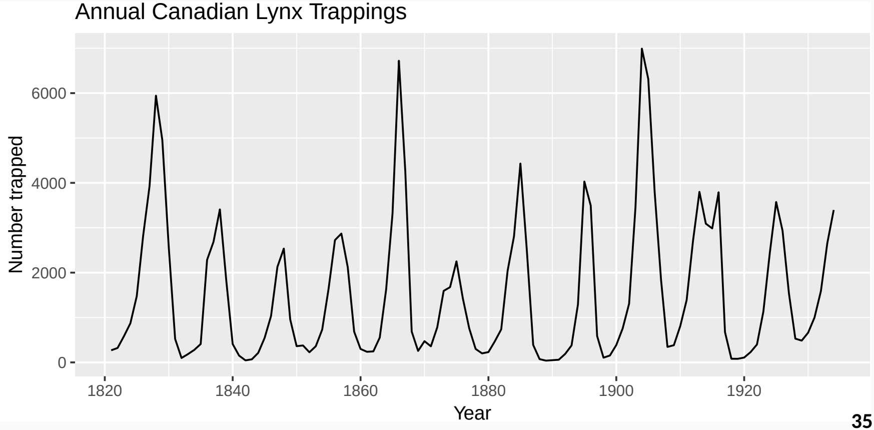
Time series patterns

```
autoplot(ustreas) +
  ggtitle("US Treasury Bill Contracts") +
  xlab("Day") + ylab("price")
```



Time series patterns

```r
autoplot(lynx) +
  ggtitle("Annual Canadian Lynx Trappings") +
  xlab("Year") + ylab("Number trapped")
```



Seasonal or cyclic?

Differences between seasonal and cyclic patterns:

- seasonal pattern constant length; cyclic pattern variable length
- average length of cycle longer than length of seasonal pattern
- magnitude of cycle more variable than magnitude of seasonal pattern

Seasonal or cyclic?

Differences between seasonal and cyclic patterns:

- seasonal pattern constant length; cyclic pattern variable length
- average length of cycle longer than length of seasonal pattern
- magnitude of cycle more variable than magnitude of seasonal pattern

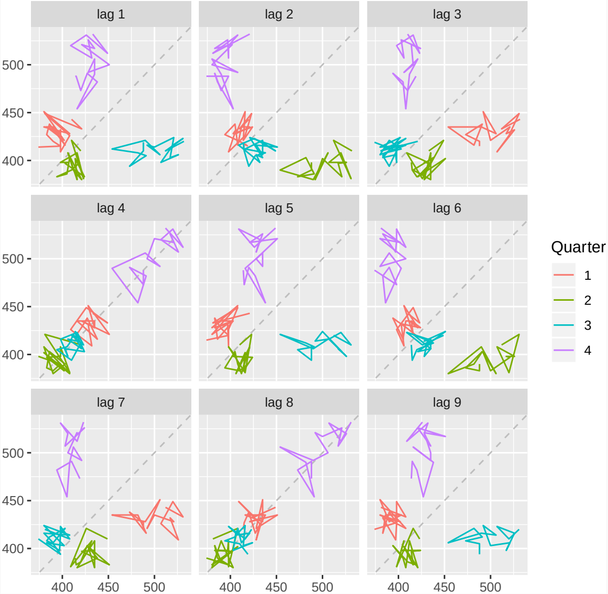
The timing of peaks and troughs is predictable with seasonal data, but unpredictable in the long term with cyclic data.

Outline

Example: Beer production

```
beer <- window(ausbeer, start=1992)
gglagplot(beer)
```

Example: Beer production



Lagged scatterplots

- Each graph shows y_t plotted against y_{t-k} for different values of k.
- The autocorrelations are the correlations associated with these scatterplots.

Autocorrelation

Covariance and **correlation**: measure extent of **linear relationship** between two variables (y and X).

Autocorrelation

Covariance and **correlation**: measure extent of **linear relationship** between two variables (y and X).

Autocovariance and **autocorrelation**: measure linear relationship between **lagged values** of a time series y.

Autocorrelation

Covariance and **correlation**: measure extent of **linear relationship** between two variables (y and X).

Autocovariance and **autocorrelation**: measure linear relationship between **lagged values** of a time series y.

\begin{tabbing} We measure the relationship between:~~= y_t and y_{t-1}\ > y_t and y_{t-2}\ > y_t and y_{t-3}\ > etc. \end{tabbing}

Autocorrelation

We denote the sample autocovariance at lag k by c_k and the sample autocorrelation at lag k by r_k. Then define

$$c_k = \frac{1}{T}\sum_{t=k+1}^{T}(y_t - \bar{y})(y_{t-k} - \bar{y})$$

$$\text{and} \qquad r_k = c_k / c_0$$

Autocorrelation

We denote the sample autocovariance at lag k by c_k and the sample autocorrelation at lag k by r_k. Then define

$$c_k = \frac{1}{T}\sum_{t=k+1}^{T}(y_t - \bar{y})(y_{t-k} - \bar{y})$$

$$\text{and} \qquad r_k = c_k/c_0$$

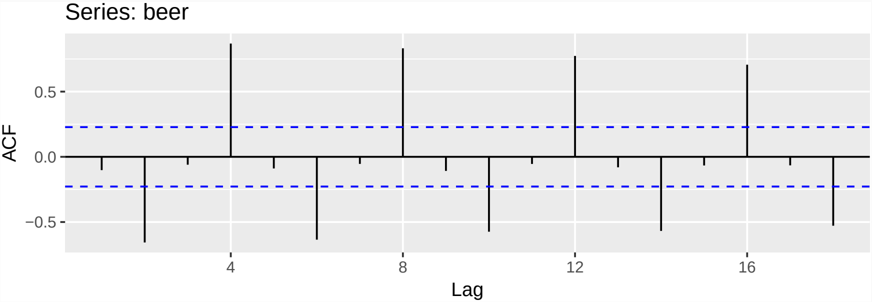
- r_1 indicates how successive values of y relate to each other
- r_2 indicates how y values two periods apart relate to each other
- r_k is *almost* the same as the sample correlation between y_t and y_{t-k}.

Autocorrelation

Results for first 9 lags for beer data:

r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9
-0.102	-0.657	-0.060	0.869	-0.089	-0.635	-0.054	0.832	-0.108

```
ggAcf(beer)
```

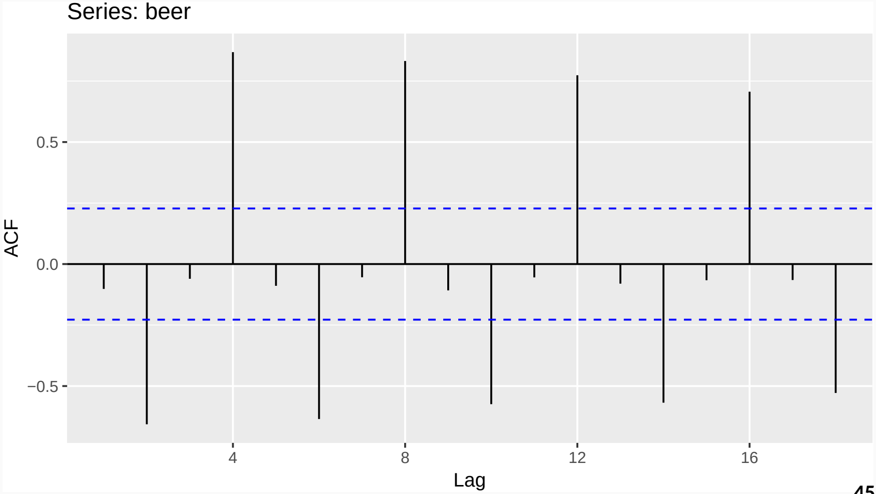


Autocorrelation

- r_4 higher than for the other lags. This is due to **the seasonal pattern in the data**: the peaks tend to be **4 quarters** apart and the troughs tend to be **2 quarters** apart.
- r_2 is more negative than for the other lags because troughs tend to be 2 quarters behind peaks.
- Together, the autocorrelations at lags 1, 2, ..., make up the *autocorrelation* or ACF.
- The plot is known as a **correlogram**

ACF

```r
ggAcf(beer)
```

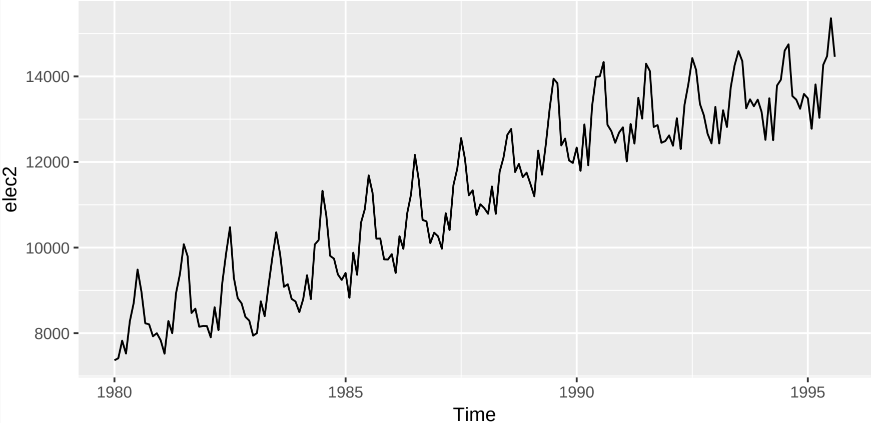


Trend and seasonality in ACF plots

- When data have a trend, the autocorrelations for small lags tend to be large and positive.
- When data are seasonal, the autocorrelations will be larger at the seasonal lags (i.e., at multiples of the seasonal frequency)
- When data are trended and seasonal, you see a combination of these effects.

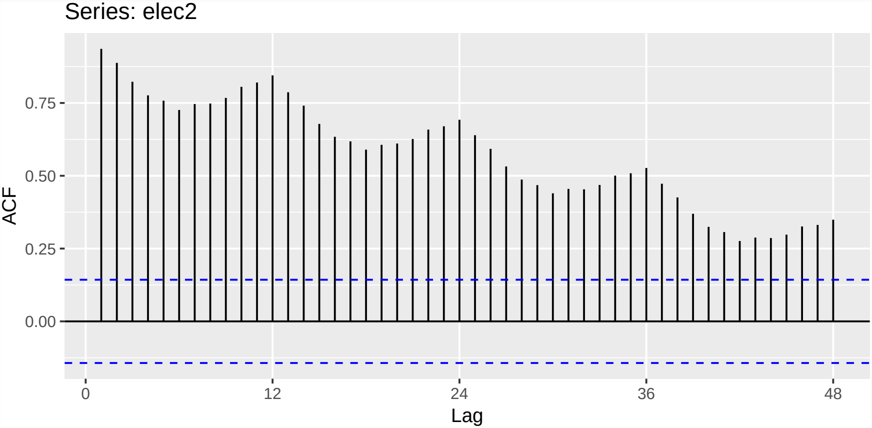
Aus monthly electricity production

```r
elec2 <- window(elec, start=1980)
autoplot(elec2)
```



Aus monthly electricity production

```r
ggAcf(elec2, lag.max=48)
```



Aus monthly electricity production

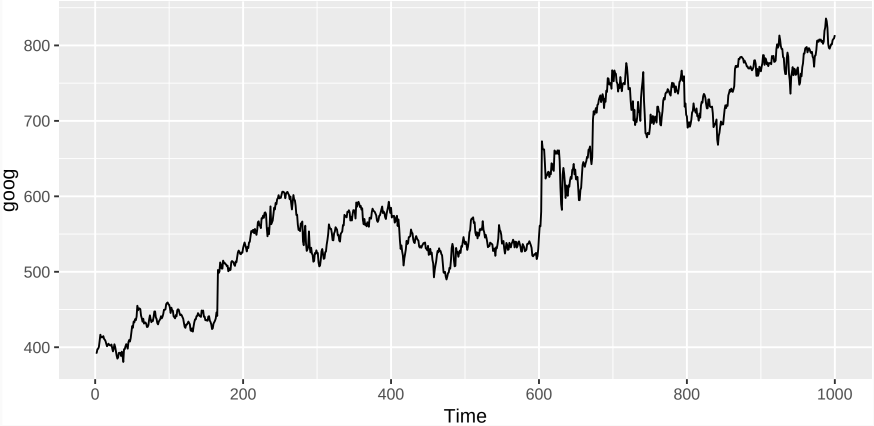
Time plot shows clear trend and seasonality.

The same features are reflected in the ACF.

- The slowly decaying ACF indicates trend.
- The ACF peaks at lags 12, 24, 36, . . . , indicate seasonality of length 12.

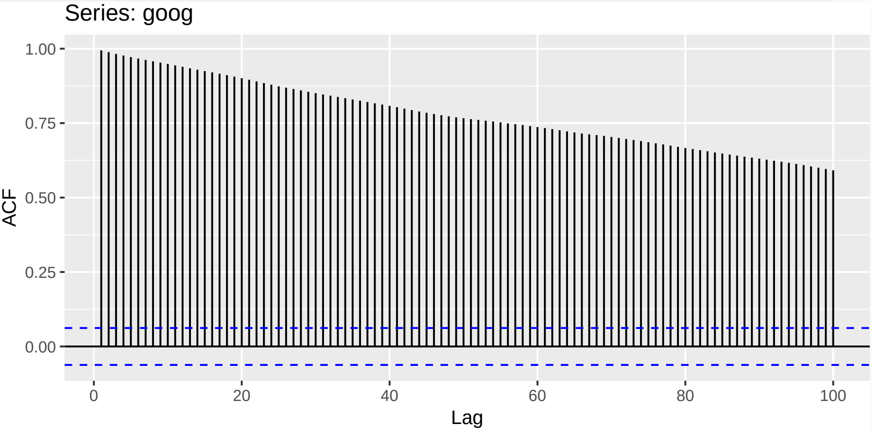
Google stock price

```r
autoplot(goog)
```



Google stock price

```
ggAcf(goog, lag.max=100)
```



Your turn

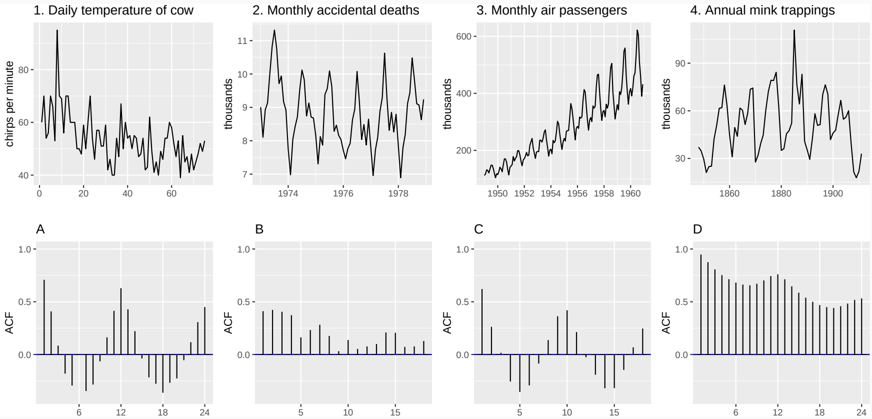
We have introduced the following graphics functions:

- `gglagplot`
- `ggAcf`

Explore the following time series using these functions. Can you spot any seasonality, cyclicity and trend? What do you learn about the series?

- `hsales`
- `usdeaths`
- `bricksq`
- `sunspotarea`
- `gasoline`

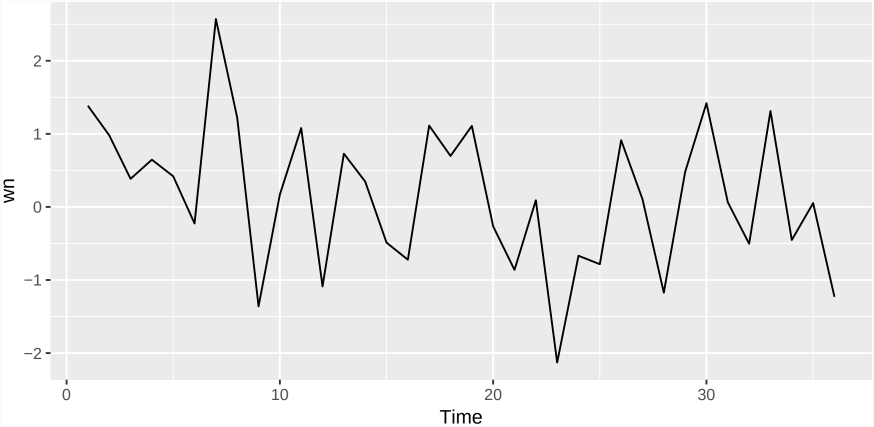
Which is which?



Outline

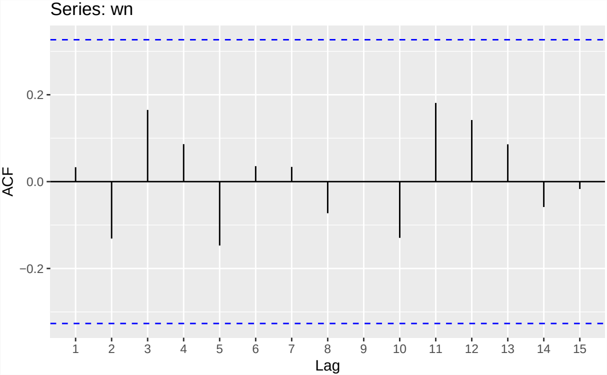
Example: White noise

```r
wn <- ts(rnorm(36))
autoplot(wn)
```



Example: White noise

r_1	0.03
r_2	-0.13
r_3	0.17
r_4	0.09
r_5	-0.15
r_6	0.04
r_7	0.03
r_8	-0.07
r_9	0.00
r_{10}	-0.13



Sample autocorrelations for white noise series.

We expect each autocorrelation to be close to zero.

Sampling distribution of autocorrelations

Sampling distribution of r_k for white noise data is asymptotically $N(0,1/T)$.

Sampling distribution of autocorrelations

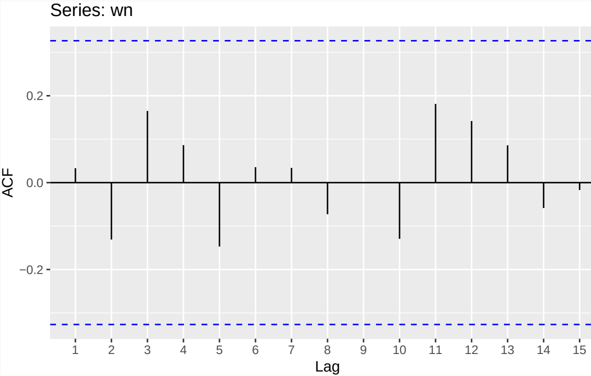
Sampling distribution of r_k for white noise data is asymptotically N$(0,1/T)$.

- 95% of all r_k for white noise must lie within $\pm 1.96/\sqrt{T}$.
- If this is not the case, the series is probably not WN.
- Common to plot lines at $\pm 1.96/\sqrt{T}$ when plotting ACF. These are the ***critical values***.

Autocorrelation

Example:

T = 36 and so critical values at $\pm 1.96/\sqrt{36} = \pm 0.327$. All autocorrelation coefficients lie within these limits, confirming that the data are white noise. (More precisely, the data cannot be distinguished from white noise.)

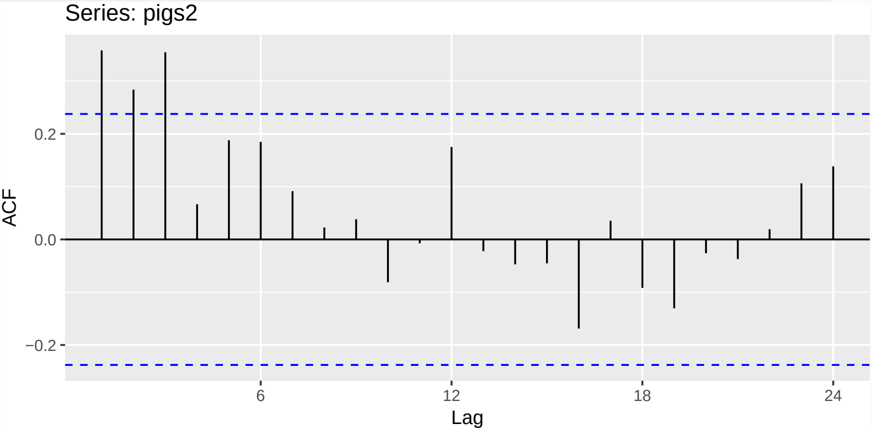


Example: Pigs slaughtered

```
pigs2 <- window(pigs, start=1990)
autoplot(pigs2) +
  xlab("Year") + ylab("thousands") +
  ggtitle("Number of pigs slaughtered in Victoria")
```

Example: Pigs slaughtered

```
ggAcf(pigs2)
```



Example: Pigs slaughtered

Monthly total number of pigs slaughtered in the state of Victoria, Australia, from January 1990 through August 1995. (Source: Australian Bureau of Statistics.)

Example: Pigs slaughtered

Monthly total number of pigs slaughtered in the state of Victoria, Australia, from January 1990 through August 1995. (Source: Australian Bureau of Statistics.)

- Difficult to detect pattern in time plot.
- ACF shows some significant autocorrelation at lags 1, 2, and 3.
- r_{12} relatively large although not significant. This may indicate some slight seasonality.

Example: Pigs slaughtered

Monthly total number of pigs slaughtered in the state of Victoria, Australia, from January 1990 through August 1995. (Source: Australian Bureau of Statistics.)

- Difficult to detect pattern in time plot.
- ACF shows some significant autocorrelation at lags 1, 2, and 3.
- r_{12} relatively large although not significant. This may indicate some slight seasonality.

These show the series is **not a white noise series**.

Your turn

You can compute the daily changes in the Google stock price using

```
dgoog <- diff(goog)
```

Does `dgoog` look like white noise?