Time series

Professor McNamara

Time series

What is a time series?
Why model time series?

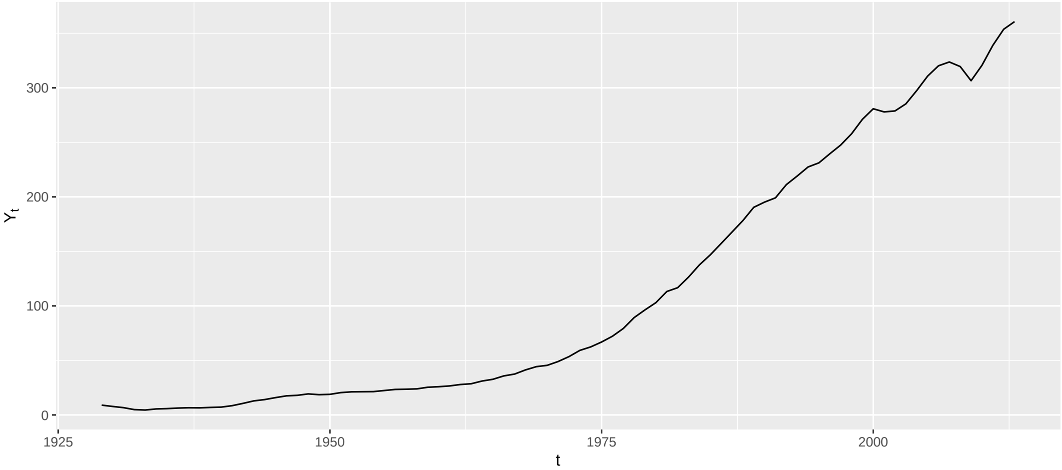
Example– clothing expenditures

```
## # A tibble: 85 x 3
##     year sales.b     t
##    <dbl>   <dbl> <int>
##  1  1929     9       1
##  2  1930     7.8     2
##  3  1931     6.7     3
##  4  1932     4.9     4
##  5  1933     4.5     5
##  6  1934     5.5     6
##  7  1935     5.8     7
##  8  1936     6.3     8
##  9  1937     6.6     9
## 10  1938     6.5    10
## # ... with 75 more rows
```

Plots

The greatest value of a picture is when it forces us to notice what we never expected to see
-John Tukey

Time series plot



What does this plot tell us?
What model might we fit?

Section 12.1

Variations on linear models

Initially, let's pretend that the only tool we have available to us is linear regression. How can we adapt linear regression to help us model time series data?

Example– clothing expenditures

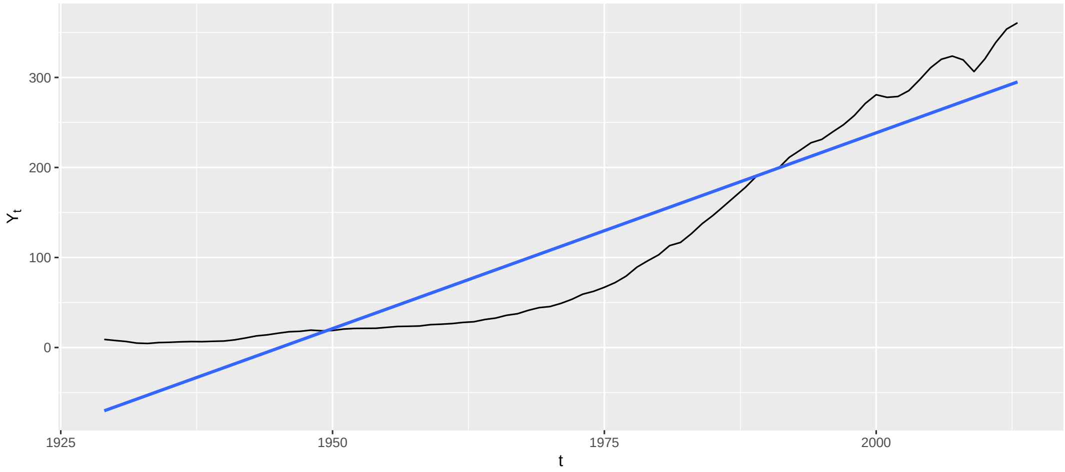
Using year as predictor,

```
##
## Call:
## lm(formula = sales.b ~ year, data = clothes)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -63.372 -42.288   1.892  35.077  79.259
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8458.2427   371.6368  -22.76   <2e-16 ***
## year            4.3484     0.1885   23.06   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 42.65 on 83 degrees of freedom
## Multiple R-squared:  0.865,  Adjusted R-squared:  0.8634
## F-statistic: 531.9 on 1 and 83 DF,  p-value: < 2.2e-16
```

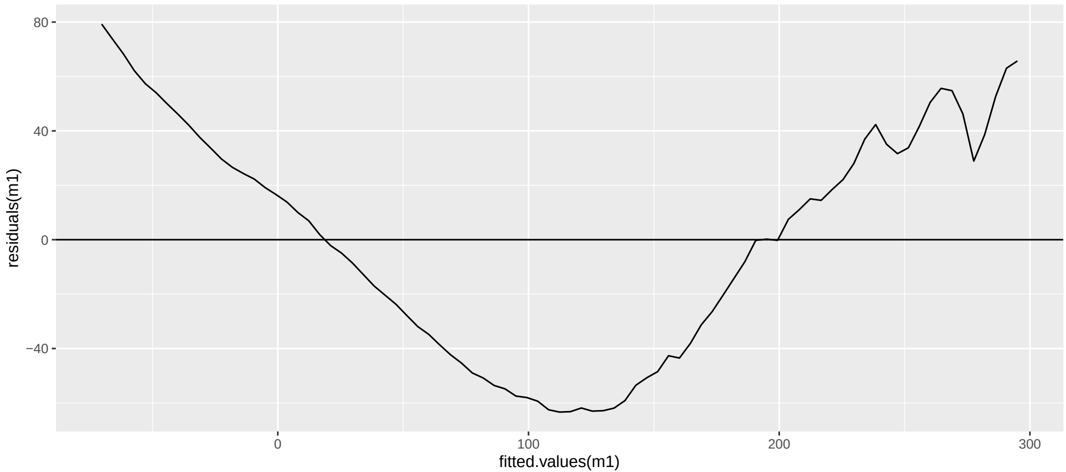
Could also use the time step as the predictor,

```
##
## Call:
## lm(formula = sales.b ~ t, data = clothes)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -63.372 -42.288   1.892  35.077  79.259
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -74.6076     9.3341  -7.993 6.72e-12 ***
## t             4.3484     0.1885  23.064  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

Plotting the fit



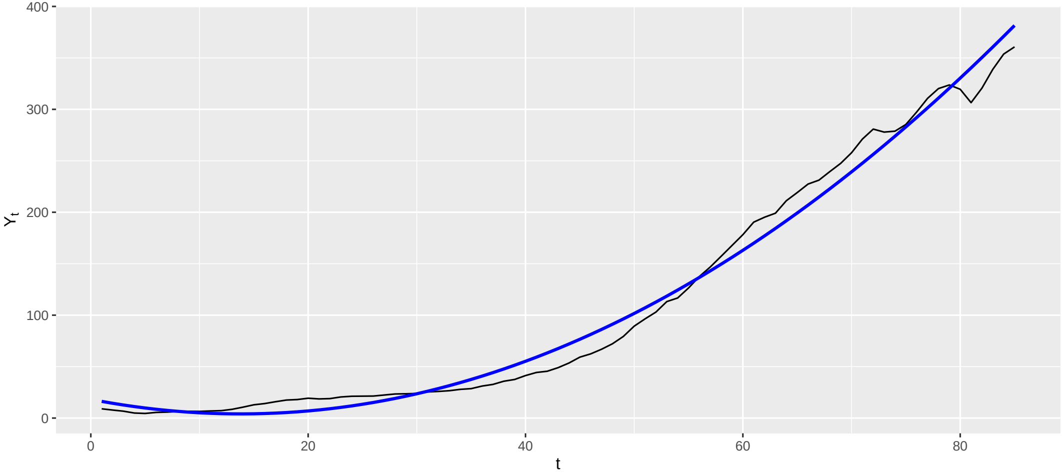
Residuals



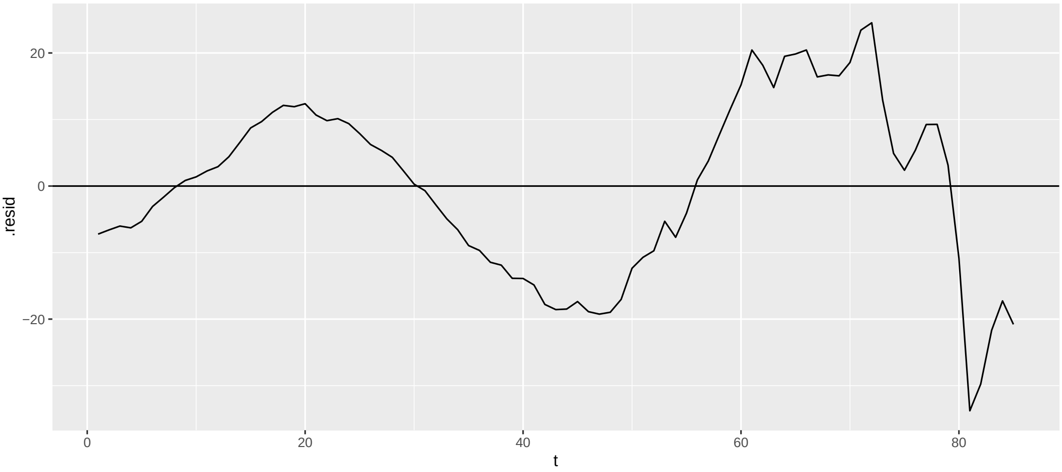
Quadratic fit?

```
##
## Call:
## lm(formula = sales.b ~ poly(t, degree = 2), data = clothes)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -33.776  -9.731   1.379   9.826  24.538
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)           112.372      1.443   77.85   <2e-16 ***
## poly(t, degree = 2)1  983.633     13.307   73.92   <2e-16 ***
## poly(t, degree = 2)2  369.390     13.307   27.76   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.31 on 82 degrees of freedom
## Multiple R-squared:  0.987,  Adjusted R-squared:  0.9867
## F-statistic:  3117 on 2 and 82 DF,  p-value: < 2.2e-16
```

Quadratic fit?



Residuals



Cosine model

$$Y = \beta_0 + \alpha \cos\left(\frac{2\pi t}{12} + \theta\right) + \epsilon$$

This model is non-linear, but it can be transformed

$$Y = \beta_0 + \alpha \cos(\theta) \cos\left(\frac{2\pi t}{12}\right) - \alpha \sin(\theta) \sin\left(\frac{2\pi t}{12}\right) + \epsilon$$

Then, instead of using t as our predictor, we use $X_{\cos} = \cos\left(\frac{2\pi t}{12}\right)$ and $X_{\sin} = \sin\left(\frac{2\pi t}{12}\right)$

$$Y = \beta_0 + \beta_1 X_{\cos} + \beta_2 X_{\sin} + \epsilon$$

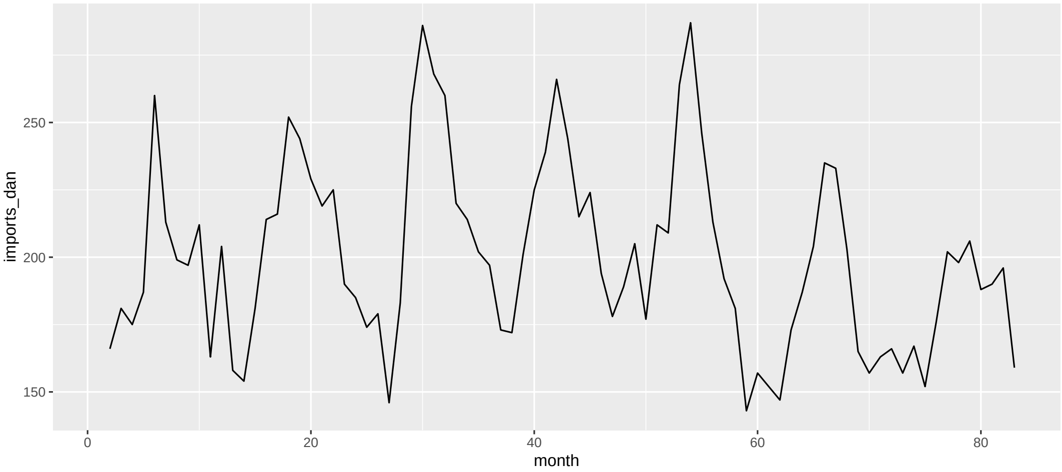
Shiny app

Play with this shiny app to get a sense for what the different coefficients do.

Example– butter imports

```
## # A tibble: 83 x 2
##    month imports_dan
##    <dbl>       <dbl>
##  1     1          NA
##  2     2         166
##  3     3         181
##  4     4         175
##  5     5         187
##  6     6         260
##  7     7         213
##  8     8         199
##  9     9         197
## 10    10         212
## # ... with 73 more rows
```

Example– butter imports

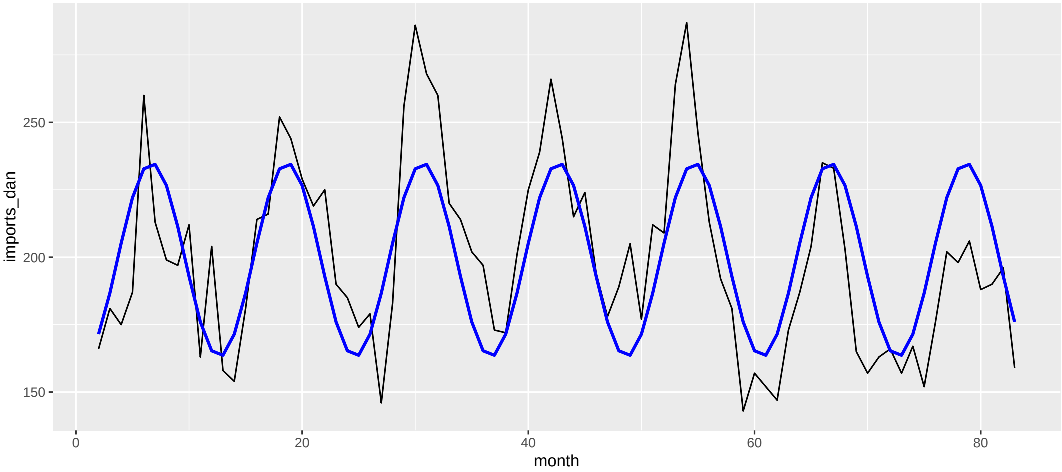


Example– butter imports

```r
butter <- butter %>%
  mutate(Xcos = cos(2 * pi * month / 12), Xsin = sin(2 * pi * month / 12))
m4 <- lm(imports_dan ~ Xcos + Xsin, data = butter)
summary(m4)
```

```
##
## Call:
## lm(formula = imports_dan ~ Xcos + Xsin, data = butter)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -46.407 -17.930   0.605  16.267  54.219
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   199.05       2.62  75.964  < 2e-16 ***
## Xcos          -33.73       3.74  -9.020 8.91e-14 ***
## Xsin          -12.36       3.67  -3.367  0.00117 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 23.71 on 79 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.5381, Adjusted R-squared:  0.5264
## F-statistic: 46.02 on 2 and 79 DF,  p-value: 5.605e-14
```

Example– butter imports



Seasonal means model

A simpler method would be to just use means for each time period.

$$Y = \beta_0 + \beta_1 Period_2 + \beta_2 Period_3 + \cdots + \beta_{k-1} Period_k + \epsilon$$

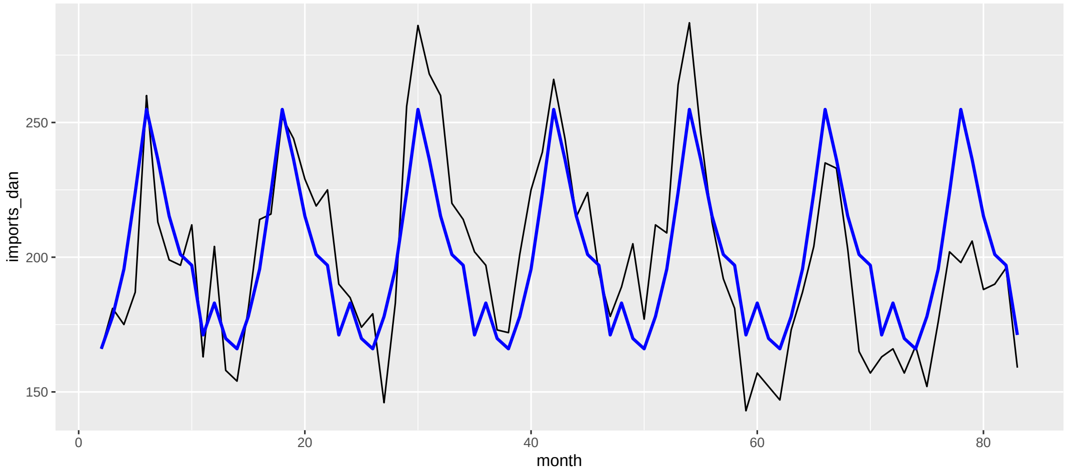
E.g.,

$$Y = \beta_0 + \beta_1 Month_{February} + \beta_2 Month_{March} + \cdots + \beta_{11} Month_{December} + \epsilon$$

Example– butter imports

```
##
## Call:
## lm(formula = imports_dan ~ as_factor(month_real), data = butter)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -56.857 -15.208  -0.143  14.750  44.714
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)              169.833      9.197  18.465  < 2e-16 ***
## as_factor(month_real)2    -3.833     12.534  -0.306 0.760638
## as_factor(month_real)3     8.167     12.534   0.652 0.516818
## as_factor(month_real)4    25.738     12.534   2.053 0.043766 *
## as_factor(month_real)5    54.167     12.534   4.322 5.02e-05 ***
## as_factor(month_real)6    85.024     12.534   6.783 3.09e-09 ***
## as_factor(month_real)7    66.452     12.534   5.302 1.27e-06 ***
## as_factor(month_real)8    45.452     12.534   3.626 0.000542 ***
## as_factor(month_real)9    31.167     12.534   2.487 0.015286 *
## as_factor(month_real)10   27.167     12.534   2.167 0.033605 *
## as_factor(month_real)11    1.310     12.534   0.104 0.917089
## as_factor(month_real)12   13.167     13.007   1.012 0.314899
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 22.53 on 70 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.6306, Adjusted R-squared:  0.5726
## F-statistic: 10.86 on 11 and 70 DF,  p-value: 2.338e-11
```

Example– butter imports



Comparing model choices

Cosine trend:

fewer parameters

```
## (Intercept)        Xcos        Xsin
##   199.04722   -33.73367   -12.35982
```

worse R^2

```
## [1] 0.5381284
```

Seasonal means:

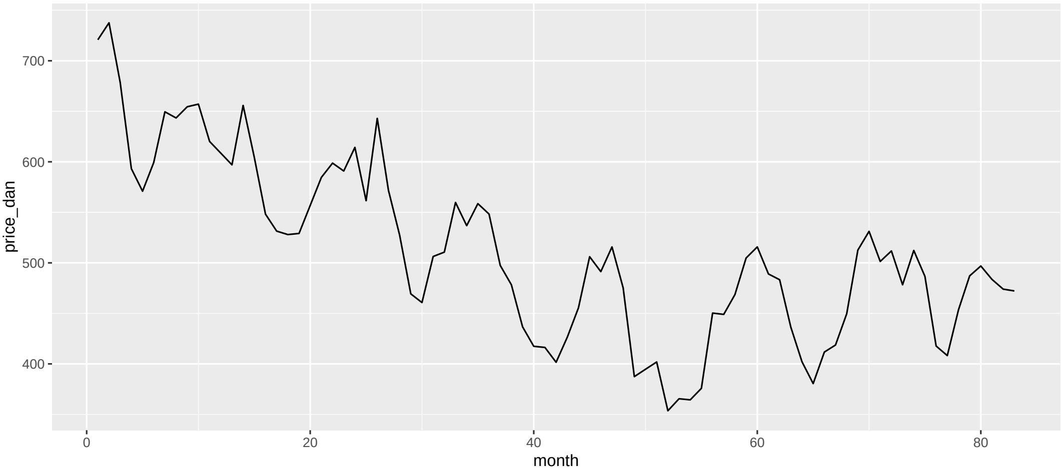
better R^2 and adjusted R^2,

```
## [1] 0.6306284
## [1] 0.5725843
```

lots of parameters. . .

```
##             (Intercept)  as_factor(month_real)2  as_factor(month_real)3
##              169.833333               -3.833333                8.166667
##  as_factor(month_real)4  as_factor(month_real)5  as_factor(month_real)6
##               25.738095               54.166667               85.023810
##  as_factor(month_real)7  as_factor(month_real)8  as_factor(month_real)9
##               66.452381               45.452381               31.166667
## as_factor(month_real)10 as_factor(month_real)11 as_factor(month_real)12
##               27.166667                1.309524               13.166667
```

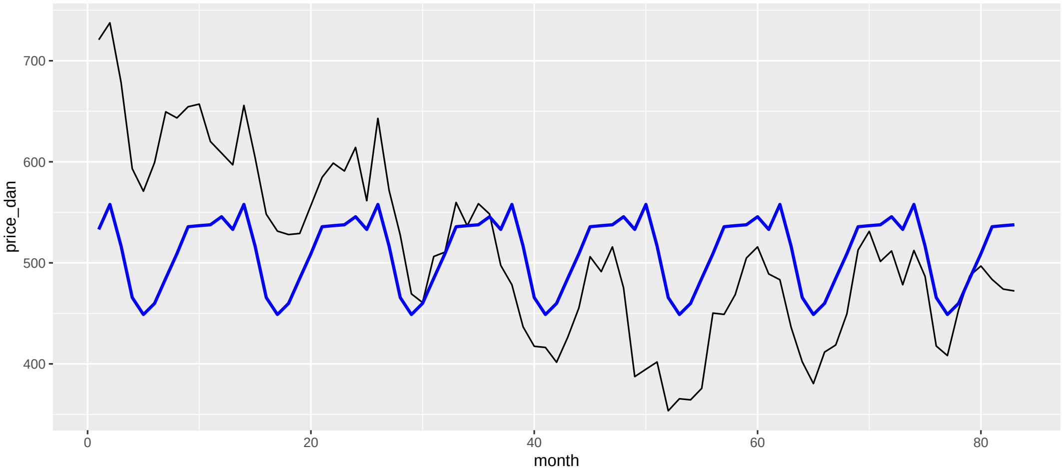
Another example– butter price



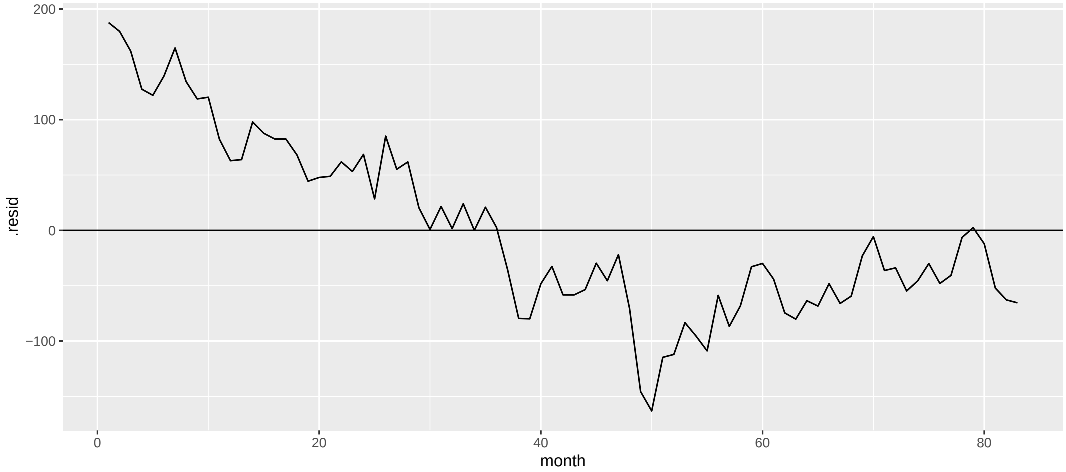
Example– butter price

```
##
## Call:
## lm(formula = price_dan ~ as_factor(month_real), data = butter)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -163.16  -58.51  -23.06   61.86  187.73
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)              533.071     32.186  16.562   <2e-16 ***
## as_factor(month_real)2    24.786     45.517   0.545   0.5878
## as_factor(month_real)3   -16.471     45.517  -0.362   0.7185
## as_factor(month_real)4   -67.400     45.517  -1.481   0.1431
## as_factor(month_real)5   -84.200     45.517  -1.850   0.0685 .
## as_factor(month_real)6   -73.186     45.517  -1.608   0.1123
## as_factor(month_real)7   -48.371     45.517  -1.063   0.2915
## as_factor(month_real)8   -24.057     45.517  -0.529   0.5988
## as_factor(month_real)9     2.686     45.517   0.059   0.9531
## as_factor(month_real)10    3.743     45.517   0.082   0.9347
## as_factor(month_real)11    4.614     45.517   0.101   0.9195
## as_factor(month_real)12   12.595     47.376   0.266   0.7911
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 85.15 on 71 degrees of freedom
## Multiple R-squared:  0.169,  Adjusted R-squared:  0.04024
## F-statistic: 1.313 on 11 and 71 DF,  p-value: 0.2357
```

Example– butter price



Residuals

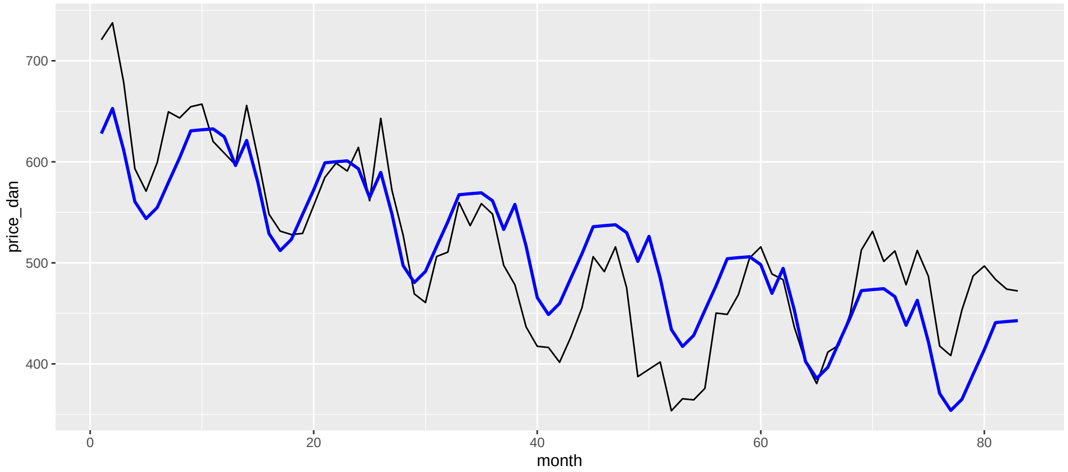


Another term

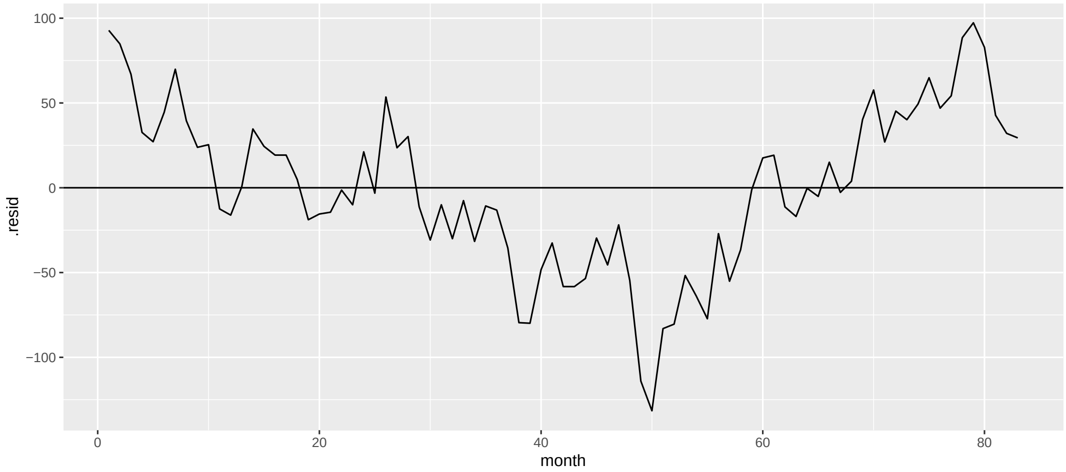
Maybe it would help to include a linear trend,

```
##
## Call:
## lm(formula = price_dan ~ as_factor(month_real) + month, data = butter)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -131.525  -30.432   -1.378   32.357   97.295
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)             630.6023    21.5633  29.244  < 2e-16 ***
## as_factor(month_real)2   27.4217    27.7876   0.987  0.32712
## as_factor(month_real)3  -11.1995    27.7908  -0.403  0.68818
## as_factor(month_real)4  -59.4921    27.7959  -2.140  0.03582 *
## as_factor(month_real)5  -73.6561    27.8032  -2.649  0.00997 **
## as_factor(month_real)6  -60.0059    27.8125  -2.158  0.03440 *
## as_factor(month_real)7  -32.5556    27.8239  -1.170  0.24595
## as_factor(month_real)8   -5.6054    27.8374  -0.201  0.84100
## as_factor(month_real)9   23.7735    27.8529   0.854  0.39627
## as_factor(month_real)10  27.4666    27.8705   0.986  0.32777
## as_factor(month_real)11  30.9740    27.8902   1.111  0.27055
## as_factor(month_real)12  25.7751    28.9461   0.890  0.37627
## month                    -2.6360     0.2401 -10.978  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 51.98 on 70 degrees of freedom
## Multiple R-squared:  0.6947, Adjusted R-squared:  0.6423
## F-statistic: 13.27 on 12 and 70 DF,  p-value: 1.05e-13
```

Another term

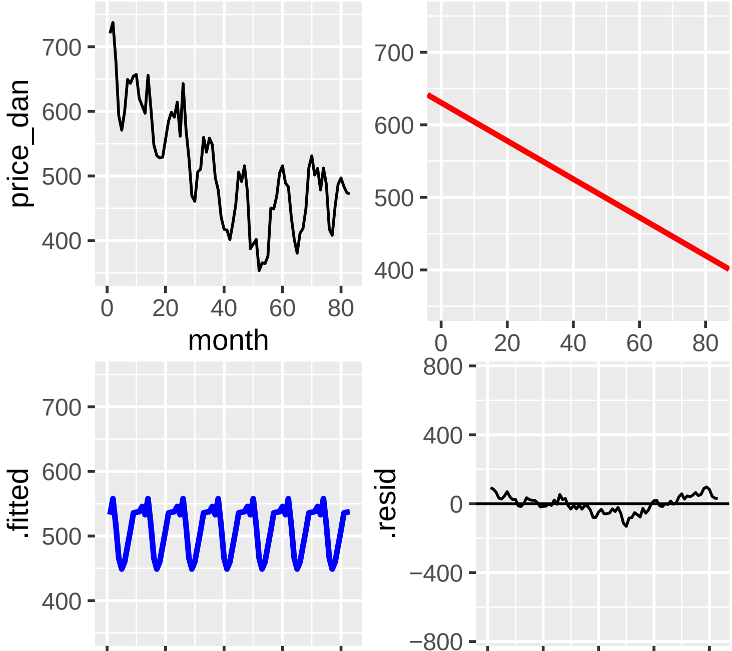


Residuals



Decomposing time series

It is useful to think of time series as a decomposition



Section 12.2

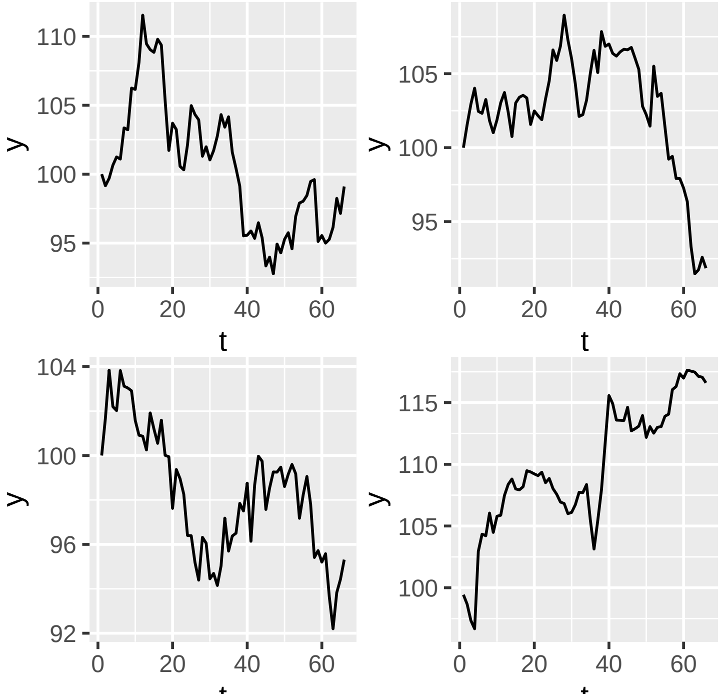
Lags and autocorrelation

Random walk

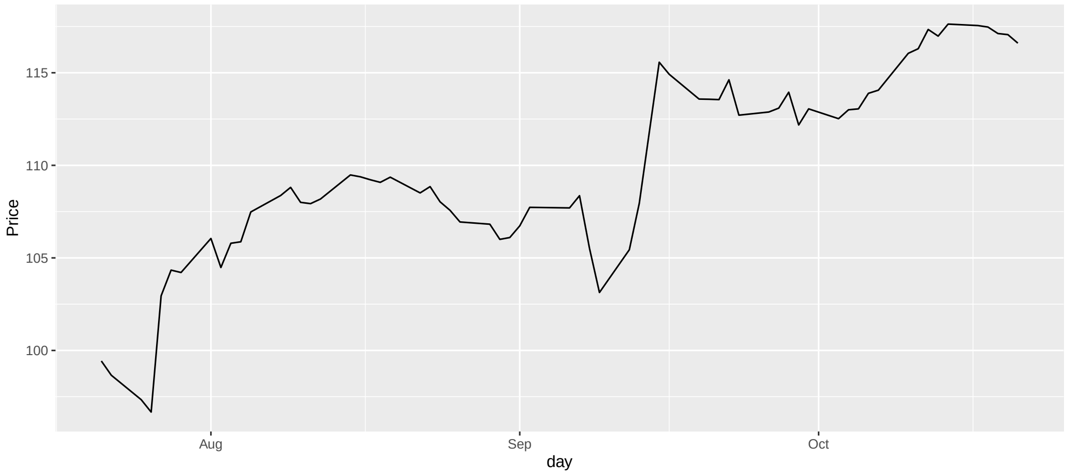
In a random walk, each value is just a random movement from the previous one.

$$Y_t = Y_{t-1} + \epsilon_t$$

Random walks



Example– Apple Stock



Differences

First difference of a time series,

$$\Delta Y_t = Y_t - Y_{t-1}$$

For a random walk with $Y_t = Y_{t-1} + \epsilon_t$,

$$\Delta Y_t = Y_t - Y_{t-1} = \epsilon_t$$

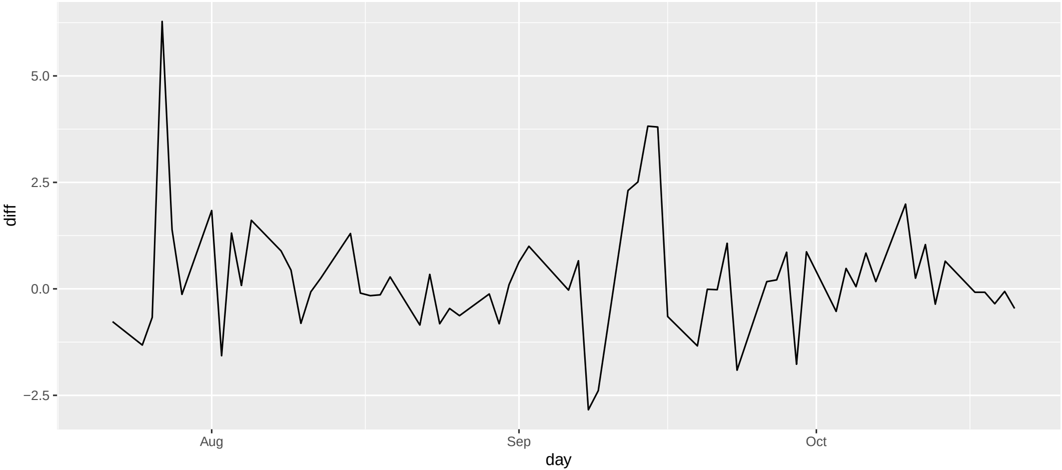
Example– Apple Stock

```r
AppleStock <- AppleStock %>%
  tibble() %>%
  mutate(diff = Price - lag(Price))
AppleStock %>%
  select(Price, diff, day)
```

```
## # A tibble: 66 x 3
##    Price  diff day
##    <dbl> <dbl> <date>
##  1  99.4 NA    2016-07-21
##  2  98.7 -0.77 2016-07-22
##  3  97.3 -1.32 2016-07-25
##  4  96.7 -0.67 2016-07-26
##  5 103.   6.28 2016-07-27
##  6 104.   1.39 2016-07-28
##  7 104.  -0.13 2016-07-29
##  8 106.   1.84 2016-08-01
##  9 104.  -1.57 2016-08-02
## 10 106.   1.31 2016-08-03
## # ... with 56 more rows
```

First differences

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```



Autocorrelation

You can measure the association between time series values that are k time units apart. For example, could also do lag 2.

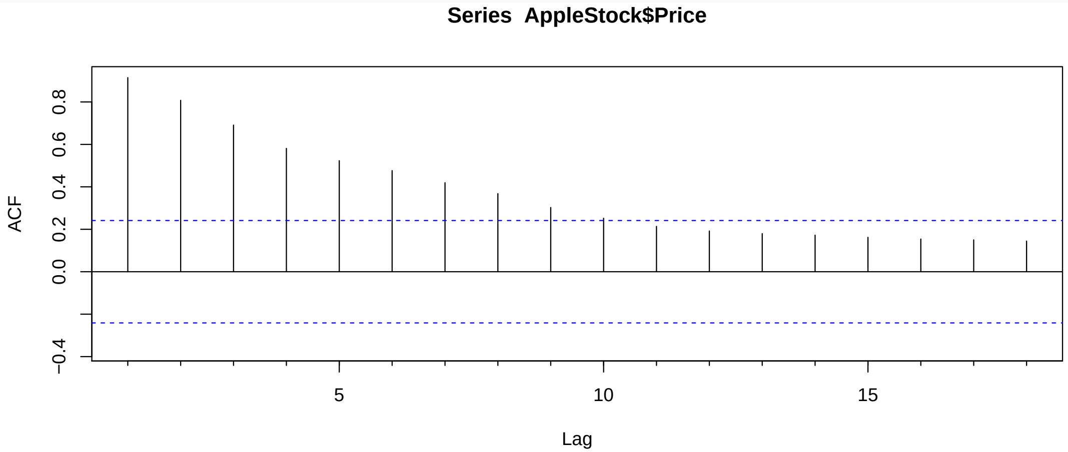
```
AppleStock <- AppleStock %>%
  mutate(diff2 = Price - lag(Price, n = 2))
AppleStock %>%
  select(Price, diff2, day)
```

```
## # A tibble: 66 x 3
##    Price  diff2 day
##    <dbl>  <dbl> <date>
##  1  99.4 NA     2016-07-21
##  2  98.7 NA     2016-07-22
##  3  97.3 -2.09  2016-07-25
##  4  96.7 -1.99  2016-07-26
##  5 103.   5.61  2016-07-27
##  6 104.   7.67  2016-07-28
##  7 104.   1.26  2016-07-29
##  8 106.   1.71  2016-08-01
##  9 104.   0.27  2016-08-02
## 10 106.  -0.260 2016-08-03
## # ... with 56 more rows
```

Computing autocorrelation

One way to compute autocorrelation is with `forecast::Acf()`

```r
Acf(AppleStock$Price)
```

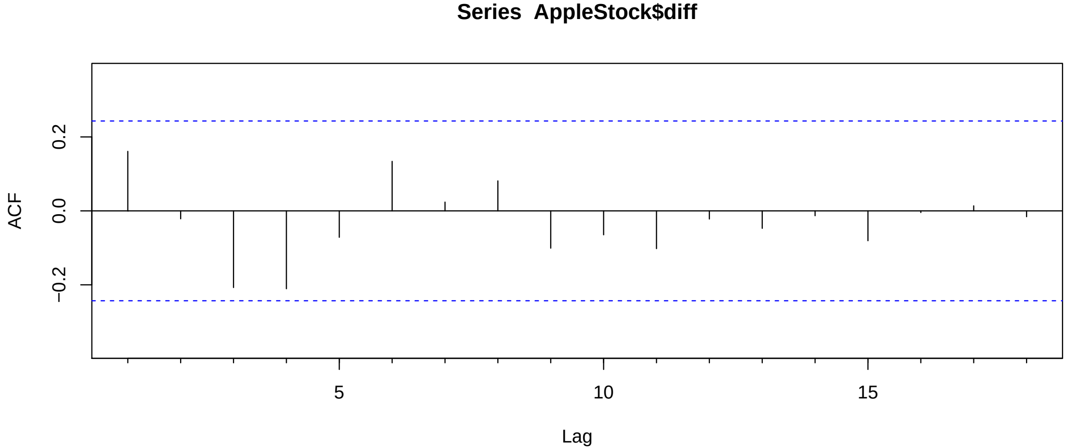



```r
Acf(AppleStock$Price, plot = FALSE)
```

```
##
## Autocorrelations of series 'AppleStock$Price', by lag
##
##     0     1     2     3     4     5     6     7     8     9    10    11    12
## 1.000 0.915 0.808 0.691 0.581 0.523 0.477 0.419 0.368 0.303 0.252 0.214 0.192
##    13    14    15    16    17    18
```

No autocorrelation

Let's compare those last few plots with an autocorrelation plot of one of the lags



Nothing makes it to the edge, so we're good!

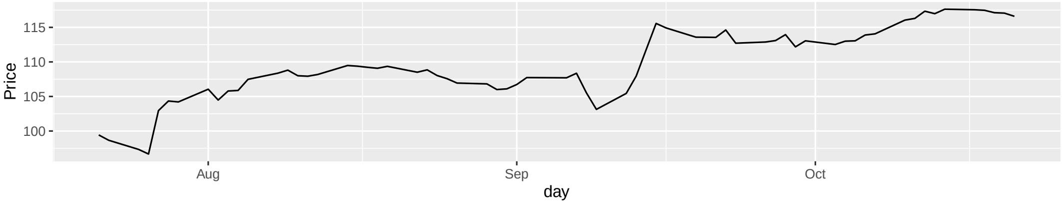
Stationarity

One reason to look at the ACF plot is to determine if a time series is "stationary."
Stationarity basically means things are staying the same over time.
Ways to see **lack of stationarity**:

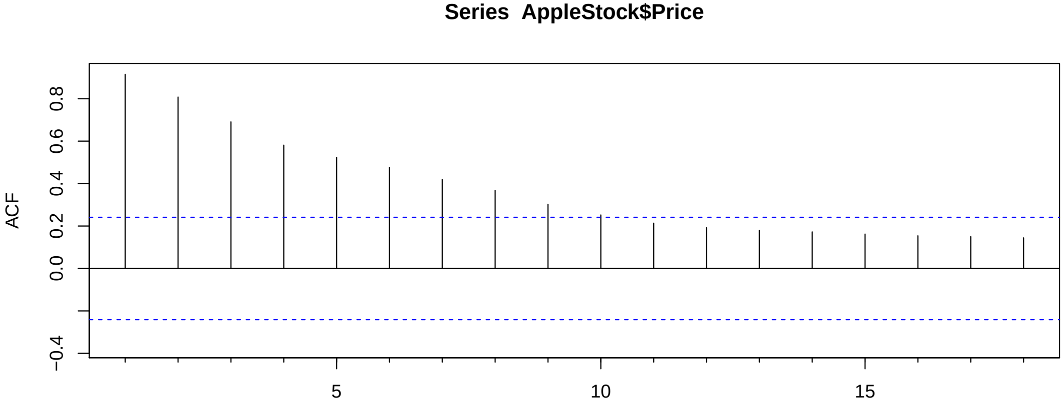
- overall increasing or decreasing trend in data plot
- linear trend in ACF plot

Example– Apple prices

Apple stock prices are **not** stationary. We can see this either in the plot of the data,



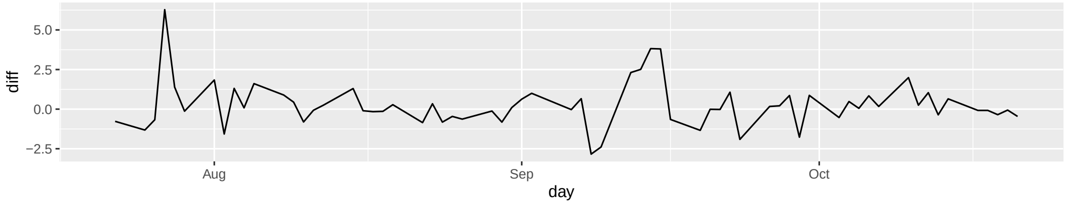
Or in the ACF plot,

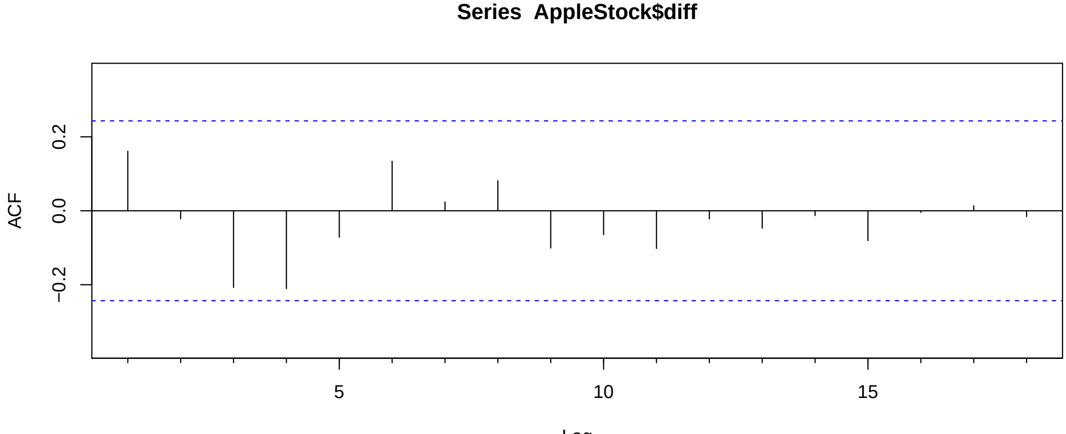


Example– Apple stock price differences

The first differences of Apple stock prices **are** stationary

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```





Seasonal differences

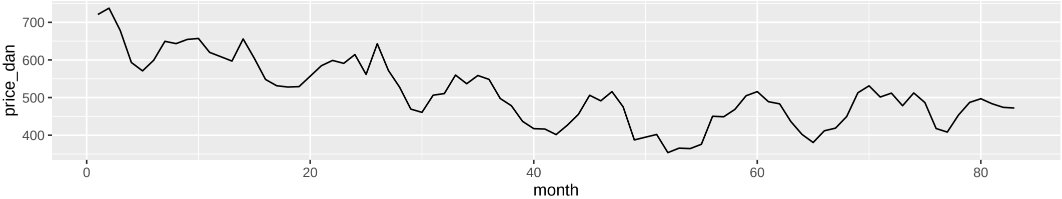
We can also have differences on regular lags, such as seasons. For example, we could find

$$\Delta_{12} Y_t = Y_t - Y_{t-12}$$

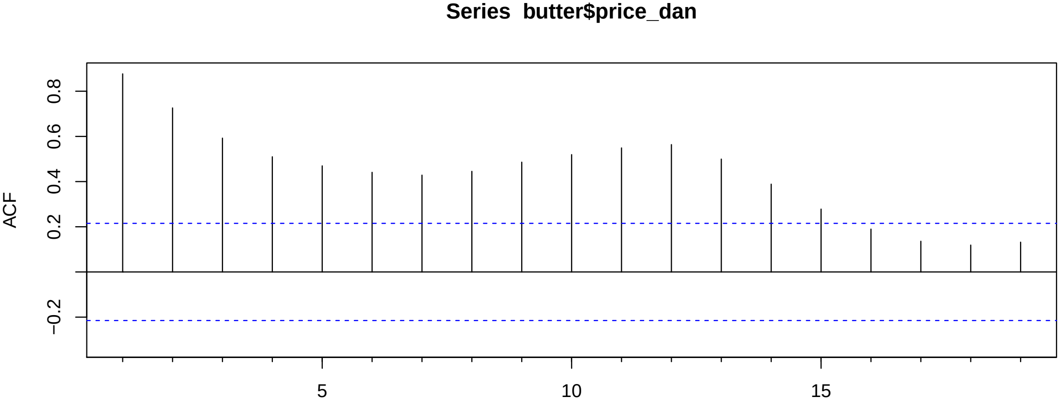
to compare to the same month from the previous year

Back to butter prices

Recall our butter prices data,

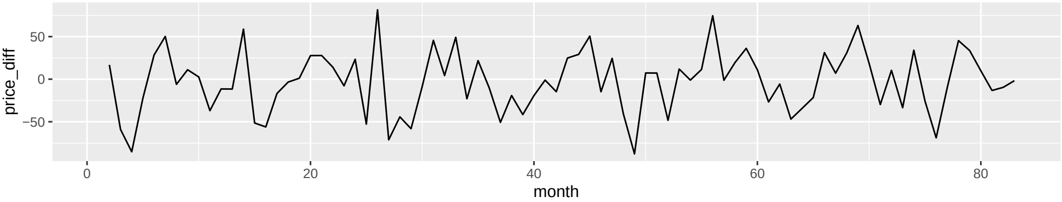


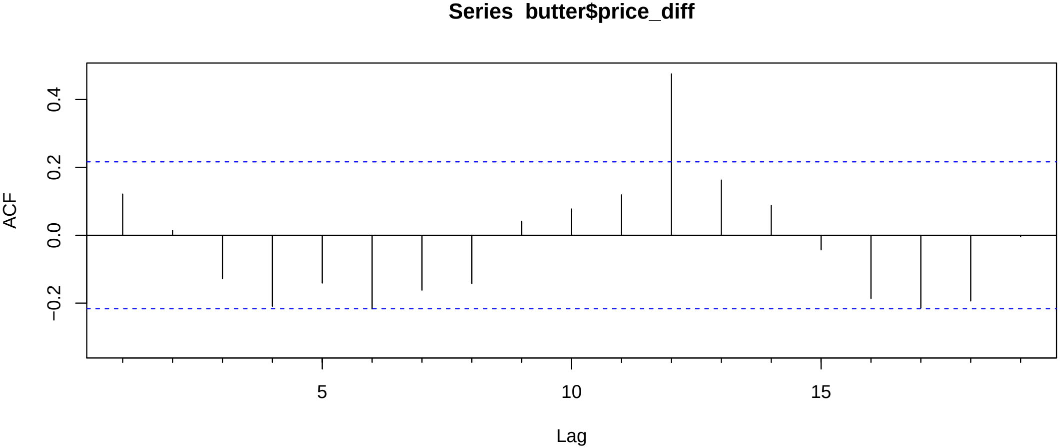
We could look at an ACF plot for this data,



Butter price differences

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```

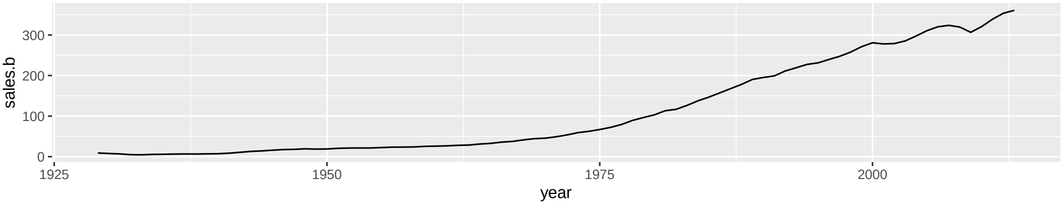


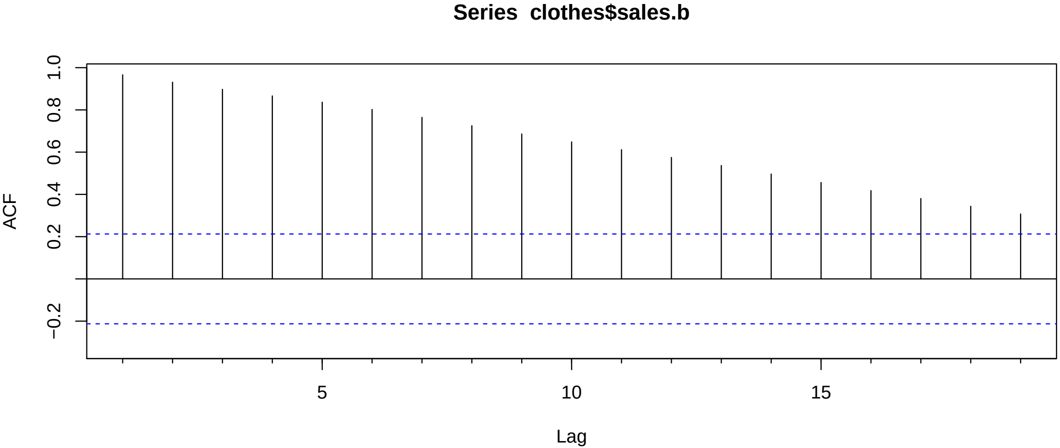


Section 12.3

- ARIMA models
  - AutoRegressive (AR)
  - Moving Average (MA)

Example– clothing expenditures

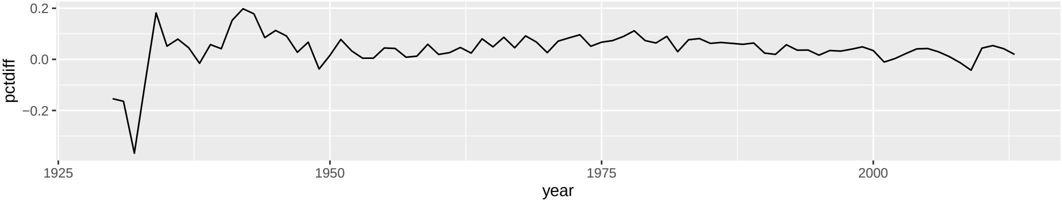


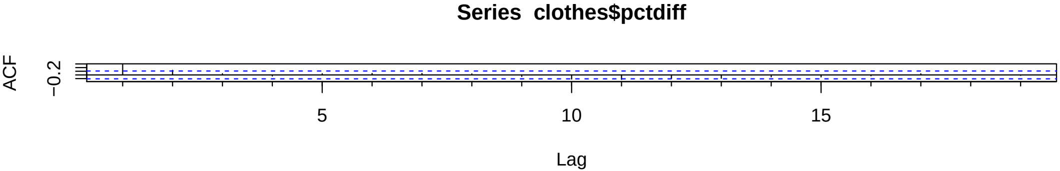


Percent difference

Instead of just subtracting the previous month to get an absolute difference, let's do a percent difference.

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```





Autoregressive model

ϕ

$$Y_t = \delta + \phi Y_{t-1} + \epsilon_t$$

Example– clothing expenditures

Find the AR(1) model

```
Arima(clothes$pctdiff, order = c(1, 0, 0), include.constant = TRUE)
```

```
## Series: clothes$pctdiff
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##          ar1    mean
##       0.6196  0.0366
## s.e.  0.0903  0.0160
##
## sigma^2 estimated as 0.003281:  log likelihood=121.81
## AIC=-237.62   AICc=-237.32   BIC=-230.32
```

Annoyingly in R, the "mean" isn't the constant term in the model. You have to do a little algebra,

$$\mu = \delta + \phi_1 \mu$$

Considering more history

We don't have to just use the previous value! We could fit $AR(p)$, with as many lags as we want.

$$Y_t = \delta + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} \epsilon_t$$

Example– clothing expenditures

Fitting the AR(2) model,

```
Arima(clothes$pctdiff, order = c(2, 0, 0), include.constant = TRUE)
```

```
## Series: clothes$pctdiff
## ARIMA(2,0,0) with non-zero mean
##
## Coefficients:
##          ar1      ar2    mean
##       0.6614  -0.0787  0.0376
## s.e.  0.1090   0.1150  0.0147
##
## sigma^2 estimated as 0.003303:  log likelihood=122.04
## AIC=-236.08   AICc=-235.58   BIC=-226.36
```

How do we know which variables are significant?

```r
Arima(clothes$pctdiff, order = c(2, 0, 0), include.constant = TRUE)
```

```
## Series: clothes$pctdiff
## ARIMA(2,0,0) with non-zero mean
##
## Coefficients:
##          ar1      ar2    mean
##       0.6614  -0.0787  0.0376
## s.e.  0.1090   0.1150  0.0147
##
## sigma^2 estimated as 0.003303:  log likelihood=122.04
## AIC=-236.08   AICc=-235.58   BIC=-226.36
```

Rule of thumb:

$$|\hat{\phi}_i / SE| > 2$$

More formal

Can use `lmtest::coeftest()`

```
##
## z test of coefficients:
##
##            Estimate Std. Error z value  Pr(>|z|)
## ar1        0.661368   0.108984  6.0685 1.291e-09 ***
## ar2       -0.078663   0.115013 -0.6839   0.49401
## intercept  0.037602   0.014654  2.5660   0.01029 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Moving Average (MA)

Okay, so we've talked about AR models, now let's talk MA models. These models consider that values in a time series might be related to the residual(s) from previous time steps.

$$Y_t = \delta + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}$$

Note: often this type of model would have minus signs holding it together, rather than plus signs. But, the `Arima()` function does things a little different, so we're being consistent with that.

Example– clothing expenditures

```r
(MA1 <- Arima(clothes$pctdiff, order = c(0, 0, 1), include.constant = TRUE))
```

```
## Series: clothes$pctdiff
## ARIMA(0,0,1) with non-zero mean
##
## Coefficients:
##          ma1    mean
##       0.7429  0.0388
## s.e.  0.0963  0.0107
##
## sigma^2 estimated as 0.003294:  log likelihood=121.47
## AIC=-236.94   AICc=-236.64   BIC=-229.65
```

```r
coeftest(MA1)
```

```
##
## z test of coefficients:
##
##           Estimate Std. Error z value  Pr(>|z|)
## ma1       0.742912   0.096256  7.7181 1.181e-14 ***
## intercept 0.038788   0.010735  3.6131 0.0003026 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

ARIMA models

We can put the AR and MA models together,

$$\begin{aligned} Y_t = \delta \quad &+ \quad \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} \epsilon_t + \epsilon_t \\ &+ \quad \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q} \end{aligned}$$

Example– clothing expenditures

```r
(ARMA11 <- Arima(clothes$pctdiff, order = c(1, 0, 1), include.constant = TRUE))
```

```
## Series: clothes$pctdiff
## ARIMA(1,0,1) with non-zero mean
##
## Coefficients:
##          ar1     ma1    mean
##       0.4817  0.2077  0.0377
## s.e.  0.2185  0.2744  0.0142
##
## sigma^2 estimated as 0.003292:  log likelihood=122.16
## AIC=-236.33   AICc=-235.82   BIC=-226.6
```

```r
coeftest(ARMA11)
```

```
##
## z test of coefficients:
##
##           Estimate Std. Error z value Pr(>|z|)
## ar1       0.481688   0.218519  2.2043 0.027501 *
## ma1       0.207720   0.274355  0.7571 0.448977
## intercept 0.037708   0.014207  2.6542 0.007949 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

One more parameter

We can also bring differences into the model,

$$\begin{aligned}\Delta^d Y_t = \delta &+ \phi_1 \Delta^d Y_{t-1} + \cdots + \phi_p \Delta^d Y_{t-p} \epsilon_t + \epsilon_t \\ &+ \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}\end{aligned}$$

Generally, we either do $d = 0$ or $d = 1$

ARIMA for clothing expenditure

Since we're going to include differences with the d term, let's go back to original sales

```
(ARIMA11 <- Arima(clothes$sales.b, order = c(1, 1, 0), include.constant = TRUE))
```

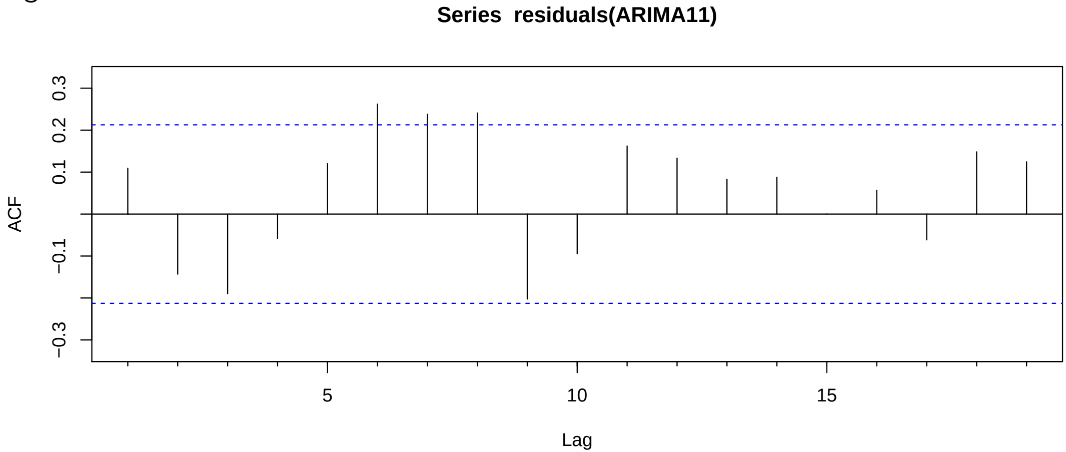
```
## Series: clothes$sales.b
## ARIMA(1,1,0) with drift
##
## Coefficients:
##          ar1   drift
##       0.6191  4.1390
## s.e.  0.0851  1.1233
##
## sigma^2 estimated as 16.36:  log likelihood=-235.81
## AIC=477.62   AICc=477.92   BIC=484.91
```

```
coeftest(ARIMA11)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error z value  Pr(>|z|)
## ar1   0.619071   0.085051  7.2788 3.368e-13 ***
## drift 4.138951   1.123329  3.6845 0.0002291 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Residual ACF

We want to make sure the residuals look random, to ensure we've modeled out any signal in the noise.



Still some seasonality there

Seasonal ARIMA

Let's just go nuts with terms... This model includes:

- p regular autoregressive terms
- d regular differences
- q moving average terms
- P seasonal autoregressive terms
- D seasonal differences
- Q seasonal moving average terms

With this much going on, we're just going to let R handle it

Seasonal ARIMA

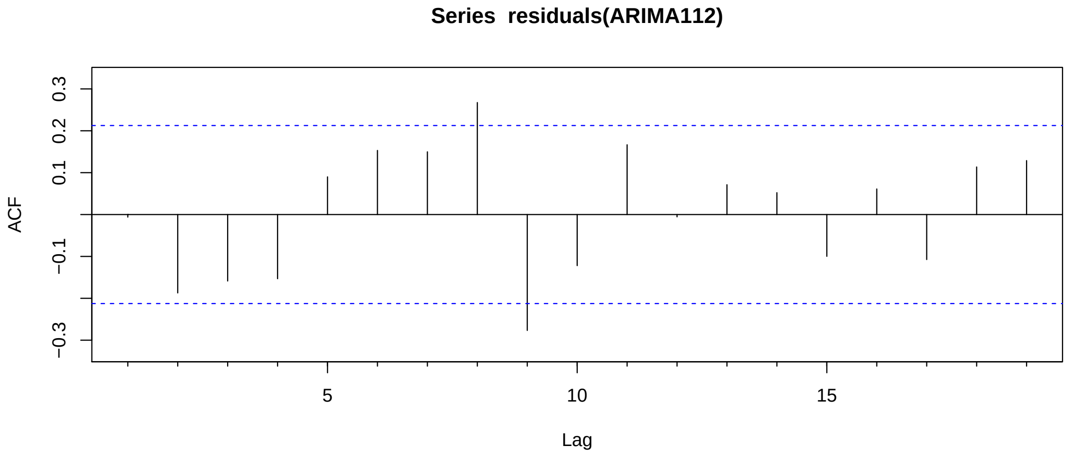
```r
(ARIMA112 <- Arima(clothes$sales.b, order = c(1, 1, 0), seasonal = list(order= c(2,0,0), period=12)))
```

```
## Series: clothes$sales.b
## ARIMA(1,1,0)(2,0,0)[12]
##
## Coefficients:
##          ar1    sar1    sar2
##       0.6499  0.3552  0.1013
## s.e.  0.0995  0.1574  0.1531
##
## sigma^2 estimated as 16.65:  log likelihood=-237.21
## AIC=482.41   AICc=482.92   BIC=492.14
```

```r
coeftest(ARIMA112)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ar1   0.64985    0.09948  6.5325 6.468e-11 ***
## sar1  0.35519    0.15736  2.2571    0.0240 *
## sar2  0.10131    0.15315  0.6615    0.5083
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Residuals



Forecasting

It's tough to make predictions, especially about the future
-Yogi Berra

Forecasting

```
ARIMA111 <- Arima(clothes$sales.b, order = c(1, 1, 0), seasonal = list(order= c(1,0,0), period=12))
forecast(ARIMA111, h=10)
```

```
##    Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 86       366.4090 361.1859 371.6320 358.4210 374.3969
## 87       372.4280 362.2929 382.5632 356.9277 387.9284
## 88       379.4049 364.4613 394.3484 356.5507 402.2591
## 89       385.9650 366.4806 405.4493 356.1663 415.7636
## 90       390.5975 366.8864 414.3085 354.3345 426.8604
## 91       392.6096 364.9815 420.2378 350.3560 434.8633
## 92       391.4758 360.2158 422.7358 343.6677 439.2839
## 93       386.8578 352.2204 421.4952 333.8845 439.8311
## 94       392.3946 354.6036 430.1856 334.5983 450.1910
## 95       399.4535 358.7043 440.2026 337.1330 461.7739
```

Plotting forecasts

```
pred1 <- forecast(ARIMA111, h=10)
plot(pred1)
```