# Introduction to Data Analysis in R

Code

# Time-series data analysis example

Let's pull some time-series data using `quantmod` package.

To download an individual time series we use a dedicated function. We downlaod VITAX which is a vanguard mutual fund for the technology sector.

Hide

```
require("quantmod")
```

```
Loading required package: quantmod
Loading required package: xts
Loading required package: zoo

Attaching package: ‘zoo’

The following objects are masked from ‘package:base’:

    as.Date, as.Date.numeric

Loading required package: TTR
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo
```

Hide

```
vitax = getSymbols('vitax', from='2005-01-01', to='2022-01-01', auto.assign=FALSE)
```

```
‘getSymbols’ currently uses auto.assign=TRUE by default, but will
use auto.assign=FALSE in 0.5-0. You will still be able to use
‘loadSymbols’ to automatically load data. getOption("getSymbols.env")
and getOption("getSymbols.auto.assign") will still be checked for
alternate defaults.

This message is shown once per session and may be disabled by setting
options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

And we examine what did we get in return.

Hide

```
list(
  'class'=class(vitax),
  'dimensions'=dim(vitax),
  'head'=head(vitax, 3)
)
```

```
$class
[1] "xts" "zoo"

$dimensions
[1] 4280    6

$head
           VITAX.Open VITAX.High VITAX.Low VITAX.Close VITAX.Volume VITAX.Adjusted
2005-01-03      23.83      23.83     23.83       23.83            0       20.54392
2005-01-04      23.30      23.30     23.30       23.30            0       20.08700
2005-01-05      23.16      23.16     23.16       23.16            0       19.96631
```

The object is of somewhat unfamiliar classes `zoo` and `xts`, however most of the data formats in R easily convert into the familiar `data.frame`. We will also only keep the price at the "Close" of the day and drop remaining columns.

Hide

```
vitax.df = fortify.zoo(vitax$VITAX.Close) # this is a crazy name for a function (one of the draw backs of open source development)
head(vitax.df, 3)
```

| | Index<br><date> | VITAX.Close<br><dbl> |
|---|---|---|
| 1 | 2005-01-03 | 23.83 |
| 2 | 2005-01-04 | 23.30 |
| 3 | 2005-01-05 | 23.16 |

3 rows

Hide

```
colnames(vitax.df) = c("date", "y")
head(vitax.df, 3)
```

| | date<br><date> | y<br><dbl> |
|---|---|---|
| 1 | 2005-01-03 | 23.83 |
| 2 | 2005-01-04 | 23.30 |
| 3 | 2005-01-05 | 23.16 |

3 rows

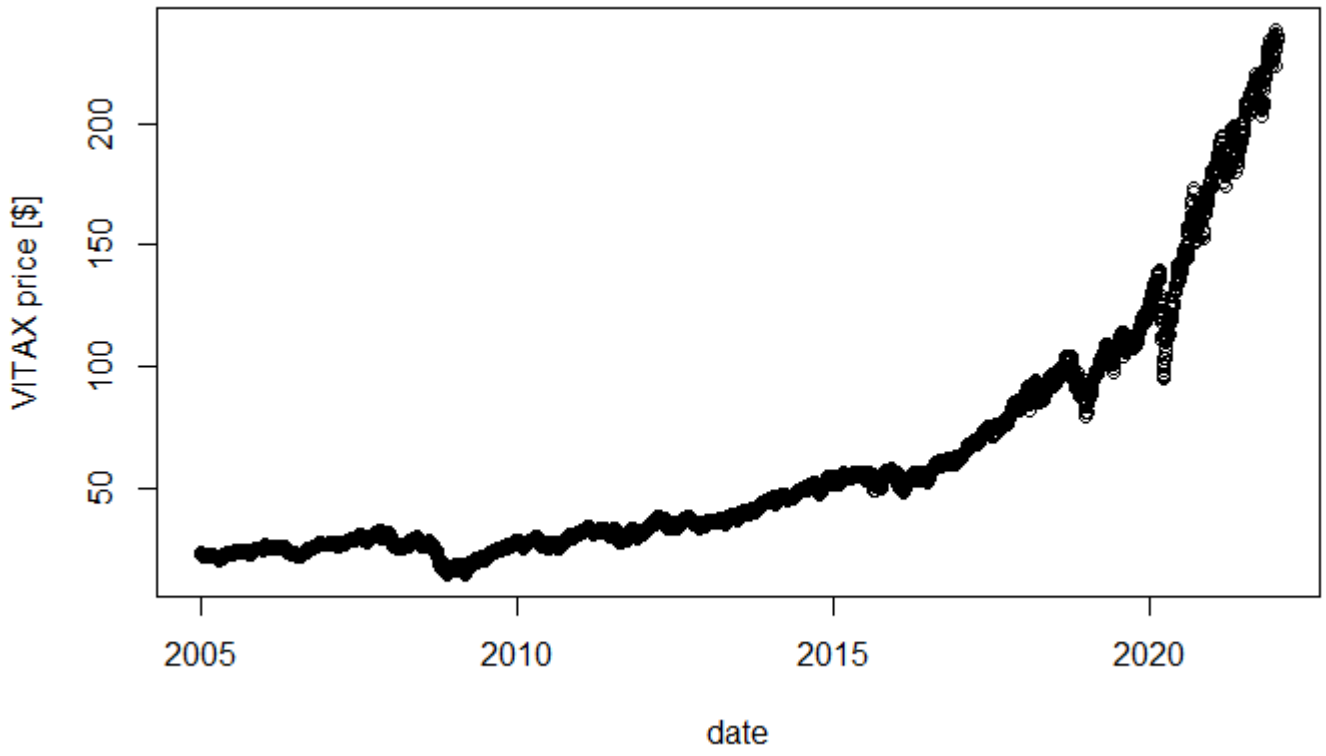
Potential questions to ask the data (some easy some not so much) list:

- what is the overall value of the technology sector across time?
- is it susceptible to wide-range economic shocks (2008 lending crash, early 2020 Covid hiatus)?
- how is the trend changing across time?
- can we measure the trend and make extrapolations/forecasts?

First (easy) thing to do is to plot.

Hide

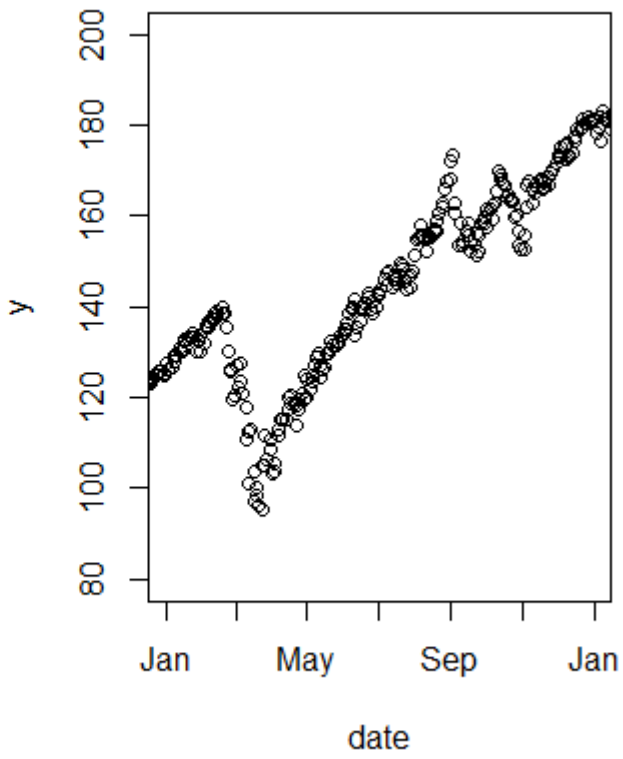
```
with(vitax.df, plot(date, y, ylab='VITAX price [$]'))
```




Definitely trending UP! How about shocks? That's easy, we just need to zoom.

Hide

```
par(mfrow=c(1,2))
with(vitax.df, plot(date, y, xlim=as.Date(c("2008-01-01", "2012-01-01")), ylim=c(10,40)))
with(vitax.df, plot(date, y, xlim=as.Date(c("2020-01-01", "2021-01-01")), ylim=c(80,200)))
```


That was easy :)

Quantification of the trend is going to be more a bit more difficult. At this point most data practitioners use models (aka approximations to reality).

Let's build some simple trend models.

Initially, we convert the date to a numeric feature. There are 2 ways of doing this:

Hide

```
head(as.numeric(vitax.df$date), 10)
```

```
 [1] 12786 12787 12788 12789 12790 12793 12794 12795 12796 12797
```

Hide

```
head(seq_along(vitax.df$date), 10)
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

Hide

```
vitax.df$time = seq_along(vitax.df$date)
head(vitax.df)
```

| | date | y | time |
|---|---|---|---|
| | <date> | <dbl> | <int> |

| | date <date> | y <dbl> | time <int> |
|---|---|---|---|
| 1 | 2005-01-03 | 23.83 | 1 |
| 2 | 2005-01-04 | 23.30 | 2 |
| 3 | 2005-01-05 | 23.16 | 3 |
| 4 | 2005-01-06 | 23.08 | 4 |
| 5 | 2005-01-07 | 23.10 | 5 |
| 6 | 2005-01-10 | 23.16 | 6 |

6 rows

The simplest model is a basic linear model.

$$y = \beta_0 + \beta_1 \cdot time + \epsilon$$

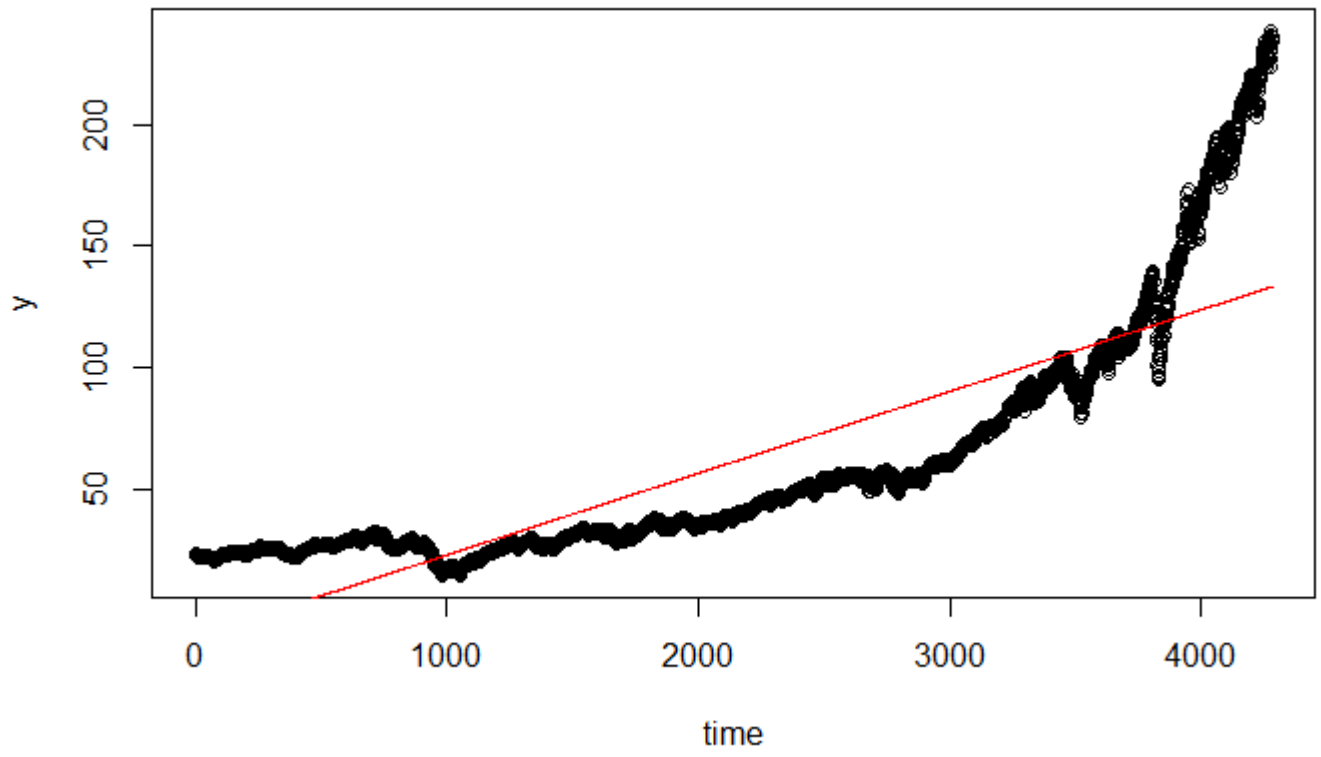
Hide

```
lin.model = lm(y ~ time, data=vitax.df)

summary(lin.model)
```

```

Call:
lm(formula = y ~ time, data = vitax.df)

Residuals:
    Min      1Q  Median      3Q     Max 
-35.210 -20.378  -8.105  18.378 105.390 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -1.045e+01  8.161e-01   -12.8   <2e-16 ***
time         3.360e-02  3.302e-04   101.8   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 26.69 on 4278 degrees of freedom
Multiple R-squared:  0.7076,    Adjusted R-squared:  0.7075 
F-statistic: 1.035e+04 on 1 and 4278 DF,  p-value: < 2.2e-16
```

Hide

```
with(vitax.df, plot(time, y))
lines(predict(lin.model), col='red')
```




Maybe adding a quadratic or cubic term helps? $y = \beta_0 + \beta_1 \cdot time + \beta_2 \cdot time^2 + \epsilon$

$y = \beta_0 + \beta_1 \cdot time + \beta_2 \cdot time^2 + \beta_3 \cdot time^3 + \epsilon$

Hide

```
quad.model = lm(y ~ time + I(time^2), data=vitax.df)
summary(quad.model)
```

```

Call:
lm(formula = y ~ time + I(time^2), data = vitax.df)

Residuals:
    Min      1Q  Median      3Q     Max 
-45.855 -10.721   2.260   7.084  53.349 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  4.196e+01  5.869e-01   71.50   <2e-16 ***
time        -3.984e-02  6.332e-04  -62.92   <2e-16 ***
I(time^2)    1.715e-05  1.432e-07  119.78   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 12.79 on 4277 degrees of freedom
Multiple R-squared:  0.9329,    Adjusted R-squared:  0.9328 
F-statistic: 2.971e+04 on 2 and 4277 DF,  p-value: < 2.2e-16
```

Hide

```
cub.model = lm(y ~ time + I(time^2) + I(time^3), data=vitax.df)
summary(cub.model)
```

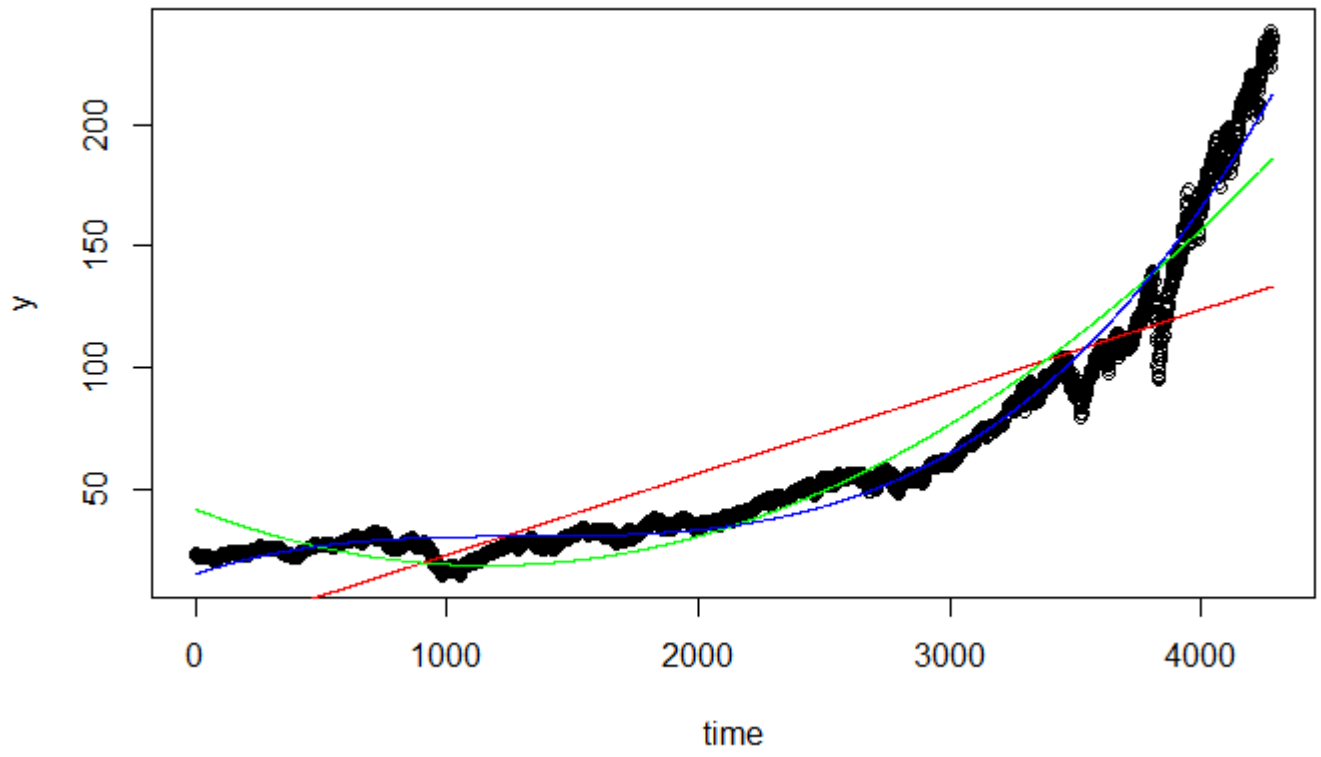
```

Call:
lm(formula = y ~ time + I(time^2) + I(time^3), data = vitax.df)

Residuals:
    Min      1Q  Median      3Q     Max 
-47.122  -3.077   0.612   4.035  26.979 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1.522e+01  4.806e-01   31.66   <2e-16 ***
time         3.508e-02  9.722e-04   36.09   <2e-16 ***
I(time^2)   -2.659e-05  5.277e-07  -50.40   <2e-16 ***
I(time^3)    6.813e-09  8.103e-11   84.08   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.854 on 4276 degrees of freedom
Multiple R-squared:  0.9747,    Adjusted R-squared:  0.9747 
F-statistic: 5.49e+04 on 3 and 4276 DF,  p-value: < 2.2e-16
```

Hide

```
with(vitax.df, plot(time, y))
lines(predict(lin.model), col='red')
```

Hide

```
lines(predict(quad.model), col='green')
lines(predict(cub.model), col='blue')
```




Let's add some higher polynomial terms (up to 6 for now).

Hide

```
null.model = lm(y ~ 1, data=vitax.df)
tetr.model = lm(y ~ poly(time, degree=4), data=vitax.df)
pent.model = lm(y ~ poly(time, degree=5), data=vitax.df)
hex.model = lm(y ~ poly(time, degree=6), data=vitax.df)
```

A neat way of comparing models that are nested within each other is ANOVA (https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/anova). Nesting occurs when a more complex model includes the less complex model in its specification. ANOVA shall tell us when to stop adding terms. Eventually we hit the limit of polynomial regression modeling.

Caveat: using ANOVA requires certain assumptions to be met. We will not delve into them now, but it's good to keep in mind that any statistical test is most powerful when the underlying assumptions are met.

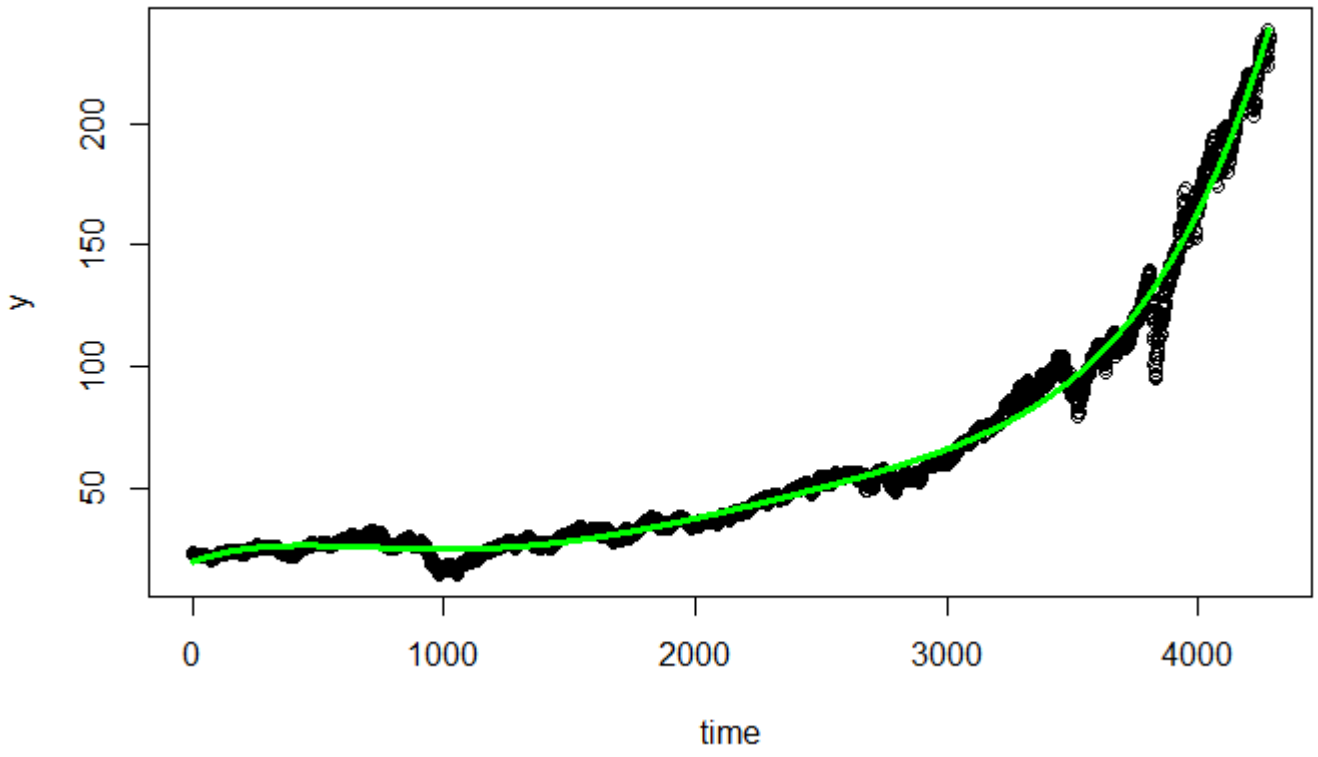
Hide

```
anova(null.model, lin.model, quad.model, cub.model, tetr.model, pent.model, hex.model)
```

```
Analysis of Variance Table

Model 1: y ~ 1
Model 2: y ~ time
Model 3: y ~ time + I(time^2)
Model 4: y ~ time + I(time^2) + I(time^3)
Model 5: y ~ poly(time, degree = 4)
Model 6: y ~ poly(time, degree = 5)
Model 7: y ~ poly(time, degree = 6)
  Res.Df      RSS Df Sum of Sq          F  Pr(>F)
1   4279 10423340
2   4278  3047659  1   7375681 3.0512e+05 < 2e-16 ***
3   4277   699851  1   2347808 9.7125e+04 < 2e-16 ***
4   4276   263774  1    436077 1.8040e+04 < 2e-16 ***
5   4275   149541  1    114233 4.7257e+03 < 2e-16 ***
6   4274   103444  1     46097 1.9070e+03 < 2e-16 ***
7   4273   103291  1       153 6.3214e+00 0.01197 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Hide

```
with(vitax.df, plot(time, y))
lines(predict(pent.model), col='green', lw=3)
```




Polynomial regression suggests that a 5th degree polynomial fits the data well.

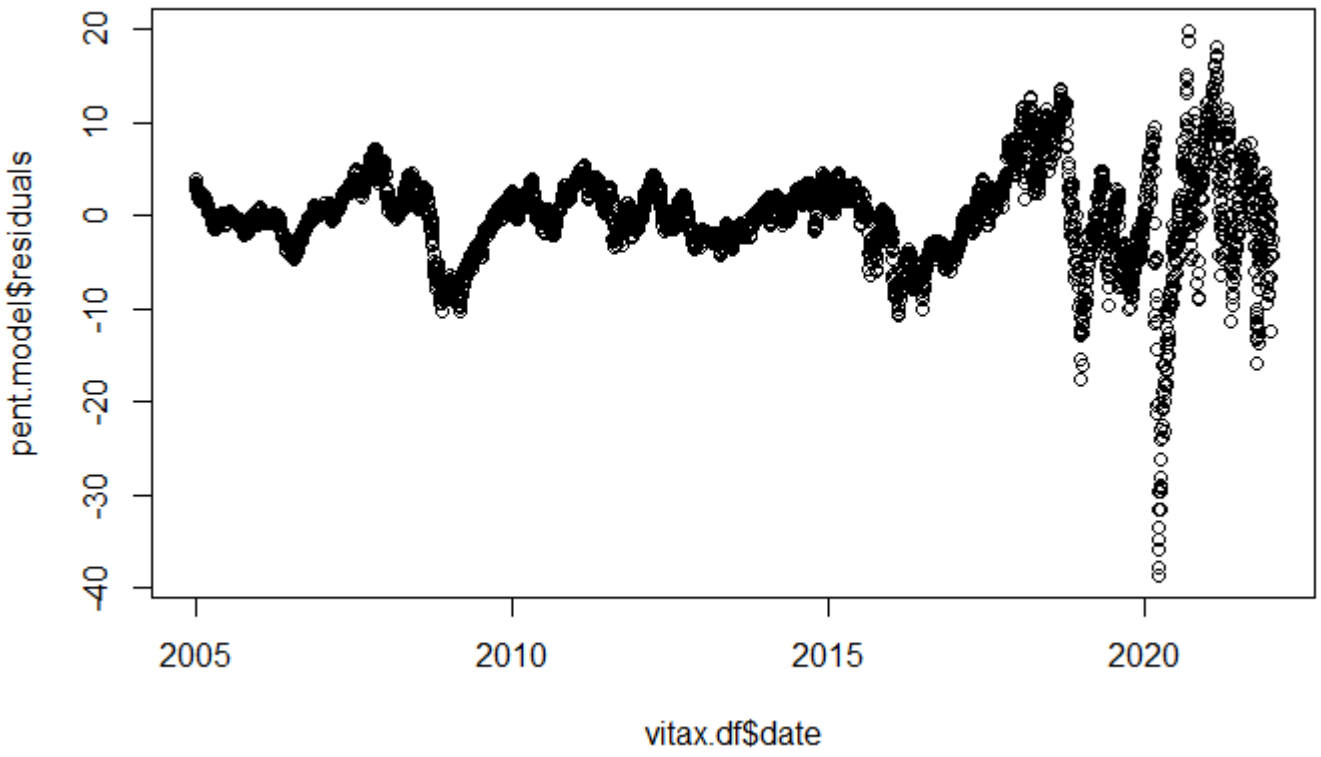
Hide

```
summary(pent.model)
```

```

Call:
lm(formula = y ~ poly(time, degree = 5), data = vitax.df)

Residuals:
    Min      1Q  Median      3Q     Max 
-38.610  -2.128   0.187   2.554  19.885 

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)    
(Intercept)               61.4694     0.0752  817.42   <2e-16 ***
poly(time, degree = 5)1 2715.8205     4.9197  552.03   <2e-16 ***
poly(time, degree = 5)2 1532.2558     4.9197  311.45   <2e-16 ***
poly(time, degree = 5)3  660.3610     4.9197  134.23   <2e-16 ***
poly(time, degree = 5)4  337.9843     4.9197   68.70   <2e-16 ***
poly(time, degree = 5)5  214.7018     4.9197   43.64   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.92 on 4274 degrees of freedom
Multiple R-squared:  0.9901,    Adjusted R-squared:  0.9901 
F-statistic: 8.528e+04 on 5 and 4274 DF,  p-value: < 2.2e-16
```

Hide

```
plot(vitax.df$date, pent.model$residuals)
```



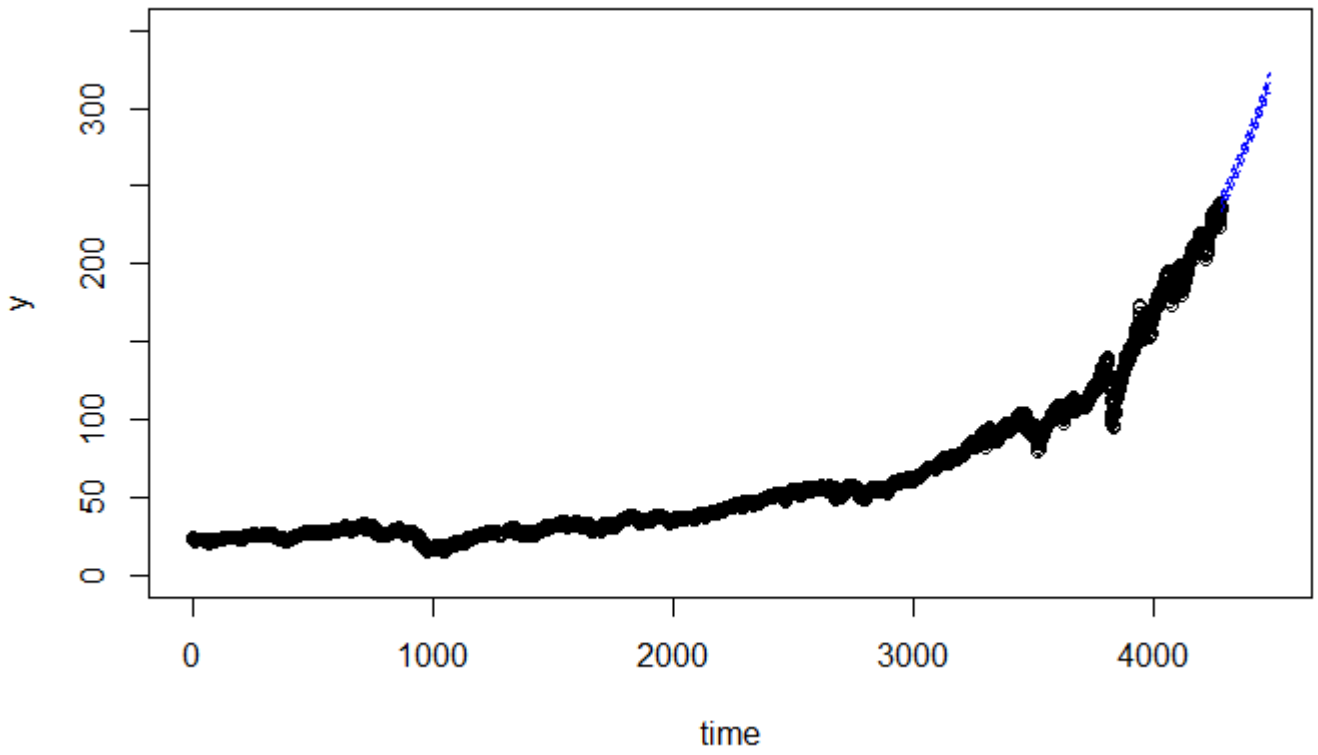

This doesn't look like such an awesome model, the residuals are bigger for more recent days. But it has a good $R^2$ and maybe we can try to make some extrapolations/forecasts from it.

Hide

```
future_times = data.frame(time=seq(4281, 4480))

future_values = predict(pent.model, future_times, se.fit=TRUE)

with(vitax.df, plot(time, y, xlim=c(1, 4480), ylim=c(0, 350)))
lines(future_times$time, future_values$fit, col='blue')
```

Hide

```
lines(future_times$time, future_values$fit + future_values$se + future_values$residual.scale, co
l='blue', lty=2)
lines(future_times$time, future_values$fit - future_values$se - future_values$residual.scale, co
l='blue', lty=2)
```




The forecast seems sensible, altough uncertainty (~$5) is very small and unrealistic at current levels. Additionally the uncertainty does not increase with time (which we would intuitively expect). Finally, the model is not very interpretable, the coefficients values are not meaningful. How can we make the modeling/analysis a bit more transparent? First and foremost the time-series data is strongly correlated, i.e., daily value of a fund depends mostly on its recent history. In time-series one often **transforms** the data prior to analysis.

Question: What do you think is a sensible transform for a time-series?

Hide

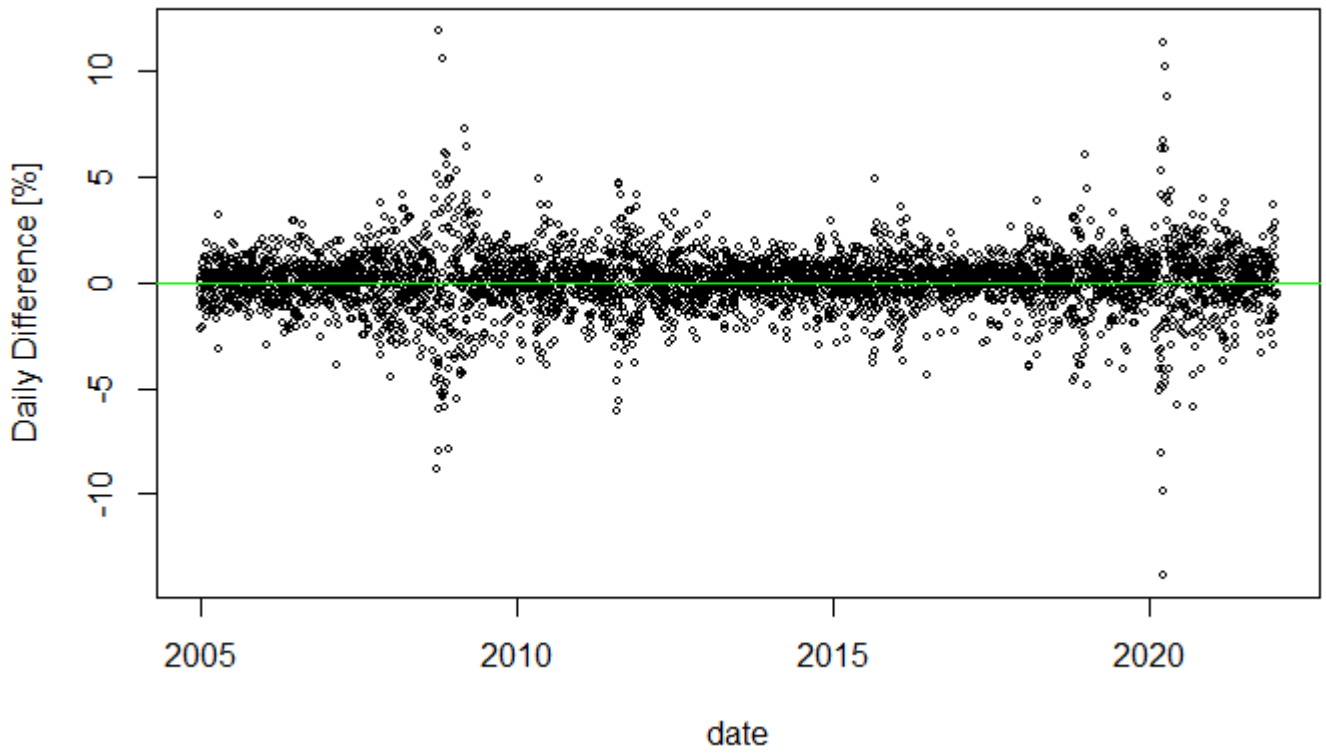
```
vitax.df$ylag1 = c(NA, vitax.df$y[1:nrow(vitax.df)-1])
vitax.df$yd = vitax.df$y - vitax.df$ylag1
vitax.df$yrd = (vitax.df$yd/vitax.df$ylag1) *100
head(vitax.df)
```

| | date <date> | y <dbl> | time <int> | ylag1 <dbl> | yd <dbl> | yrd <dbl> |
|---|---|---|---|---|---|---|
| 1 | 2005-01-03 | 23.83 | 1 | *NA* | *NA* | *NA* |
| 2 | 2005-01-04 | 23.30 | 2 | 23.83 | -0.530001 | -2.22409148 |
| 3 | 2005-01-05 | 23.16 | 3 | 23.30 | -0.139999 | -0.60085410 |
| 4 | 2005-01-06 | 23.08 | 4 | 23.16 | -0.080000 | -0.34542314 |
| 5 | 2005-01-07 | 23.10 | 5 | 23.08 | 0.020000 | 0.08665511 |
| 6 | 2005-01-10 | 23.16 | 6 | 23.10 | 0.060000 | 0.25974026 |

6 rows

Hide

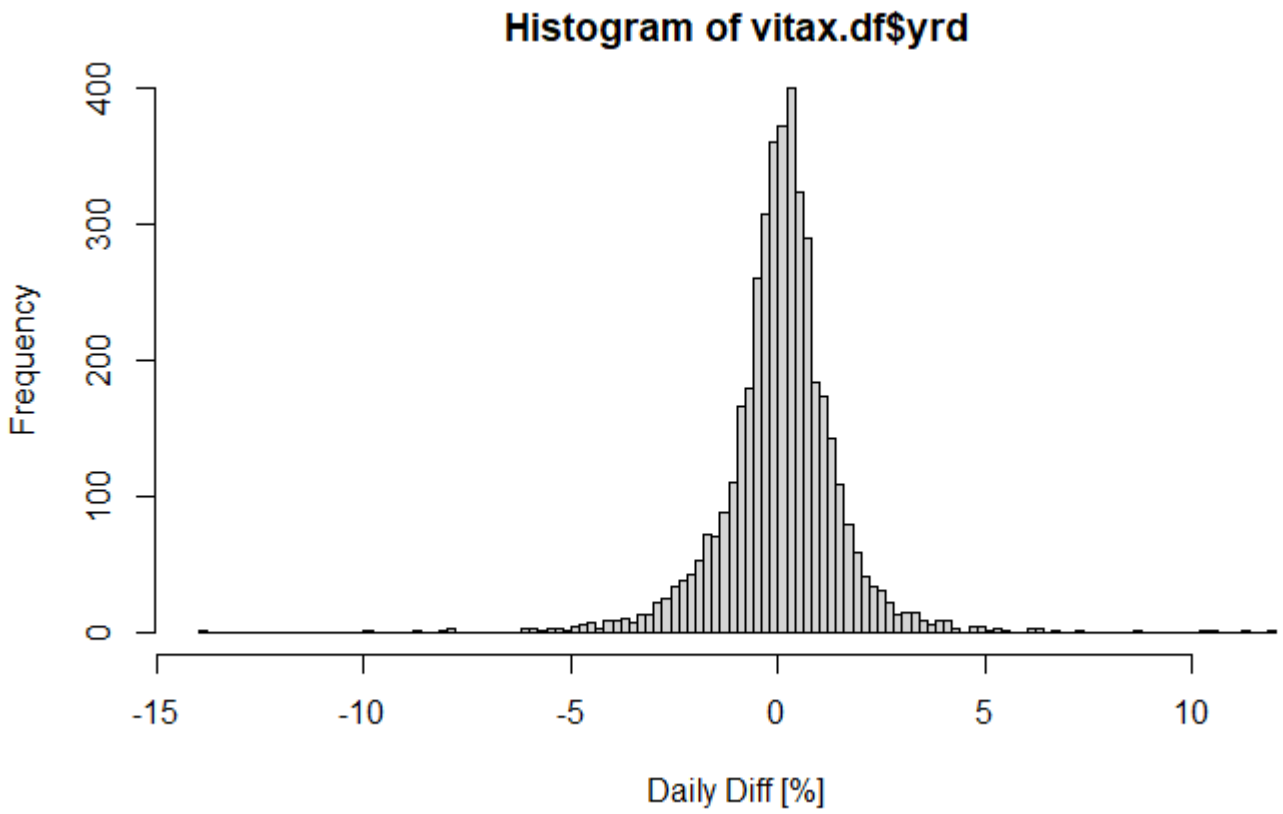
```
with(vitax.df, plot(date, yrd, ylab="Daily Difference [%]", cex=0.5))
abline(0,0, lty=1, col='green')
```




Relative differencing removes the effect of scale and shifts the focus on daily changes.

Hide

```
hist(vitax.df$yrd, breaks=100, xlab='Daily Diff [%]')
```




The distribution of differences is bell-like, although has quite long tails. Since we know that the overall trend is positive, the distribution of differences must be shifted above 0.

Question: How can we smooth these differences to see a larger picture?

Hide

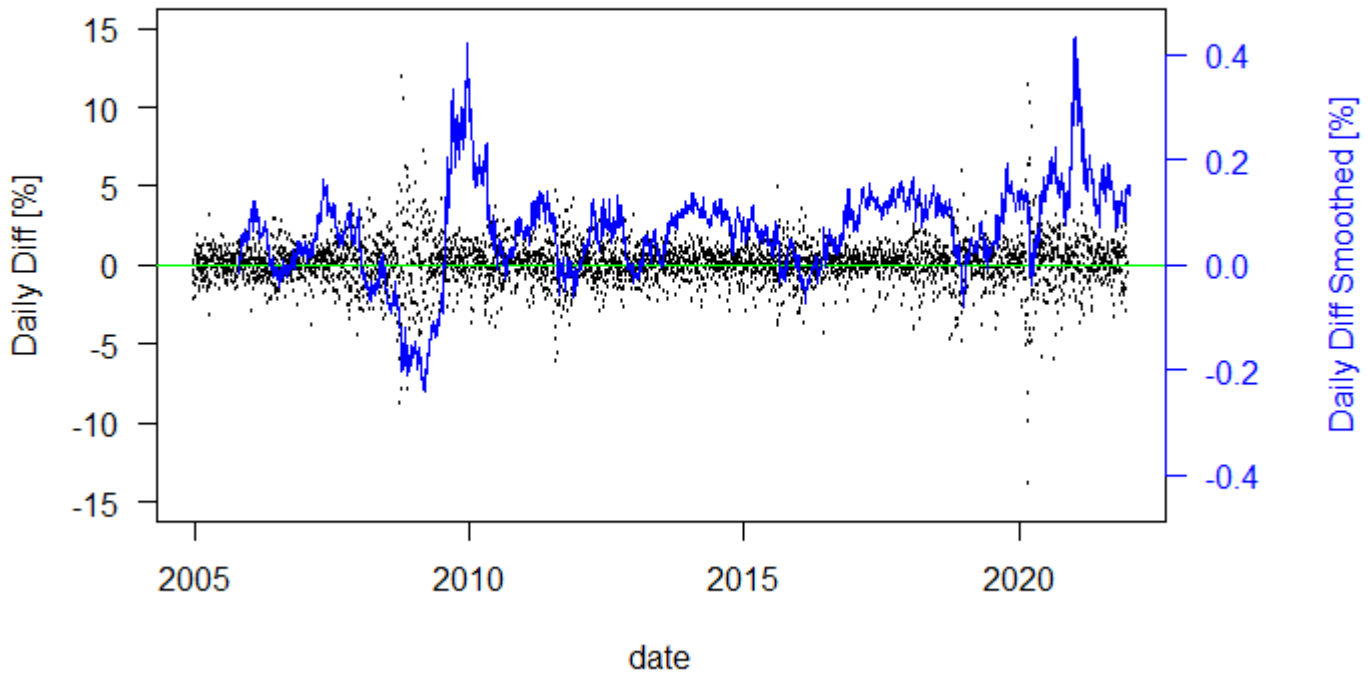
```
# take a relative difference with a larger time window

n=200
yrd_smooth = (diff(vitax.df$y, n) / vitax.df$y[(n+1):nrow(vitax.df) - n])*100/n
vitax.df$yrd_smooth = c(rep(NA, n), yrd_smooth)

# plot it
par(mar=c(5, 4, 4, 6) + 0.1)
with(vitax.df, plot(date, yrd, cex=0.3, ylab='Daily Diff [%]', las=1, ylim=c(-15,15)))
abline(0,0, lty=1, col='green')
```

Hide

```
par(new=TRUE) # add to the current already filled plot
with(vitax.df, plot(date, yrd_smooth, col='blue', type='l', axes=F, xlab='', ylab='', ylim=c(-0.
45,0.45)))
```

Hide

```
# draw separate axes
mtext('Daily Diff Smoothed [%]', side=4, col='blue', line=4)
axis(4, col='blue', col.axis='blue', las=1, )
```




This is a different and quite informative picture of the time-series.

- Longer period derivatives smooth out the noise and better illustrate the pace of growth
  - tech has been growing fast in various periods 2009-2011 following the 2008 recession
  - ~2014 and even faster ~2017-2018
  - post Covid growth is the fastest of all with no strong signs of tapering

## Time-series dedicated modeling

R has a multitude of dedicated libraries to make predictions on time-series. A common one is called `forecast`.

Auto.arima methodology automatically handles the choice of differentiation and the auto-regressive or smoothing components that best describe the dynamics of a given time series.

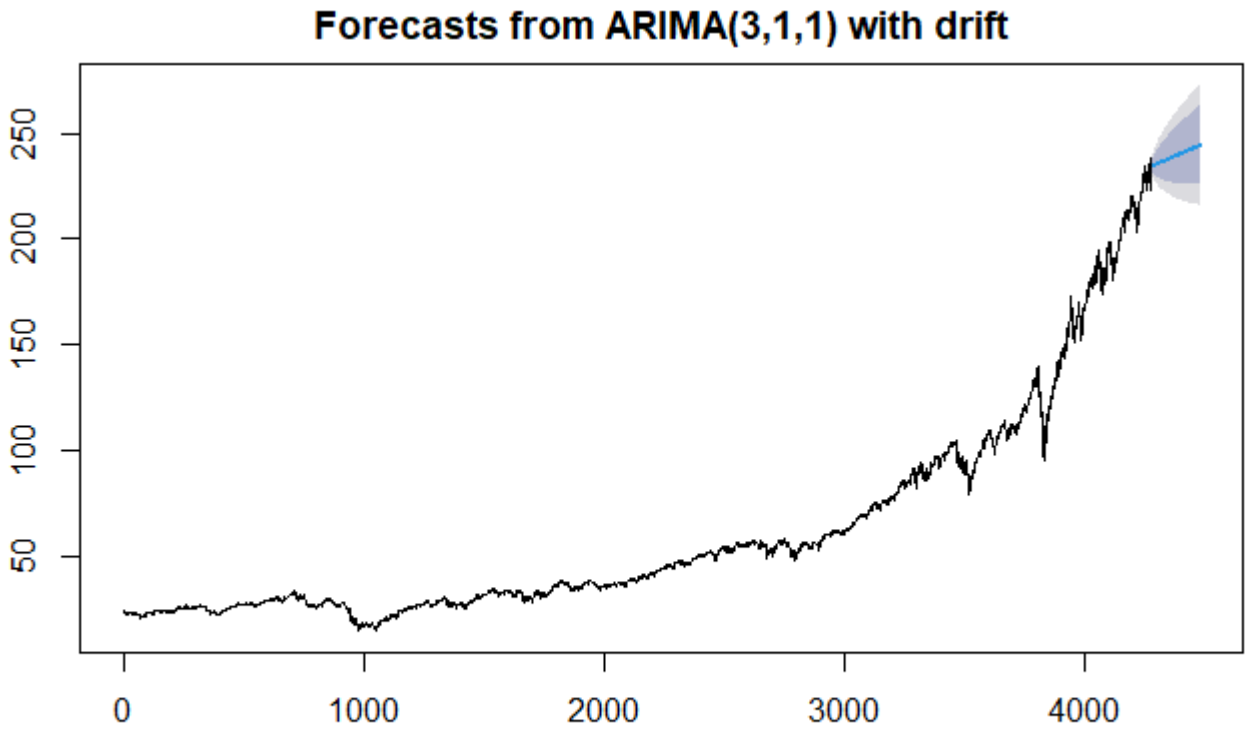
Hide

```
require(forecast)
```

```
Loading required package: forecast
Warning: package 'forecast' was built under R version 4.1.2
```

Hide

```
# a common rudimentary time-series foreacsting method is ARIMA (autoregressive-integrated-moving
-average)
arima.model = forecast::auto.arima(vitax$VITAX.Close, max.p=3, max.d=1, max.q=3,
                                   seasonal = FALSE, stationary = FALSE)

# with the fitted model we can make predictions
arima.forecast = forecast::forecast(arima.model, h=200)
plot(arima.forecast)
```




## Key Points

1. Every data problem is unique to the process that generated the data, e.g.,

- physical process- one needs to know the laws of nature to describe it)
- human (or society) induced process - often no or very little valid theories exist

2. One can model the data in various ways, linear regression is often a good start but may not be very satisfactory and result in an incomprehensible set of coefficients/weights, often dedicated modeling techniques exist for specific data.
3. Simple transformations may help illustrate useful information even without explicit modeling