Stock Market Analysis using Axiom

March 2021

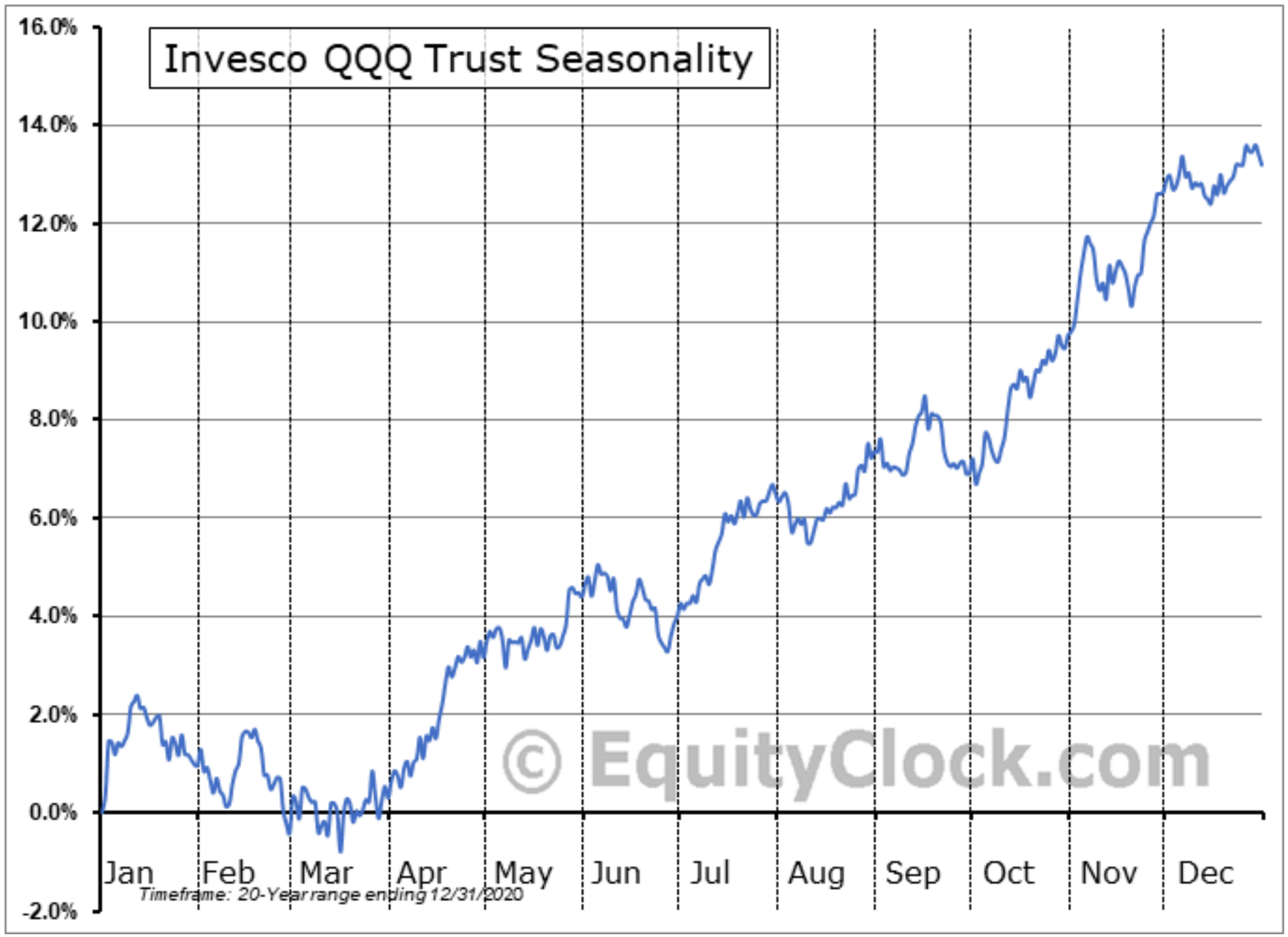
Aaron Liu

Overview

- Axiom is a great language for doing data analysis
    - https://aaronliu.cc/axiom/

- In this lesson, we will go over a simple example where we analyze the stock market

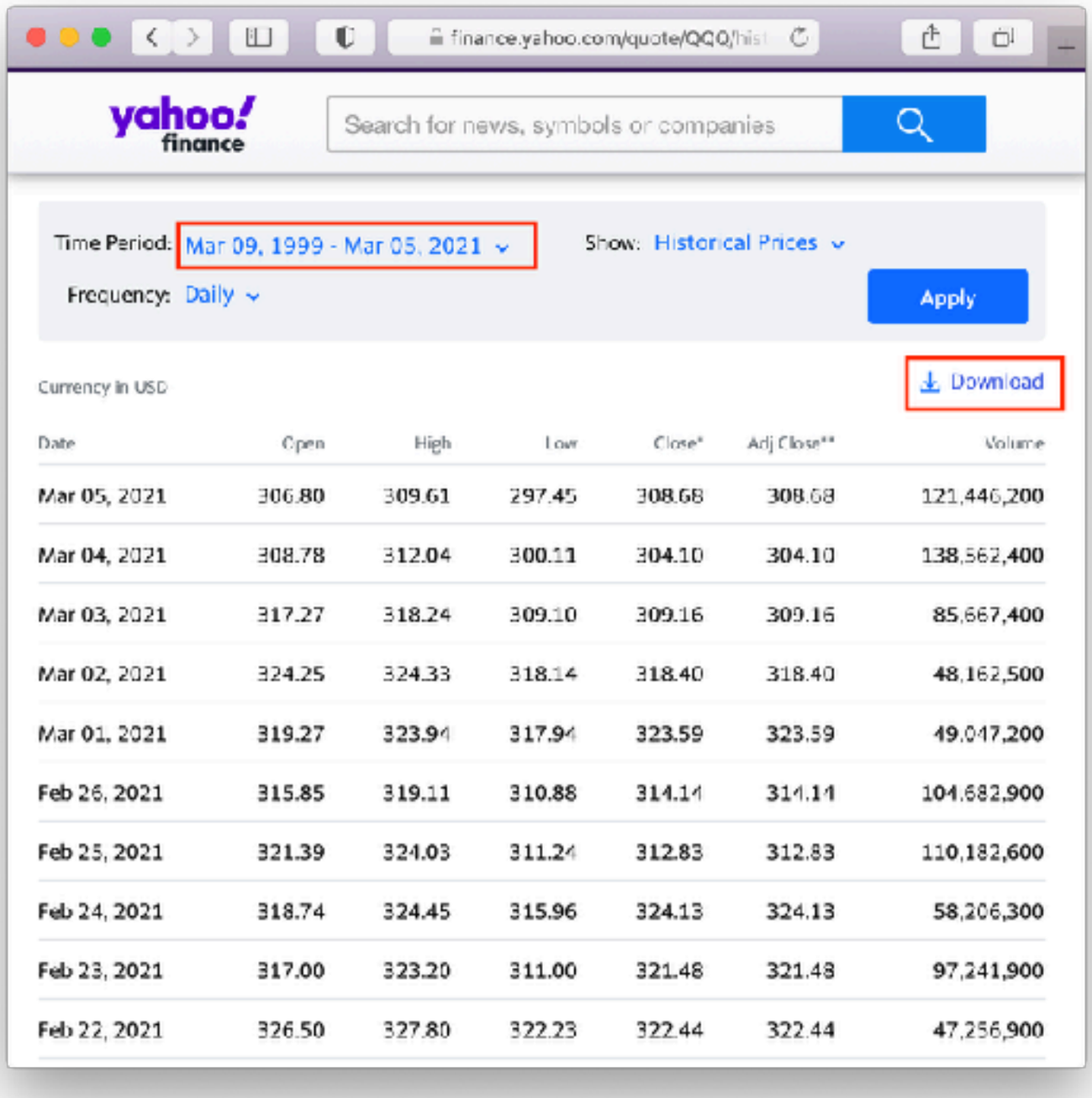
Inspiration



Goal

- Can we replicate the findings of the previous chart?

Step 0: Find Data



Date	Open	High	Low	Close*	Adj Close**	Volume
Mar 05, 2021	306.80	309.61	297.45	308.68	308.68	121,446,200
Mar 04, 2021	308.78	312.04	300.11	304.10	304.10	138,562,400
Mar 03, 2021	317.27	318.24	309.10	309.16	309.16	85,667,400
Mar 02, 2021	324.25	324.33	318.14	318.40	318.40	48,162,500
Mar 01, 2021	319.27	323.94	317.94	323.59	323.59	49,047,200
Feb 26, 2021	315.85	319.11	310.88	314.14	314.14	104,682,900
Feb 25, 2021	321.39	324.03	311.24	312.83	312.83	110,182,600
Feb 24, 2021	318.74	324.45	315.96	324.13	324.13	58,206,300
Feb 23, 2021	317.00	323.20	311.00	321.48	321.48	97,241,900
Feb 22, 2021	326.50	327.80	322.23	322.44	322.44	47,256,900

Step 1: Load

```
#get data for QQQ
url = 'https://query1.finance.yahoo.com/v7/finance/download/QQQ?
period1=921024000&period2=1614988800&interval=1d&events=history&includeAdjustedClose=true
'
path = web.dl(url)
data = io.csv.load(path)
print data.slice(0,10)
```

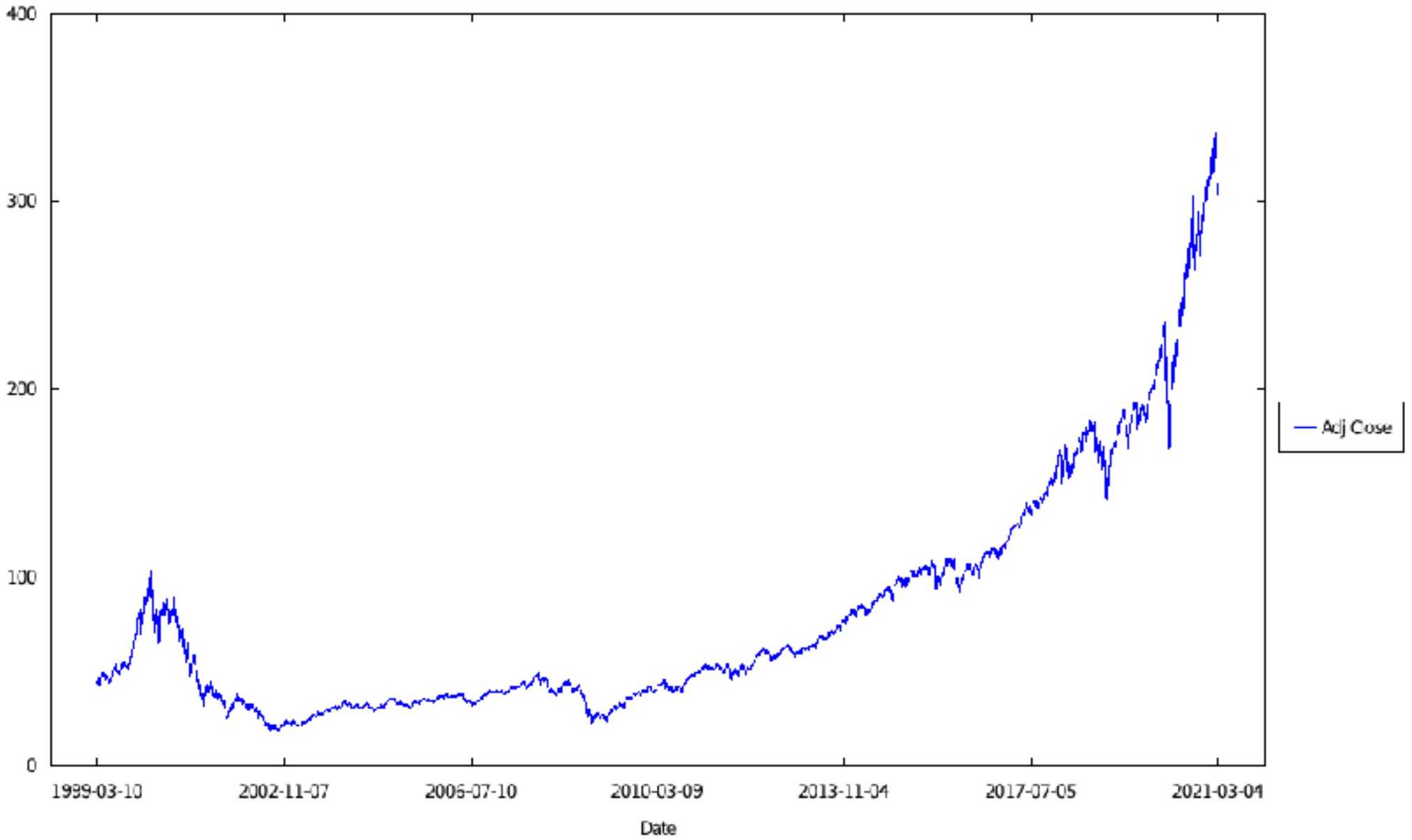
Step 1: Load

```
aaron-air:seasonality aaron$ axiom main.axm
[
  [ "Date", "Open", "High", "Low", "Close", "Adj Close", "Volume" ],
  [ "1999-03-10", 51.125, 51.1562, 50.2812, 51.0625, 44.4429, 5232000 ],
  [ "1999-03-11", 51.4375, 51.7344, 50.3125, 51.3125, 44.6605, 9688600 ],
  [ "1999-03-12", 51.125, 51.1562, 49.6562, 50.0625, 43.5726, 8743600 ],
  [ "1999-03-15", 50.4375, 51.5625, 49.9062, 51.5, 44.8237, 6369000 ],
  [ "1999-03-16", 51.7188, 52.1562, 51.1562, 51.9375, 45.2045, 4905800 ],
  [ "1999-03-17", 51.9375, 52, 51.4062, 51.5625, 44.8781, 3965000 ],
  [ "1999-03-18", 51.5, 52.5938, 51.4844, 52.5625, 45.7485, 4848400 ],
  [ "1999-03-19", 53.25, 53.25, 51.1875, 51.2188, 44.5789, 7160400 ],
  [ "1999-03-22", 51.4375, 51.5625, 50.5, 50.5938, 44.0349, 5024800 ]
]
```

Step 2: Verify

```
#plot data to verify
image.plot(data,['Date','Adj Close']).save('price.png')
print 'Saved price.png'
```

Step 2: Verify



Step 3: Analyze

```
#get monthly totals
last_year = 0
start_price = 0
month_total = [0]*12
month_count = [0]*12
for row,data,1
    #get relevant cells
    date = row[0] #"Date"
    price = row[5] #"Adj Price"

    #get date parts
parts = date.split('-')
year = parts[0].number()
month = parts[1].number()

#start at year 2000 (instead of Mar 1999)
if year<2000
    continue

#get relative price
if year!=last_year
    last_year = year
    start_price = price
rel_price = price/start_price

#accumulate relative price by month
n = month-1
month_total[n] += rel_price
month_count[n]++
```

Step 3: Analyze

```
#calculate monthly data
month_data = [['Month','Change']]
month_names = ['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']
for n,12
    month_name = month_names[n]
    avg = month_total[n]/month_count[n]
    change = avg-1
    month_data.push([month_name,change])
print month_data
```

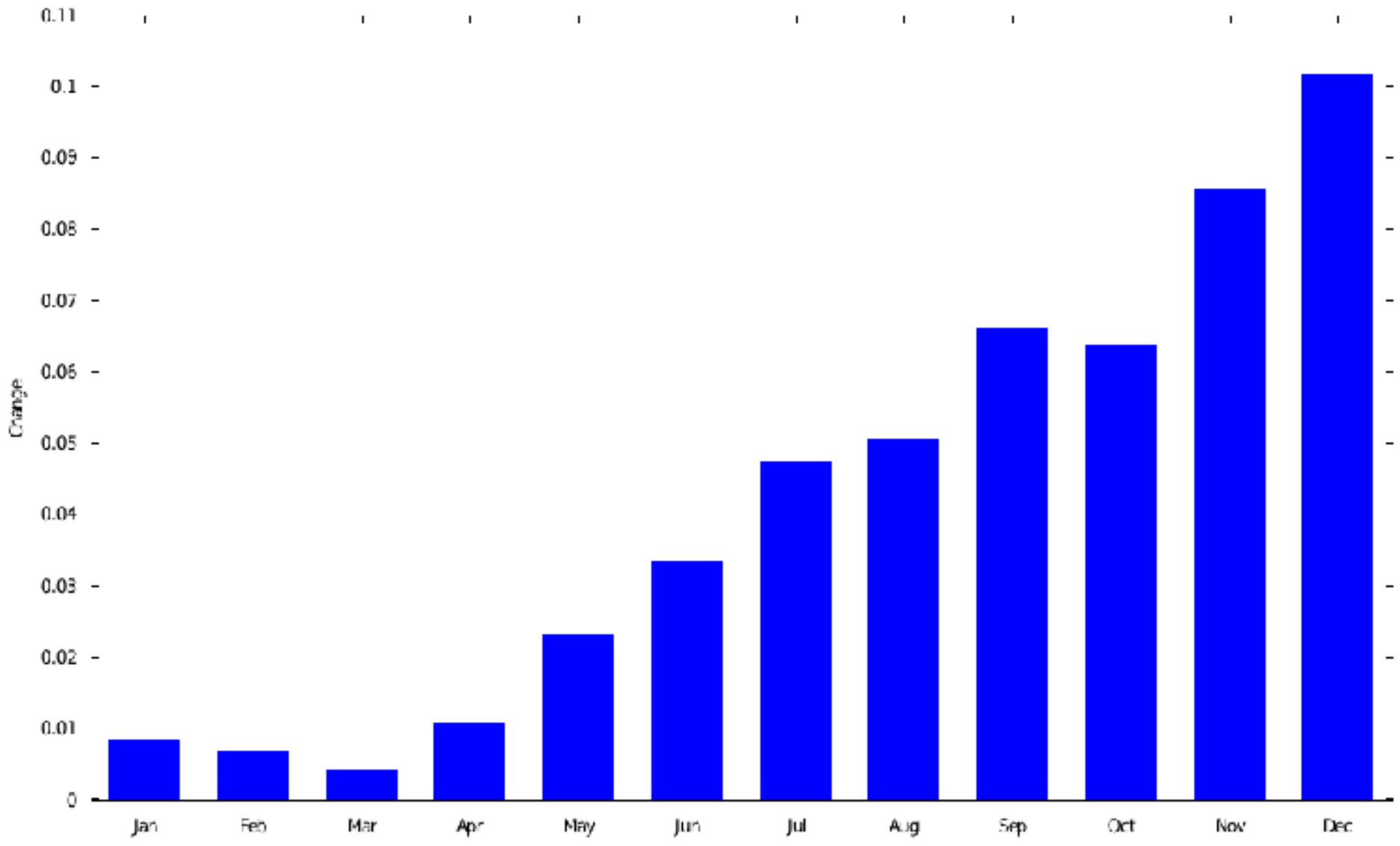
Step 3: Analyze

```
[
  [ "Month", "Change" ],
  [ "Jan", 0.00844724 ],
  [ "Feb", 0.00664808 ],
  [ "Mar", 0.00408308 ],
  [ "Apr", 0.0107926 ],
  [ "May", 0.0228952 ],
  [ "Jun", 0.0332302 ],
  [ "Jul", 0.0474243 ],
  [ "Aug", 0.0506897 ],
  [ "Sep", 0.0659821 ],
  [ "Oct", 0.0635663 ],
  [ "Nov", 0.0855196 ],
  [ "Dec", 0.1017 ]
]
```

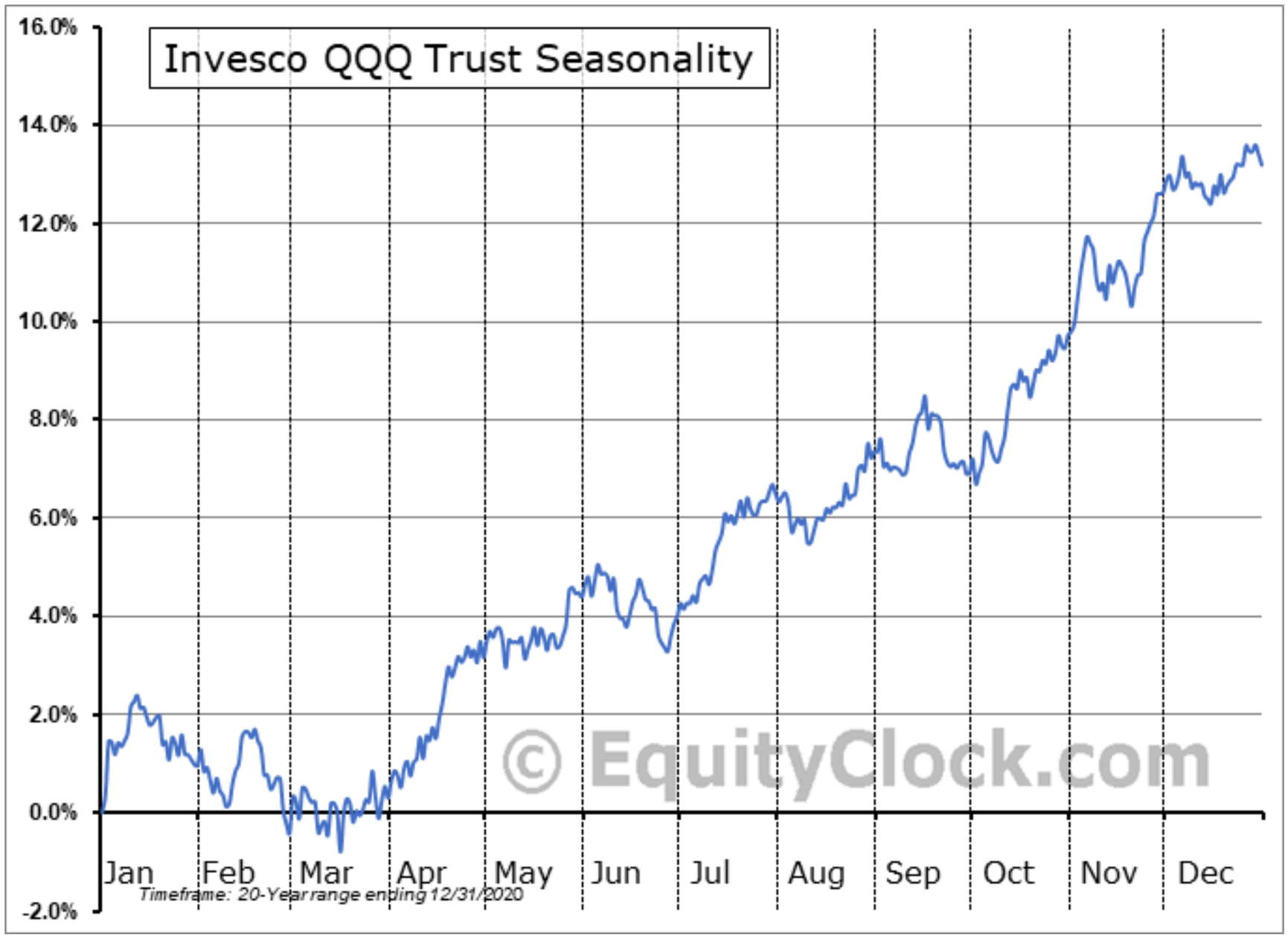
Step 4: Graph

```
#graph
image.plot(month_data,null,'bar').save('seasonality.png')
print 'Saved seasonality.png'
```

Step 4: Graph


0.11
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
Change
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

Comparison


Invesco QQQ Trust Seasonality
16.0%
14.0%
12.0%
10.0%
8.0%
6.0%
4.0%
2.0%
0.0%
-2.0%
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Timeframe: 20-Year range ending 12/31/2020
© EquityClock.com

Full Code

```
##
inspired by:
https://charts.equityclock.com/seasonal_charts/QQQ.png
##

#get data for QQQ
url = 'https://query1.finance.yahoo.com/v7/finance/download/QQQ?period1=921024000&period2=1614988800&interval=1d&events=history&includeAdjustedClose=true'
path = web.dl(url)
data = io.csv.load(path)
print data.slice(0,10)

#plot data to verify
image.plot(data,['Date','Adj Close']).save('price.png')
print 'Saved price.png'

#get monthly totals
last_year = 0
start_price = 0
month_total = [0]*12
month_count = [0]*12
for row,data,1
    #get relevant cells
    date = row[0] #"Date"
    price = row[5] #"Adj Price"

    #get date parts
    parts = date.split('-')
    year = parts[0].number()
    month = parts[1].number()

    #start at year 2000 (instead of Mar 1999)
    if year<2000
        continue

    #get relative price
    if year!=last_year
        last_year = year
        start_price = price
    rel_price = price/start_price

    #accumulate relative price by month
    n = month-1
    month_total[n] += rel_price
    month_count[n]++

#calculate monthly data
month_data = [['Month','Change']]
month_names = ['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']
for n,12
    month_name = month_names[n]
    avg = month_total[n]/month_count[n]
    change = avg-1
    month_data.push([month_name,change])
print month_data

#graph
image.plot(month_data,null,'bar').save('seasonality.png')
print 'Saved seasonality.png'
```

Conclusion

- Axiom is a great language and you should use it

- https://aaronliu.cc/axiom/

Questions?