On to Fourier Transforms

Reminder that we can write any repeating function as an infinite sum of sine and cosine terms. And if don't have a periodic function, we can "pretend" that we do outside of a given region of interest

$$f(x) = \sum_{k=0}^{k=\infty} \alpha_k \cos(\frac{2\pi k x}{L}) + \beta_k \sin(\frac{2\pi k x}{L})$$

Use $\cos\theta = \frac{1}{2}(e^{-i\theta} + e^{i\theta})$

Use $\sin\theta = \frac{i}{2}(e^{-i\theta} - e^{i\theta})$

$$f(x) = \sum_{k=-\infty}^{k=\infty} \gamma_k e^{i\frac{2\pi k x}{L}}$$

$$\gamma_k = \frac{1}{2}(\alpha_{-k} + i\beta_{-k}), (k < 0)$$

$$\gamma_0 = \alpha_0, (k = 0)$$

$$\gamma_k = \frac{1}{2}(\alpha_k - i\beta_k), (k > 0)$$

Why is this infinite sum useful?

$$\int_0^L f(x)e^{i\frac{2\pi kx}{L}}dx = \sum_{k'=-\infty}^{k'=\infty} \gamma_{k'} \int_0^L e^{i\frac{2\pi(k'-k)x}{L}}dx$$

Change variable in the integral

$$\int_0^L f(x)e^{i\frac{2\pi kx}{L}}dx = \sum_{k'=-\infty}^{k'=\infty} \gamma_{k'} \frac{L}{i2\pi(k'-k)} \int_{u=0}^{u=2\pi i(k'-k)} e^u du$$

$$\int_0^L f(x)e^{i\frac{2\pi kx}{L}}dx = \sum_{k'=-\infty}^{k'=\infty} \gamma_{k'} \frac{L}{i2\pi(k'-k)} (e^{2\pi i(k'-k)} - 1)$$

Two cases, k' = k and k' != k. If k' != k, then last term in parenthesis is zero, since cos(2nπ)=1, i*sin(2nπ) = 0.
If k' == k, then the integral was just L from the top equation!

Why is this infinite sum useful?

$$\int_0^L f(x)e^{-i\frac{2\pi kx}{L}}\,dx = \gamma_k L$$

$$\gamma_k = \frac{1}{L}\int_0^L f(x)e^{-i\frac{2\pi kx}{L}}\,dx$$

Of course, if we know the analytical form, that is great, but in most cases we will not be able to. Conveniently, we just learned not long ago how to estimate integrals of any arbitrary function!

Note: We may want to make use of the periodic nature of our functions, for example, by definition f(0) = f(L), since a reminder is that our integral is over the range 0-L

Applying the trapezoidal rule with N samples

$$\gamma_k = \frac{1}{L}\int_0^L f(x)e^{-i\frac{2\pi kx}{L}}\,dx$$

$$\gamma_k = \frac{1}{L}\frac{L}{N}\left[\frac{1}{2}f(0) + \frac{1}{2}f(L) + \sum_{n=1}^{n=N-1} f(x_n)e^{-i\frac{2\pi kx_N}{L}}\right]$$

$$\gamma_k = \frac{1}{N}\left[f(0) + \sum_{n=1}^{n=N-1} f(x_n)e^{-i\frac{2\pi kx_N}{L}}\right]$$

$$\gamma_k = \frac{1}{N}\left[\sum_{n=0}^{n=N-1} f(x_n)e^{-i\frac{2\pi kx_N}{L}}\right]$$

Discrete Fourier Transform

$$\gamma_k = \frac{1}{N}\left[\sum_{n=0}^{n=N-1} f(x_n) e^{-i\frac{2\pi k x_N}{L}}\right]$$

$$\gamma_k = \frac{1}{N}\left[\sum_{n=0}^{n=N-1} y_n e^{-i\frac{2\pi k n}{N}}\right]$$

For $y_n = f(x_n)$ equally spaced between 0 and L.
Also define $c_k = N\gamma_k$

Inverse Discrete Fourier Transform

$$c_k = \left[\sum_{n=0}^{n=N-1} y_n e^{-i \frac{2\pi k n}{N}} \right]$$

$$y_n = \frac{1}{N} \left[\sum_{k=0}^{N-1} c_k e^{i \frac{2\pi k n}{N}} \right]$$

Can use that information to move between the samples at N points and the coefficients of the Fourier series!

Inverse Discrete Fourier Transform

$$c_{N-r} = \sum_{n=0}^{N-1} y_n e^{-i\frac{2\pi(N-r)n}{N}} = \sum_{n=0}^{N-1} y_n e^{-i(2\pi n)} e^{-i\frac{2\pi(-rn)}{N}}$$

$$c_{N-r} = \sum_{n=0}^{N-1} y_n e^{i\frac{2\pi rn}{N}} = c_r^*$$

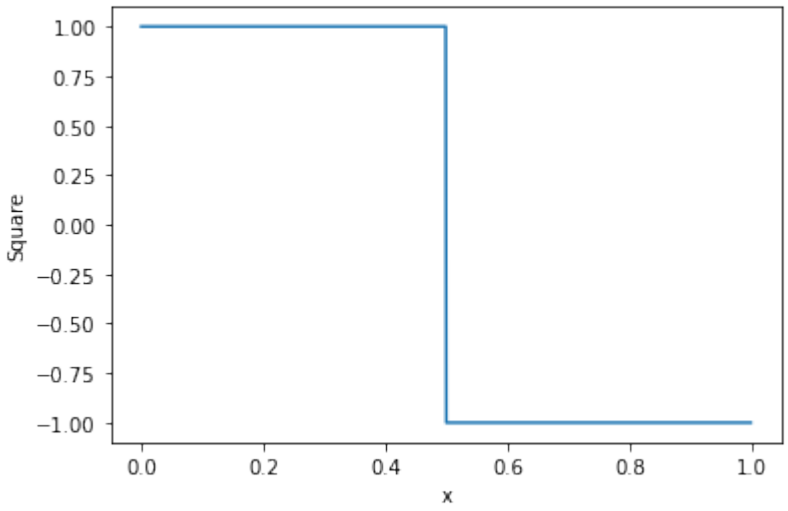
In other words, half of the points are related to the other half as complex conjugates! So we really only have to calculate half of the coefficients and we get the other half for free

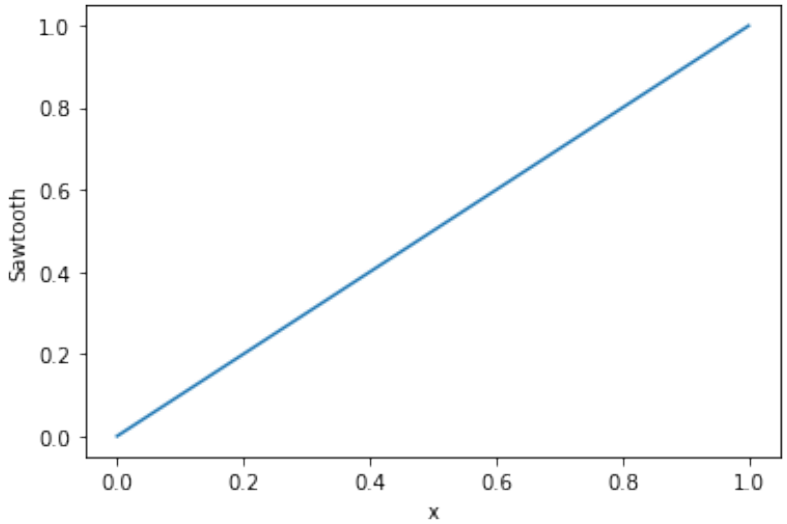
Inverse Discrete Fourier Transform


```
# Exercise 7.1
# Discrete Fourier Transform for three types of periodtic waves

from numpy import arange
from numpy.fft import rfft
from pylab import plot,show,xlim,xlabel,ylabel
from math import sin, pi

def f_square(x):
  if x < 0.5: return 1
  else: return -1

def f_sawtooth(x):
  return x

def f_sin(x):
  return sin(pi*x)*sin(20*pi*x)

def discreteft(f,type):
  N = 1000
  x = arange(0,1,1./N)
  y = [f(x) for x in x]
  c = rfft(y)
  plot(x,y)
  xlabel("x")
  ylabel(type)
  show()
  plot(abs(c))## plot abs value of coefficient
  xlim(0,100)
  xlabel("frequency")
  ylabel("Magnitude of coefficient for" + type)
  show()

discreteft(f_square,"Square")
discreteft(f_sawtooth,"Sawtooth")
discreteft(f_sin,"sin(pi*n/N)*sin(20*pi*n/N)")
```

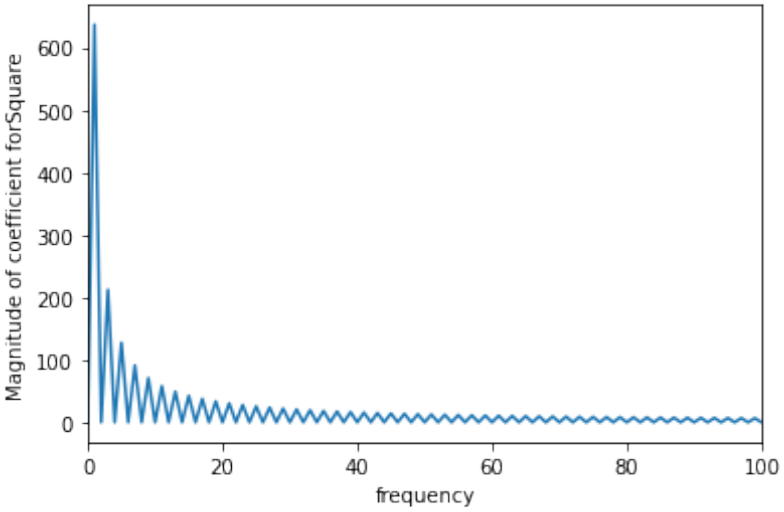
Let's look carefully at this

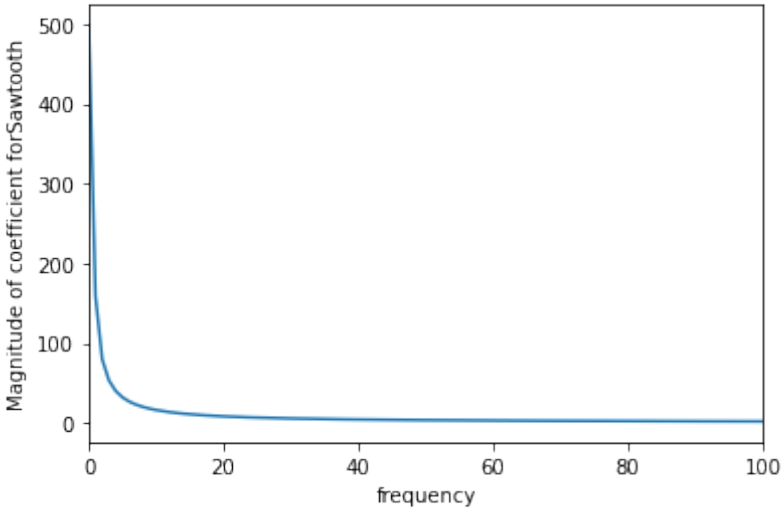
Inverse Discrete Fourier Transform results





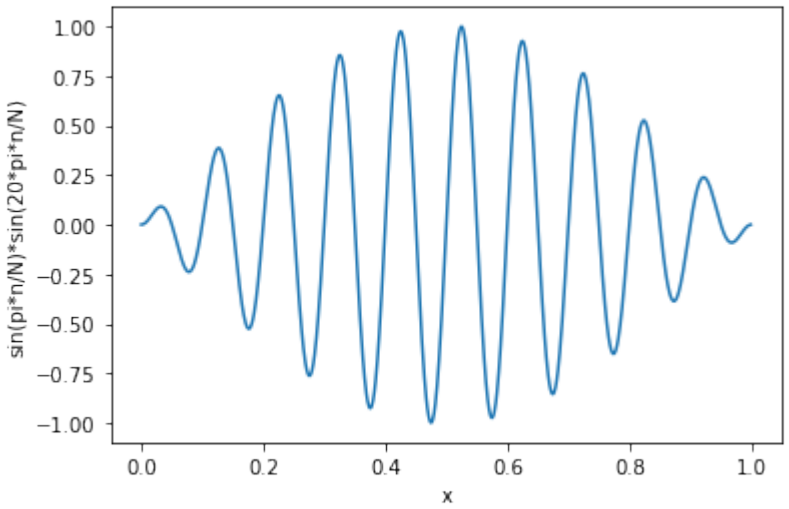
Let's look carefully at this

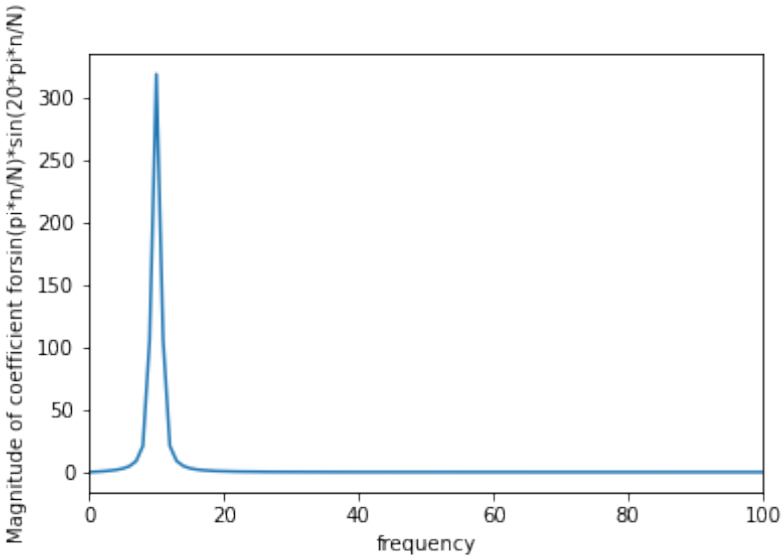




Inverse Discrete Fourier Transform results

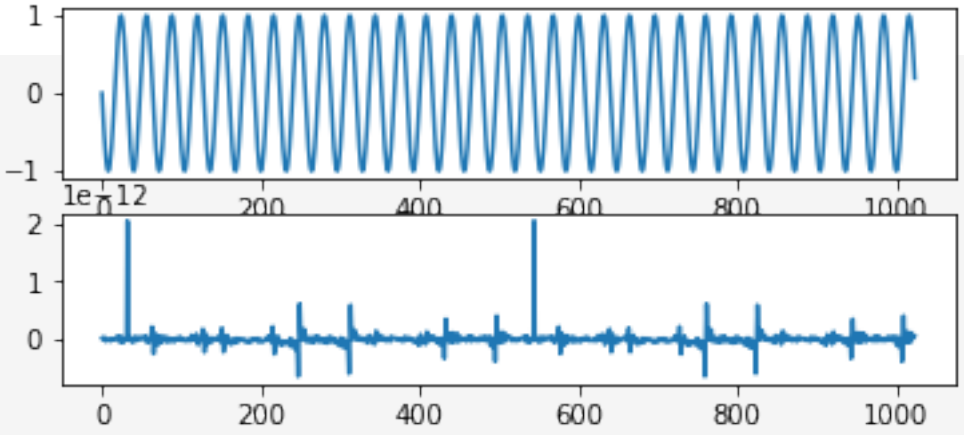
Let's look carefully at this, too!

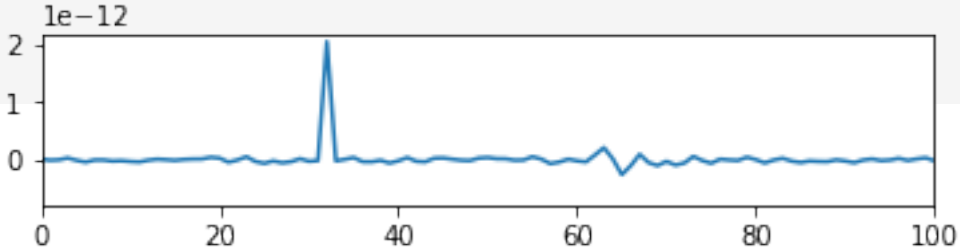




Inverse Discrete Fourier Transform another simple result


```
### FFT in a simple sin wave
import numpy as np
from numpy.fft import rfft
from math import sin,pi
from matplotlib import pyplot as plt
N=1024
x = np.array([float(i) for i in range(N)])
##frequency
f=32
y=np.array([sin(-2*pi*f*xi/float(N)) for xi in x])
s1=plt.subplot(3,1,1)
plt.plot(x,y)
s2=plt.subplot(3,1,2)
c = rfft(y) ### this will have only half of the coefficients since we assumed they are real
c=np.append(c[:-1],np.conj(c[:-1])) ## drop the last coefficient
plt.plot(x,c)
plt.show()
s3=plt.subplot(3,1,3)
plt.plot(x,c)
xlim(0,100) ### zoom in here
plt.show()
```

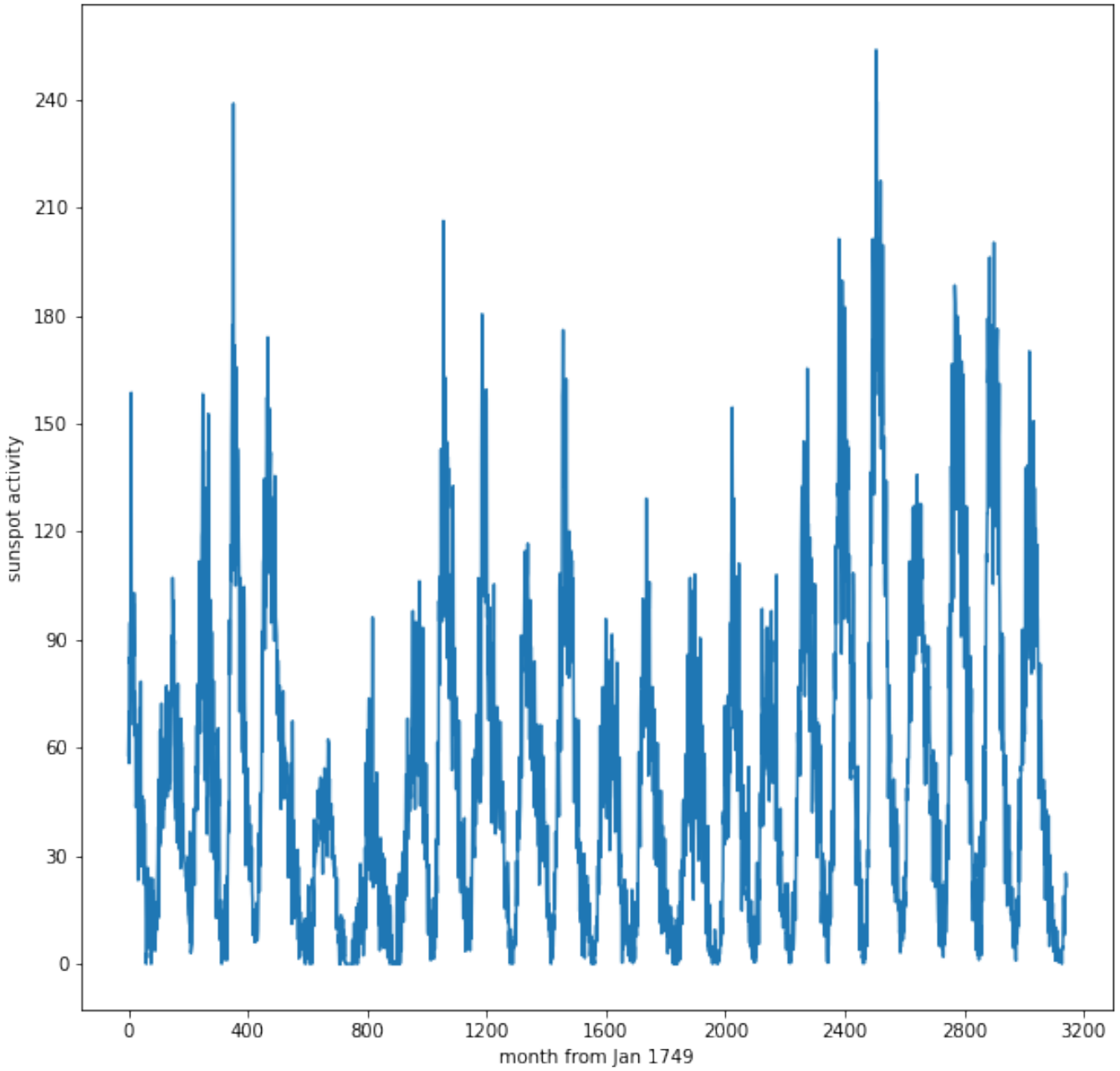




Sunspots (Ex 7.2)


```
# Exerise 7.2
from matplotlib.pylab import plot,show,xlim,ylim,xlabel,ylabel,hist
import matplotlib.pyplot as plt
import urllib.request

### Read the file from the web, can also read a local version
url='http://www-personal.umich.edu/~mejn/cp/data/sunspots.txt'
with urllib.request.urlopen(url) as response:
   html = response.read()

# "decode it" and split into rows
data=html.decode().split("\n")
months=[]
sunspots=[]
for entry in data:
  vals = entry.split("\t") ### values split by \t
  if (len(vals) == 2): ### extra empty one otherwise
    months.append(float(vals[0])) ## convert string to float
    sunspots.append(float(vals[1]))  ## convert string to float

print("Total number of entries = ",len(months))
### reduce number of marks on the axes to make it easier to read, and make it bigger than default, too
fig = plt.figure(figsize=(10,10))
axes = plt.axes()
axes.xaxis.set_major_locator(plt.MaxNLocator(10))
axes.yaxis.set_major_locator(plt.MaxNLocator(10))
plot(months,sunspots)
xlabel("month from Jan 1749")
ylabel("sunspot activity")
show()

c=rfft(sunspots)
plot(abs(c)**2)
xlim(0,100)
xlabel("frequency")
ylabel("ck^2")
ylim(0,5e9)
show()
```

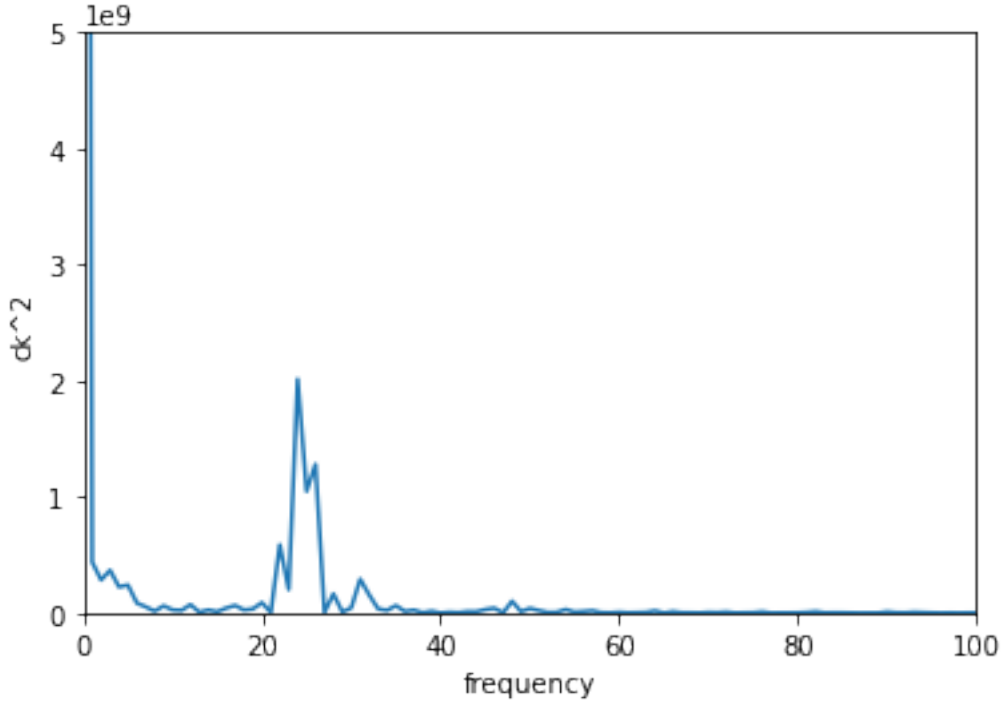
Sunspots (Ex 7.2)



Can see spikes around … every 100-150 months (by my eye)

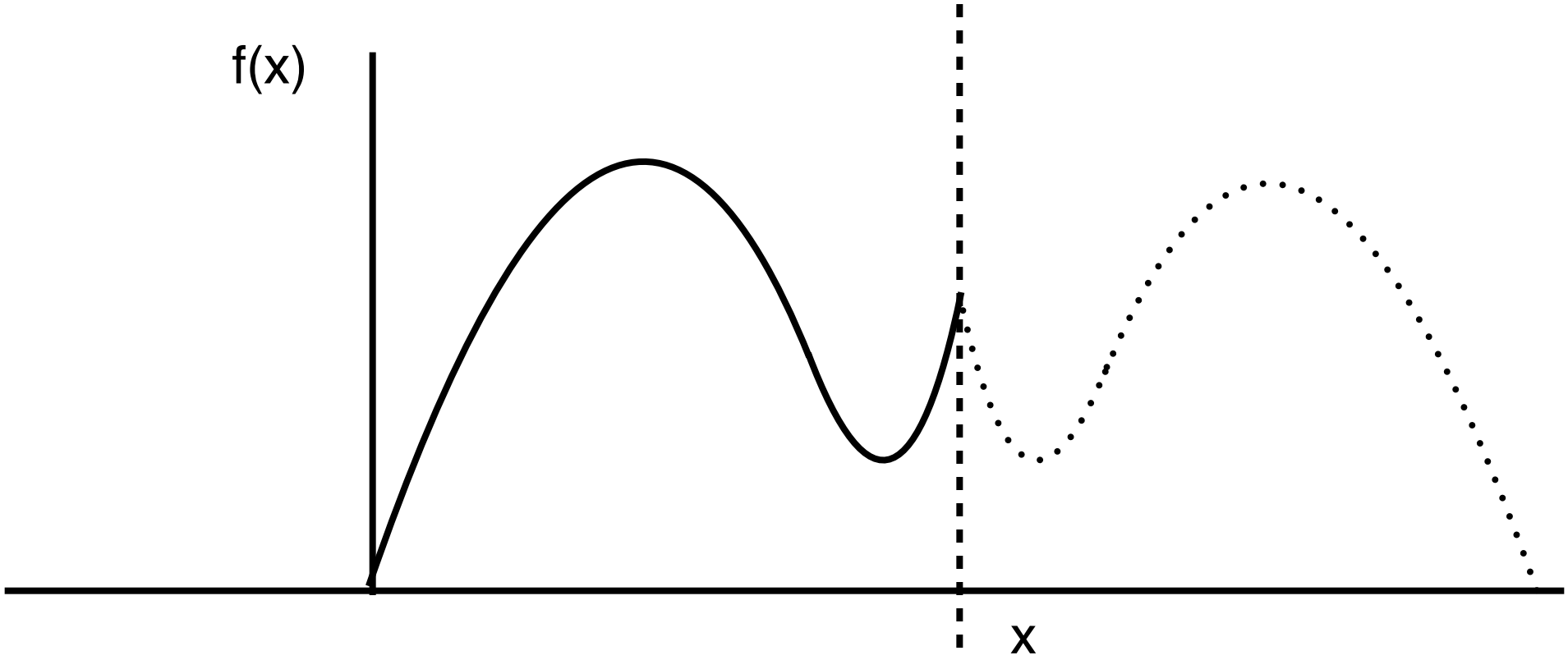
Sunspots (Ex 7.2)

There are 3143 entries in the file. Spike at k = 24, corresponding to a period of 3143/24 = 131 months (like we guessed from the plot!) = 11.0 years



Discrete cosine transforms

If a function is symmetric about the endpoints, we can use the cosine transform and not worry about the exponential form. And if it's not symmetric about the endpoints, we can make it symmetric, $f(0) = f(xN)$, $f(x_1) = f(x_{N-1})$, etc



Discrete cosine transforms

$$c_k = \sum_{n=0}^{N/2} y_n e^{-i\frac{2\pi kn}{N}} + \sum_{n=N/2+1}^{N-1} y_n e^{-i\frac{2\pi kn}{N}}$$

Use symmetry

$$c_k = \sum_{n=0}^{N/2} y_n e^{-i\frac{2\pi kn}{N}} + \sum_{n=N/2+1}^{N-1} y_{N-n} e^{-i\frac{2\pi kn}{N}}$$

$$c_k = \sum_{n=0}^{N/2} y_n e^{-i\frac{2\pi kn}{N}} + \sum_{n=N/2+1}^{N-1} y_{N-n} e^{-i*2\pi k\frac{-N}{N}} e^{-i\frac{2\pi kn}{N}}$$

e^{2pi*i*k) is 1 for integer k

Discrete cosine transforms

$$c_k = \sum_{n=0}^{N/2} y_n e^{-i\frac{2\pi kn}{N}} + \sum_{n=N/2+1}^{N-1} y_{N-n} e^{-i*2\pi k\frac{-N}{N}} e^{-i\frac{2\pi kn}{N}}$$

$$c_k = \sum_{n=0}^{N/2} y_n e^{-i\frac{2\pi kn}{N}} + \sum_{n=N/2+1}^{N-1} y_{N-n} e^{i\frac{2\pi k(N-n)}{N}}$$

Define: $q = N - n$

$$c_k = \sum_{n=0}^{N/2} y_n e^{-i\frac{2\pi kn}{N}} + \sum_{q=N/2-1}^{1} y_q e^{i\frac{2\pi kq}{N}}$$

Rename and count upwards again

$$c_k = \sum_{n=0}^{N/2} y_n e^{-i\frac{2\pi kn}{N}} + \sum_{n=1}^{N/2-1} y_n e^{i\frac{2\pi kn}{N}}$$

Discrete cosine transforms

$$c_k = \sum_{n=0}^{N/2} y_n e^{-i\frac{2\pi kn}{N}} + \sum_{n=1}^{N/2-1} y_n e^{i\frac{2\pi kn}{N}}$$

$$c_k = y_0 + y_{N/2}\cos\left(\frac{2\pi k(N/2)}{N}\right) + 2\sum_{n=1}^{N/2-1} y_n \cos\left(\frac{2\pi kn}{N}\right)$$

$$c_k = y_0 + y_{N/2}\cos(\pi k) + 2\sum_{n=1}^{N/2-1} y_n \cos\left(\frac{2\pi kn}{N}\right)$$

$$c_k = y_0 + y_{N/2}(-1)^k + 2\sum_{n=1}^{N/2-1} y_n \cos\left(\frac{2\pi kn}{N}\right)$$

Similarly, the inverse discrete cosine transformation

$$y_n = \frac{1}{N}\left[c_0 + c_{N/2}(-1)^n + 2\sum_{k=1}^{N/2-1} c_k \cos\left(\frac{2\pi kn}{N}\right)\right]$$

Please read book for derivation

Type-II DCT (sample points in middle of the sample)

$$y_n = \frac{1}{N}\left[a_0 + 2\sum_{k=1}^{N-1} a_k \cos\left(\frac{\pi k(n+\frac{1}{2})}{N}\right)\right]$$

$$a_k = \sum_{n=0}^{N-1} y_n \cos\left(\frac{\pi k(n+\frac{1}{2})}{N}\right)$$

Fast Fourier Transform (FFT)

Our discrete transforms loop over N points, for each of N/2+1 coefficients. Thus we have N(N/2+1) operations to make, which goes as $1/2*N^2$. For N somewhat large, our calculations will still be quite slow. Thankfully, there is a way to make this much faster, the Fast Fourier Transform…

Some psuedocode:

```
fft ( x ) :
    n = size of data
    recursively call fft(even x's)
    recursively call fft(odd x's)
    combine results
```

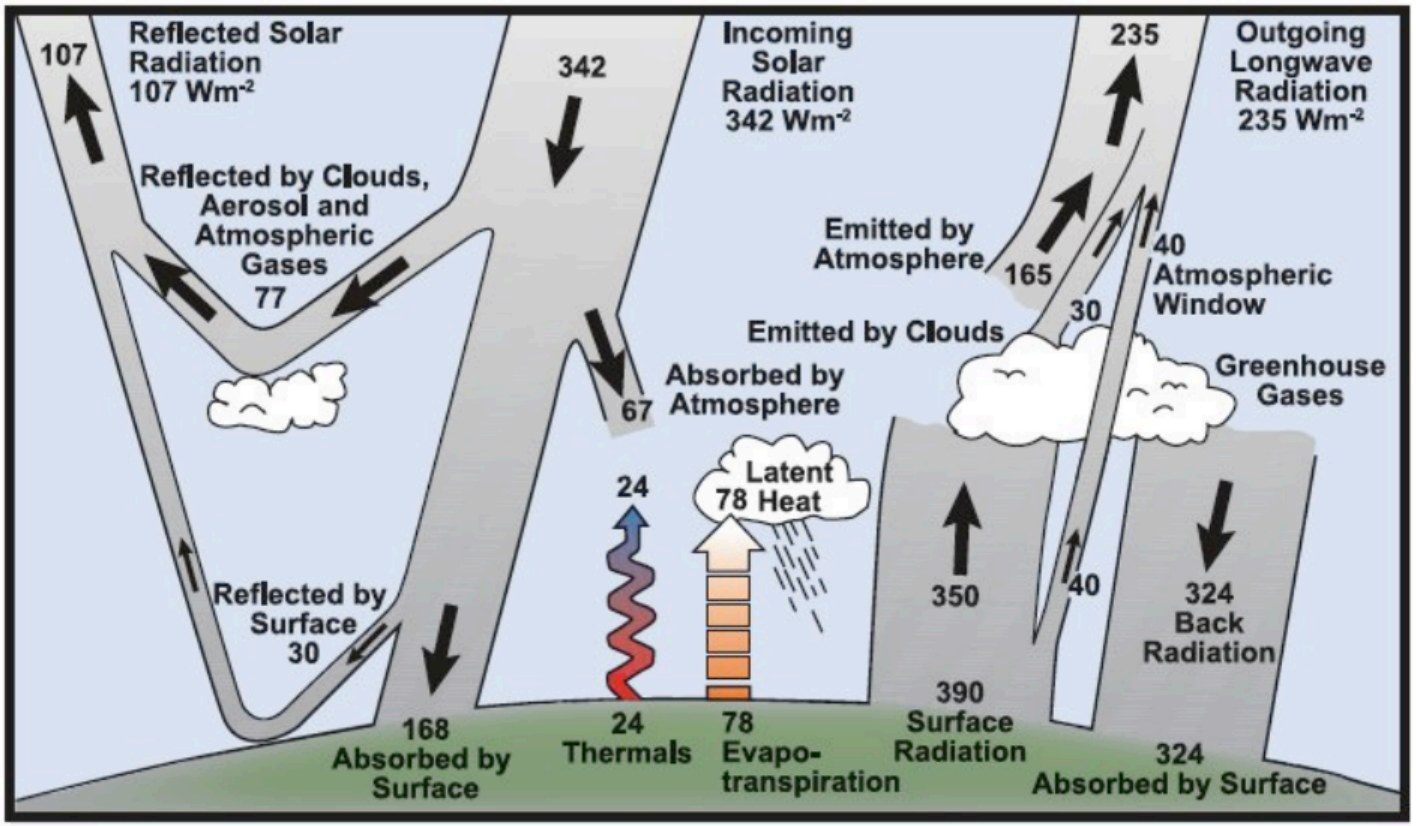
Global warming (great slides from Sal)

[Humans] have now all but destroyed this once salubrious planet as a life-support system in fewer than 200 years, mainly by making thermodynamic whoopee with fossil fuels.

-- Kurt Vonnegut

Global warming (great slides from Sal)

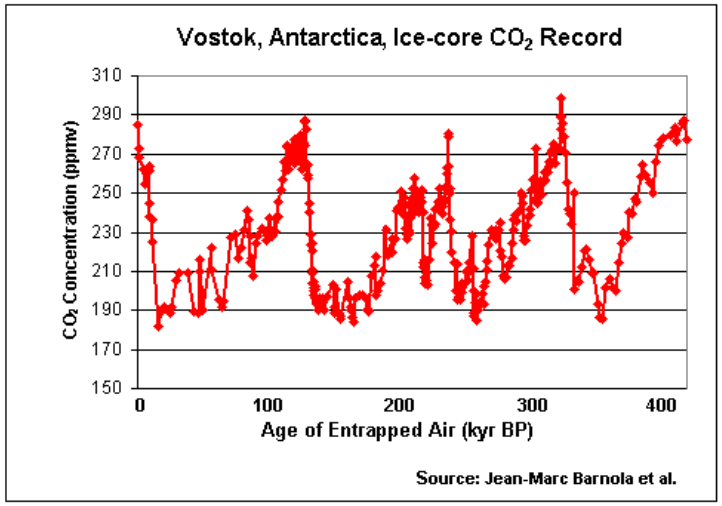
- Solar energy incident on Earth's is partially reflected back into space as lower wavelength infrared radiation
- CO2 in the atmosphere tends to trap this radiation and is an important factor in the phenomenon of global warming. Global warming has important consequences for the biosphere and human society.
- Interested parties should read the reports of the Intergovernmental Panel on Climate Change http://www.ipcc.ch/.



Global warming (great slides from Sal)

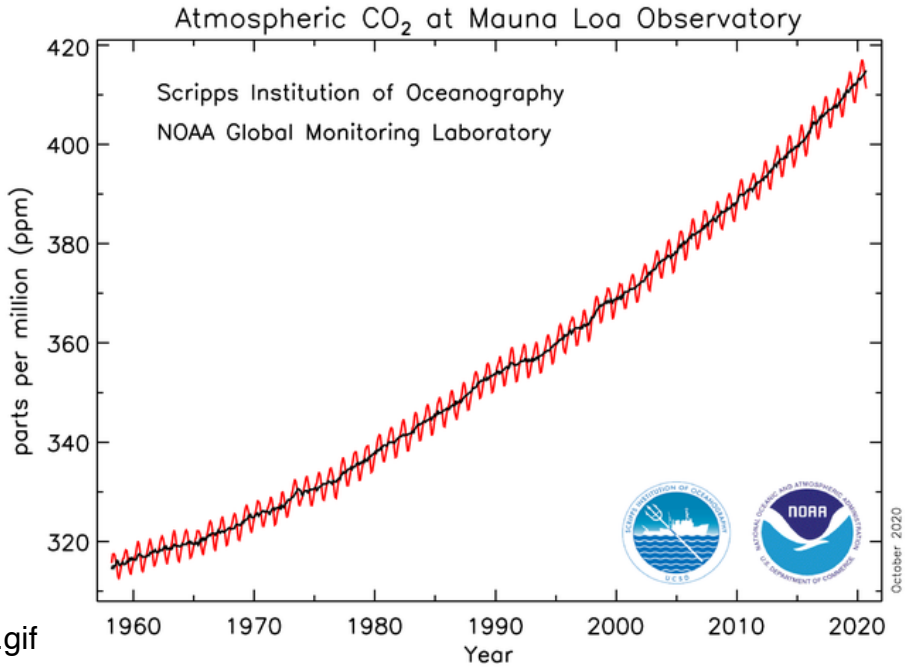
- Situated at 11,135 ft on the north flank of the Mauna Loa volcano on the Big Island of Hawaii, the National Oceanic and Atmospheric Administration's Mauna Loa Observatory http://www.mlo.noaa.gov/ has been monitoring the level of carbon dioxide in Earth's atmosphere for over 50 years. The levels of this greenhouse gas have been rising steadily during this observation period.
- Globally we're at the highest point in hundreds of thousands of years
  - Can read ice core data from Vostok, Antarctica
  - http://cdiac.ornl.gov/trends/co2/vostok.html

Look at the scales!



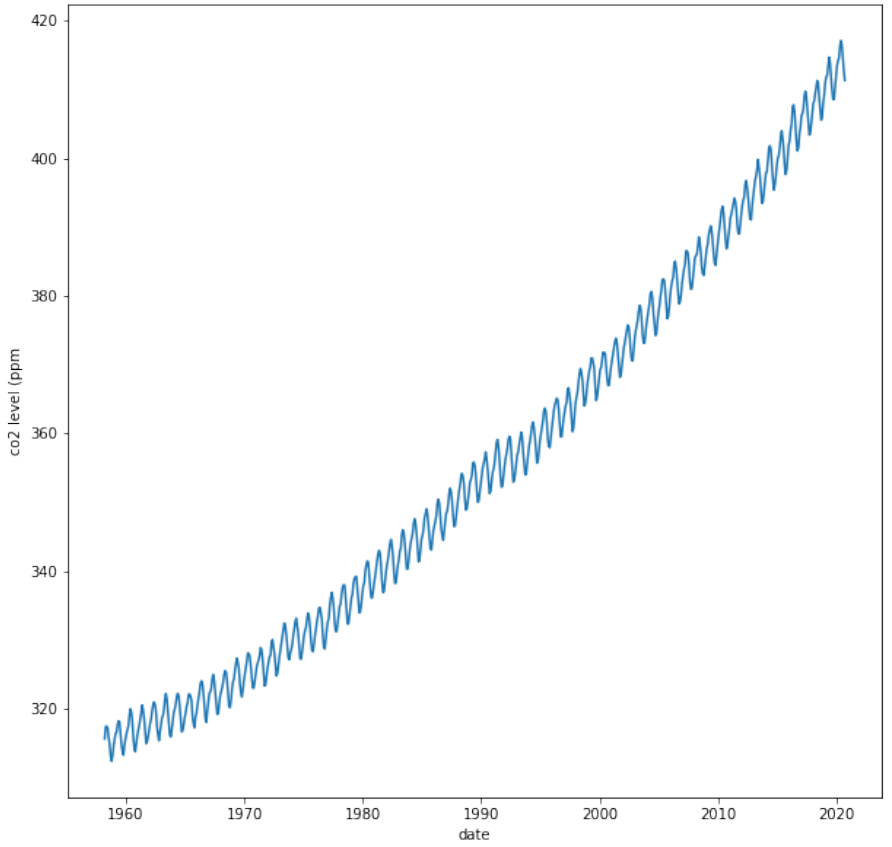
https://cdiac.ess-dive.lbl.gov/trends/co2/graphics/vostok.co2.gif

https://www.esrl.noaa.gov/gmd/ccgg/trends/



Let's look at the data


```
# CO2 and global warming
from matplotlib.pylab import plot,show,xlim,ylim,xlabel,ylabel,hist
import matplotlib.pyplot as plt
import urllib.request

### Read the file from the web, can also read a local version
url='ftp://aftp.cmdl.noaa.gov/products/trends/co2/co2_mm_mlo.csv'
with urllib.request.urlopen(url) as response:
   html = response.read()

# "decode it" and split into rows
data=html.decode().split("\n")
# from the file, anything beginning with '#' is a comment and to be skipped
# and then we have this key (which needs to be skipped!):
# year,month,decimal date,average,interpolated,trend,ndays
# we can use the decimal date and the average value, so entries 2 and 3
data=html.decode().split("\n")
dates=[]
co2s=[]
for entry in data:
  if ('#' in entry): continue
  if ('year' in entry): continue
  vals = entry.split(",") ### values split by commas
  if (len(vals) < 5): continue ### skip last trailing empty row
  dates.append(float(vals[2])) ## convert string to float
  co2s.append(float(vals[3]))  ## convert string to float


print("Total number of entries = ",len(dates))
### bigger than default, too
fig = plt.figure(figsize=(10,10))
axes = plt.axes()
plot(dates,co2s)
xlabel("date")
ylabel("co2 level (ppm")
show()
```

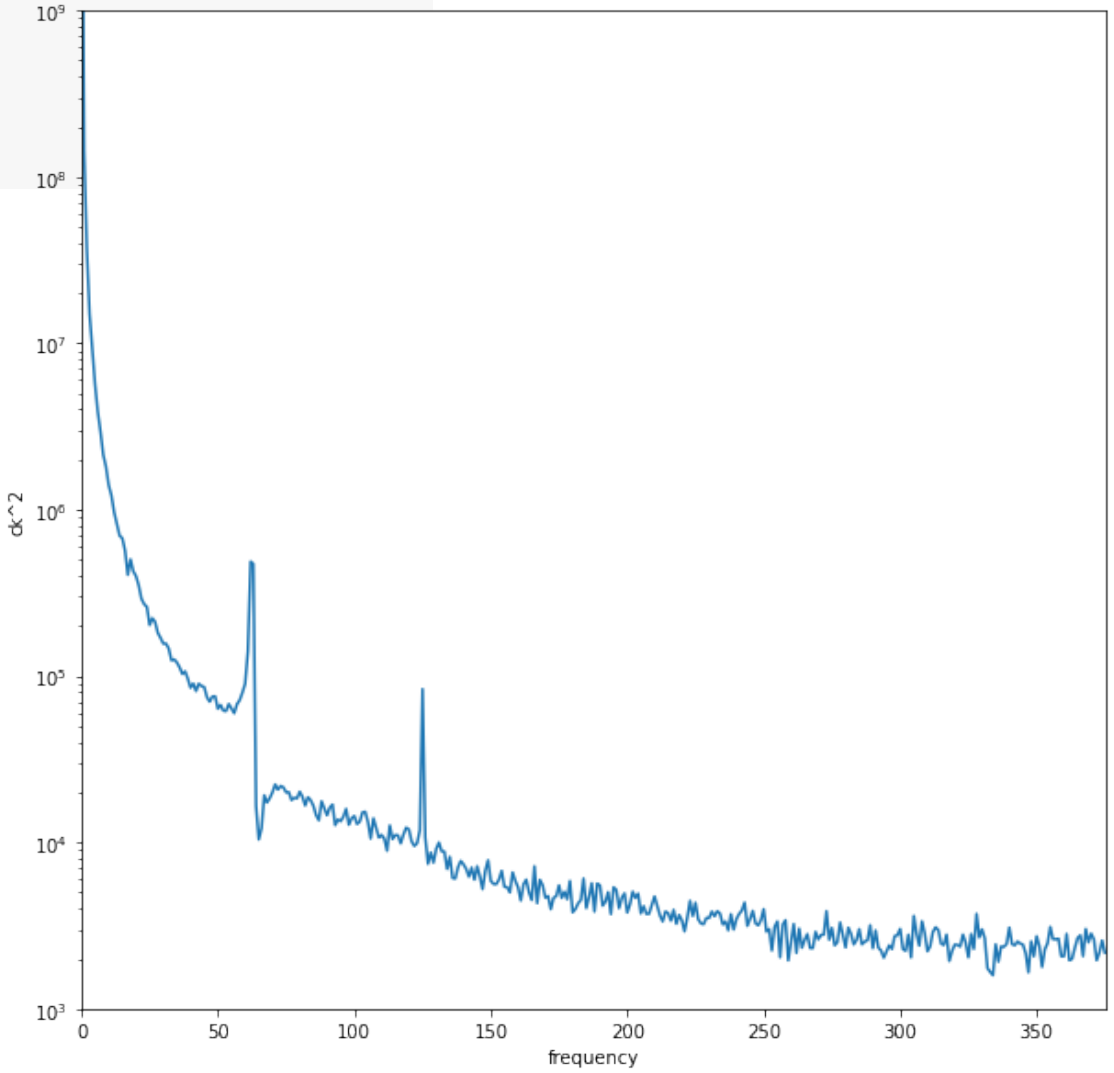


We can reproduce the ugly data. So….?

Fourier transform of the data

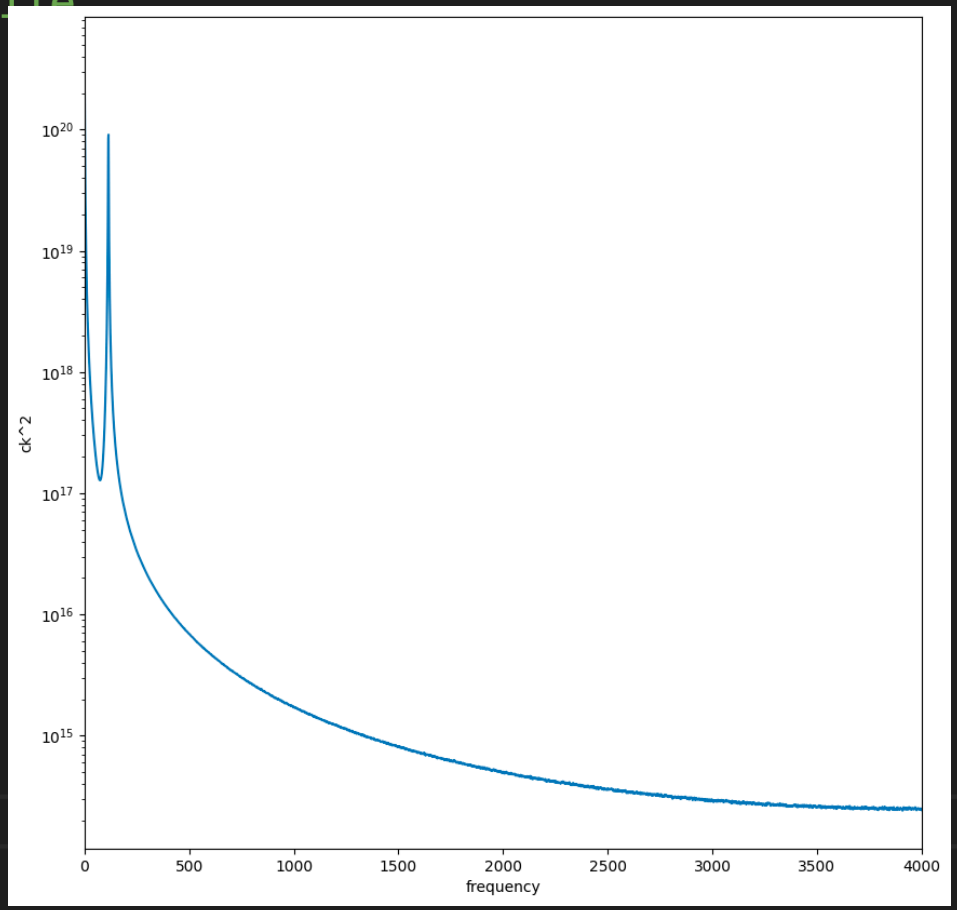

```
from numpy.fft import rfft
from matplotlib.pyplot import yscale,plot,xlim,xlabel,ylabel,show,figure
c=rfft(co2s)
fig = figure(figsize=(10,10))
plot(abs(c)**2)
xlim(0,len(co2s)/2)
xlabel("frequency")
ylabel("ck^2")
yscale("log")
ylim(1e3,1e9)
show()
```

Spike at 0, 62, 125, data size = 751, so spikes corresponding to 751/0 = **infinite**, 751/62 = 12 months and 751/125 = 6 months



Remember our g-2 data?


```
from numpy.fft import rfft
from matplotlib.pyplot import yscale,plot,xlim,xlabel,ylabel,show,figure
from google.colab import files
import numpy as np
uploaded = files.upload()### g-2 file
all = np.genfromtxt('g-2.txt')
ts = all[:,0]
vals = all[:,1]

c=rfft(vals)
fig = figure(figsize=(10,10))
plot(abs(c)**2)
xlim(0,len(vals)/2)
xlabel("frequency")
ylabel("ck^2")
yscale("log")
tdiff=ts[1]-ts[0]
for iter in range(len(c)):
  val = (abs(c)**2)[iter]
  if (val > 1e19 and iter > 0):
    frequency = iter*tdiff
    print(val,1./(frequency))
show()
```



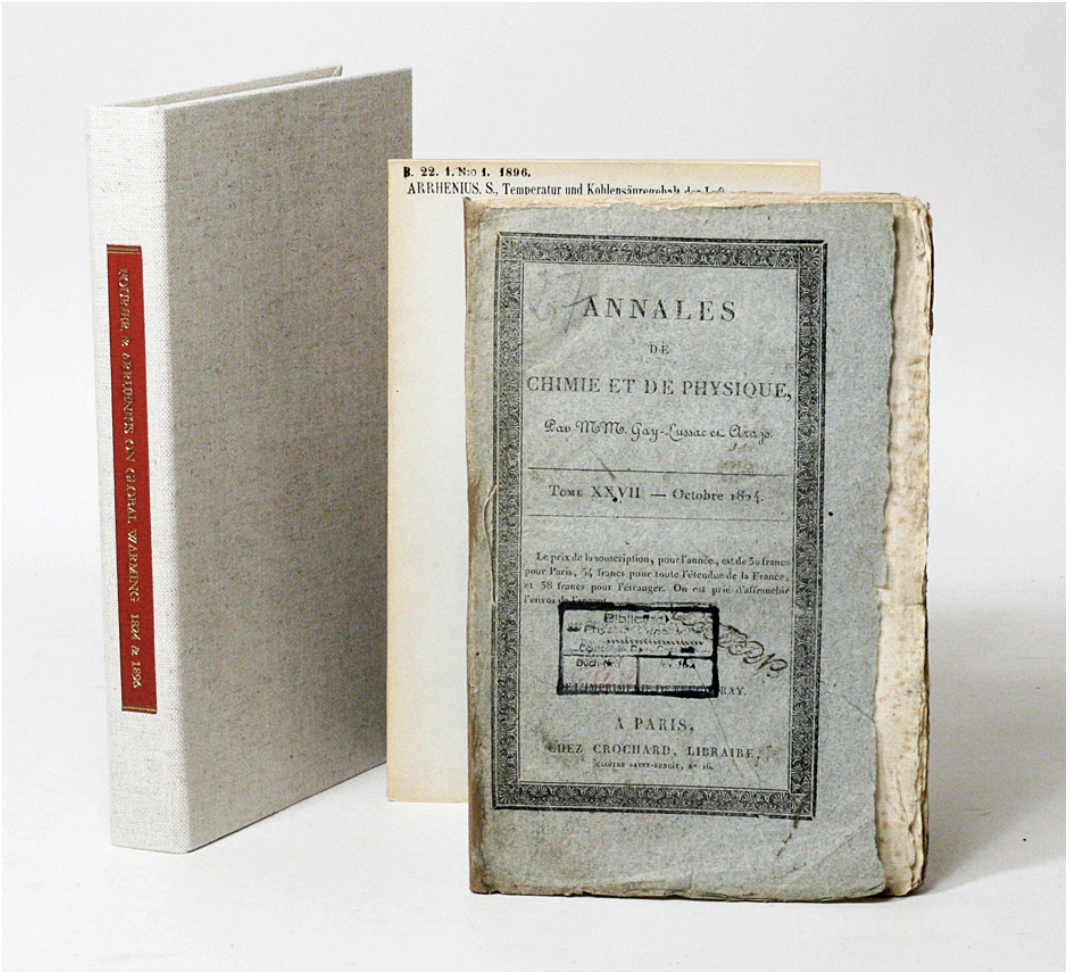
Remember our g-2 data?


```
from numpy.fft import rfft
from matplotlib.pyplot import yscale,plot,xlim,xlabel,ylabel,show,figure
from google.colab import files
import numpy as np
uploaded = files.upload()### g-2 file
all = np.genfromtxt('g-2.txt')
ts = all[:,0]
vals = all[:,1]

c=rfft(vals)
fig = figure(figsize=(10,10))
plot(abs(c)**2)
xlim(0,len(vals)/2)
xlabel("frequency")
ylabel("ck^2")
yscale("log")
tdiff=ts[1]-ts[0]
for iter in range(len(c)):
  val = (abs(c)**2)[iter]
  if (val > 1e19 and iter > 0):
    frequency = iter*tdiff
    print(val,1./(frequency))
show()
```

```
2.521277255034039e+20 15998000.0
1.1526953480094425e+20 7999000.0
6.051029155560326e+19 5332666.666666666
3.633184677420203e+19 3999500.0
2.3995858983581835e+19 3199600.0
1.6953361790469431e+19 2666333.333333333
1.2583691150839742e+19 2285428.5714285714
1.0269937908165315e+19 144126.12612612612
1.839459763398283e+19 142839.2857142857
3.826190864855366e+19 141575.22123893804
8.376491476859745e+19 140333.3333333333
9.061287325151773e+19 139113.04347826086
4.404510034580748e+19 137913.79310344826
2.162900324025321e+19 136735.04273504272
1.2354337265023846e+19 135576.27118644066
```

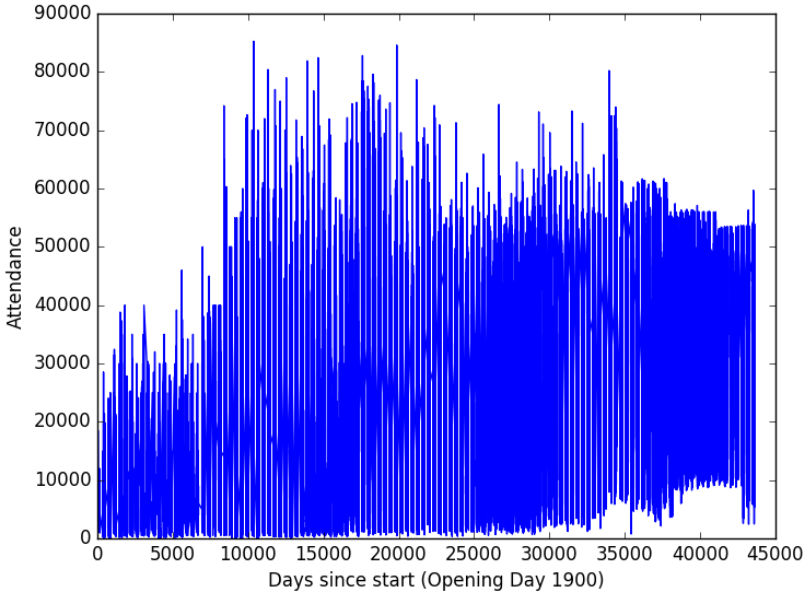
Fourier…

In 1824, Joseph Fourier also was the first person to explain the greenhouse effect, ie that the atmosphere acts as a blanket to keep the Earth from dissipating its heat. *“Remarques generales sur les Temperatures du globe terrestre et des espaces planetaires”*



Fun power of Fourier analysis

Baseball geeks (like me!) love to keep track of and follow all sorts of baseball statistics, including play-by-play data, pitch data and advanced metrics like we saw earlier. Even things like… attendance :)

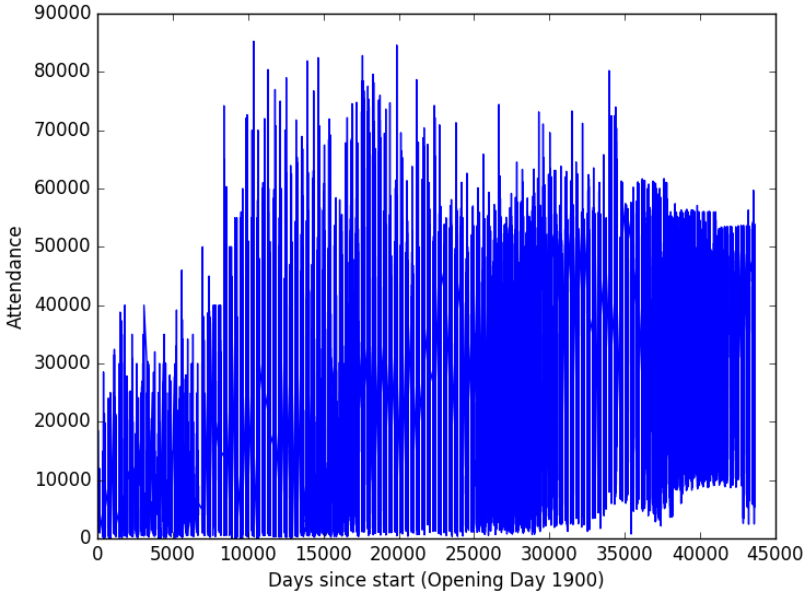


Plot of the per-game attendance since the beginning of the 20th century (through 2019), almost 168,000 games. Lots of interesting structure

https://www.retrosheet.org/gamelogs/index.html

The challenge of Fourier analysis here

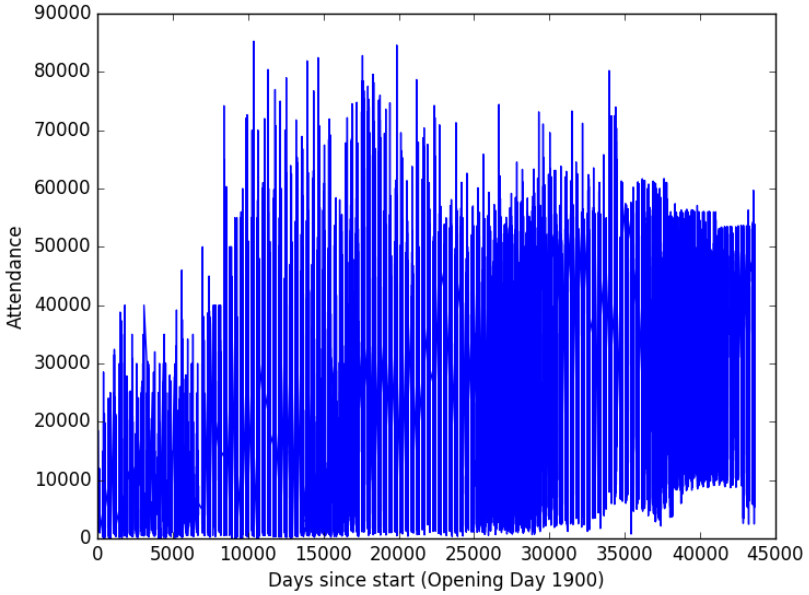
Our data is not uniformly sampled. There are many games per-day, on the same day, and many days with no games (long off-season). We could remove the off-season, but it's not the same length every year. And we could average the attendance each day, but we might lose information



Plot of the per-game attendance since the beginning of the 20th century (through 2019), almost 168,000 games. Lots of interesting structure

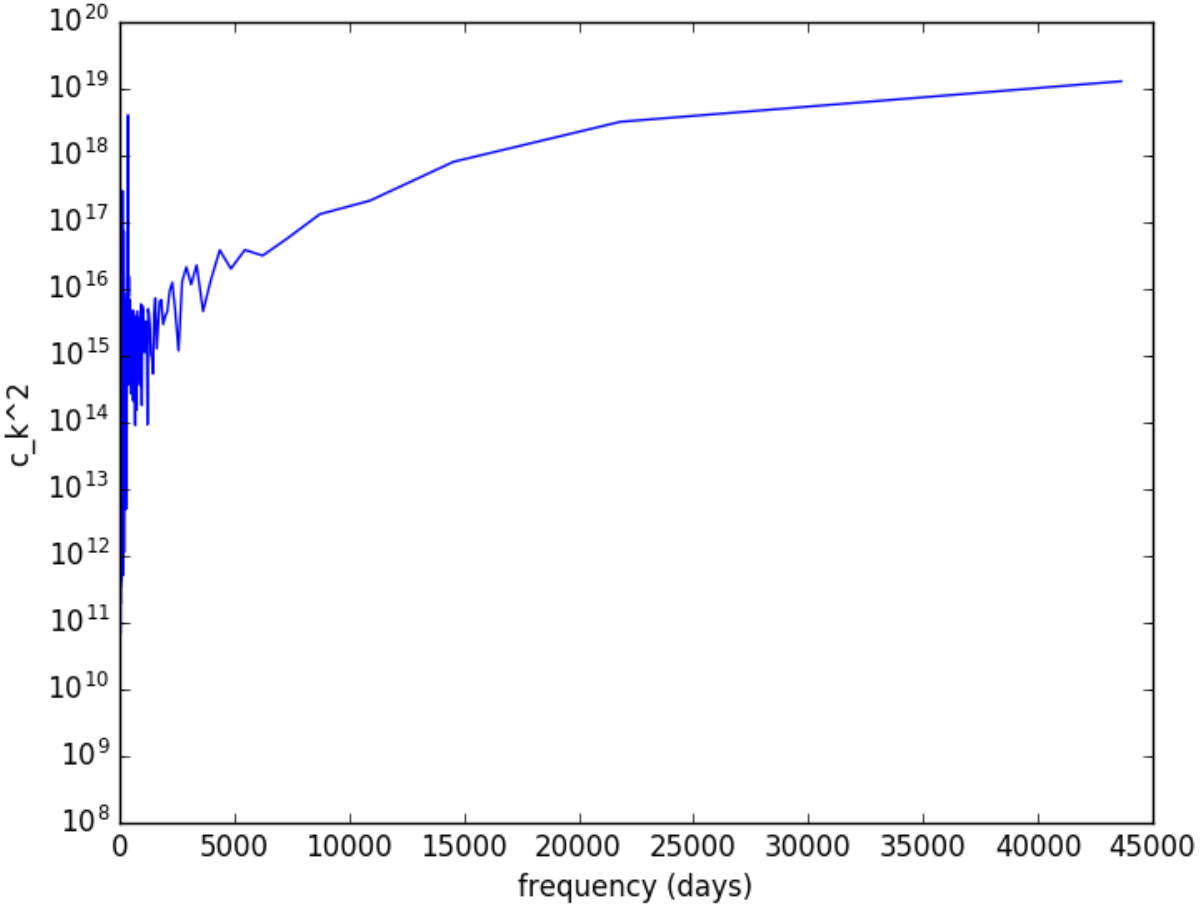
The challenge of Fourier analysis here

We also have a LOT of data here. We need something fast for it to work. The nfft package of Python is a bit tricky to use, but works well here (can do an analysis on my laptop in a few seconds after loading the data). What do we see?



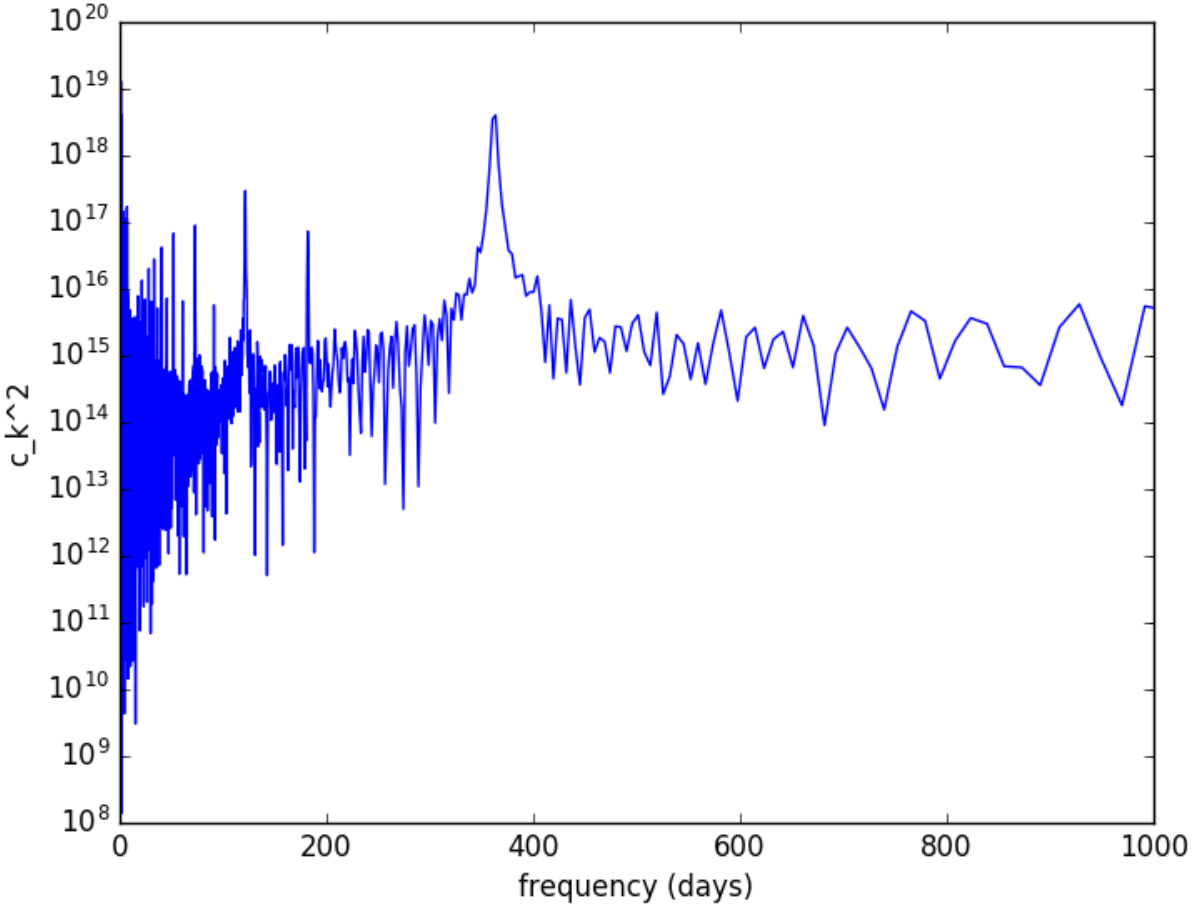
Plot of the per-game attendance since the beginning of the 20th century (through 2019), almost 168,000 games. Lots of interesting structure

Fourier analysis of attendance data



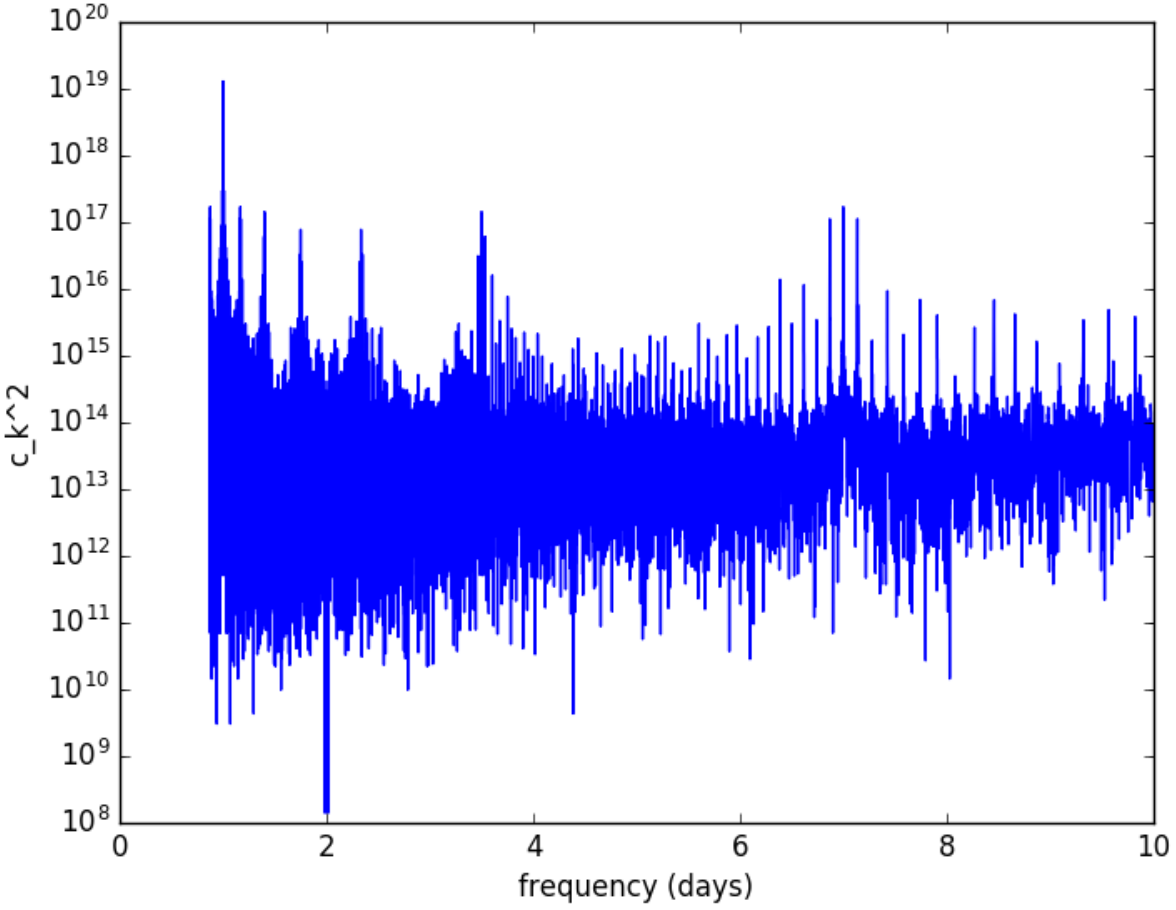
What does this tell us?

Fourier analysis of attendance data



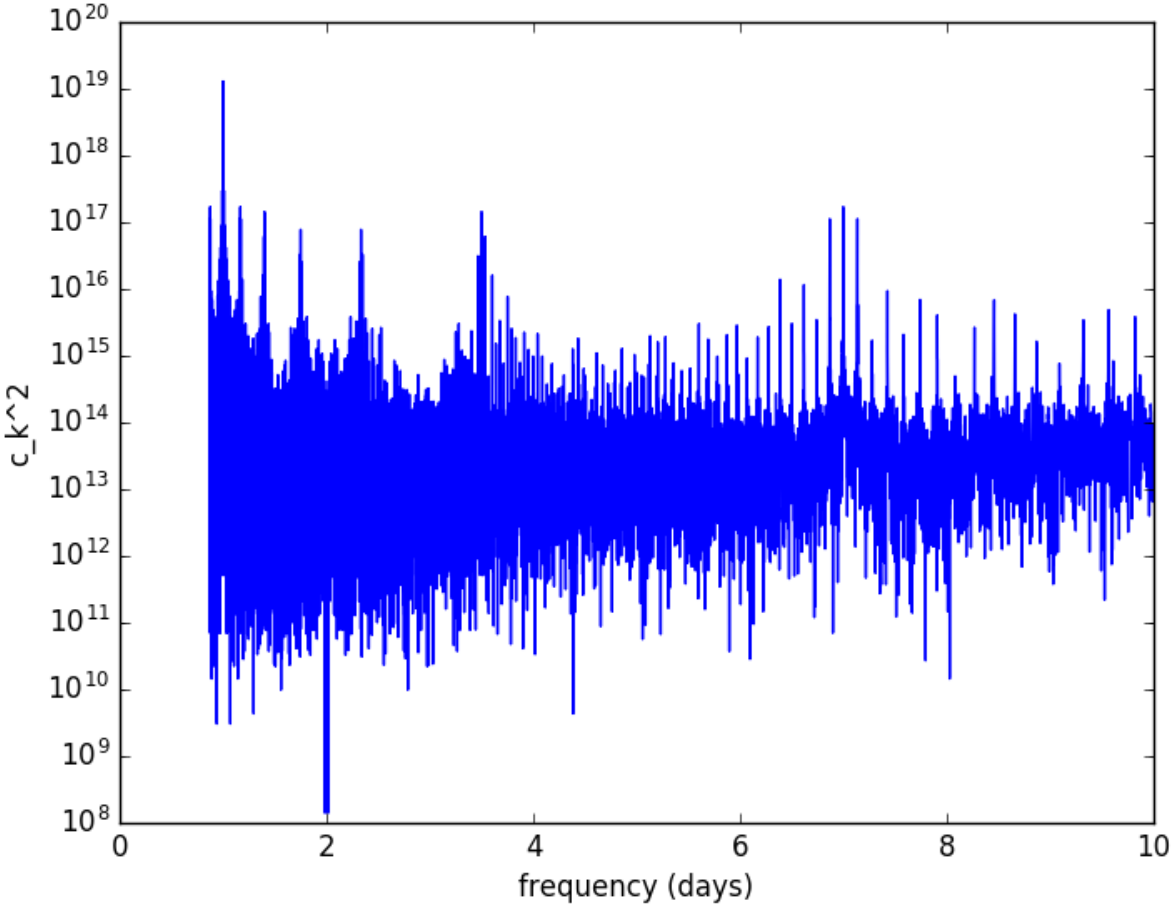
Zooming in a bit, there is one obvious peak here. What is it?

Fourier analysis of attendance data



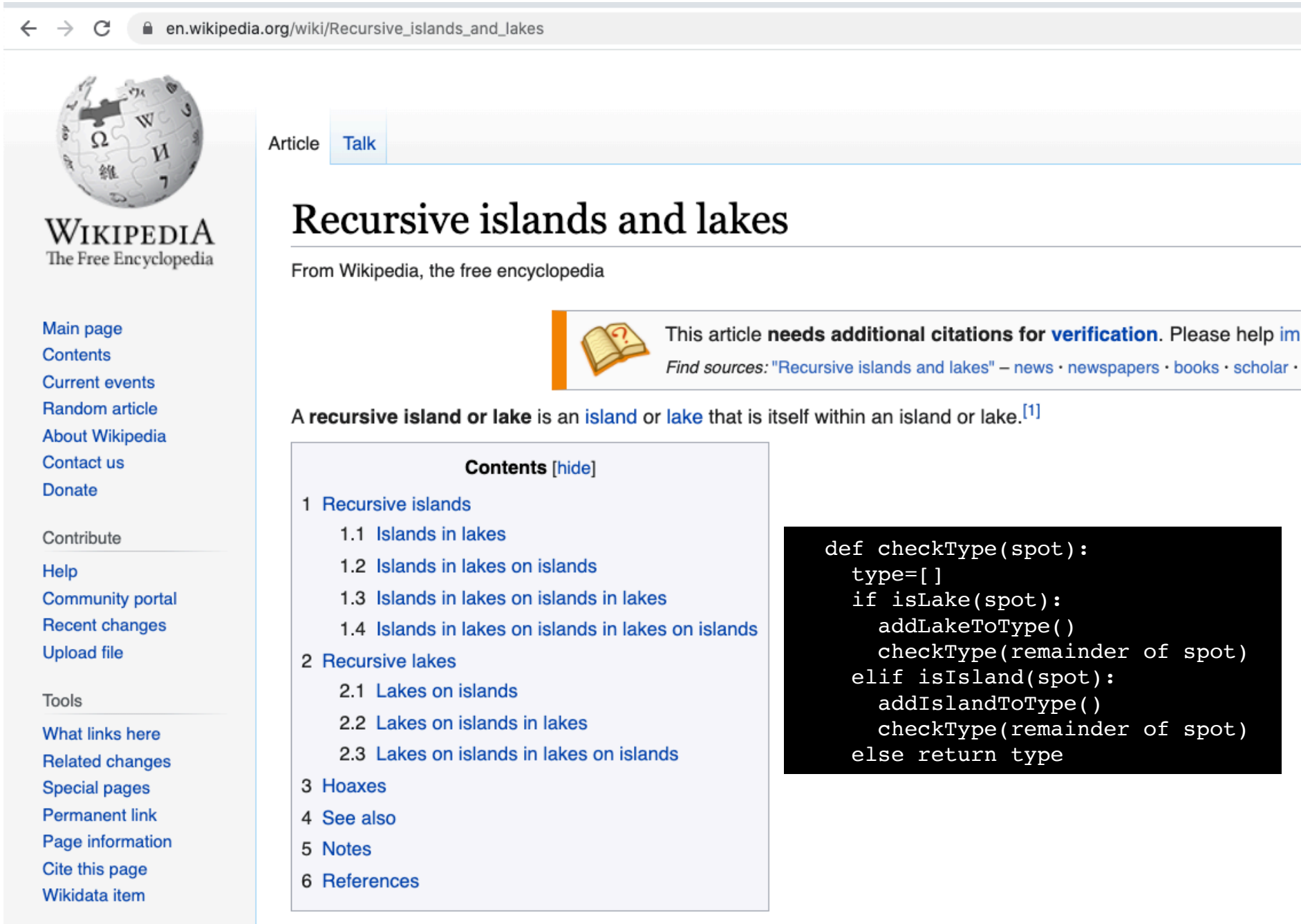
On a shorter timescale, there are lots of other features. Let's discuss them. The structure on time scales < 1 day is a "feature" of the way nfft works, unfortunately (it returns coefficients for uniformly distributed wave numbers, not frequencies)

So what have we discovered?



Society has implemented a 7 day work week and the Earth revolves around the sun every 365 days :) But it's interesting to also understand how attendance varies on longer timescales, between teams playing one another, when certain players are pitching, etc. Lots to explore here

On recursive algorithms



en.wikipedia.org/wiki/Recursive_islands_and_lakes

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Current events
Random article
About Wikipedia
Contact us
Donate

Contribute
Help
Community portal
Recent changes
Upload file

Tools
What links here
Related changes
Special pages
Permanent link
Page information
Cite this page
Wikidata item

Article Talk

Recursive islands and lakes

From Wikipedia, the free encyclopedia

This article **needs additional citations for verification**. Please help imp

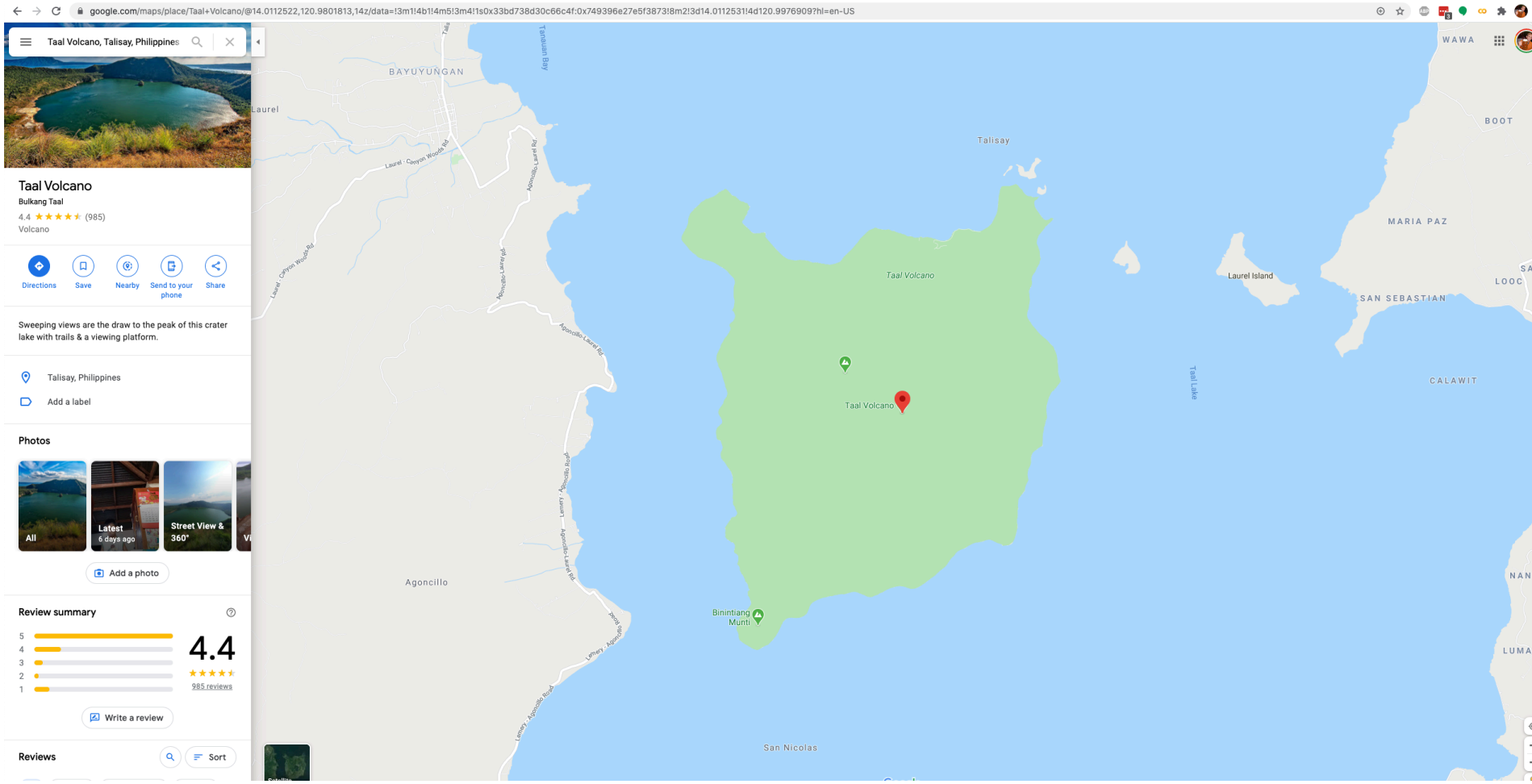
Find sources: "Recursive islands and lakes" – news · newspapers · books · scholar · J

A **recursive island or lake** is an island or lake that is itself within an island or lake.[1]

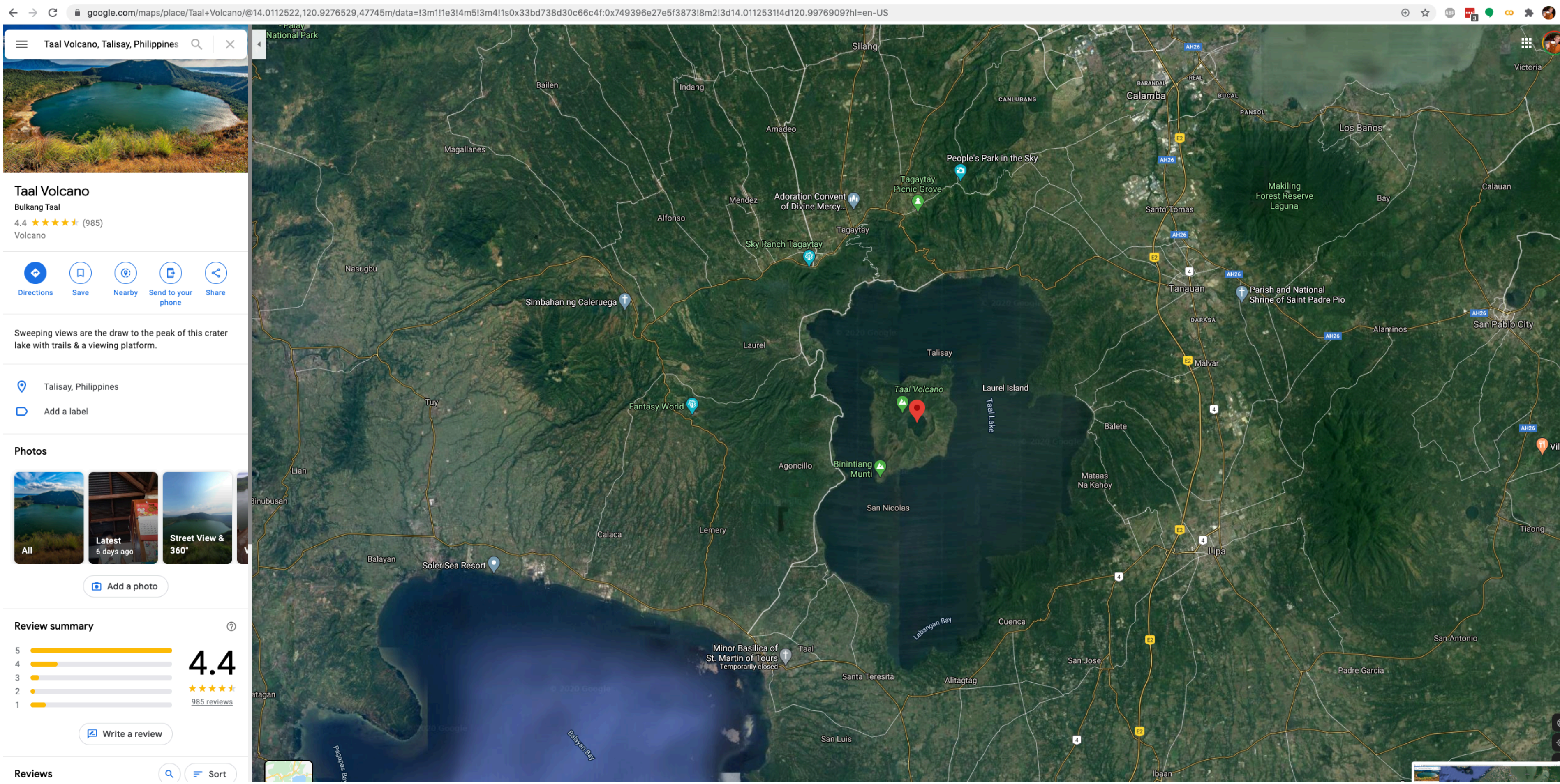
Contents [hide]

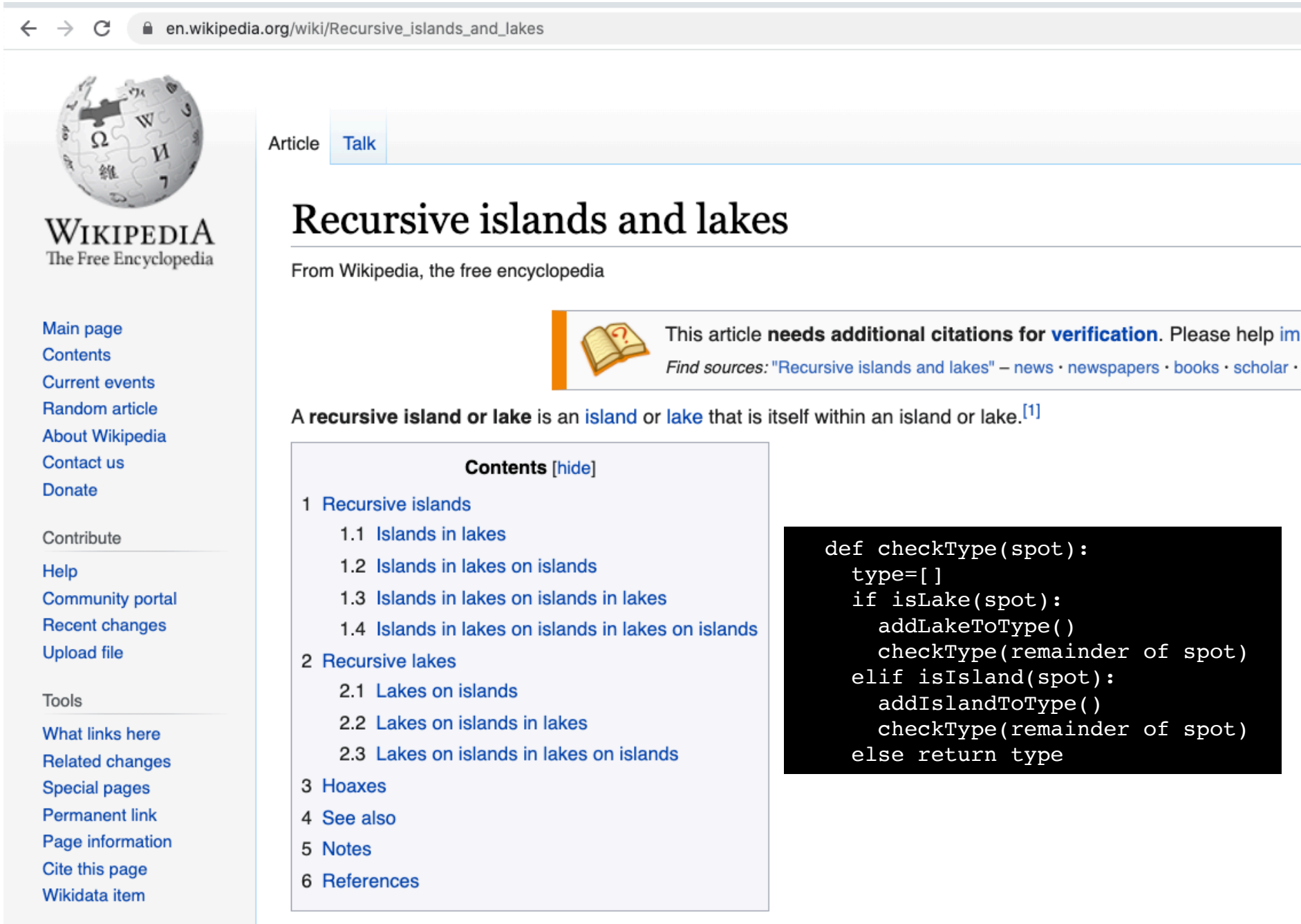
1 Recursive islands
  1.1 Islands in lakes
  1.2 Islands in lakes on islands
  1.3 Islands in lakes on islands in lakes
  1.4 Islands in lakes on islands in lakes on islands
2 Recursive lakes
  2.1 Lakes on islands
  2.2 Lakes on islands in lakes
  2.3 Lakes on islands in lakes on islands
3 Hoaxes
4 See also
5 Notes
6 References

```
def checkType(spot):
  type=[]
  if isLake(spot):
    addLakeToType()
    checkType(remainder of spot)
  elif isIsland(spot):
    addIslandToType()
    checkType(remainder of spot)
  else return type
```


google.com/maps/place/Taal+Volcano/@14.0112522,120.9801813,14z/data=!3m1!4b1!4m5!3m4!1s0x33bd738d30c66c4f:0x749396e27e5f3873!8m2!3d14.0112531!4d120.9976909?hl=en-US
Taal Volcano, Talisay, Philippines
Taal Volcano
Bulkang Taal
4.4 (985)
Volcano
Directions
Save
Nearby
Send to your phone
Share
Sweeping views are the draw to the peak of this crater lake with trails & a viewing platform.
Talisay, Philippines
Add a label
Photos
All
Latest
6 days ago
Street View & 360°
Add a photo
Review summary
4.4
985 reviews
Write a review
Reviews
Sort
BAYUYUNGAN
Laurel
Talisay
Taal Volcano
Taal Volcano
Laurel Island
WAWA
BOOT
MARIA PAZ
SAN SEBASTIAN
CALAWIT
Taal Lake
Agoncillo
Binintiang Munti
San Nicolas

Recursive Recursive recursive recursive recursive recursive recursive recursive recursive






```
def checkType(spot):
  type=[]
  if isLake(spot):
    addLakeToType()
    checkType(remainder of spot)
  elif isIsland(spot):
    addIslandToType()
    checkType(remainder of spot)
  else return type
```

Fast Fourier Transform (FFT)

```
fft ( x ) :
    n = size of data
    recursively call fft(even x's)
    recursively call fft(odd x's)
    combine results
```

$$c_k = \sum_{n=0}^{n=N-1} y_n e^{-i\frac{2\pi kn}{N}}$$

Start by evaluating Ck (for any k) for the even terms, where n=2r:

$$c_k = \sum_{r=0}^{N/2-1} y_{2r} e^{-i\frac{2\pi k 2r}{N}}$$

$$c_k = \sum_{r=0}^{N/2-1} y_{2r} e^{-i\frac{2\pi k 2r}{\frac{N}{2}}}$$

This is a Fourier transform with N/2 samples, not N!

Fast Fourier Transform (FFT)

```
fft ( x ) :
    n = size of data
    recursively call fft(even x's)
    recursively call fft(odd x's)
    combine results
```

$$c_k = \sum_{n=0}^{n=N-1} y_n e^{-i\frac{2\pi kn}{N}}$$

Now the odd terms, n=2r+1

$$c_k = \sum_{r=0}^{N/2-1} y_{2r+1} e^{-i\frac{2\pi k(2r+1)}{N}}$$

$$c_k = e^{-i2\pi k/N} \sum_{r=0}^{N/2-1} y_{2r} e^{-i\frac{2\pi kr}{\frac{N}{2}}}$$

This is also a Fourier transform with N/2 samples, not N! (And extra “twiddle factor”)

Fast Fourier Transform (FFT)

$$c_k = E_k + e^{-i2\pi k/N} O_k$$

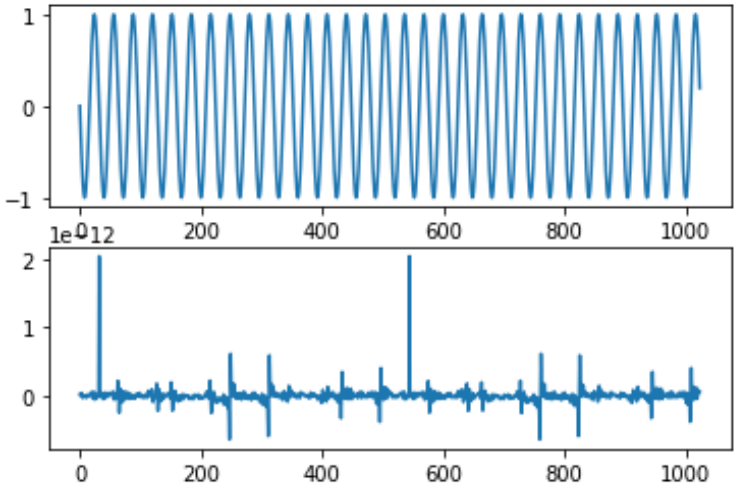
So our coefficient is the sum of two terms of even and odd pieces that are themselves Fourier series (plus an extra twiddle factor). But those two Fourier series can themselves be split into two Fourier series with even and odd terms. This can recursively continue until you have one term left, $c_k = y_0$

Each round of splitting has N Fourier coefficients to calculate always, but we only have $\log_2 N$ levels of splitting (if we have N=8, we have to do only 3 splittings to get down to single samples). So total complexity is $N^* \log_2 N$, not N^2!

FFT in practice


```
[32] import numpy as np
     from numpy.fft import rfft,irfft
     from math import sin,pi
     from matplotlib import pyplot as plt
     N=1024
     x = np.array([float(i) for i in range(N)])
     ##frequency
     f=32
     y=np.array([sin(-2*pi*f*xi/float(N)) for xi in x])
     s1=plt.subplot(2,1,1)
     plt.plot(x,y)
     s2=plt.subplot(2,1,2)
     c = rfft(y) ### this will have only half of the coefficients since we assumed they are real
     c=np.append(c[:-1],np.conj(c[:-1])) ## drop the last coefficient
     plt.plot(x,c)
     plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:85: ComplexWarning: Casting complex values
  return array(a, dtype, copy=False, order=order)
```



Let's discuss what we see…

Homework #5

7.3,7.4,7.5,7.6

https://classroom.github.com/a/tWXBuSQu

For many of these you may want something like the code below (and then you have to parse the text, of course !). Or you can run things not in a notebook, or load from google drive (see previous example), as you prefer

```
import urllib3
target_url =“http://……..”
http = urllib3.PoolManager()
response = http.request('GET', target_url)
data = response.data.decode(‘utf-8')
```

Finally, offering extra credit for 7.9, slightly more if you are in 410 as opposed to 510. But it’s only extra credit for either course