Physics 305 Programming Assignment 1

- Rene L. Principe Jr.
- 2015-04622

Importing Libraries

In [1]:
```
from tqdm import tqdm
import warnings
warnings.filterwarnings('ignore')

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns

from scipy.stats import norm, describe
from scipy.optimize import curve_fit

import statsmodels.api as sm
from matplotlib import gridspec
```

Simulating fractional Brownian motion (fBm)

`Lambda Function`

In [2]:
```
# set process parameters
q = 11
N = 2**q + 1 #number of datapoints
M = 2*N - 2
delta = 0.002
print("q: %d, N: %d, M: %d" % (q, N, M))
```

```
q: 11, N: 2049, M: 4096
```

`fbc(n, G)`

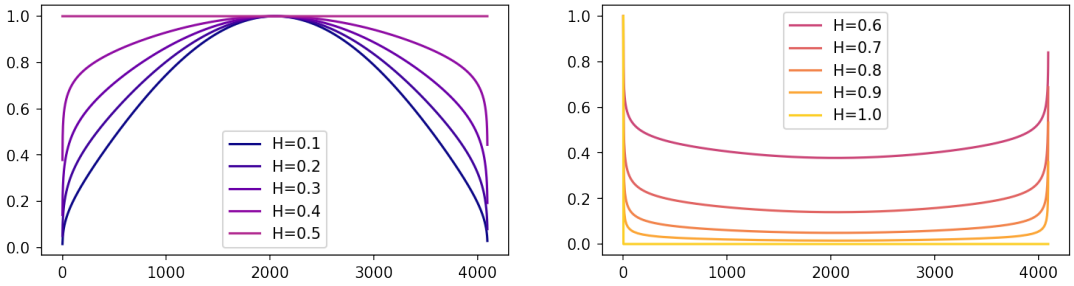
$(n+1)^G + \frac{(n-1)^G - 2n^G}{2}$

In [3]:
```
# define functions
def fbc(n,G):
    return ((n+1)**G + np.abs(n-1)**G - 2*n**G)/2.

def lambda_func(H,N):
    M = 2*N - 2
    C = np.zeros(M)
```

```
    G = 2*H
    for i in np.arange(N):
        C[i] = fbc(i,G) # fill in first N out of M values of C (i=0 to N-
    C[N:] = C[1:N-1][::-1]
    return np.real(np.fft.fft(C))**0.5
```

Visualizing `lambda_func(H,N)`

In [4]:
```
plt.figure(dpi = 150, figsize = (12,3))

for i in range(5):
    plt.subplot(121)
    H = i*0.1 + 0.1
    lambda_function = lambda_func(H, 2049)
    plt.plot(lambda_function/lambda_function.max(), ls = '-', label = "H=
             color = plt.cm.plasma(i/10))

    plt.legend()

    plt.subplot(122)
    H = i*0.1 + 0.6
    lambda_function = lambda_func(H, 2049)
    plt.plot(lambda_function/lambda_function.max(), ls = '-', label = "H=
             color = plt.cm.plasma((i+5)/10))
    plt.legend()

plt.show()
```

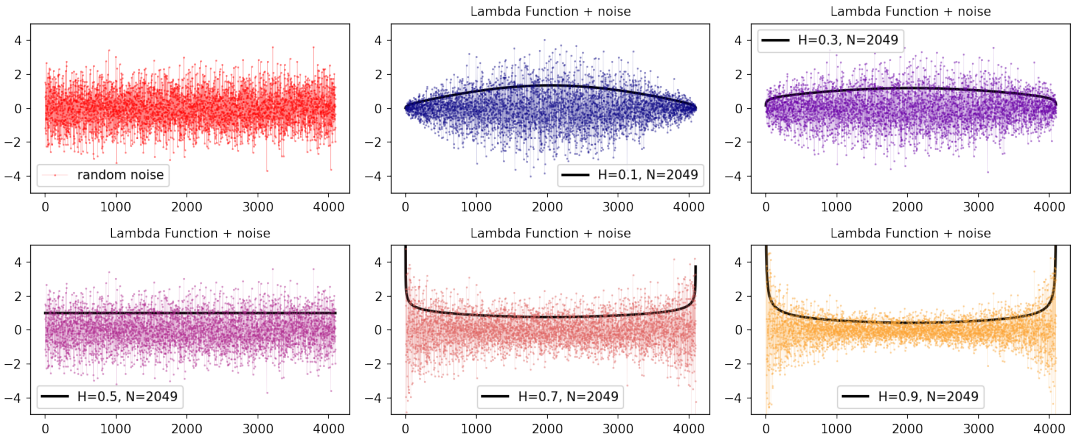


Noise Function

In [5]:
```
noise = np.random.normal(size=(M))
```

In [6]:
```
plt.figure(dpi = 150, figsize = (12,5))
plt.subplot(231)
plt.plot(noise, 'r.-', label = "random noise", lw = 0.2, alpha = 0.5, ms
plt.legend()
plt.ylim(-5,5)

for i in range(5):
    plt.subplot(2,3, i + 2)
    H = i*0.2 + 0.1
    lambda_function = lambda_func(H, 2049)*noise
    plt.plot(lambda_func(H, 2049), 'k-', lw = 2, label = "H=%.1f, N=%d" %
    plt.plot(lambda_function, '.-',
             lw = 0.1, alpha = 0.5, ms = 1,
             color = plt.cm.plasma(i/5))
    plt.ylim(-5,5)
```

```
    plt.title("Lambda Function + noise", fontsize = 10)
    plt.legend()

plt.tight_layout()
plt.show()
```



An ordinary Brownian Motion would have a purely random noise, however, fractional brownian motion exhibits memory. In this case, we simulate "having a memory" by forcing a noise into the lamda function. As shown above, different Hurst parameter H values correspond to how the fractional noise is eventually shaped.

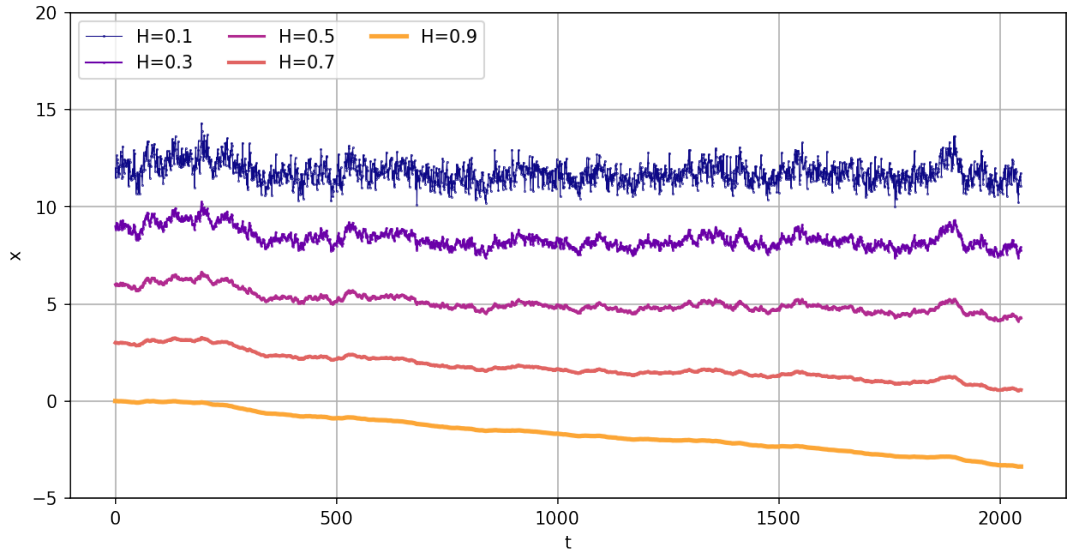
The H shapes the Lambda function accordingly in which lowering H magnifies fluctuations through time while larger H deminishes. This will latter on manifest as the diffusive property parameter of the fBm.

In order to simulate fBm, we take the cumulative sum of these fluctuations as shown in the figures below.

fBm simulation by taking `cumsum()`

In [7]:
```
# initialize
fGnsamples = np.zeros((5,N))

# generate fractional Gaussian noise samples
for i in np.arange(5):
    H = 0.2*(i+1) - 0.1
    lambda_res = lambda_func(H,N)
    a = np.fft.ifft(noise)*lambda_res
    b = np.real(np.fft.fft(a))
    fGnsamples[i,0] = 15-3*(i+1)            # apply offset in starting va
    fGnsamples[i,1:N] = delta**H*b[0:N-1]

# take cumulative sums to get the fractional Brownian motion samples
fBmsamples = np.transpose(np.cumsum(fGnsamples,axis=1)) # cumulative sum

# get values of H
Hval = (np.arange(5) + 1)*0.2 - 0.1

# plot fractional Brownian motion samples
plt.figure(dpi = 150
           , figsize = (10,5))
plt.grid()
for i in np.arange(len(Hval)):
```

```
    plt.plot(fBmsamples[:,i], label="H=%.1f" % Hval[i],
             marker = '.', ls = '-', lw = 0.5*(i+1), ms = 0.7,
             color = plt.cm.plasma(i/5))
    plt.ylim(-5, 20)
    plt.legend(loc="upper left", ncol=3)
    plt.xlabel("t")
    plt.ylabel("x")
```



As metioned, lowering H magnifies fluctuations through time while larger H deminishes it in the Lmabda function. Upon taking the cumulative sum, we can see that fluctuations are indeed evident on small H while the noise is reduced on big H. These different regimes depend on the value of H:

- 0 < H < 1/2: subdiffusion
- 1/2 < H < 1: superdiffusion
- H = 1/2: normal diffusion (Brownian motion)

Generating five fBm samples

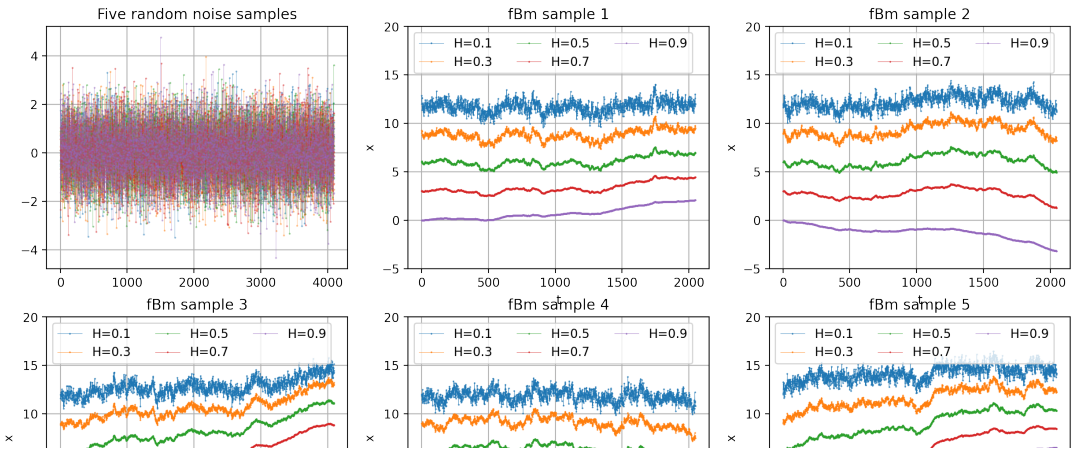
In [8]:

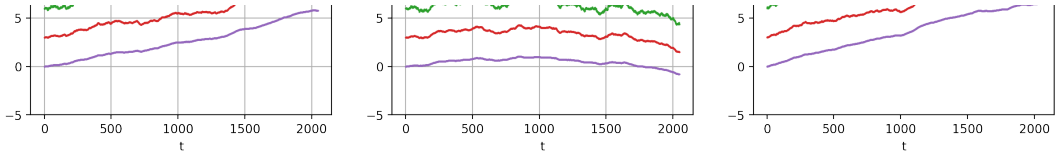
```
def fBm_generator(noise):
    # initialize
    fGnsamples = np.zeros((5,N))

    # generate fractional Gaussian noise samples
    for i in np.arange(5):
        H = 0.2*(i+1) - 0.1
        lambda_res = lambda_func(H,N)
        a = np.fft.ifft(noise)*lambda_res
        b = np.real(np.fft.fft(a))
        fGnsamples[i,0] = 15-3*(i+1)            # apply offset in startin
        fGnsamples[i,1:N] = delta**H*b[0:N-1]

    # take cumulative sums to get the fractional Brownian motion samples
    fBmsamples = np.transpose(np.cumsum(fGnsamples,axis=1)) # cumulative
    return fBmsamples
```

```
    return fBmsamples
```

In [9]:
```
np.random.seed(seed=11)
n_samp = 5
# Initialize array to hold the noise samples
noise_samp = np.zeros((M, n_samp))
for i in range(n_samp):
    noise = np.random.normal(size=(M))
    noise_samp[:,i] = noise
```

In [10]:
```
plt.figure(dpi = 200, figsize = (15,8))

plt.subplot(2,3,1)
plt.title("Five random noise samples")
plt.plot(noise_samp, '.-', lw = 0.2, alpha = 0.5, ms = 1)

# Initialize list to hold the fBm samples
fBm_simulations = []

for i, noise in enumerate(noise_samp.T):
    fBmsamples = fBm_generator(noise)
    fBm_simulations.append(fBmsamples)
    plt.grid()
    plt.subplot(2,3,i+2)
    for j in np.arange(len(Hval)):
        plt.title("fBm sample " + str(i+1))
        plt.plot(fBmsamples[:,j], label="H=%.1f" % Hval[j],
                 marker = '.', ls = '-', lw = 0.3, ms = 0.7)
        plt.ylim(-5, 20)
        plt.xlabel("t")
        plt.ylabel("x")
        plt.legend(loc="upper left", ncol=3)
plt.show()
```





Here, five sample datasets of fBm samples are generated. The same workflow is implemented on five iterations of random noise, however, a seed in the random noise generator was imposed to generate consistent results.

Calculating PDFs

Pre-defined functions

In [11]:
```
def get_sample_dx(x, delta):
    x_trunc = x[:-1*delta]
    x_shift = x[delta:]
    dx = x_shift - x_trunc
    return dx

def get_pdf(x, delta, bin_edges, norm=True):
    dx = get_sample_dx(x, delta)
    pdf, junk = np.histogram(dx, bins = bin_edges, density=norm)
    return pdf

def gaussian(x, sigma2, N):
    fac = N/(2.*np.pi*sigma2)**0.5
    return fac*np.exp(-1.*(x)**2/2./sigma2)

# define function that performs fit
def fit_pdf(bin_centers, y, yerr=np.array([]), initial=[1., 1.], maxfev=5
    if(len(yerr)==0): # no uncertainties given
        popt, pcov = curve_fit(gaussian, bin_centers, y, initial, maxfev=
    else:
        popt, pcov = curve_fit(gaussian, bin_centers, y, initial, sigma=y
    sigma2, N = popt[0], popt[1]
    err_sigma2, err_N = pcov[0,0]**0.5, pcov[1,1]**0.5
    return sigma2, err_sigma2, N, err_N
```

Selecting Test Data

Sample 1, H = 0.1, τ = 30

In [12]:
```
sample = 0
H_n = 0
H_test = Hval[H_n]
test_data = fBm_simulations[sample][:, H_n]
tau_test = 30
```

Obtaining displacements

In [13]:
```
test_dx = get_sample_dx(test_data, tau_test)
```

Probability Distribution Functions

In [14]:
```
xlimit = 5
```

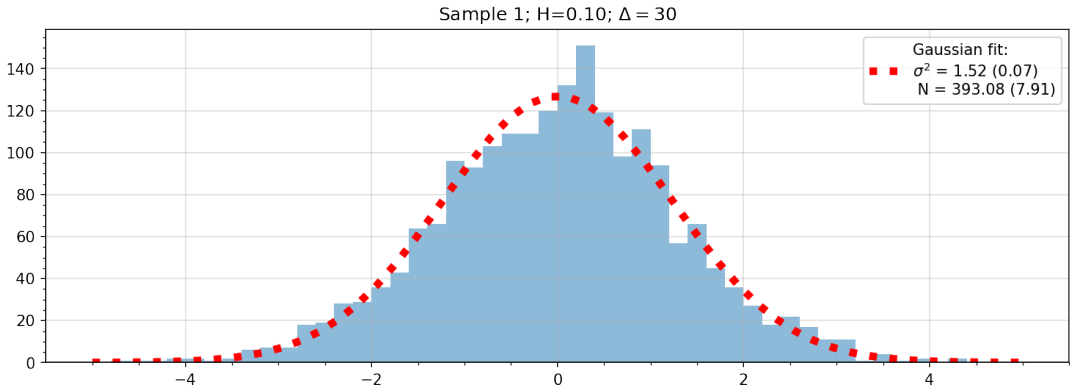
In [14]:
```
xlimit = 5.
n_bins = 50
bin_edges = np.linspace(xlimit*-1., xlimit, n_bins+1)
pdf = get_pdf(test_dx, tau_test, bin_edges, norm=False)
```

Gaussian Fit

In [15]:
```
bin_centers = 0.5*(bin_edges[:-1] + bin_edges[1:])
sigma2, err_sigma2, N, err_N = fit_pdf(bin_centers, pdf)
xx = np.linspace(xlimit*-1., xlimit)
yy = gaussian(xx, sigma2, N)
```

Plotting the fBm PDF and Gaussian fit

In [16]:
```
plt.figure(dpi = 150, figsize = (12,4))
plt.grid(alpha = 0.4)
plt.stairs(pdf, bin_edges, fill=True, alpha = 0.5)
plt.plot(xx,yy, 'r:', lw = 5, label = "Gaussian fit: \n" \
        + r"$\sigma^2$ = %.2f (%.2f)" % (sigma2, err_sigma2) \
         + "\n N = %.2f (%.2f)" % (N, err_N))
plt.minorticks_on()
plt.title(r"Sample %d; H=%.2f; $\Delta=%d$" % (sample+1, H_test, tau_test
plt.legend()
plt.show()
```



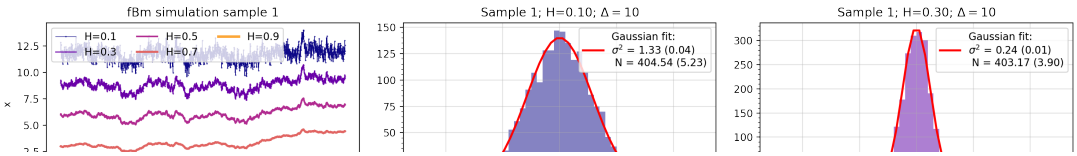
As observed, there is a normal distribution in the displacements in the subdiffusive fBm sample. A gaussian fit is overlayed in red broken line given the standard deviation of the histogram.

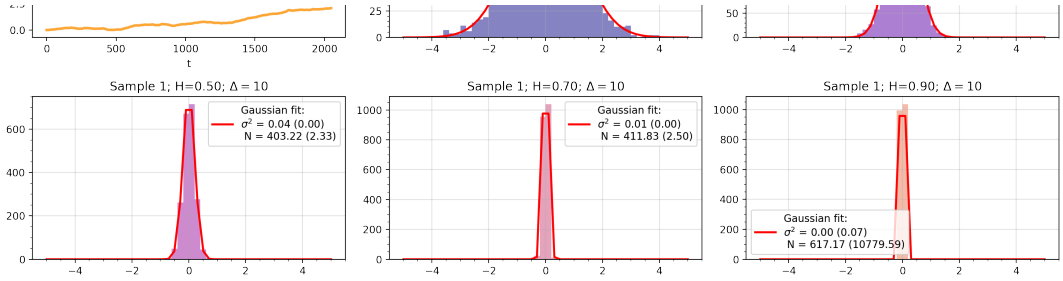
Automate Plotting

In [17]:
```
def plot_PDF_fBm(sample, H_n, tau_test, xlimit, n_bins, c):
    H_test = Hval[H_n]
    test_data = fBm_simulations[sample][:, H_n]

    test_dx = get_sample_dx(test_data, tau_test)

    bin_edges = np.linspace(xlimit*-1., xlimit, n_bins+1)
    pdf = get_pdf(test_dx, tau_test, bin_edges, norm=False)

    bin_centers = 0.5*(bin_edges[:-1] + bin_edges[1:])
    sigma2, err_sigma2, N, err_N = fit_pdf(bin_centers, pdf)
    xx = np.linspace(xlimit*-1., xlimit)
    yy = gaussian(xx, sigma2, N)

    plt.grid(alpha = 0.4)
    plt.stairs(pdf, bin_edges, fill=True, alpha = 0.5, color = c)
    plt.plot(xx,yy, 'r-', lw = 2, label = "Gaussian fit: \n" \
            + r"$\sigma^2$ = %.2f (%.2f)" % (sigma2, err_sigma2) \
             + "\n N = %.2f (%.2f)" % (N, err_N))
    plt.minorticks_on()
    plt.title(r"Sample %d; H=%.2f; $\Delta=%d$" % (sample+1, H_test, tau_
    plt.minorticks_on()

    plt.legend(loc = 0)
#     plt.ylim(0,1)
```

fBm PDFs for different Hurst Exponents

In [18]:
```
# generate plot for 1 sample
i_samp = 0 # select sample to plot (by index)
plt.figure(dpi = 200, figsize = (15,6))

plt.subplot(2,3,1)
for i in np.arange(len(Hval)):
    plt.plot(fBm_simulations[i_samp][:, i], label="H=%.1f" % Hval[i],
             marker = '.', ls = '-', lw = 0.5*(i+1), ms = 0.7,
             color = plt.cm.plasma(i/5))
plt.xlabel("t")
plt.ylabel("x")
plt.title("fBm simulation sample " + str(i_samp+1))
plt.legend(ncol=3)

for i in np.arange(5):
    plt.subplot(2,3,i+2)
    plot_PDF_fBm(sample = i_samp, H_n = i, tau_test = 10, xlimit = 5, n_b

plt.subplots_adjust(bottom=0.1, right=0.8, top=0.9, hspace=0.5, wspace=0.
plt.tight_layout()
```





Getting the PDFs for varying Hurst exponents H show that indeed, a great variability in the displacements is evident in the subdiffussive regime, while superdiffusive fBm samples tend to have a narrower distribution, meaning, the displacements are incremental.

fBm PDFs for different τ

Setting log-spaced τ values

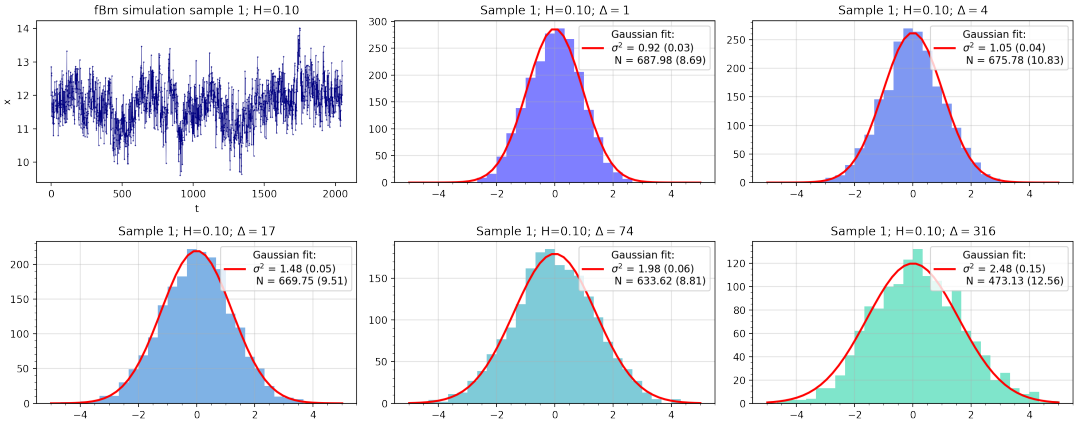
In [19]:
```
n_delta_init = 5 # later, can increase this
logdelta_min = 0
logdelta_max = 2.5
delta_vals = np.unique(np.floor(np.logspace(logdelta_min, logdelta_max, n
delta_vals
```

Out[19]:
```
array([  1,   4,  17,  74, 316])
```

fBm Sample 1, H = 0.1, increasing τ values

In [20]:
```
# generate plot for 1 sample
i_samp = 0 # select sample to plot (by index)
H_n = 0

plt.figure(dpi = 200, figsize = (15,6))

plt.subplot(2,3,1)
plt.plot(fBm_simulations[i_samp][:, H_n],
         marker = '.', ls = '-', lw = 0.3, ms = 0.7,
         color = plt.cm.jet(H_n/5))
plt.xlabel("t")
plt.ylabel("x")
```

```
plt.title("fBm simulation sample %d; H=%.2f" % (i_samp+1, Hval[H_n]))
# plt.legend(ncol=3)

for i in np.arange(5):
    plt.subplot(2,3,i+2)
    tau = delta_vals[i]
    plot_PDF_fBm(sample = i_samp, H_n = H_n, tau_test = tau, xlimit = 5,
#     plt.ylim(0, 0.5)

plt.subplots_adjust(bottom=0.1, right=0.8, top=0.9, hspace=0.5, wspace=0.
plt.tight_layout()
```



Here we investigate if observations still hold true for varying lag times. As shown above for $H = 0.1$, increasing the lag times generate roughly the same normally distributed PDFs for a subdiffusive fBm sample. Meaning, displacements vary greatly regardeless of the observation window.

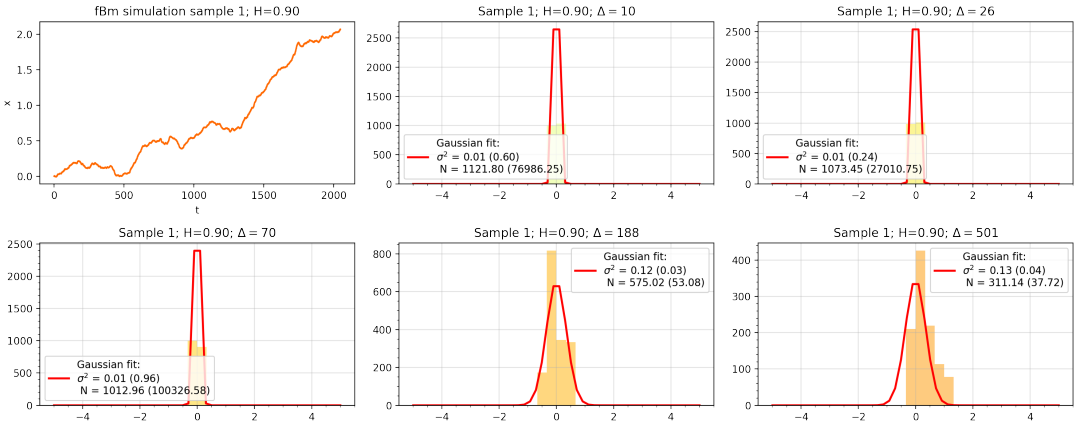
Sample 1, H = 0.9, increasing τ values

In [21]:
```
n_delta_init = 5 # later, can increase this
logdelta_min = 1
logdelta_max = 2.7
delta_vals = np.unique(np.floor(np.logspace(logdelta_min, logdelta_max, n
delta_vals
```

Out[21]:
```
array([ 10,  26,  70, 188, 501])
```

In [22]:
```
# generate plot for 1 sample
i_samp = 0 # select sample to plot (by index)
H_n = 4

plt.figure(dpi = 200, figsize = (15,6))

plt.subplot(2,3,1)
plt.plot(fBm_simulations[i_samp][:, H_n],
         marker = '.', ls = '-', lw = 0.3*H_n, ms = 0.7,
         color = plt.cm.jet(H_n/5))
plt.xlabel("t")
plt.ylabel("x")
plt.title("fBm simulation sample %d; H=%.2f" % (i_samp+1, Hval[H_n]))
# plt.legend(ncol=3)

for i in np.arange(5):
    plt.subplot(2,3,i+2)
    tau = delta_vals[i]
    plot_PDF_fBm(sample = i_samp, H_n = H_n, tau_test = tau, xlimit = 5,
#     plt.ylim(0,0.8)
```

```
plt.subplots_adjust(bottom=0.1, right=0.8, top=0.9, hspace=0.5, wspace=0.
plt.tight_layout()
```



For $H = 0.9$, displacements are very low for small lag times and this manifests as narrow PDFs due to small standard deviations. However, increasing the lag times result to a subtle increase in the standard deviations since the displacements are now more apparent. Hence, we can see how PDFs are wider on higher lag times. Since this is a superdifussive sample, an affinity to go a particular + or - direction is apparent hence, a certain skewness in the PDF is seen for higher lag times.

Calculating MSD(τ) for the five samples

In this portion, MSD calculation summarizes the behavior of displacements throughout a wider set of lag times.

In [23]:
```
n_delta_init = 22
# n_delta_init = M
logdelta_min = 0
logdelta_max = 3.3
delta_vals = np.unique(np.floor(np.logspace(logdelta_min, logdelta_max, n
delta_vals
```

Out[23]:
```
array([   1,    2,    4,    6,    8,   12,   18,   25,   37,   53,   76,
        110,  158,  227,  326,  469,  673,  967, 1389, 1995])
```

MSD for H = 0.1

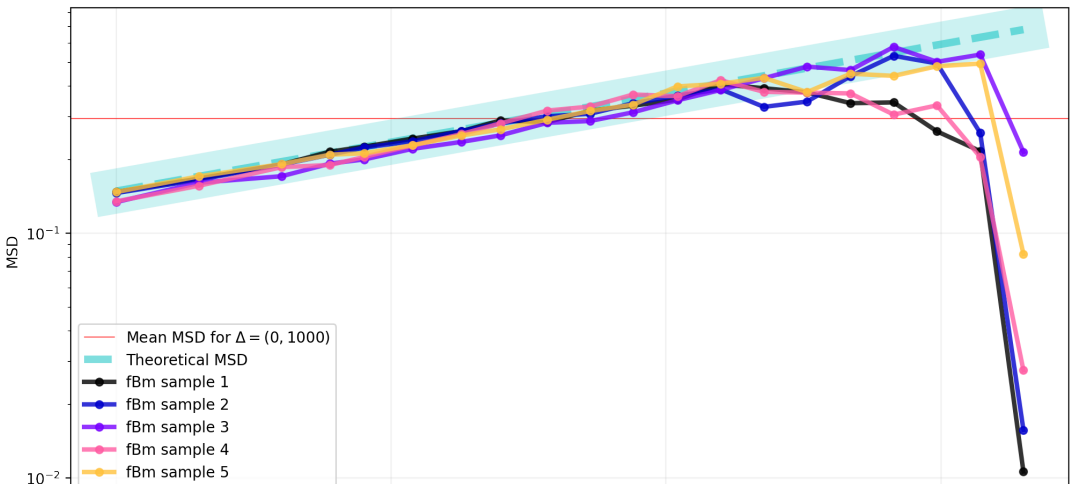
In [24]:
```
n_samp = 5
H_n = 0
n_tau = len(delta_vals)

dx_tau = np.empty((M, n_tau, n_samp))*np.nan
msd_tau = np.empty((n_tau, n_samp))*np.nan

for i_sample in range(n_samp):
    for i, tau in tqdm(enumerate(delta_vals)):
        test_data = fBm_simulations[i_sample][:, H_n]
        test_dx = get_sample_dx(test_data, tau)
        dx_tau[:len(test_dx), i, i_sample] = test_dx
        dx2_sum = np.nansum(test_dx**2) # returns sum treating NaNs as ze
        denom = M-delta_vals[i]
        msd_tau[i, i_sample] = dx2_sum/denom
```

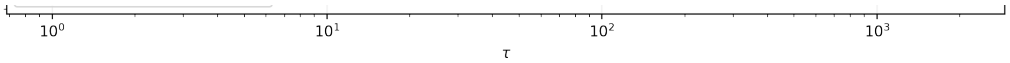
```
20it [00:00, 6707.13it/s]
20it [00:00, 7344.90it/s]
20it [00:00, 9775.79it/s]
20it [00:00, 4838.00it/s]
20it [00:00, 9385.33it/s]
```

In [25]:
```
from scipy.special import gamma

def msd_theo_fbm_N(t, H, N):
    gamma_term = gamma(H+0.5)
    denom = 2*H*gamma_term**2
    return N*t**(2*H)/denom
```

In [26]:
```
MSD_theo = msd_theo_fbm_N(delta_vals, 0.1, M)
k = MSD_theo[1]/msd_tau[:,i_sample][1] #scaling factor
```

In [27]:
```
plt.figure(dpi = 200, figsize = (12,6))
plt.grid(alpha = 0.2)

plt.axhline(np.mean(msd_tau[:-3,:]), color = 'r', ls= '-', lw = 0.5, labe
plt.plot(delta_vals, MSD_theo/k, 'c-', lw = 30, alpha = 0.2)
plt.plot(delta_vals, MSD_theo/k, 'c--', lw = 5, alpha = 0.5, label = 'The

for i_sample in range(n_samp):
    plt.plot(delta_vals, msd_tau[:,i_sample], '.-', alpha = 0.8, ms = 10,
             label = "fBm sample %d" % (i_sample+1),
             color = plt.cm.gnuplot2(i_sample/5))

plt.minorticks_on()
plt.xscale("log")
plt.yscale("log")
plt.xlabel(r"$\tau$")
plt.ylabel("MSD")
plt.legend()
plt.show()
```





For small Hurst exponent ($H = 0.1$), the MSD steadily increased but roughly stayed on around the same value (red broken line which is the average MSD) across the five fBm samples. This agrees with the discussion when plotting the PDFs that the displacements of fluctuations remain fairly consistent regardless of the lag time.

Fitting the theoretical MSD given by

$$MSD_{theoretical} = \frac{t^{2H}}{2H[\Gamma(H+\frac{1}{2})]^2} \tag{1}$$

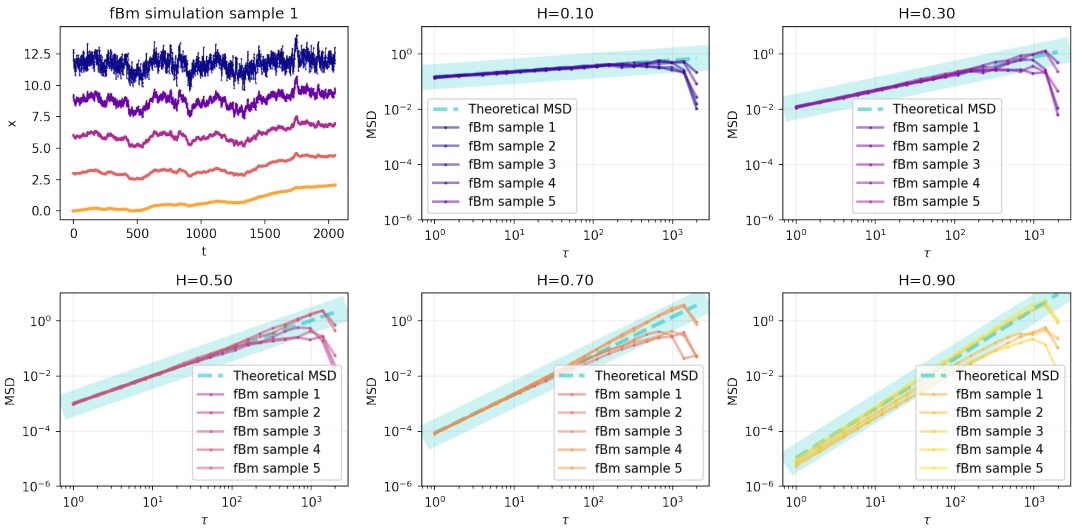
where t is the respective lag time, we can see how the simulated fBm samples' empirical MSD is in close correspondence with the theoretical MSD fit (in cyan line).

Automate Plotting for different H values

In [28]:
```
n_delta_init = 22
logdelta_min = 0
logdelta_max = 3.3
delta_vals = np.unique(np.floor(np.logspace(logdelta_min, logdelta_max, n
delta_vals

def plot_MSD_fBm(n_samp, H_n, delta_vals):

    n_tau = len(delta_vals)
    dx_tau = np.empty((M, n_tau, n_samp))*np.nan
    msd_tau = np.empty((n_tau, n_samp))*np.nan

    for i_sample in range(n_samp):
        for i, tau in tqdm(enumerate(delta_vals)):
            test_data = fBm_simulations[i_sample][:, H_n]
            test_dx = get_sample_dx(test_data, tau)
            dx_tau[:len(test_dx), i, i_sample] = test_dx
            dx2_sum = np.nansum(test_dx**2) # returns sum treating NaNs a
            denom = M-delta_vals[i]
            msd_tau[i, i_sample] = dx2_sum/denom

    MSD_theo = msd_theo_fbm_N(delta_vals, Hval[H_n], M)
    k = MSD_theo[1]/msd_tau[:,i_sample][1] #scaling factor
    plt.plot(delta_vals, MSD_theo/k, 'c-', lw = 20, alpha = 0.2)
    plt.plot(delta_vals, MSD_theo/k, 'c--', lw = 3, alpha = 0.5, label =

    plt.grid(alpha = 0.2)
    for i_sample in range(n_samp):
        plt.plot(delta_vals, msd_tau[:,i_sample], '.-', alpha = 0.5, ms =
                 label = "fBm sample %d"  % (i_sample+1), color = plt.cm.

    plt.minorticks_on()
    plt.xscale("log")
    plt.yscale("log")
    plt.xlabel(r"$\tau$")
    plt.ylabel("MSD")
    plt.title("H=%.2f" % Hval[H_n])
    plt.legend()
```

In [29]:
```
# generate plot for 1 sample
i_samp = 0 # select sample to plot (by index)
plt.figure(dpi = 150, figsize = (12,6))
plt.subplot(2,3,1)
for i in np.arange(len(Hval)):
    plt.plot(fBm_simulations[i_samp][:, i], label="H=%.1f" % Hval[i],
             marker = '.', ls = '-', lw = 0.5*(i+1), ms = 0.7,
             color = plt.cm.plasma(i/5))
plt.xlabel("t")
plt.ylabel("x")
plt.title("fBm simulation sample " + str(i_samp+1))
# plt.legend(ncol=3)

for i in np.arange(5):
    plt.subplot(2,3,i+2)
    plot_MSD_fBm(n_samp = 5, H_n = i, delta_vals = delta_vals)
    plt.ylim(1e-6,10)

plt.subplots_adjust(bottom=0.1, right=0.8, top=0.9, hspace=0.5, wspace=0.
plt.tight_layout()
plt.show()
```

```
20it [00:00, 16288.56it/s]
20it [00:00, 7388.24it/s]
20it [00:00, 8997.76it/s]
20it [00:00, 6472.19it/s]
20it [00:00, 7217.87it/s]
20it [00:00, 11786.72it/s]
20it [00:00, 7494.51it/s]
20it [00:00, 10472.67it/s]
20it [00:00, 7933.24it/s]
20it [00:00, 9288.68it/s]
20it [00:00, 7558.67it/s]
20it [00:00, 10126.28it/s]
20it [00:00, 8257.32it/s]
20it [00:00, 6170.82it/s]
20it [00:00, 8598.41it/s]
20it [00:00, 14225.21it/s]
20it [00:00, 7716.50it/s]
20it [00:00, 7587.38it/s]
20it [00:00, 6980.62it/s]
20it [00:00, 10061.90it/s]
20it [00:00, 11074.07it/s]
20it [00:00, 10654.91it/s]
20it [00:00, 9866.63it/s]
20it [00:00, 8015.87it/s]
20it [00:00, 6565.91it/s]
```



Plotting the MSD for H values, we can see a steady increase in the slope which agrees with the observation when plotting the PDFs. The samples in the subdiffusive regime has a fairly consistent in MSD despite the lag time. Meanwhile, the samples in the superdiffusive regime starts with very small displacements and increases in MSD with higher lag times, which manifests as a linearly increasing behaviour in the MSD log-log plots.

Across all the samples for all Hurst exponents, the MSD closely falls within the theoretical expectation as shown by the overlaid cyan lines.

Application on Black Marble NTL dataset

Importing Libraries

In [30]:
```
from tqdm import tqdm
import itertools
import statsmodels.api as sm
import warnings
warnings.filterwarnings('ignore')
from matplotlib import gridspec
import matplotlib.dates as mdates
from matplotlib.dates import DateFormatter
import datetime
from scipy.stats import norm, describe

import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

Loading extracted NTL data

In [31]:
```
data_path = "../Project/data/VNP46A2_Regional/NCR_NTL_VNP46A2.csv"
```

Read saved CSV file

In [32]:
```
data = pd.read_csv(data_path, parse_dates=True, index_col = 0)
data.head()
```

Out[32]:

date	DNB_BRDF_Corrected_NTL	DNB_Lunar_Irradiance	Gap_Filled_DNB_BRDF_Correcte
2012-01-19	21.733953	0.5	21.9
2012-01-20	28.488267	0.5	27.8
2012-01-21	19.911453	0.5	21.1
2012-01-22	18.401624	0.5	18.2
2012-01-23	18.080945	0.5	17.6

In [33]:
```
data.index
```

Out[33]:
```
DatetimeIndex(['2012-01-19', '2012-01-20', '2012-01-21', '2012-01-22',
               '2012-01-23', '2012-01-24', '2012-01-25', '2012-01-26',
               '2012-01-27', '2012-01-28',
               ...
               '2023-06-02', '2023-06-03', '2023-06-04', '2023-06-05',
               '2023-06-06', '2023-06-07', '2023-06-08', '2023-06-09',
               '2023-06-10', '2023-06-11'],
              dtype='datetime64[ns]', name='date', length=4142, freq=None
)
```

In [34]:
```
n_init = len(data)
print("Size of original dataset: %d" % n_init)
```

```
Size of original dataset: 4142
```

Descriptive statistics

In [35]:
```
data.describe()
```

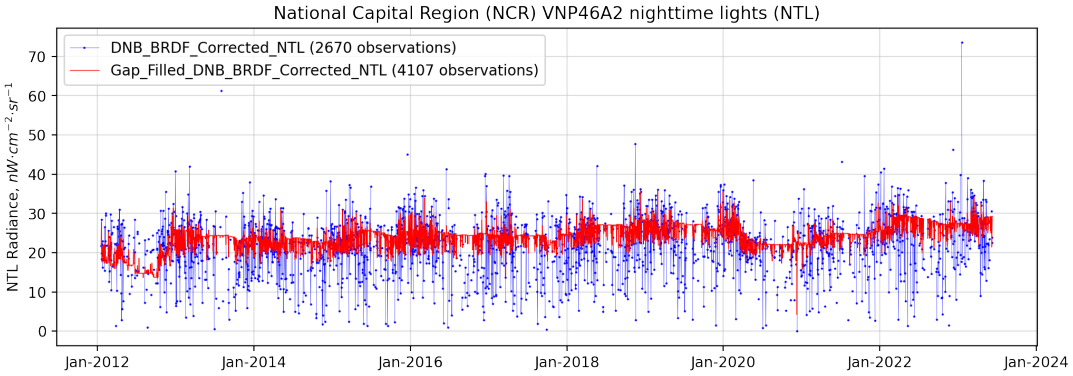
Out[35]:

	DNB_BRDF_Corrected_NTL	DNB_Lunar_Irradiance	Gap_Filled_DNB_BRDF_Correcte
count	2670.000000	4107.000000	4107.0
mean	22.044298	26.312285	23.8
std	7.017390	36.803359	3.2
min	0.032562	0.500000	4.2
25%	18.622556	0.500000	22.0

50%	22.592008	0.500000	24.2
75%	26.418989	44.950000	25.7
max	73.589233	164.198530	35.9

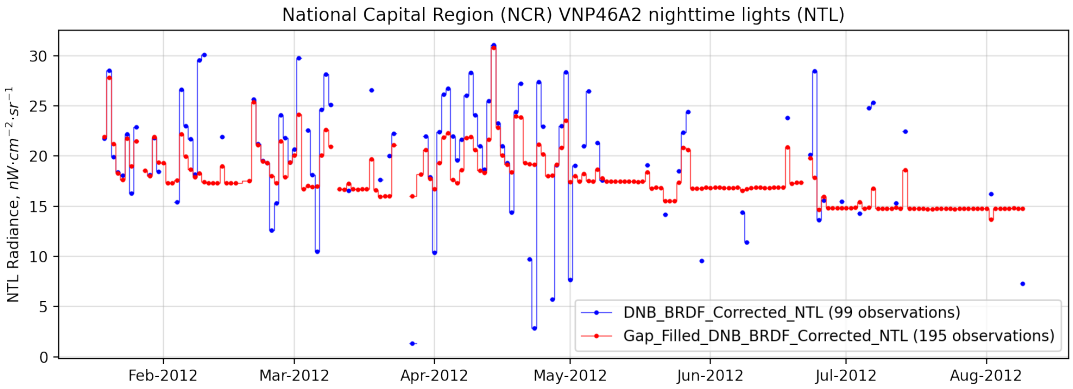
Plotting the timeseries data

In [36]:
```
plt.figure(dpi = 200, figsize = (12,4))
plt.grid(alpha = 0.4)

NTL = data['DNB_BRDF_Corrected_NTL']
NTL_GF = data['Gap_Filled_DNB_BRDF_Corrected_NTL']

plt.plot(NTL, 'b.-', lw = 0.2, ms = 1,
         label = 'DNB_BRDF_Corrected_NTL (' + str(NTL.count()) + ' observ
plt.plot(NTL_GF, 'r-', lw = 0.5,
         label = 'Gap_Filled_DNB_BRDF_Corrected_NTL (' + str(NTL_GF.count
plt.ylabel('NTL Radiance, $nW·cm^{-2}·sr^{-1}$')

plt.title('National Capital Region (NCR) VNP46A2 nighttime lights (NTL)')

plt.gcf().autofmt_xdate
dtFmt = mdates.DateFormatter("%b-%Y") # define the formatting
plt.gca().xaxis.set_major_formatter(dtFmt) # apply the format to the desi
plt.legend()
plt.show()
```



In [37]:
```
view = 200

plt.figure(dpi = 200, figsize = (12,4))
plt.grid(alpha = 0.4)

NTL_ = data['DNB_BRDF_Corrected_NTL'][:view]
NTL_GF_ = data['Gap_Filled_DNB_BRDF_Corrected_NTL'][:view]

plt.plot(NTL_, 'bo-', lw = 0.5, ms = 2, drawstyle = 'steps-mid',
         label = 'DNB_BRDF_Corrected_NTL (' + str(NTL_.count()) + ' obser
plt.plot(NTL_GF_, 'ro-', lw = 0.5,  ms = 2, drawstyle = 'steps-mid',
         label = 'Gap_Filled_DNB_BRDF_Corrected_NTL (' + str(NTL_GF_.coun
plt.ylabel('NTL Radiance, $nW·cm^{-2}·sr^{-1}$')

plt.title('National Capital Region (NCR) VNP46A2 nighttime lights (NTL)')

plt.gcf().autofmt_xdate
```

```
dtFmt = mdates.DateFormatter( %b-%Y ) # define the formatting
plt.gca().xaxis.set_major_formatter(dtFmt) # apply the format to the desi
plt.legend()
plt.show()
```



Missing datapoints

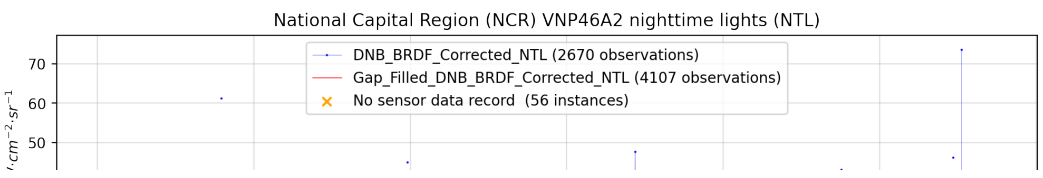
In [38]:
```
date_range = pd.date_range(start='2012-01-19', end='2023-06-12', freq='D'
NTL_reindexed = NTL.reindex(date_range)
NTL_GF_reindexed = NTL_GF.reindex(date_range)

NTL_missing_dates = NTL_reindexed[NTL_reindexed.isna()]
NTL_GF_missing_dates = NTL_GF_reindexed[NTL_GF_reindexed.isna()]
```

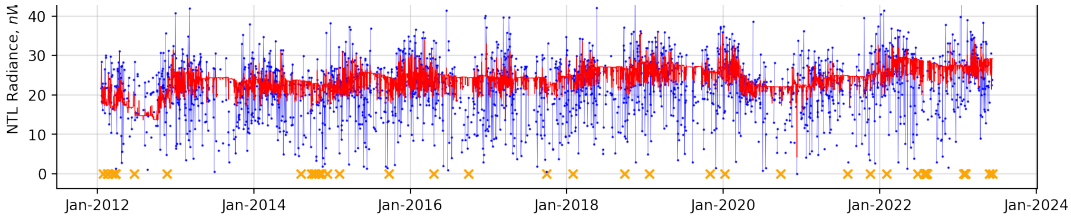
In [39]:
```
missing_days = len(date_range) - NTL_GF.count()
cloud_contaminated =  NTL.count() - missing_days
```

In [40]:
```
print("Of the " + str(len(date_range)) + " days, " + str(missing_days)+"
print("Meanwhile, "+ str(cloud_contaminated) + " days are cloud contamina
```

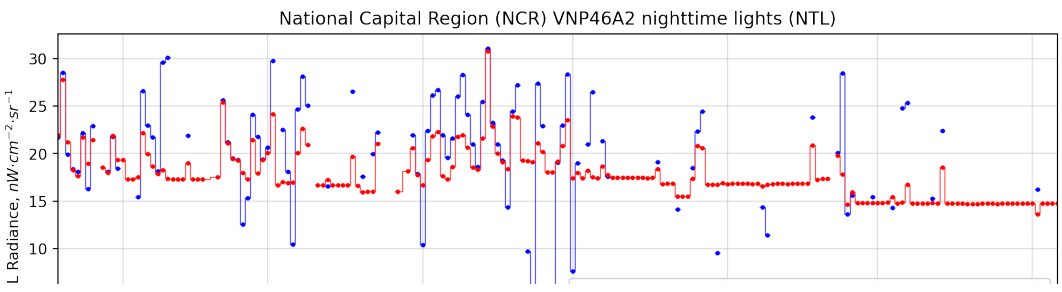
```
Of the 4163 days, 56 observations are missing.
Meanwhile, 2614 days are cloud contaminated, hence, gap-filled.
```

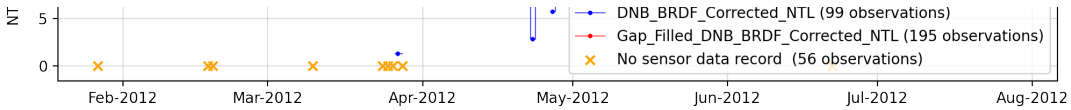
In [41]:
```
plt.figure(dpi = 200, figsize = (12,4))
plt.grid(alpha = 0.4)


plt.plot(NTL, 'b.-', lw = 0.2, ms = 1,
         label = 'DNB_BRDF_Corrected_NTL (' + str(NTL.count()) + ' observ
plt.plot(NTL_GF, 'r-', lw = 0.5,
         label = 'Gap_Filled_DNB_BRDF_Corrected_NTL (' + str(NTL_GF.count
plt.ylabel('NTL Radiance, $nW·cm^{-2}·sr^{-1}$')

plt.title('National Capital Region (NCR) VNP46A2 nighttime lights (NTL)')

plt.gcf().autofmt_xdate
dtFmt = mdates.DateFormatter("%b-%Y") # define the formatting
plt.gca().xaxis.set_major_formatter(dtFmt) # apply the format to the desi
plt.scatter(NTL_GF_missing_dates.index, [0] * len(NTL_GF_missing_dates),
            label='No sensor data record  (' + str(missing_days) + ' inst
plt.legend()
plt.show()
```





In [42]:
```
view = 200

plt.figure(dpi = 200, figsize = (12,4))
plt.grid(alpha = 0.4)

plt.plot(NTL_, 'bo-', lw = 0.5, ms = 2, drawstyle = 'steps-mid',
         label = 'DNB_BRDF_Corrected_NTL (' + str(NTL_.count()) + ' obser
plt.plot(NTL_GF_, 'ro-', lw = 0.5,  ms = 2, drawstyle = 'steps-mid',
         label = 'Gap_Filled_DNB_BRDF_Corrected_NTL (' + str(NTL_GF_.coun
plt.ylabel('NTL Radiance, $nW·cm^{-2}·sr^{-1}$')

plt.title('National Capital Region (NCR) VNP46A2 nighttime lights (NTL)')

plt.gcf().autofmt_xdate
dtFmt = mdates.DateFormatter("%b-%Y") # define the formatting
plt.gca().xaxis.set_major_formatter(dtFmt) # apply the format to the desi


plt.scatter(NTL_GF_missing_dates.index, [0] * len(NTL_GF_missing_dates[:v
            label='No sensor data record  (' + str(missing_days) + ' obse
plt.xlim(date_range[0], date_range[view])
plt.legend()
plt.show()
```





Generating PDFs

In [43]:
```
data_sample = NTL_GF
n = len(data_sample)
```

We select τ = 7 (weekly scale)

In [44]:
```
tau = 7 #7 days
dt = 1 #1 day interval

delta = int(np.round(tau/dt))
print("tau: %d, Delta: %d" % (tau, delta))
```

```
tau: 7, Delta: 7
```

In [45]:
```
data_trunc = data_sample[:-1*delta]
data_shift = data_sample[delta:]
dx = data_shift.values - data_trunc.values
len(dx), len(data_trunc), len(data_shift)
```

Out[45]:
```
(4135, 4135, 4135)
```

In [46]:
```
data_trunc.head(), data_shift.head()
```

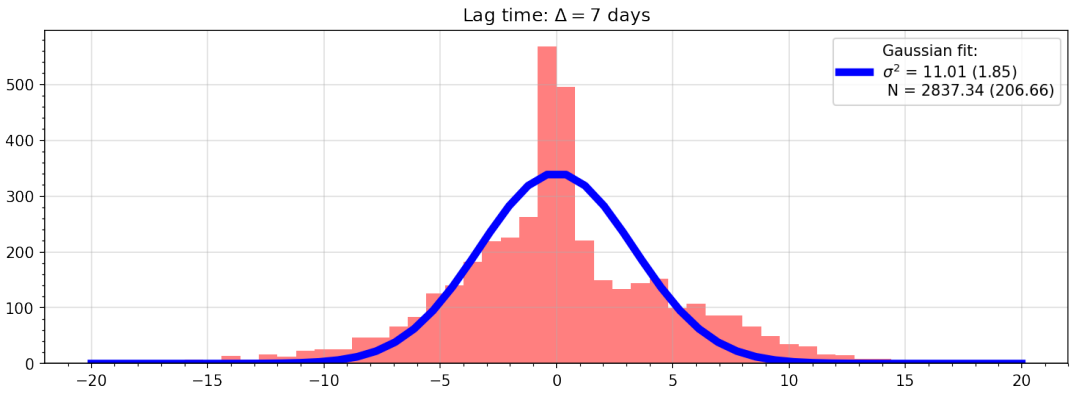
Out[46]:
```
(date
 2012-01-19    21.923586
 2012-01-20    27.802954
 2012-01-21    21.199934
 2012-01-22    18.264756
 2012-01-23    17.632629
 Name: Gap_Filled_DNB_BRDF_Corrected_NTL, dtype: float64,
 date
 2012-01-26    21.449830
 2012-01-27          NaN
 2012-01-28    18.568147
 2012-01-29    17.997314
 2012-01-30    21.906167
 Name: Gap_Filled_DNB_BRDF_Corrected_NTL, dtype: float64)
```

In [47]:
```
# view = 200
# plt.figure(dpi = 150, figsize = (15,3))
# plt.grid()
# plt.plot(data_trunc[:view].values, 'b.-', alpha = 0.5, label = 'truncat
# plt.plot(data_shift[:view].values, 'g.-', alpha = 0.5, label = 'shifted
# plt.plot(dx[:view-tau], drawstyle = 'steps-mid', color = 'red', label =
# plt.legend()
# plt.show()

# plt.figure(dpi = 150, figsize = (15,3))
# plt.grid()
# plt.plot(data_trunc.values, 'b.-', alpha = 0.5, ms = 1, lw = 0.5, label
# plt.plot(data_shift.values, 'g.-', alpha = 0.5, ms = 1, lw = 0.5, label
# plt.plot(dx[:-tau], drawstyle = 'steps-mid', lw = 0.5, color = 'red', l
# plt.legend()
# plt.show()
```

In [48]:
```
# # Generate normal distribution
# xx_sd = np.sqrt(tau)
# xx = np.linspace(-5, 5)*xx_sd # gridded points from -5 to 5 in units of
# xx_mean = 0.
# yy = norm.pdf(xx, xx_mean, xx_sd)

# plt.figure(figsize=(8,3), dpi = 150)
# plt.grid(alpha = 0.4)
# # Plot PDF of displacements
# plt.hist(dx, density=True, bins="auto", label="NTL displacements histog
# plt.xlabel("Delta x")
# plt.ylabel("PDF")

# # # Overlay normal distribution
# plt.plot(xx, yy, 'k-', label="N(%.1f,%.2f)" % (xx_mean, xx_sd))
# plt.ylabel("PDF")
# plt.xlabel(r"$\Delta x$")
# plt.legend(loc="best")

# plt.minorticks_on()
# plt.xlabel(r"$\Delta x$")
# plt.ylabel("PDF")
# plt.title(r"Lag time: $\Delta=$%d days" % (tau))
```

In [49]:
```
NTL_dx = get_sample_dx(data_sample.values, delta)

xlimit = 20.
n_bins = 50
bin_edges = np.linspace(xlimit*-1., xlimit, n_bins+1)
pdf = get_pdf(NTL_dx, tau, bin_edges, norm=False)

bin_centers = 0.5*(bin_edges[:-1] + bin_edges[1:])
sigma2, err_sigma2, N, err_N = fit_pdf(bin_centers, pdf)
xx = np.linspace(xlimit*-1., xlimit)
yy = gaussian(xx, sigma2, N)
```

In [50]:
```
plt.figure(dpi = 150, figsize = (12,4))
plt.grid(alpha = 0.4)
plt.stairs(pdf, bin_edges, fill=True, alpha = 0.5, color = 'red')
plt.plot(xx,yy, 'b-', lw = 5, label = "Gaussian fit: \n" \
        + r"$\sigma^2$ = %.2f (%.2f)" % (sigma2, err_sigma2) \
         + "\n N = %.2f (%.2f)" % (N, err_N))
plt.minorticks_on()
plt.title(r"Lag time: $\Delta=$%d days" % (tau))
plt.legend()
plt.show()
```

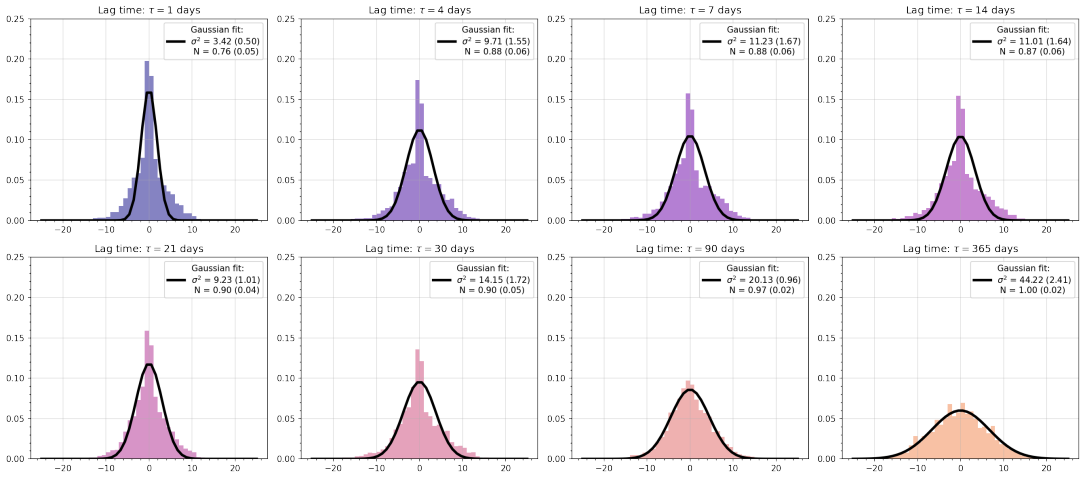


Relevant Lag times (1, 4, 7, 14, 21, 30, 90, 365)

In [51]:
```
tau_vals_1 = np.array([1, 4, 7, 14, 21, 30, 90, 365])
```

In [52]:
```
def plot_PDF(data_sample, tau, xlimit, n_bins, y_max, c):
    NTL_dx = get_sample_dx(data_sample.values, tau)
    bin_edges = np.linspace(xlimit*-1., xlimit, n_bins+1)
    pdf = get_pdf(NTL_dx, tau, bin_edges, norm=True)

    bin_centers = 0.5*(bin_edges[:-1] + bin_edges[1:])
    sigma2, err_sigma2, N, err_N = fit_pdf(bin_centers, pdf)
    xx = np.linspace(xlimit*-1., xlimit)
    yy = gaussian(xx, sigma2, N)

    plt.grid(alpha = 0.4)
    plt.stairs(pdf, bin_edges, fill=True, alpha = 0.5, color = c)
    plt.plot(xx,yy, 'k-', lw = 3, label = "Gaussian fit: \n" \
            + r"$\sigma^2$ = %.2f (%.2f)" % (sigma2, err_sigma2) \
             + "\n N = %.2f (%.2f)" % (N, err_N))
    plt.minorticks_on()
    plt.title(r"Lag time: $\tau=$%d days" % (tau))
    plt.legend(loc = 0)
    plt.ylim(0, y_max)
```

In [53]:
```
plt.figure(dpi = 150, figsize = (18,8))

for i, tau in enumerate(tau_vals_1):
    plt.subplot(2,4,i+1)
    plot_PDF(data_sample, tau, xlimit = 25, n_bins = 50, y_max = 0.25, c

plt.subplots_adjust(bottom=0.1, right=0.8, top=0.9, hspace=0.5, wspace=0.
plt.tight_layout()
```

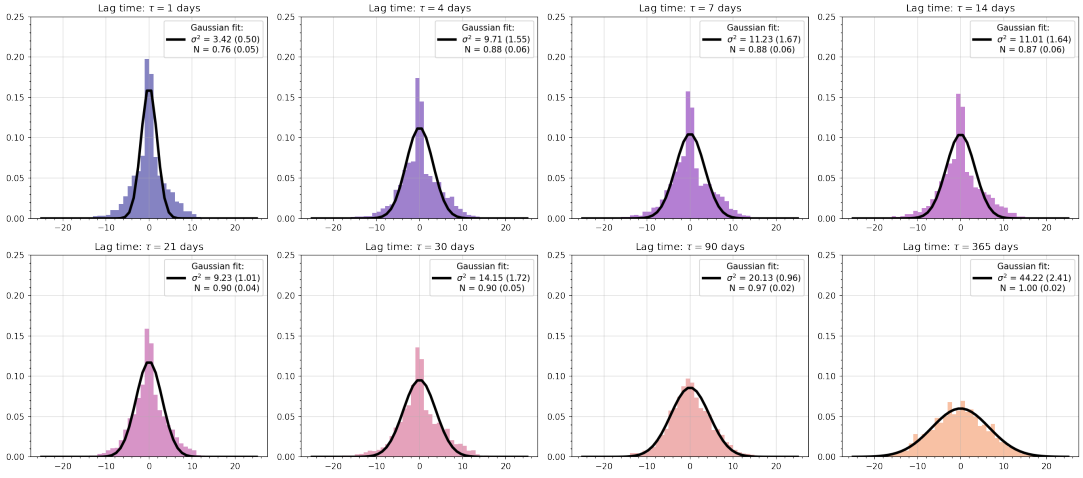


Log-spaced time intervals (1, 3, 10, 31, 100, 316, 1000, 3162)

In [54]:
```
n_delta_init = 8 # later, can increase this
logdelta_min = 0
logdelta_max = 3.5
delta_vals = np.unique(np.floor(np.logspace(logdelta_min, logdelta_max, n
delta_vals
```

Out[54]:
```
array([   1,    3,   10,   31,  100,  316, 1000, 3162])
```

In [55]:
```
plt.figure(dpi = 150, figsize = (18,8))

for i, tau in enumerate(tau_vals_1):
    plt.subplot(2,4,i+1)
    plot_PDF(data_sample, tau, xlimit = 25, n_bins = 50, y_max = 0.25, c

plt.subplots_adjust(bottom=0.1, right=0.8, top=0.9, hspace=0.5, wspace=0.
plt.tight_layout()
```



Generating MSDs

Discrete Lag Times

In [56]:
```
n_delta_init = 30 # later, can increase this
logdelta_min = 0
logdelta_max = 3.5
delta_vals = np.unique(np.floor(np.logspace(logdelta_min, logdelta_max, n
```

```
delta_vals

n = len(data)
n_samp = 1
n_tau = len(delta_vals)
dx_tau = np.empty((n, n_tau))*np.nan
for i, tau in tqdm(enumerate(delta_vals)):
    delta = delta_vals[i]
    data_trunc = data_sample[:-1*delta]
    data_shift = data_sample[delta:]
    dx = data_shift.values - data_trunc.values
    dx_tau[:len(dx), i] = dx.T
```

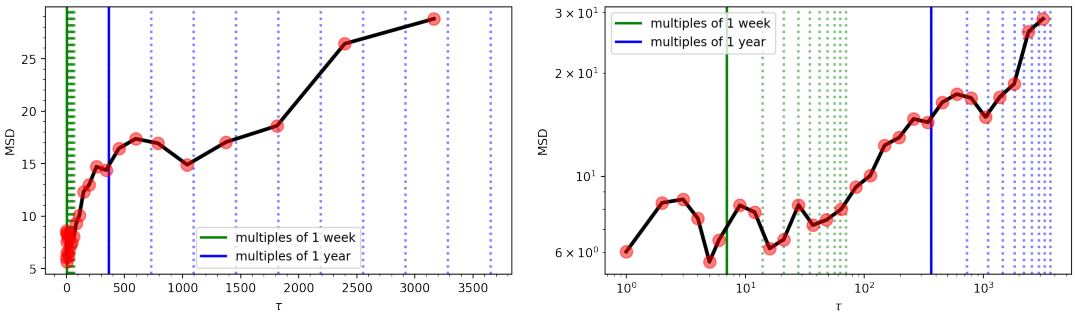
```
28it [00:00, 7449.45it/s]
```

In [57]:
```
msd_tau = np.empty((n_tau, n_samp))*np.nan
for i, tau in enumerate(delta_vals):
    dx2_sum = np.nansum(dx_tau[:, i]**2) # returns sum treating NaNs as z
    denom = n-delta_vals[i]
    msd_tau[i, i_samp] = dx2_sum/denom
```

In [58]:
```
def show_markers():
    for year in np.arange(1,11)*7: plt.axvline(year, ls=':', color = 'g',
    plt.axvline(7, ls='-', color = 'g', lw = 2, label = 'multiples of 1 w
    for year in np.arange(1,11)*365: plt.axvline(year, ls=':', color = 'b
    plt.axvline(365, ls='-', color = 'b', lw = 2, label = 'multiples of 1
```

In [59]:
```
plt.figure(dpi = 200, figsize = (15,4))
plt.subplot(121)
show_markers()
plt.plot(delta_vals, msd_tau, 'k-', lw = 3)
plt.plot(delta_vals, msd_tau, 'ro', alpha = 0.5,  ms = 10)
plt.xlabel(r"$\tau$")
plt.ylabel("MSD")
plt.legend()
plt.minorticks_on()


plt.subplot(122)
show_markers()
plt.plot(delta_vals, msd_tau, 'k-', lw = 3)
plt.plot(delta_vals, msd_tau, 'ro', alpha = 0.5,  ms = 10)
```

```
msd_theo = delta_vals
plt.xscale("log")
plt.yscale("log")
plt.xlabel(r"$\tau$")
plt.ylabel("MSD")
plt.legend()
plt.minorticks_on()

plt.show()
```



Here we can see a particular periodicity in the MSD plots, however, take note that we are using a discretize set of delta values that are log-spaced. NTL is an output of a complex system of nocturnal lighting from artificial and natural sources hence, I reckon that each timescale (and thus lag-time) must be investigated by plotting its corresponding MSD.

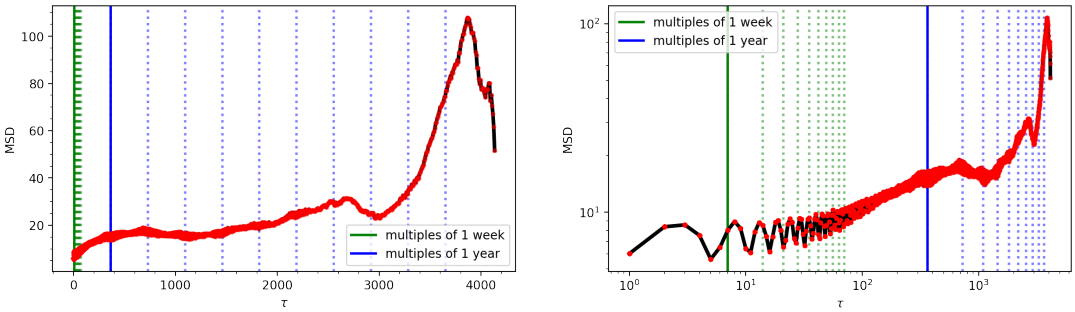
Continuous Lag Times

In [60]:

```
n_tau = len(data_sample)
tau_vals = np.logspace(0,np.log10(n_tau), n_tau)
delta_vals = np.round(tau_vals/dt).astype(int)

n = len(data)
n_samp = 1
n_tau = len(tau_vals)
dx_tau = np.empty((n, n_tau))*np.nan
for i, tau in tqdm(enumerate(tau_vals)):
    delta = delta_vals[i]
    data_trunc = data_sample[:-1*delta]
    data_shift = data_sample[delta:]
    dx = data_shift.values - data_trunc.values
    dx_tau[:len(dx), i] = dx.T

msd_tau = np.empty((n_tau, n_samp))*np.nan
for i, tau in enumerate(tau_vals):
    dx2_sum = np.nansum(dx_tau[:, i]**2) # returns sum treating NaNs as z
    denom = n-delta_vals[i]
    msd_tau[i, i_samp] = dx2_sum/denom

plt.figure(dpi = 200, figsize = (15,4))
plt.subplot(121)
```

```
plt.subplot(121)
show_markers()
plt.plot(delta_vals, msd_tau, 'k-', lw = 3)
plt.plot(delta_vals, msd_tau, 'ro', alpha = 0.5,  ms = 3)
plt.xlabel(r"$\tau$")
plt.ylabel("MSD")
plt.legend()
plt.minorticks_on()


plt.subplot(122)
show_markers()
plt.plot(delta_vals, msd_tau, 'k-', lw = 3)
plt.plot(delta_vals, msd_tau, 'ro', alpha = 0.5,  ms = 3)
msd_theo = tau_vals
plt.xscale("log")
plt.yscale("log")
plt.xlabel(r"$\tau$")
plt.ylabel("MSD")
plt.legend()
plt.minorticks_on()

plt.show()
```

```
4142it [00:00, 10374.04it/s]
```



Markers indicating multiples of the week and year are displayed as the magenta and cyan lines, respectively. Noticeably, a linear and exponential regime in the MSD is evident in the weekly and yearly timescales.

Automating the plots

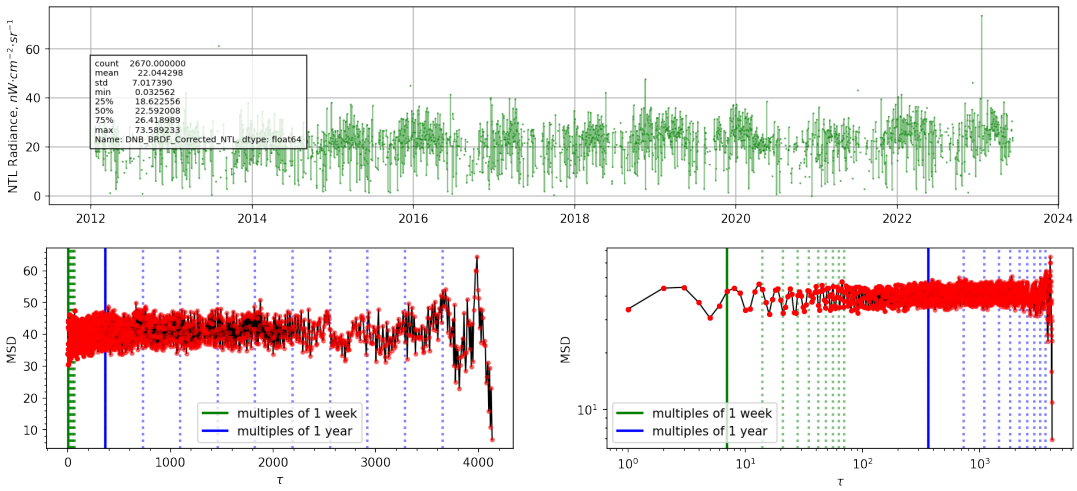
In [61]:
```
def MSD_plotted(data):
    plt.figure(dpi = 150, figsize = (15,3))
    plt.grid()
    plt.plot(data, 'g.-', drawstyle = 'steps-mid', alpha = 0.5, ms = 1, l
    plt.text(datetime.date(2012, 1, 19), data.mean(), str(data.describe()
             fontsize=7, bbox=dict(facecolor='white', alpha=0.8))
    plt.ylabel('NTL Radiance, $nW·cm^{-2}·sr^{-1}$')

    n_tau = len(data)
    tau_vals = np.logspace(0,np.log10(n_tau), n_tau)
    delta_vals = np.round(tau_vals/dt).astype(int)

    n = len(data)
    n_samp = 1
    n_tau = len(tau_vals)
    dx_tau = np.empty((n, n_tau))*np.nan

    for i, tau in tqdm(enumerate(tau_vals)):
        delta = delta_vals[i]
        data_trunc = data[:-1*delta]
        data_shift = data[delta:]
        dx = data_shift.values - data_trunc.values
        dx_tau[:len(dx), i] = dx.T

    msd_tau = np.empty((n_tau, n_samp))*np.nan
    for i, tau in enumerate(tau_vals):
        dx2_sum = np.nansum(dx_tau[:, i]**2)
        denom = n-delta_vals[i]
        msd_tau[i, i_samp] = dx2_sum/denom

    plt.figure(dpi = 150, figsize = (15,3))

    plt.subplot(121)
    show_markers()
    plt.plot(delta_vals, msd_tau, 'k-', lw = 1)
    plt.plot(delta_vals, msd_tau, 'ro', alpha = 0.5,  ms = 3)
    plt.xlabel(r"$\tau$")
    plt.ylabel("MSD")
```

```
    plt.legend()
    plt.minorticks_on()


    plt.subplot(122)
    show_markers()
    plt.plot(delta_vals, msd_tau, 'k-', lw = 1)
    plt.plot(delta_vals, msd_tau, 'ro', alpha = 0.5,  ms = 3)
    msd_theo = tau_vals
    plt.xscale("log")
    plt.yscale("log")
    plt.xlabel(r"$\tau$")
    plt.ylabel("MSD")
    plt.legend()
    plt.minorticks_on()

    plt.show()
```

[MSD] NTL data

In [62]:
```
MSD_plotted(data['DNB_BRDF_Corrected_NTL'])
```

```
4142it [00:00, 10610.10it/s]
```

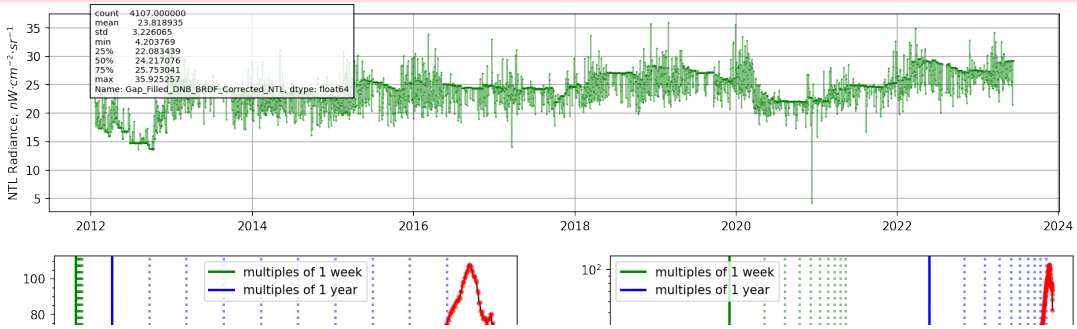


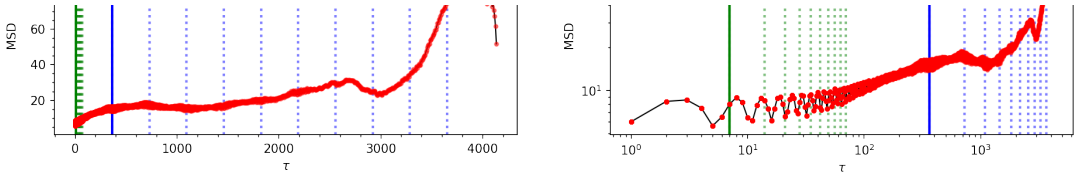
Using the untreated raw data renders a seemingly subdiffusive MSD plot, however, it's reliability is still in question especially that roughly half of the data is missing due to cloud cover.

[MSD] Gap Filled NTL data

In [63]:
```
MSD_plotted(NTL_GF)
```

```
4142it [00:00, 10406.14it/s]
```



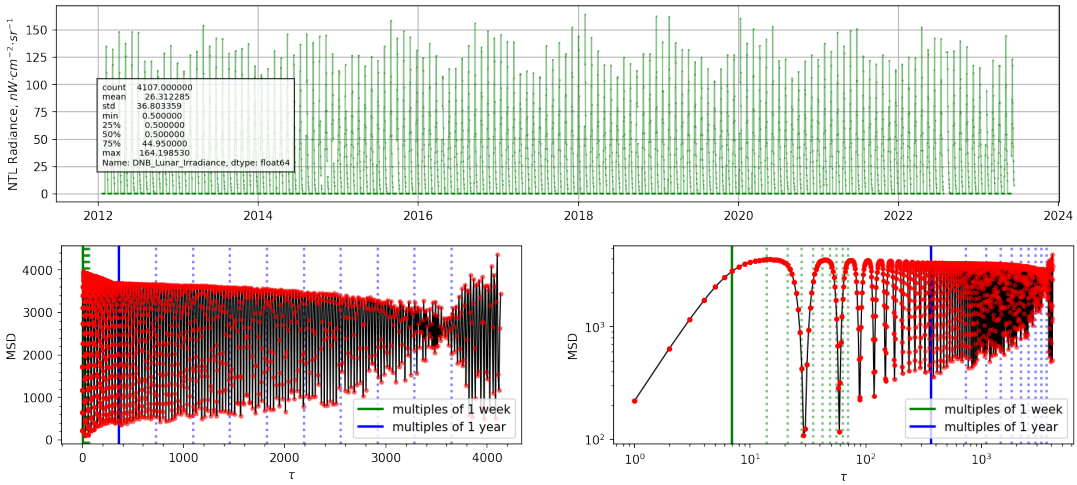


With Gap Filling algorithm on Black Marble, the data is now more continious and corrected for all the other effects. We can see here a periodic fluctuation in MSD but overall an increasing trend through larger lag times. NTL has been established to represent long term economic progression hence, this increase in trend is captures by setting larger values of τ.

[MSD] Lunar Irradiance data

In [64]:
```
MSD_plotted(data['DNB_Lunar_Irradiance'])
```

```
4142it [00:00, 10651.43it/s]
```

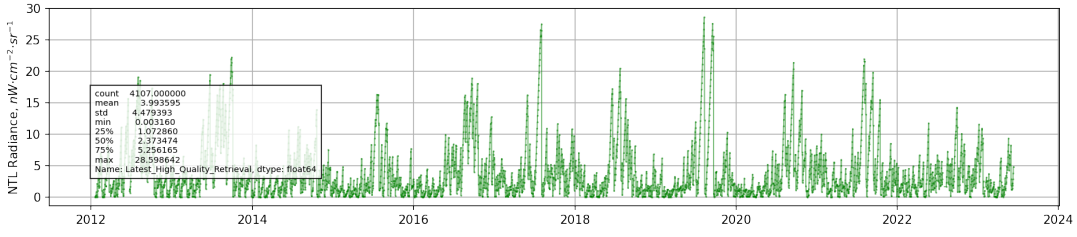


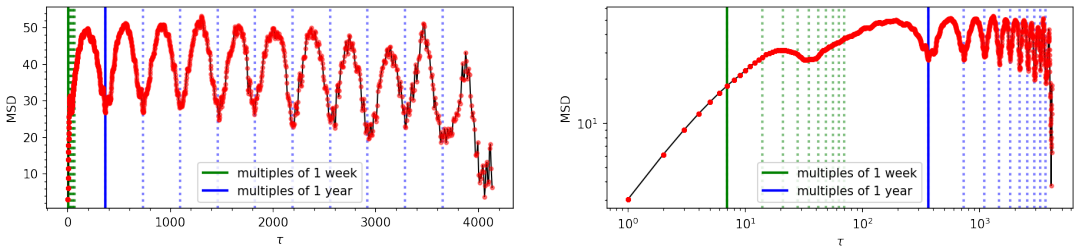
A layer of VNP46A2 accounts for the lunar effects that is used for correction and here we can see that the MSD periodically drops on multiples of 4 weeks, which is representative of one lunar cycle.

[MSD] Latest High Quality Retrieval data (in unit of days)

In [65]:
```
MSD_plotted(data['Latest_High_Quality_Retrieval'])
```

```
4142it [00:00, 9777.65it/s]
```





MSD for Regional Gap Filled NTL data

In this portion, the MSD for the NTL values aggregated on a regional scale is shown and characteristic behaviors were observed. However, a further data processing is needed to clean some of the datasets with extreme outliers which eventually affected the MSD plots.

In [66]:
```
import os

data_path = "../Project/data/VNP46A2_regional/"
regional_NTL = os.listdir(data_path)
regional_NTL
# "../data/
```

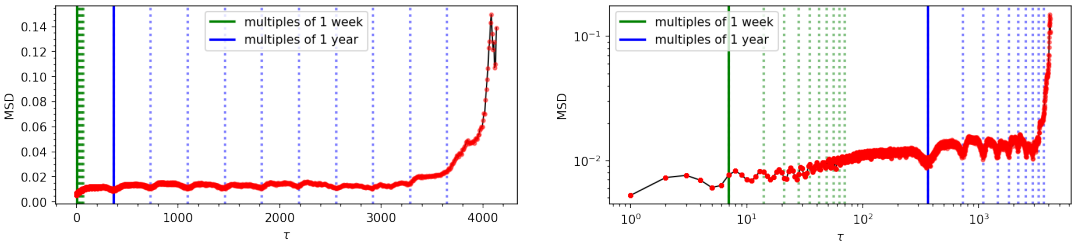
Out[66]:
```
['Region XII_NTL_VNP46A2.csv',
 'Region VI_NTL_VNP46A2.csv',
 'Region II_NTL_VNP46A2.csv',
 'ARMM_NTL_VNP46A2.csv',
 'Region IV-A_NTL_VNP46A2.csv',
 'Region I_NTL_VNP46A2.csv',
 'Region XI_NTL_VNP46A2.csv',
 'Region IX_NTL_VNP46A2.csv',
 'Region III_NTL_VNP46A2.csv',
 'NCR_NTL_VNP46A2.csv',
 'Region XIII_NTL_VNP46A2.csv',
 'Region VII_NTL_VNP46A2.csv',
 'Region VIII_NTL_VNP46A2.csv',
 'Region X_NTL_VNP46A2.csv',
 'CAR_NTL_VNP46A2.csv']
```

In [67]:
```
regions = []
MSD_reg_NTL = []

for NTL_data in tqdm(regional_NTL):

    regions.append(str(NTL_data[:-16]))

    dataframe = pd.read_csv(data_path+NTL_data, parse_dates=True, index_c
    data = dataframe['Gap_Filled_DNB_BRDF_Corrected_NTL']

    date_range = pd.date_range(start='2012-01-19', end='2023-06-12', freq
    data_reindexed = data.reindex(date_range)
    NTL_missing_dates = data_reindexed[data_reindexed.isna()]
    no_obs = len(NTL_missing_dates)

    plt.figure(dpi = 150, figsize = (15,3))
    plt.grid()
    plt.plot(data, 'g.-', drawstyle = 'steps-mid', alpha = 0.5, ms = 1, l
    plt.text(datetime.date(2012, 1, 19), data.mean()*2, str(data.describe
             fontsize=7, bbox=dict(facecolor='white', alpha=0.8))
```

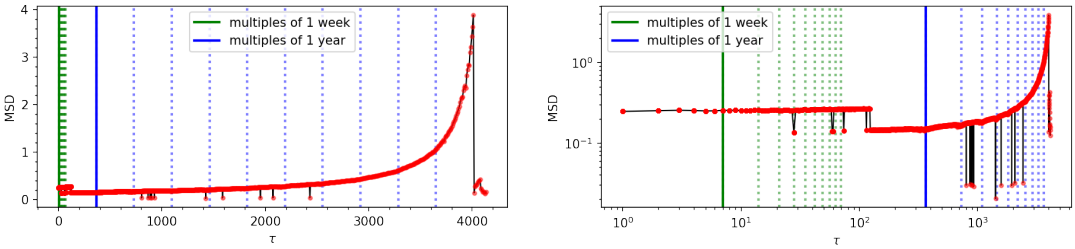
```
    plt.title(NTL_data[:-4] + ' (Days with NTL data: ' + str(len(data) -
    plt.scatter(NTL_missing_dates.index, [0] * no_obs, color='orange', ma
            label='No sensor data record  (' + str(no_obs) + ' observatio
#     plt.savefig("../results/" + str(NTL_data[:-4]) + ".png")
    plt.show()

    n_tau = len(data)
    tau_vals = np.logspace(0,np.log10(n_tau), n_tau)
    delta_vals = np.round(tau_vals/dt).astype(int)

    n = len(data)
    n_samp = 1
    n_tau = len(tau_vals)
    dx_tau = np.empty((n, n_tau))*np.nan
    msd_tau = np.empty((n_tau, n_samp))*np.nan


    for i, tau in tqdm(enumerate(tau_vals)):
        delta = delta_vals[i]
        data_trunc = data[:-1*delta]
        data_shift = data[delta:]
        dx = data_shift.values - data_trunc.values
        dx_tau[:len(dx), i] = dx.T
        dx2_sum = np.nansum(dx**2)
        denom = n-delta_vals[i]
        msd = dx2_sum/denom
        msd_tau[i, i_samp] = msd

#     plt.plot(msd_tau)
#     plt.show()

#         plt.legend(loc = 2)


    msd_tau = np.empty((n_tau, n_samp))*np.nan
    for i, tau in enumerate(tau_vals):
        dx2_sum = np.nansum(dx_tau[:, i]**2)
        denom = n-delta_vals[i]
        msd_tau[i, i_samp] = dx2_sum/denom

    MSD_reg_NTL.append(msd_tau)


    plt.figure(dpi = 150, figsize = (15,3))
    plt.subplot(121)
    show_markers()
    plt.plot(delta_vals, msd_tau, 'k-', lw = 1)
    plt.plot(delta_vals, msd_tau, 'ro', alpha = 0.5,  ms = 3)
    plt.xlabel(r"$\tau$")
    plt.ylabel("MSD")
    plt.legend()
    plt.minorticks_on()



    plt.subplot(122)
    show_markers()
    plt.plot(delta_vals, msd_tau, 'k-', lw = 1)
    plt.plot(delta_vals, msd_tau, 'ro', alpha = 0.5,  ms = 3)
    msd_theo = tau_vals
    plt.xscale("log")
    plt.yscale("log")
    plt.xlabel(r"$\tau$")
    plt.ylabel("MSD")
    plt.legend()
    plt.minorticks_on()
```

```
    plt.minorticks_on()
#     plt.savefig("../results/" + str(NTL_data[:-4]) + "_MSD.png")
    plt.show()
```

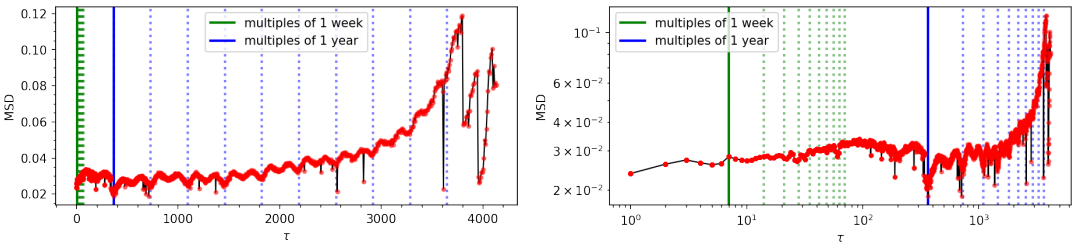
```
  0%|                                                    | 0/15 [00:00<?,
?it/s]
0it [00:00, ?it/s]
659it [00:00, 6585.85it/s]
1390it [00:00, 7011.25it/s]
2149it [00:00, 7275.10it/s]
2958it [00:00, 7595.21it/s]
4142it [00:00, 7622.14it/s]
```




```
  7%|▋                                                   | 1/15 [00:01<00:25,  1.
80s/it]
0it [00:00, ?it/s]
771it [00:00, 7704.52it/s]
1547it [00:00, 7736.54it/s]
2324it [00:00, 7748.99it/s]
3102it [00:00, 7758.83it/s]
4142it [00:00, 7898.08it/s]
```

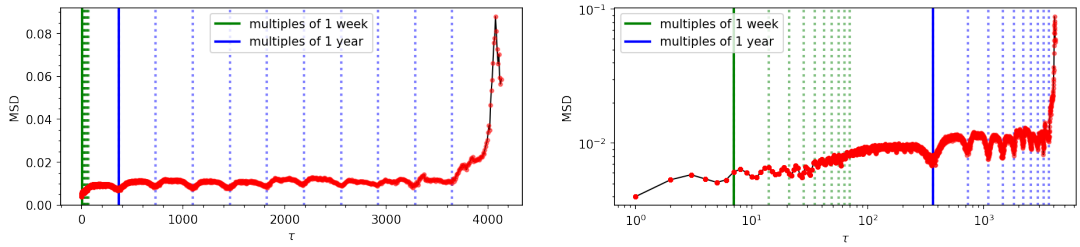



```
 13%|█▎                                                  | 2/15 [00:03<00:21,  1.
68s/it]
0it [00:00, ?it/s]
707it [00:00, 7065.63it/s]
1462it [00:00, 7345.07it/s]
2209it [00:00, 7400.15it/s]
2973it [00:00, 7494.53it/s]
4142it [00:00, 7712.86it/s]
```

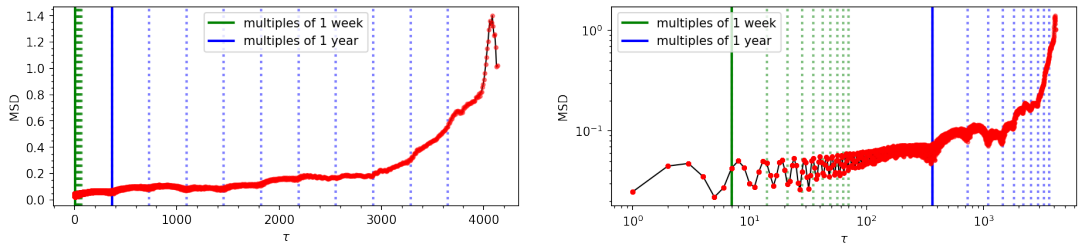



```
 20%|██                                                  | 3/15 [00:05<00:20,  1.
67s/it]
0it [00:00, ?it/s]
```

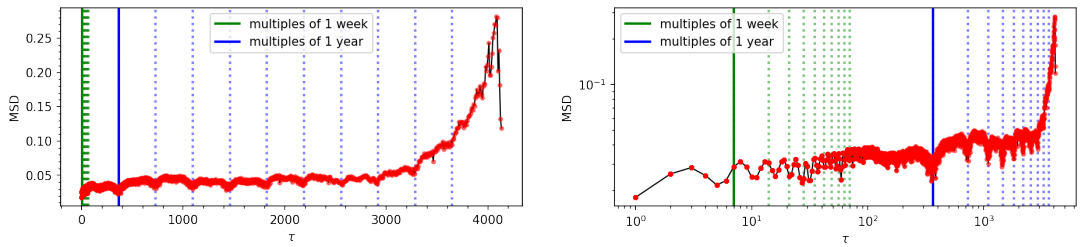
```
703it [00:00, 7027.68it/s]
1449it [00:00, 7281.26it/s]
2199it [00:00, 7379.41it/s]
2962it [00:00, 7476.03it/s]
4142it [00:00, 7654.90it/s]
```



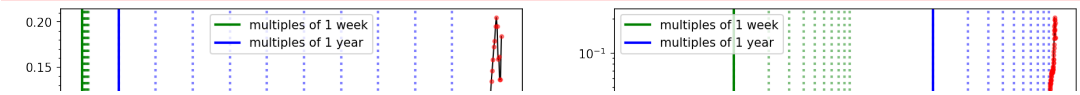

```
 27%|███████████                              | 4/15 [00:06<00:18,  1.
67s/it]
0it [00:00, ?it/s]
678it [00:00, 6772.50it/s]
1396it [00:00, 7007.86it/s]
2134it [00:00, 7175.29it/s]
2929it [00:00, 7479.40it/s]
4142it [00:00, 7659.18it/s]
```

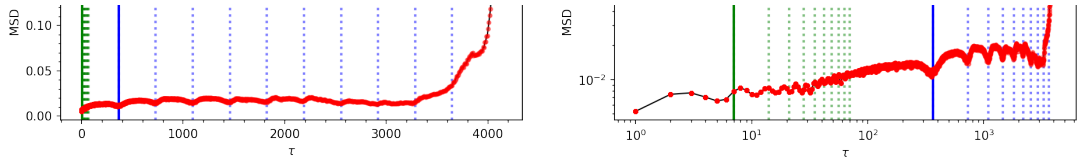



```
 33%|██████████████                           | 5/15 [00:08<00:16,  1.
65s/it]
0it [00:00, ?it/s]
776it [00:00, 7756.18it/s]
1552it [00:00, 7749.77it/s]
2327it [00:00, 7665.45it/s]
3132it [00:00, 7814.72it/s]
4142it [00:00, 7940.40it/s]
```

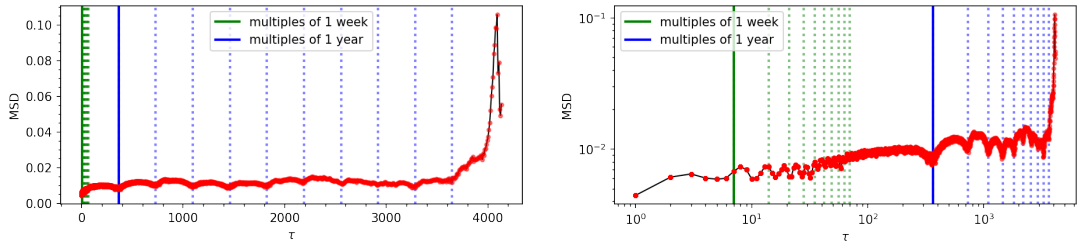



```
 40%|█████████████████                        | 6/15 [00:09<00:14,  1.
61s/it]
0it [00:00, ?it/s]
754it [00:00, 7537.58it/s]
1508it [00:00, 7433.04it/s]
2261it [00:00, 7473.01it/s]
3034it [00:00, 7572.38it/s]
4142it [00:00, 7780.31it/s]
```

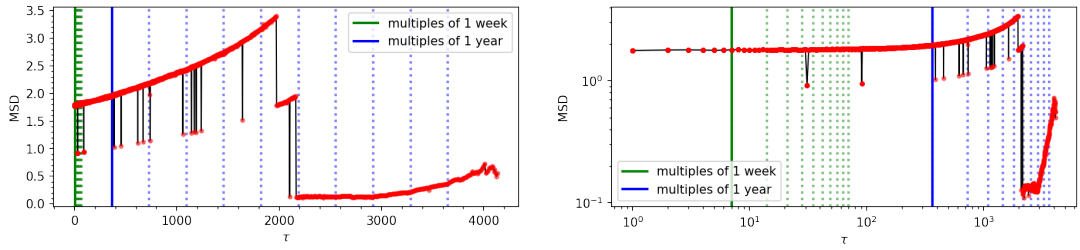




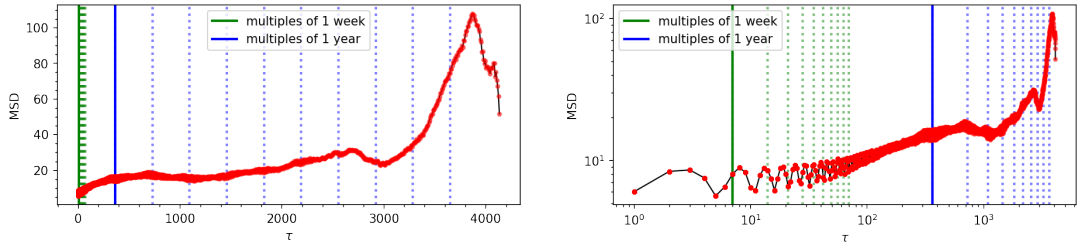

```
 47%|                                                  | 7/15 [00:11<00:12,  1.
59s/it]
0it [00:00, ?it/s]
749it [00:00, 7489.19it/s]
1545it [00:00, 7761.01it/s]
2356it [00:00, 7918.89it/s]
4142it [00:00, 8277.79it/s]
```




```
 53%|                                                  | 8/15 [00:13<00:11,  1.
59s/it]
0it [00:00, ?it/s]
755it [00:00, 7548.95it/s]
1510it [00:00, 7443.11it/s]
2255it [00:00, 7415.85it/s]
3028it [00:00, 7535.70it/s]
4142it [00:00, 7729.13it/s]
```

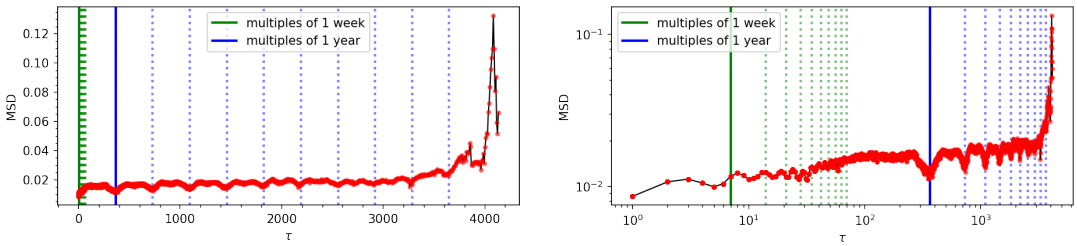



```
 60%|                                                  | 9/15 [00:14<00:09,  1.
59s/it]
0it [00:00, ?it/s]
696it [00:00, 6959.31it/s]
1482it [00:00, 7485.58it/s]
2231it [00:00, 7475.21it/s]
3007it [00:00, 7584.94it/s]
4142it [00:00, 7754.67it/s]
```

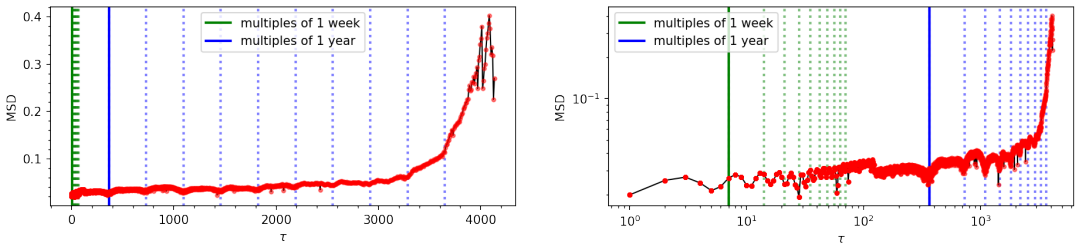



```
 67%|                                                  | 10/15 [00:16<00:08,  1.
61s/it]
```

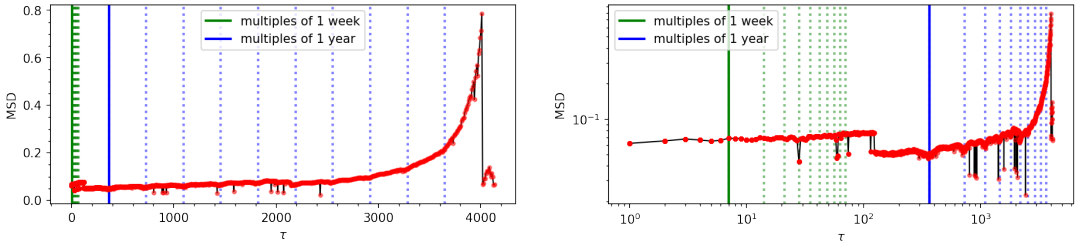
```
0it [00:00, ?it/s]
718it [00:00, 7171.59it/s]
1470it [00:00, 7373.45it/s]
2208it [00:00, 7355.99it/s]
2994it [00:00, 7553.18it/s]
4142it [00:00, 7723.94it/s]
```



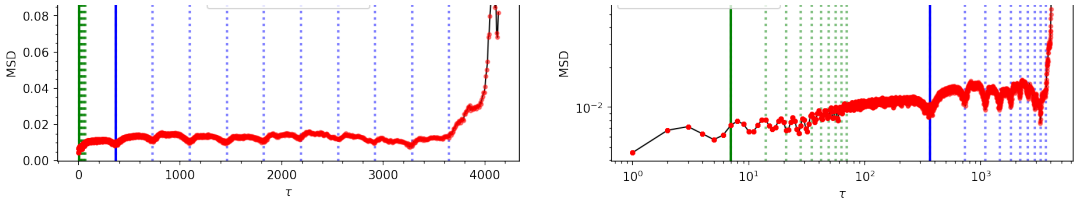

```
 73%|███████████████████████████████                | 11/15 [00:17<00:06,  1.
61s/it]
0it [00:00, ?it/s]
701it [00:00, 7005.03it/s]
1451it [00:00, 7286.97it/s]
2211it [00:00, 7425.86it/s]
3002it [00:00, 7613.95it/s]
4142it [00:00, 7773.50it/s]
```



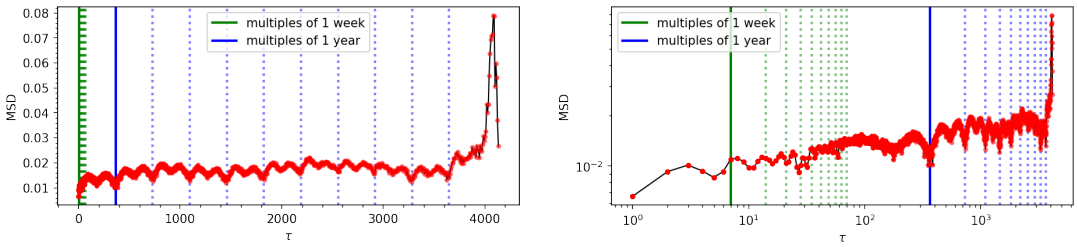

```
 80%|████████████████████████████████               | 12/15 [00:19<00:04,  1.
59s/it]
0it [00:00, ?it/s]
767it [00:00, 7661.79it/s]
1547it [00:00, 7740.68it/s]
2361it [00:00, 7919.08it/s]
3179it [00:00, 8019.50it/s]
4142it [00:00, 8062.51it/s]
```




```
 87%|███████████████████████████████████            | 13/15 [00:21<00:03,  1.
60s/it]
0it [00:00, ?it/s]
714it [00:00, 7131.37it/s]
1428it [00:00, 7056.42it/s]
2134it [00:00, 4521.99it/s]
2863it [00:00, 5332.33it/s]
4142it [00:00, 5796.87it/s]
```



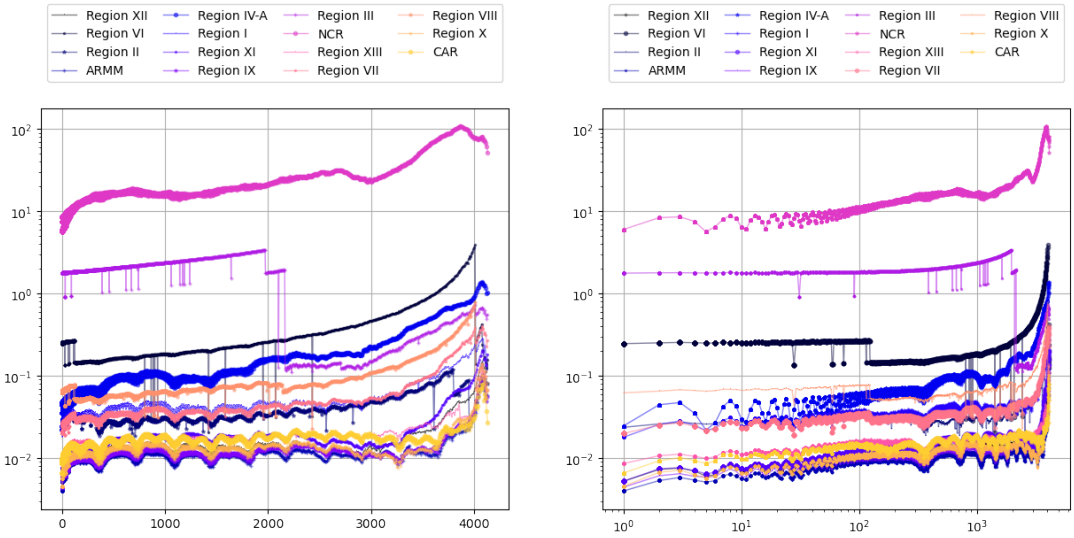

```
 93%|███████████████████████████████████████▉    | 14/15 [00:22<00:01,  1.
65s/it]
0it [00:00, ?it/s]
783it [00:00, 7828.59it/s]
1583it [00:00, 7928.67it/s]
2391it [00:00, 7994.66it/s]
4142it [00:00, 8333.69it/s]
```



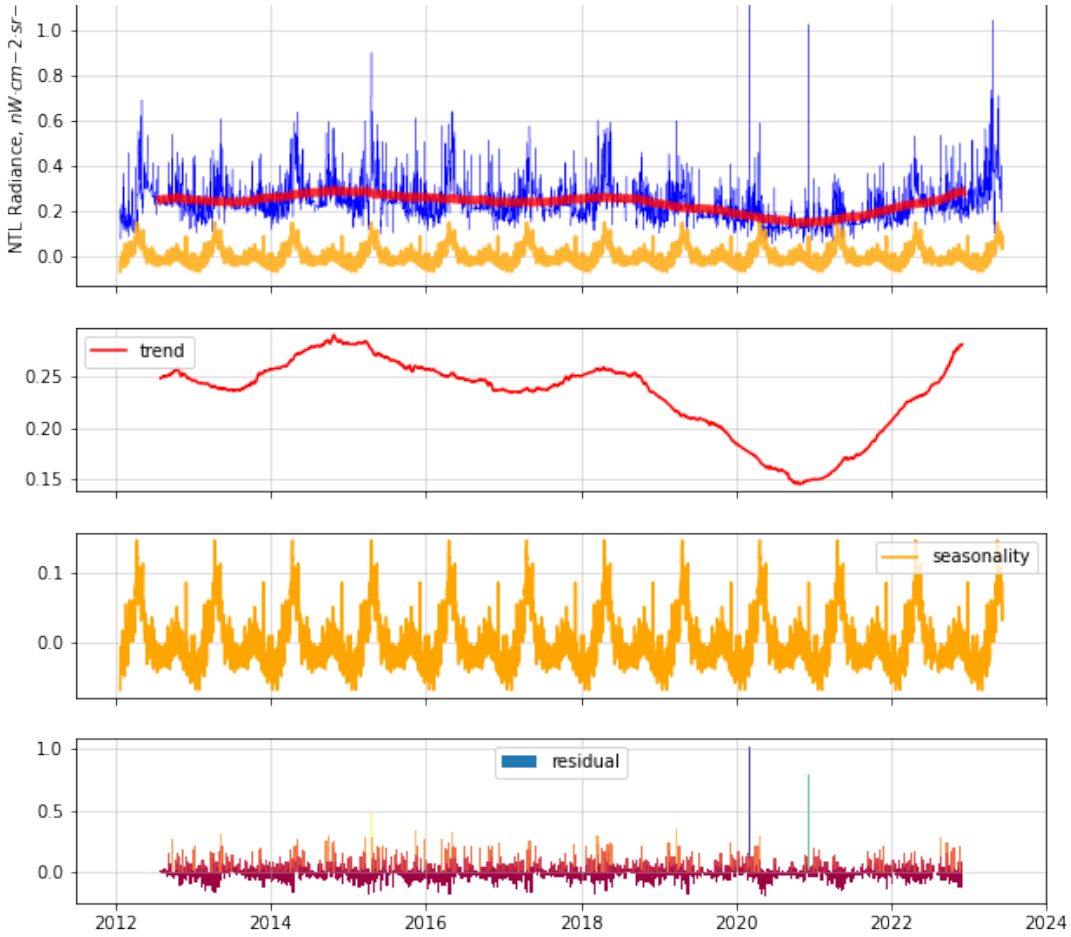

```
100%|███████████████████████████████████████| 15/15 [00:24<00:00,  1.
63s/it]
```

In [69]:
```
import itertools
marker = itertools.cycle((',', '+', '.', 'o', '*'))
```

In [70]:
```
# show_markers()
plt.figure(dpi = 100, figsize = (15,6))
for i in range(len(regions)):
    plt.subplot(121)
    plt.plot(delta_vals, MSD_reg_NTL[i], '-', label = str(regions[i]), lw
             color = plt.cm.gnuplot2(i/17), ms = 3, alpha = 0.5, marker =
    plt.legend(fontsize=10, ncol = 4, bbox_to_anchor= (0, 1.05, 1, 0.5),
               bbox_transform=plt.gca().transAxes, fancybox = True, mode
    plt.grid()
#     plt.xscale("log")
    plt.yscale("log")
    plt.subplot(122)
    plt.plot(delta_vals, MSD_reg_NTL[i], '-', label = str(regions[i]), lw
             color = plt.cm.gnuplot2(i/17), ms = 3, alpha = 0.5,  marker
    plt.legend(fontsize=10, ncol = 4, bbox_to_anchor= (0, 1.05, 1, 0.5),
               bbox_transform=plt.gca().transAxes, fancybox = True, mode
    plt.grid()
    plt.xscale("log")
    plt.yscale("log")
plt.show()
```

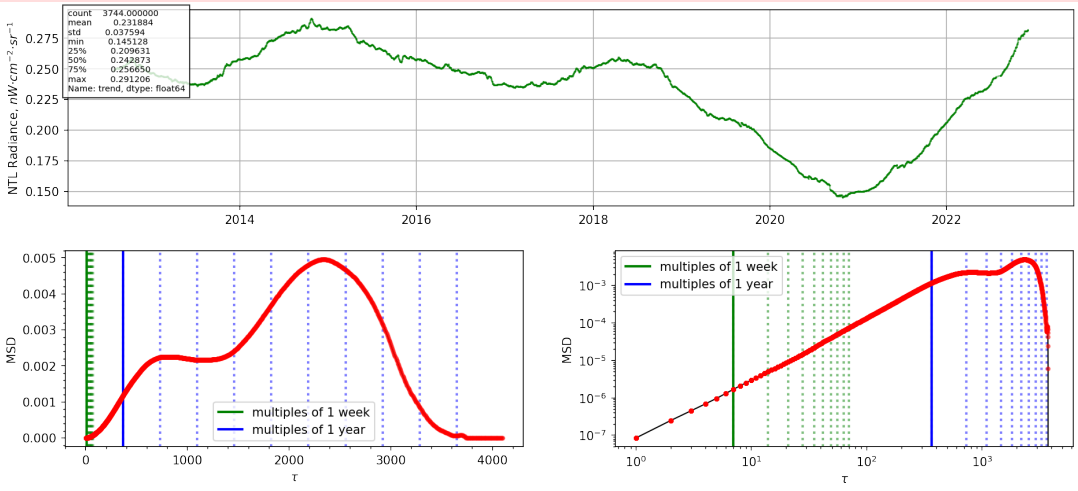


In [72]:
```
date = data.dropna().index
y = data.dropna()
decomposition = sm.tsa.seasonal_decompose(y, model='additive', period=365

fig, ax = plt.subplots(nrows=4,ncols=1, sharex='col',
                       gridspec_kw={'height_ratios':[2, 1, 1 ,1]},
                       figsize=(10,10))

ax[0].grid(alpha = 0.5)
# ax[0].plot(date, y-0.3, 'k', lw = 0.5, drawstyle = 'steps-mid')
ax[0].plot(date, y, 'b-', lw = 0.3, drawstyle = 'steps-mid', label = 'Dai
ax[0].plot(date, decomposition.seasonal, color = 'orange', alpha = 0.8, l
ax[0].plot(date, decomposition.trend, 'r', alpha = 0.8, lw = 5)
# ax[0].bar(date, decomposition.resid, width = 10,
#         color = plt.cm.Spectral(decomposition.resid), alpha = 0.5)
ax[0].set_ylabel('NTL Radiance, $nW ·cm-2·sr-1$')
ax[0].legend()

ax[1].grid(alpha = 0.5)
ax[1].plot(date, decomposition.trend, 'r', label = 'trend')
ax[1].legend()

ax[2].grid(alpha = 0.5)
ax[2].plot(date, decomposition.seasonal, color = 'orange',  label = 'seas
ax[2].legend()

ax[3].grid(alpha = 0.5)
residual = decomposition.resid
ax[3].bar(date, residual, width = 10, label = 'residual')
ax[3].bar(date, residual, width = 10,
        color = plt.cm.Spectral(residual))
plt.legend()
plt.show()
```



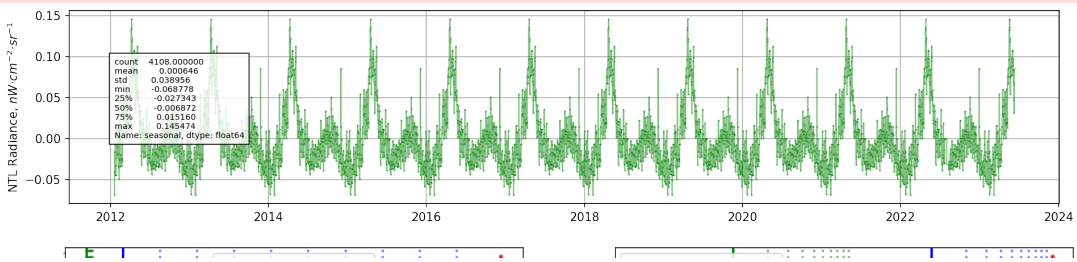
In [73]:
```
MSD_plotted(decomposition.trend)
```

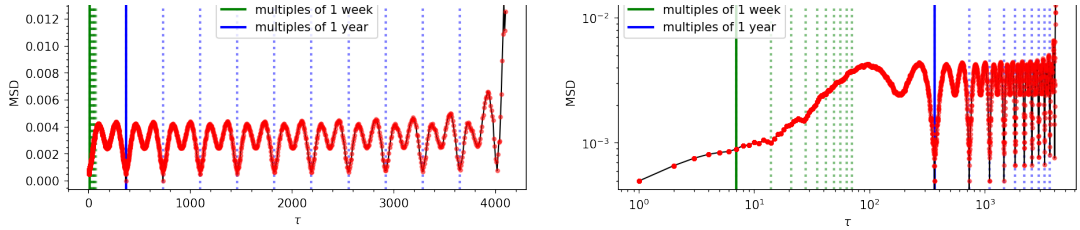
```
4108it [00:00, 10016.82it/s]
```



In [74]:
```
MSD_plotted(decomposition.seasonal)
```

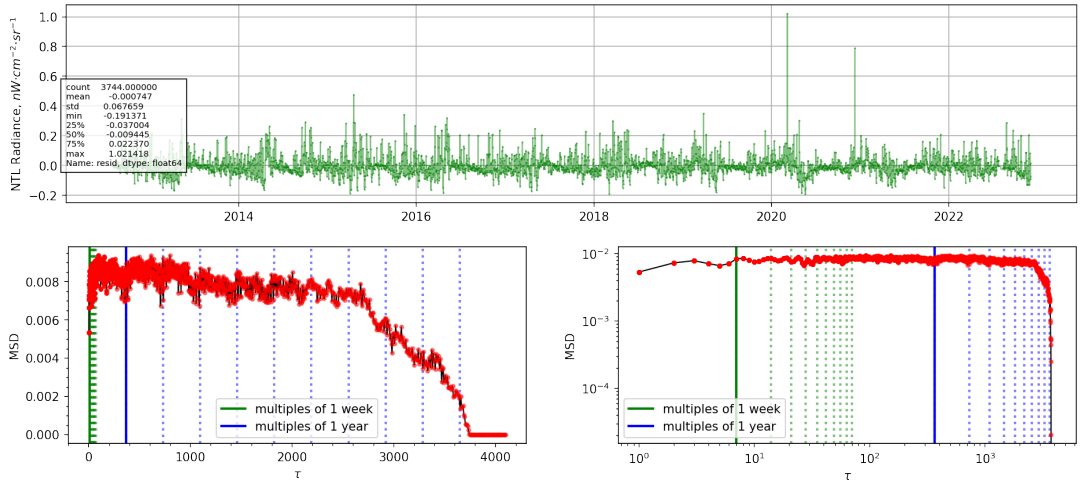
```
4108it [00:00, 10037.28it/s]
```





In [75]:
```
MSD_plotted(decomposition.resid)
```

```
4108it [00:00, 9653.23it/s]
```



STL Disaggregation

In [76]:
```
date_range = pd.date_range(start='2012-01-19', end='2023-06-12', freq='D'
```

In [77]:
```
NTL_dataframe = pd.DataFrame(index=date_range, columns = regions).fillna(
NTL_dataframe.head()
```

Out[77]:

	Region XII	Region VI	Region II	ARMM	Region IV-A	Region I	Region XI	Region IX	Region III	NCR	R
2012-01-19	0	0	0	0	0	0	0	0	0	0	
2012-01-20	0	0	0	0	0	0	0	0	0	0	
2012-01-21	0	0	0	0	0	0	0	0	0	0	
2012-01-22	0	0	0	0	0	0	0	0	0	0	
2012-01-23	0	0	0	0	0	0	0	0	0	0	

In [78]:
```
for i, NTL_data in tqdm(enumerate(regional_NTL)):
    region_name = str(NTL_data[:-16])
    dataframe = pd.read_csv(data_path+NTL_data, parse_dates=True, index_c
    data = dataframe['Gap_Filled_DNB_BRDF_Corrected_NTL']
    NTL_dataframe[region_name] = data

NTL_dataframe = NTL_dataframe.dropna()
NTL_dataframe.head()
```

```
15it [00:00, 85.63it/s]
```

Out[78]:

	Region XII	Region VI	Region II	ARMM	Region IV-A	Region I	Region XI	Region IX
2012-01-19	0.146781	0.210236	0.127336	0.074327	0.770527	0.365072	0.188997	0.129037
2012-01-20	0.140117	0.215037	0.160783	0.115419	1.108608	0.459410	0.237956	0.129962
2012-01-21	0.155769	0.268322	0.087568	0.135370	0.739408	0.222583	0.208777	0.127228
2012-01-22	0.138169	0.223430	0.068581	0.094497	0.601730	0.175084	0.199801	0.122536
2012-01-23	0.138214	0.199196	0.079029	0.094515	0.608278	0.236953	0.199162	0.124118

In [79]:
```
trend = NTL_dataframe.copy()
seasonality = NTL_dataframe.copy()
residual = NTL_dataframe.copy()

for loc in tqdm(NTL_dataframe.columns, desc = 'STL Decomposition'):
    data = NTL_dataframe[loc].values
    decomposition = sm.tsa.seasonal_decompose(data, model='additive', per
    trend[loc] = decomposition.trend
    seasonality[loc] = decomposition.seasonal
    residual[loc] = decomposition.resid

NTL_DF = pd.concat([NTL_dataframe,trend, seasonality, residual], axis = 1
NTL_DF
```

```
STL Decomposition: 100%|██████████| 15/15 [00:00<00:00, 148.
85it/s]
```

Out[79]:

	Region XII	Region VI	Region II	ARMM	Region IV-A	Region I	Region XI	Regio IX
2012-01-19	0.146781	0.210236	0.127336	0.074327	0.770527	0.365072	0.188997	0.12903
2012-01-20	0.140117	0.215037	0.160783	0.115419	1.108608	0.459410	0.237956	0.12996

2012-01-21	0.155769	0.268322	0.087568	0.135370	0.739408	0.222583	0.208777	0.12722
2012-01-22	0.138169	0.223430	0.068581	0.094497	0.601730	0.175084	0.199801	0.12253
2012-01-23	0.138214	0.199196	0.079029	0.094515	0.608278	0.236953	0.199162	0.12411
...	...	...	...	...	...	...	...	.
2023-06-07	0.501796	0.643477	0.417970	0.338322	1.762237	0.491732	0.624199	0.33755
2023-06-08	0.478718	0.605783	0.463181	0.346712	1.613582	0.686410	0.626030	0.37560
2023-06-09	0.499206	0.608347	0.423989	0.333517	1.912941	0.732638	0.624136	0.36762
2023-06-10	0.498856	0.614001	0.393224	0.330853	1.949542	0.781676	0.607354	0.36600
2023-06-11	0.640943	0.614652	0.391984	0.396416	1.945457	0.738287	0.743967	0.37179

4104 rows × 60 columns

In [80]:
```
def MSD_calculator(data):
    n_tau = len(data)
    dt = 1
    delta_vals = np.linspace(1,n_tau, n_tau).astype(int)
# #     delta_vals = np.round(tau_vals/dt).astype(int)
#     print(delta_vals)

    n = len(data)
    n_tau = len(delta_vals)
    dx_tau = np.empty((n, n_tau))*np.nan
    msd_tau = np.empty(n_tau)*np.nan

    for i, tau in enumerate(delta_vals):
        delta = delta_vals[i]
        data_trunc = data[:-1*delta]
        data_shift = data[delta:]
        dx = data_shift.values - data_trunc.values
        dx_tau[:len(dx), i] = dx.T
        dx2_sum = np.nansum(dx**2)
        denom = n-delta_vals[i]
        msd = dx2_sum/denom
        msd_tau[i] = msd

    MSD = pd.DataFrame(msd_tau, index = delta_vals, columns = ['MSD'])
#     MSD = np.array([delta_vals, msd_tau])

    return MSD
```

In [81]:
```
MSD_calculator(NTL_DF['Trend']['CAR'])
```

Out[81]:

	MSD
1	8.314524e-08

1	8.314524e-08
2	2.421837e-07
3	4.447013e-07
4	6.817673e-07
5	9.632611e-07
...	...
4100	0.000000e+00
4101	0.000000e+00
4102	0.000000e+00
4103	0.000000e+00
4104	NaN

4104 rows × 1 columns

In [82]:
```
MSD_trend_df = pd.DataFrame(index = range(1, len(date_range)), columns =

for i, loc in tqdm(enumerate(MSD_trend_df.columns), desc = 'Calculating M
    MSD_tau = MSD_calculator(NTL_DF['Trend'][loc])
    MSD_trend_df[loc] = MSD_tau.reset_index(drop=True)

MSD_trend_df.head()
```

```
Calculating MSD: 15it [00:07,  1.90it/s]
```

Out[82]:

	Region XII	Region VI	Region II	ARMM	Region IV-A	Region I	Region XI	R
1	1.992672e-07	0.000002	3.786992e-07	1.572980e-07	0.000001	6.251899e-07	2.550835e-07	1.7
2	3.689058e-07	0.000003	6.532642e-07	2.957874e-07	0.000003	1.142573e-06	4.933433e-07	3.3
3	5.733104e-07	0.000005	9.641656e-07	4.621703e-07	0.000004	1.731744e-06	7.915162e-07	5.2
4	8.231981e-07	0.000006	1.318004e-06	6.621248e-07	0.000006	2.429742e-06	1.159496e-06	7.5
5	1.126491e-06	0.000008	1.719132e-06	8.991818e-07	0.000009	3.283163e-06	1.604349e-06	1.0

In [85]:
```
MSD_seasonality_df = pd.DataFrame(index = range(1, len(date_range)), colu

for i, loc in tqdm(enumerate(MSD_seasonality_df.columns), desc = 'Calcula
    MSD_tau = MSD_calculator(NTL_DF['Seasonality'][loc])
    MSD_seasonality_df[loc] = MSD_tau.reset_index(drop=True)

MSD_seasonality_df
```

```
Calculating MSD: 15it [00:07,  2.11it/s]
```

Out[85]:

	Region XII	Region VI	Region II	ARMM	Region IV-A	Region I	Region XI	Region IX
1	0.000518	0.000888	0.002000	0.000385	0.003564	0.001494	0.000542	0.000541
2	0.000513	0.000949	0.002097	0.000428	0.003813	0.001625	0.000528	0.000527
3	0.000538	0.000808	0.002095	0.000439	0.002964	0.001534	0.000539	0.000525
4	0.000472	0.000718	0.001960	0.000415	0.002264	0.001468	0.000532	0.000497
5	0.000490	0.000830	0.002008	0.000376	0.002570	0.001525	0.000544	0.000483
...	...	...	...	...	...	...	...	...
4158	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4159	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4160	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4161	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4162	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

4162 rows × 15 columns

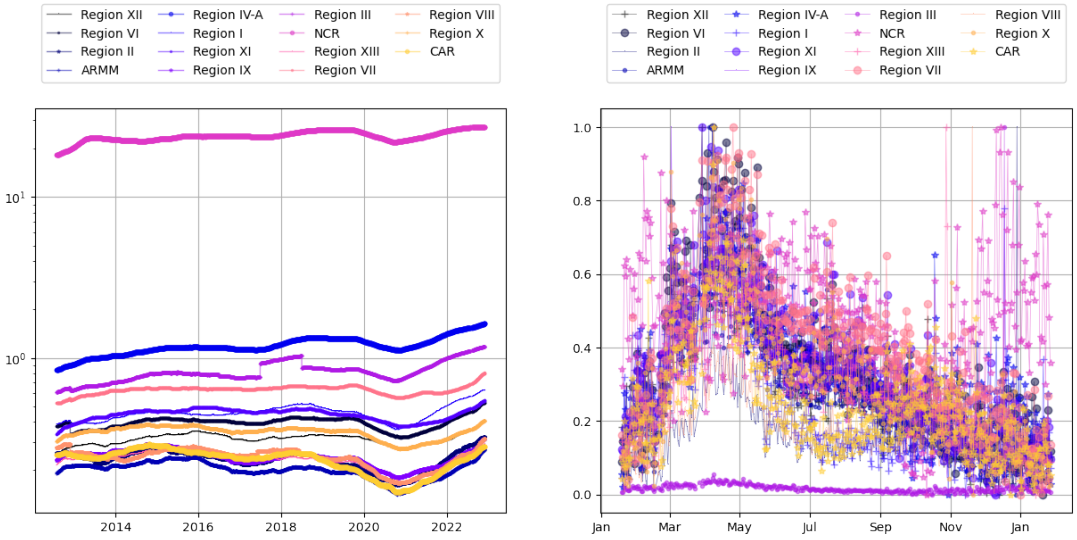
Trend and Seasonality

In [93]:

```
plt.figure(dpi = 100, figsize = (15,6))

for i, loc in tqdm(enumerate(MSD_trend_df.columns), desc = 'Plotting MSD'
    plt.subplot(121)
    plt.plot(NTL_DF['Trend'][loc], '-', label = loc, lw = 1,
             color = plt.cm.gnuplot2(i/17), ms = 3, alpha = 0.5, marker =
    plt.legend(fontsize=10, ncol = 4, bbox_to_anchor= (0, 1.05, 1, 0.5),
               bbox_transform=plt.gca().transAxes, fancybox = True, mode
    plt.yscale('log')
    plt.grid()

    plt.subplot(122)

    df = NTL_DF['Seasonality'][loc]
    normalized_df=(df-df.min())/(df.max()-df.min())

    plt.plot(normalized_df[:365], '-', label = loc, lw = 0.5,
             color = plt.cm.gnuplot2(i/17), alpha = 0.5, marker = next(ma
    plt.legend(fontsize=10, ncol = 4, bbox_to_anchor= (0, 1.05, 1, 0.5),
               bbox_transform=plt.gca().transAxes, fancybox = True, mode
```

```
    plt.gcf().autofmt_xdate
    dtFmt = mdates.DateFormatter("%b") # define the formatting
    plt.gca().xaxis.set_major_formatter(dtFmt) # apply the format to the
    plt.grid()
```

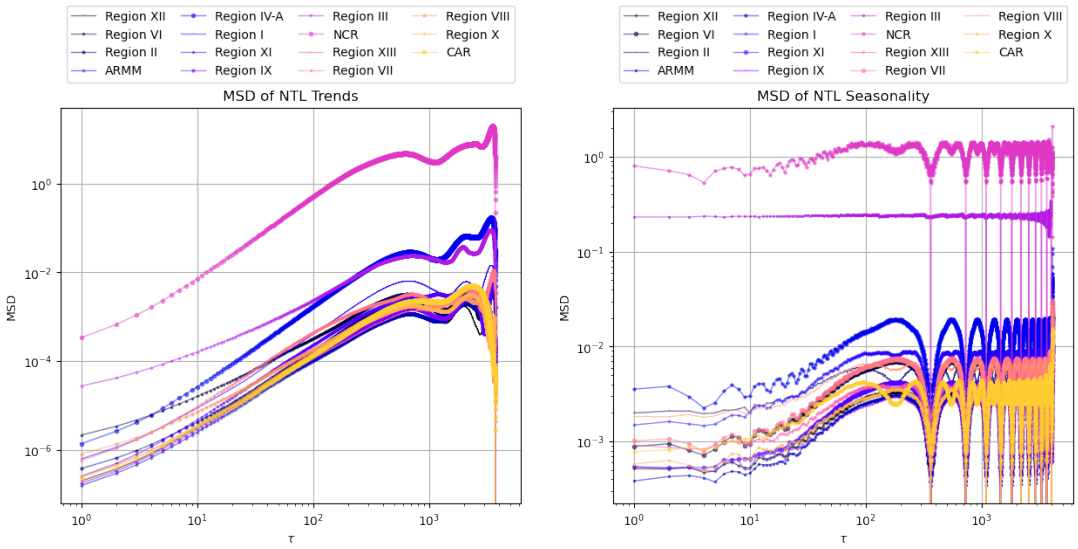
```
Plotting MSD: 15it [00:00, 29.94it/s]
```



MSD of Trend and Seasonality

In [95]:
```
plt.figure(dpi = 100, figsize = (15,6))

for i, loc in tqdm(enumerate(MSD_trend_df.columns), desc = 'Plotting MSD'
    plt.subplot(121)
    plt.plot(MSD_trend_df[loc], '-', label = loc, lw = 1,
             color = plt.cm.gnuplot2(i/17), ms = 3, alpha = 0.5, marker =
    plt.legend(fontsize=10, ncol = 4, bbox_to_anchor= (0, 1.05, 1, 0.5),
               bbox_transform=plt.gca().transAxes, fancybox = True, mode
    plt.xscale('log')
    plt.yscale('log')
    plt.title('MSD of NTL Trends')
    plt.xlabel(r"$\tau$")
    plt.ylabel("MSD")
    plt.grid()

    plt.subplot(122)
    plt.plot(MSD_seasonality_df[loc], '-', label = loc, lw = 1,
             color = plt.cm.gnuplot2(i/17), ms = 3, alpha = 0.5, marker =
    plt.legend(fontsize=10, ncol = 4, bbox_to_anchor= (0, 1.05, 1, 0.5),
               bbox_transform=plt.gca().transAxes, fancybox = True, mode
    plt.xscale('log')
    plt.yscale('log')
    plt.title('MSD of NTL Seasonality')
    plt.xlabel(r"$\tau$")
    plt.ylabel("MSD")
    plt.grid()
```

```
Plotting MSD: 15it [00:00, 66.61it/s]
```


Region XII
Region VI
Region II
ARMM
Region IV-A
Region I
Region XI
Region IX
Region III
NCR
Region XIII
Region VII
Region VIII
Region X
CAR
MSD of NTL Trends
MSD of NTL Seasonality
MSD
τ

In [ ]: