SIOC 221A: HW3

Author: Victoria Boatwright

I acknowledge my collaborators Turner Johnson and Grant Meiners.

In [9]:
```
import numpy as np
import matplotlib.pyplot as plt
import netCDF4
import datetime as dt
import matplotlib.dates as mdates
import pandas as pd
import xarray as xr
import scipy.stats as stats
import scipy
import cmocean
```

In [10]:
```
# could also make function for downloading data

def pd2np(data,var_name,start_num=1):
    array = np.array(data[var_name][start_num:][:])
    array = np.array([float(vv) for vv in array])
    return array

# variables of interest: wave height, wind speed, & air temp from Nation

wvht = np.array([])
wspd = np.array([])
atmp = np.array([])
wtmp = np.array([])
dates = np.array([])

path = '/home/vboat/Documents/SIO/fall23/sioc221a/week4/'
file = ['buoy46047_year2015.txt', 'buoy46047_year2016.txt']
for f in file:
    fn = path+f
    data = pd.read_csv(fn,sep=r'\s+',header=0, na_values=['999.0','99.0'
    variables = data.columns
    # start after column 1 because that includes the units
    # units = data.row[1]
    yy = data['#YY'][:]; MM = data['MM'][:]; dd = data['DD'][:]; hh = da
    datetimes = [dt.datetime(year=int(yy[ii]),month=int(MM[ii]),day=int(


    # now load data
    vars_of_interest = ['WDIR','WSPD','GST','WVHT','PRES','ATMP','WTMP']
    wdir = np.array(data['WDIR'][1:][:]); wdir = np.array([float(vv) for
    # made into a function after that
    windspeed = pd2np(data,'WSPD',1)
    gust = pd2np(data,'GST',1)
    waveheight = pd2np(data,'WVHT',1)
    pressure = pd2np(data,'PRES',1)
    airtemp = pd2np(data,'ATMP',1)
    watertemp = pd2np(data,'WTMP',1)

    wvht = np.append(wvht,waveheight)
    wspd = np.append(wspd,windspeed)
    wtmp = np.append(wtmp,watertemp)
    atmp = np.append(atmp,airtemp)
    dates = np.append(dates,datetimes)

df = pd.DataFrame({'dates':dates,'wvht':wvht,'wspd':wspd,'wtmp':wtmp,'atm
```

In [11]:
```
data
```

Out[11]:

	#YY	MM	DD	hh	mm	WDIR	WSPD	GST	WVHT	DPD	APD	MWD	PRES	ATM
0	#yr	mo	dy	hr	mn	degT	m/s	m/s	m	sec	sec	degT	hPa	deg
1	2015	12	31	23	50	335	3.7	4.8	1.47	12.90	7.27	297	1018.4	14.
2	2016	01	01	00	50	345	3.8	4.8	1.58	12.90	7.34	306	1018.6	14.
3	2016	01	01	01	50	346	4.1	5.1	1.43	12.90	7.28	292	1018.8	14.
4	2016	01	01	02	50	5	3.0	3.7	1.34	12.12	7.26	293	1018.8	14.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
8700	2016	12	31	18	50	284	5.5	6.8	1.99	13.79	7.38	312	1015.9	14.
8701	2016	12	31	19	50	303	6.7	8.0	1.82	13.79	7.32	321	1015.1	14.
8702	2016	12	31	20	50	273	4.8	6.8	1.85	14.81	7.13	316	1014.1	13.
8703	2016	12	31	21	50	297	7.8	11.1	1.95	13.79	6.96	328	1013.5	13.
8704	2016	12	31	22	50	303	7.7	9.0	1.81	13.79	6.27	310	1013.4	12.

8705 rows × 18 columns

The missing data is stored in some weird floats of 99, and 999 - for temperature, it's '999.0'; for visibility, it's '99.0', and for the tide, it's '99.00'. Since the data is stored in a text file, there is no metadata (that I found easily) about the fill_value for missing data, so it would be a bit difficult to identify in the dataset other than by eyeballing.

I did find 1 day that was almost all fill_values by eyeballing through the dataset, revealing that the missing data is filled by some form of 99 or 999 (row is given below):

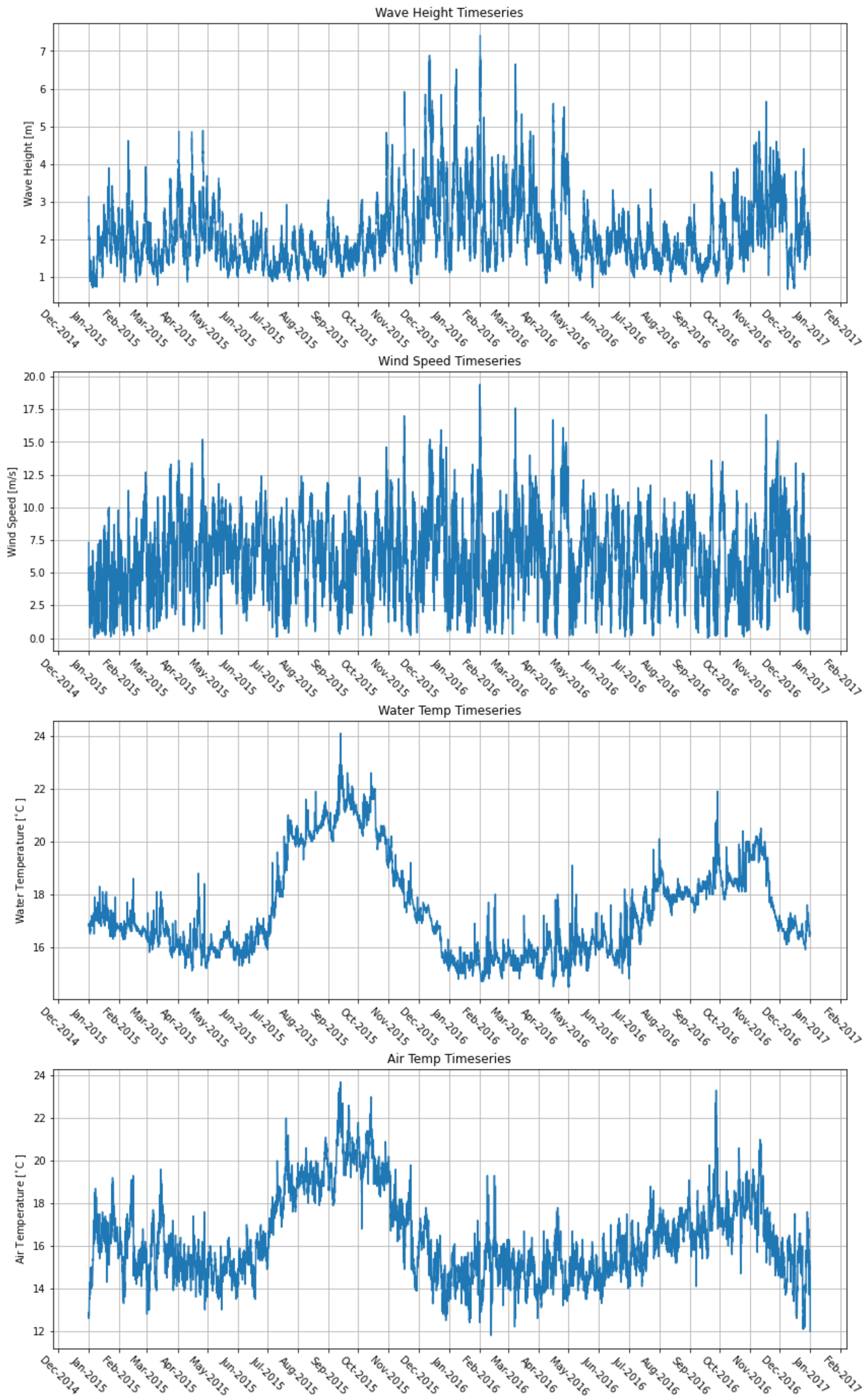
2015 01 20 20 50 314 8.4 99.0 99.00 99.00 99.00 999 9999.0 999.0 999.0 999.0 99.0 99.00

Q1: Visual evaluation.

Plot the time series of wind speed, wave height, water temperature, and air temperature from 46047.

In [12]:
```
fig,axes = plt.subplots(4,1,figsize=(14,24))
voi = [wvht,wspd,wtmp,atmp] # variables of interest
ylabels = ['Wave Height [m]','Wind Speed [m/s]',r'Water Temperature [$^{
fig.suptitle('Timeseries for Buoy 46047 Data over 2015-2016',y=0.95)
titles = ['Wave Height Timeseries','Wind Speed Timeseries','Water Temp T
for vv,ax in enumerate(axes):
    ax.grid()
    ax.plot(dates,voi[vv])
    ax.set_ylabel(ylabels[vv])
    ax.set_title(titles[vv])
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%b-%Y'))
    ax.xaxis.set_major_locator(mdates.MonthLocator())
    ax.tick_params(axis='x', labelrotation = -45)

plt.subplots_adjust(hspace=0.25)
plt.show()
```

Timeseries for Buoy 46047 Data over 2015-2016



Q2: Monthly means.

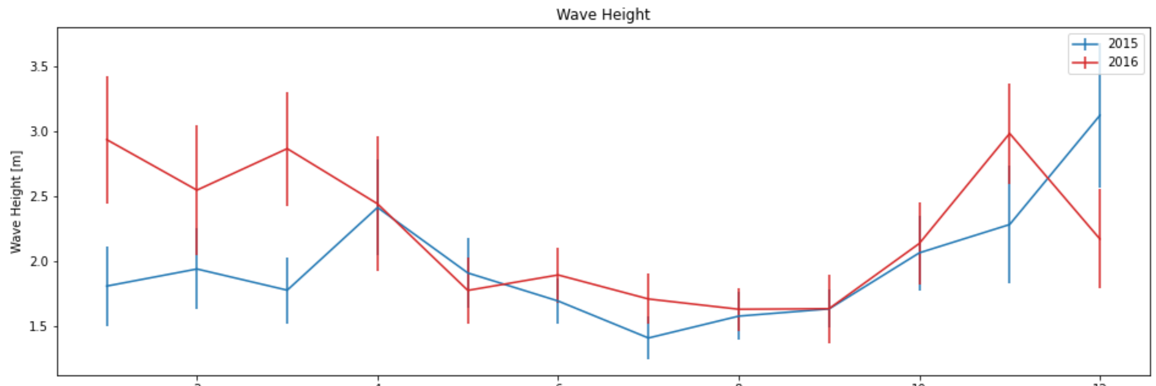
Average the data to produce monthly means for 2015 and 2016. Plot the means for each month and standard error of the mean. Data are provided at varying frequencies, but consecutive data are not independent. For the purposes of this problem set, let’s assume that the data provide one independent sample every 7 days.

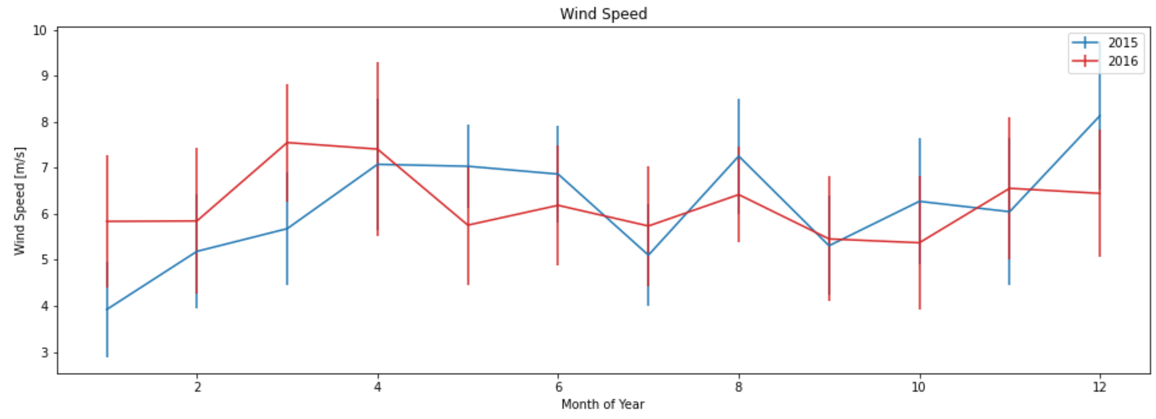
In [13]:
```
# number of observations per month
30.5/7
```

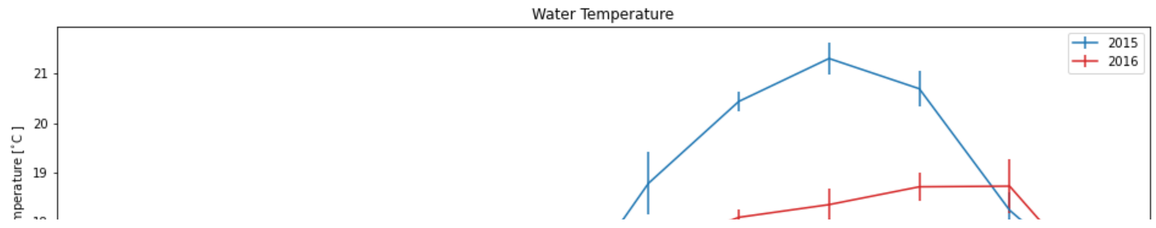
Out[13]:
```
4.357142857142857
```

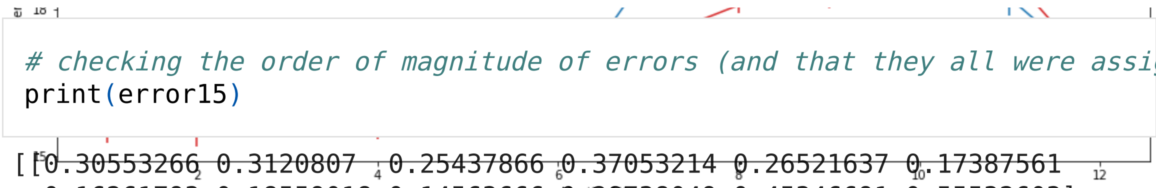
In [14]:
```
months = np.arange(1,13)

# organize arrays into [wvht, wspd, wtmp, atmp]
means15 = np.zeros((4,12)); means15[:,:] = np.nan
means16 = np.zeros((4,12)); means16[:,:] = np.nan

# standard error of the mean
err15 = np.zeros((4,12)); err15[:,:] = np.nan
err16 = np.zeros((4,12)); err16[:,:] = np.nan

# was getting some nan values from the scipy calculated sem (standard er
error15 = np.zeros((4,12)); error15[:,:] = np.nan
error16 = np.zeros((4,12)); error16[:,:] = np.nan

# assuming only 1 independent sample every 7 days (per our month data) -
n_samples = 30.5/7

for mm in months:
    # 2015 stats by month:
    # calculate number of obs per month:
    N15 = len(df.loc[(df['dates'].dt.month==mm) & (df['dates'].dt.year==

    # wave height
    means15[0,mm-1] = np.nanmean(df.loc[(df['dates'].dt.month==mm) & (df
    err15[0,mm-1] = scipy.stats.sem(df.loc[(df['dates'].dt.month==mm) &
    # this would be me calculating my own error bars without scipy

    sem = np.std(df.loc[(df['dates'].dt.month==mm) & (df['dates'].dt.yea
    error15[0,mm-1] = sem

    # wave speed
    means15[1,mm-1] = np.nanmean(df.loc[(df['dates'].dt.month==mm) & (df
    err15[1,mm-1] = scipy.stats.sem(df.loc[(df['dates'].dt.month==mm) &

    sem = np.std(df.loc[(df['dates'].dt.month==mm) & (df['dates'].dt.yea
    error15[1,mm-1] = sem

    # water temp
    means15[2,mm-1] = np.nanmean(df.loc[(df['dates'].dt.month==mm) & (df
    err15[2,mm-1] = scipy.stats.sem(df.loc[(df['dates'].dt.month==mm) &
    sem = np.std(df.loc[(df['dates'].dt.month==mm) & (df['dates'].dt.yea
    error15[2,mm-1] = sem

    # air temp
    means15[3,mm-1] = np.nanmean(df.loc[(df['dates'].dt.month==mm) & (df
    err15[3,mm-1] = scipy.stats.sem(df.loc[(df['dates'].dt.month==mm) &
    sem = np.std(df.loc[(df['dates'].dt.month==mm) & (df['dates'].dt.yea
    error15[3,mm-1] = sem

    # 2016 stats by month:
    N16 = len(df.loc[(df['dates'].dt.month==mm) & (df['dates'].dt.year==

    # wave height
    means16[0,mm-1] = np.nanmean(df.loc[(df['dates'].dt.month==mm) & (df
    err16[0,mm-1] = scipy.stats.sem(df.loc[(df['dates'].dt.month==mm) &
    sem = np.std(df.loc[(df['dates'].dt.month==mm) & (df['dates'].dt.yea
    error16[0,mm-1] = sem
    # wave speed
    means16[1,mm-1] = np.nanmean(df.loc[(df['dates'].dt.month==mm) & (df
    err16[1,mm-1] = scipy.stats.sem(df.loc[(df['dates'].dt.month==mm) &
    sem = np.std(df.loc[(df['dates'].dt.month==mm) & (df['dates'].dt.yea
    error16[1,mm-1] = sem
    # water temp
    means16[2,mm-1] = np.nanmean(df.loc[(df['dates'].dt.month==mm) & (df
    err16[2,mm-1] = scipy.stats.sem(df.loc[(df['dates'].dt.month==mm) &
    sem = np.std(df.loc[(df['dates'].dt.month==mm) & (df['dates'].dt.yea
    error16[2,mm-1] = sem
    # air temp
    means16[3,mm-1] = np.nanmean(df.loc[(df['dates'].dt.month==mm) & (df
    err16[3,mm-1] = scipy.stats.sem(df.loc[(df['dates'].dt.month==mm) &
    sem = np.std(df.loc[(df['dates'].dt.month==mm) & (df['dates'].dt.yea
    error16[3,mm-1] = sem


ylabels = ['Wave Height [m]','Wind Speed [m/s]',r'Water Temperature [$^{\
titles = ['Wave Height','Wind Speed','Water Temperature','Air Temperatur

fig,axes = plt.subplots(4,1,figsize=(16,24))
fig.suptitle('Monthly Statistics: Means and Error Bars',fontweight='bold
for vv,ax in enumerate(axes):
    #ax.plot(months,means15[vv,:],color='tab:blue',label='2015')
    #ax.plot(months,means16[vv,:],color='tab:red',label='2016')
    ax.errorbar(months,means15[vv,:],yerr=error15[vv,:],color='tab:blue'
    ax.errorbar(months,means16[vv,:],yerr=error16[vv,:],color='tab:red',
    ax.set_xlabel('Month of Year')
    ax.set_ylabel(ylabels[vv])
    ax.set_title(titles[vv])
    ax.legend(loc='upper right')
```

Monthly Statistics: Means and Error Bars





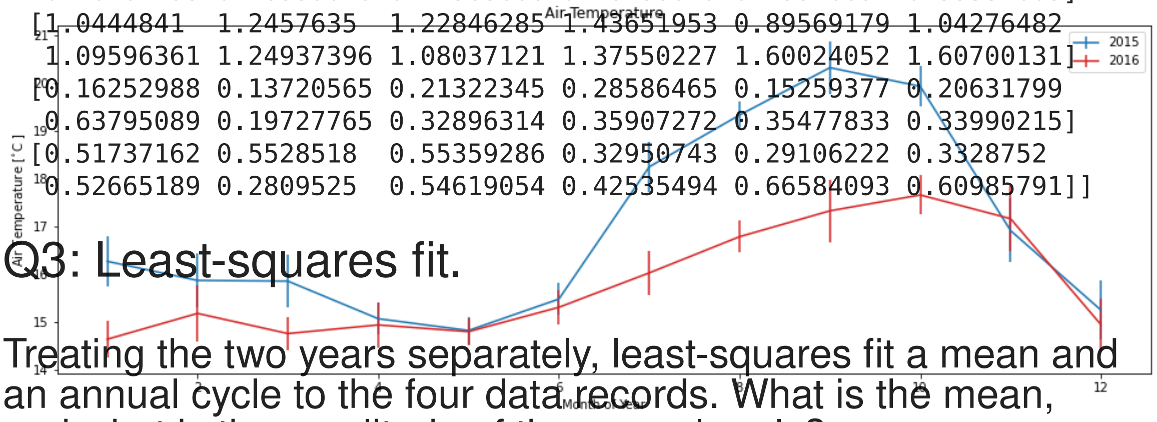




In [15]:

```
# checking the order of magnitude of errors (and that they all were assi
print(error15)
```

```
[[0.30553266 0.3120807  0.25437866 0.37053214 0.26521637 0.17387561
  0.16261793 0.18559018 0.14563666 0.28738049 0.45346691 0.55532603]
 [1.0444841  1.2457635  1.22846285 1.43651953 0.89569179 1.04276482
  1.09596361 1.24937396 1.08037121 1.37550227 1.60024052 1.60700131]
 [0.16252988 0.13720565 0.21322345 0.28586465 0.15259377 0.20631799
  0.63795089 0.19727765 0.32896314 0.35907272 0.35477833 0.33990215]
 [0.51737162 0.5528518  0.55359286 0.32950743 0.29106222 0.3328752
  0.52665189 0.2809525  0.54619054 0.42535494 0.66584093 0.60985791]]
```



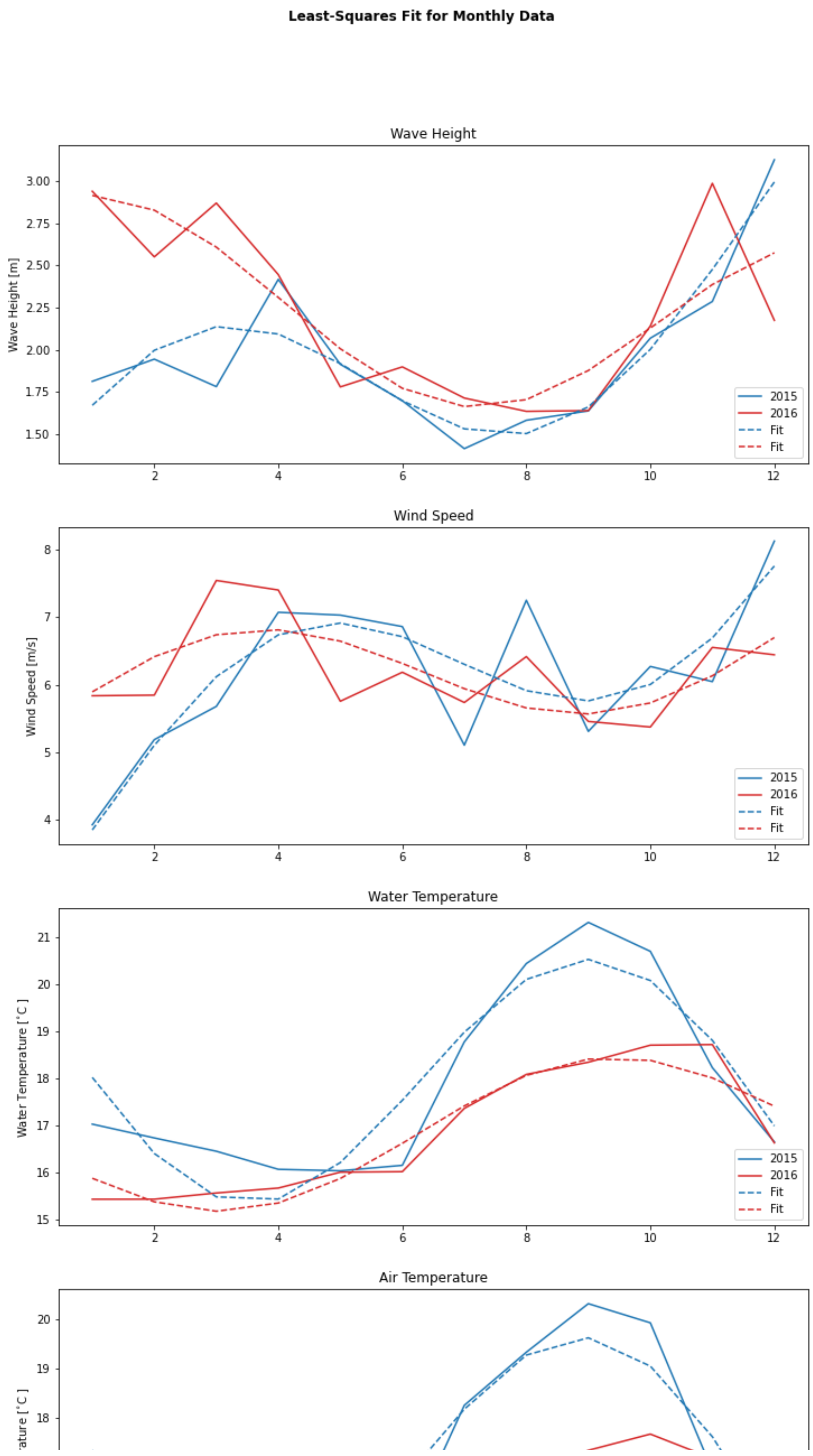
Q3: Least-squares fit.

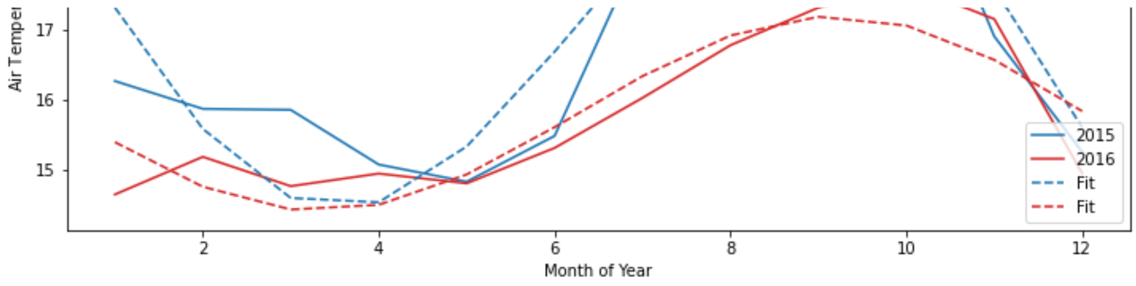
Treating the two years separately, least-squares fit a mean and an annual cycle to the four data records. What is the mean, and what is the amplitude of the annual cycle?

(Total amplitude should be determined from the square root of the sum of the squares of the sine and cosine amplitudes.) Are the fitted coefficients similar for the two years?

In [20]:
```
num_months = 12
time = np.arange(1,13) # months

variables = ['Wave Height','Wind Speed','Water Temp','Air Temp']
fit15 = np.zeros((4,12)); fit15[:,:] = np.nan
params15 = np.zeros((4,4)); params15[:,:] = np.nan
fit16 = np.zeros((4,12)); fit16[:,:] = np.nan
params16 = np.zeros((4,4)); params16[:,:] = np.nan

total_amp15 = np.zeros((4)); total_amp15[:] = np.nan
total_amp16 = np.zeros((4)); total_amp16[:] = np.nan

for vv,variable in enumerate(variables):
    # calculate least squares fit for 2015 data
    data = means15[vv,:]
    A2 = np.array([np.ones(num_months), time, np.sin(2*np.pi*time/12), np
    x = np.dot(np.linalg.inv(np.dot(A2.T, A2)), np.dot(A2.T, data))
    fit = np.dot(A2, x)

    total_amplitude = np.sqrt(x[2]**2 + x[3]**2)

    # save
    fit15[vv,:] = fit
    params15[vv,:] = x
    total_amp15[vv] = total_amplitude

    print(f'2015 Mean from {variable} fit: {x[0]} \nLinear trend from fi
    print('\n')
    # calculate least squares fit for 2016 data
    data = means16[vv,:]
    A2 = np.array([np.ones(num_months), time, np.sin(2*np.pi*time/12), np
    x = np.dot(np.linalg.inv(np.dot(A2.T, A2)), np.dot(A2.T, data))
    fit = np.dot(A2, x)

    total_amplitude = np.sqrt(x[2]**2 + x[3]**2)

    # save
    fit16[vv,:] = fit
    params16[vv,:] = x
    total_amp16[vv] = total_amplitude



    print(f'2016 Mean from {variable} fit: {x[0]} \nLinear trend from fi
    print('\n')






ylabels = ['Wave Height [m]','Wind Speed [m/s]',r'Water Temperature [$^{
titles = ['Wave Height','Wind Speed','Water Temperature','Air Temperatur

fig,axes = plt.subplots(4,1,figsize=(12,24))
fig.suptitle('Least-Squares Fit for Monthly Data',fontweight='bold',y=0.
for vv,ax in enumerate(axes):
    ax.plot(months,means15[vv,:],color='tab:blue',label='2015')
    ax.plot(months,means16[vv,:],color='tab:red',label='2016')
    ax.plot(months,fit15[vv,:],color='tab:blue',linestyle='--',label='Fi
    ax.plot(months,fit16[vv,:],color='tab:red',linestyle='--',label='Fit
    if vv==3:
        ax.set_xlabel('Month of Year')
    ax.set_ylabel(ylabels[vv])
    ax.set_title(titles[vv])
    ax.legend(loc='lower right')
```

```
2015 Mean from Wave Height fit: 1.0040512546562113
Linear trend from fit: 0.14916005514847575
Amplitude of sine:0.6856854841612314
Amplitude of cosine: 0.2017530378390564


2016 Mean from Wave Height fit: 2.3824174231373156
Linear trend from fit: -0.023215063603672537
Amplitude of sine:0.2966178542853637
Amplitude of cosine: 0.47181024412209904


2015 Mean from Wind Speed fit: 3.3418423748416473
Linear trend from fit: 0.43284989217152514
Amplitude of sine:1.4782876989151745
Amplitude of cosine: -0.7739270535379958


2016 Mean from Wind Speed fit: 5.445294675624503
Linear trend from fit: 0.11804224173854294
Amplitude of sine:0.9421133057950399
Amplitude of cosine: -0.16212285619072198


2015 Mean from Water Temp fit: 19.492138217164666
Linear trend from fit: -0.24773188218533448
Amplitude of sine:-3.2646913798820805
Amplitude of cosine: 0.4715597826856701


2016 Mean from Water Temp fit: 16.35255545132901
Linear trend from fit: 0.07371328243404207
Amplitude of sine:-1.39425525567205
Amplitude of cosine: 0.17289611391644755


2015 Mean from Air Temp fit: 19.02448650657732
Linear trend from fit: -0.31950469309979823
Amplitude of sine:-3.4727816759835406
Amplitude of cosine: 0.4140869859759775


2016 Mean from Air Temp fit: 15.984458655768492
Linear trend from fit: -0.028846322408774938
Amplitude of sine:-1.4699406004279894
Amplitude of cosine: 0.20143402209097871
```


Least-Squares Fit for Monthly Data
Wave Height
Wave Height [m]
Wind Speed
Wind Speed [m/s]
Water Temperature
Water Temperature [˚C]
Air Temperature
ature [˚C]
2015
2016
Fit
Fit



The fitted coefficients are more or less similar between years, depending on the dataset.

For wave height, the 2016 mean is double the 2015 mean. The sine wave amplitude is bigger in 2015, whereas the cosine amplitude is greater in 2016.

For wave speed, the 2016 mean is greater than the 2015 mean. The sine and cosine amplitudes are bigger in 2015.

For water temperature , the 2015 mean is slightly greater than the 2016 mean. The 2015 sine and cosine amplitudes are bigger than those in 2016.

For air temperature , the 2015 mean is slightly greater than the 2016 mean. The 2015 sine and cosine amplitudes are bigger than those in 2016, both by about double.

In [21]:
```
for vv,var in enumerate(variables):
    print('2015 v 2016')
    print(f'Total Amplitude: {var} = {total_amp15[vv]} v = {total_amp16[v
```

```
2015 v 2016
Total Amplitude: Wave Height = 0.7147509156809175 v = 0.5573033805203481
2015 v 2016
Total Amplitude: Wind Speed = 1.6686214684469112 v = 0.9559609309252632
2015 v 2016
Total Amplitude: Water Temp = 3.2985721811903583 v = 1.404934441237933
2015 v 2016
Total Amplitude: Air Temp = 3.4973819638411987 v = 1.4836782110829647
```

Q4: Least-squares fit a semi-annual cycle.

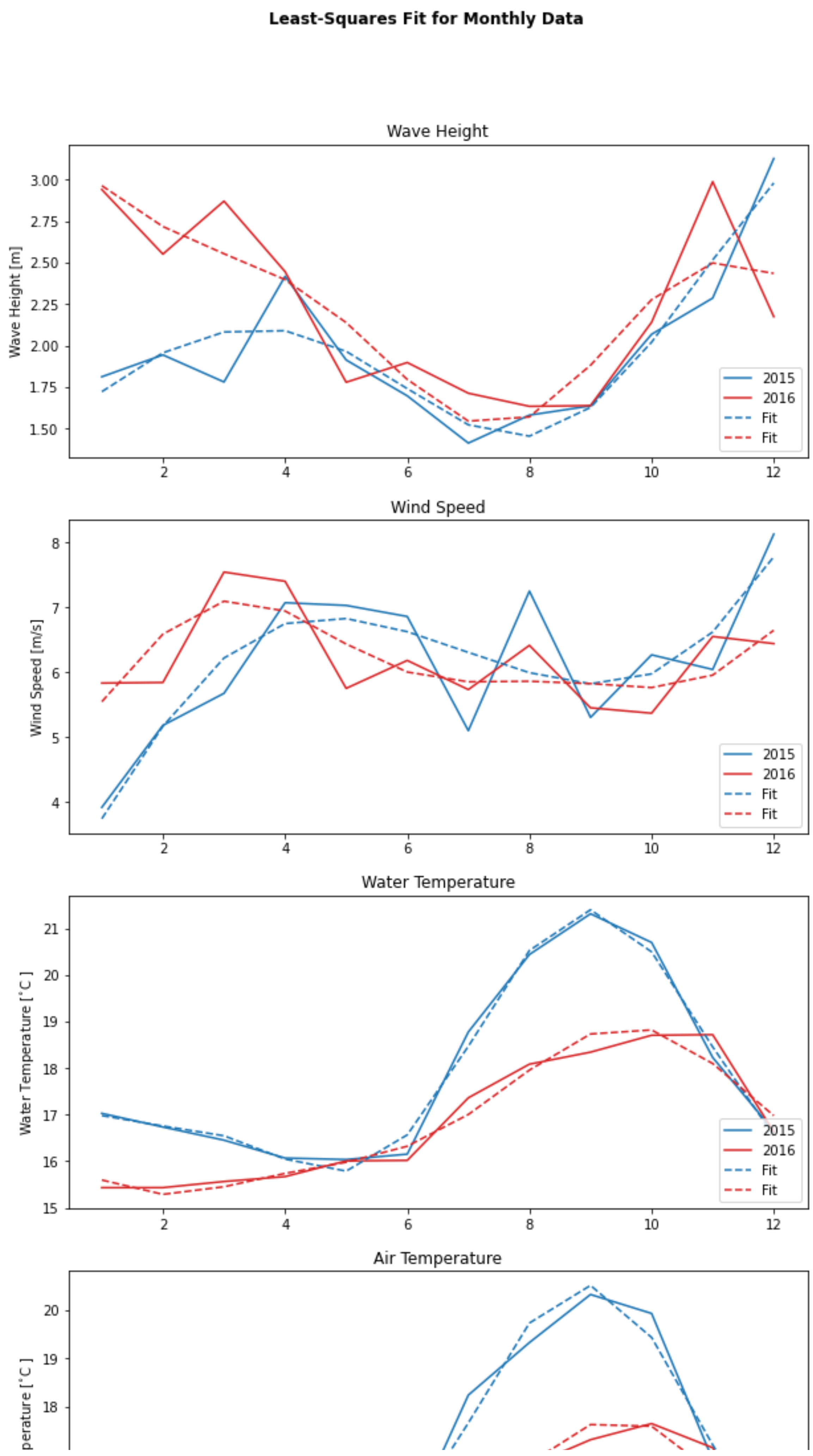
Augment your annual cycle least-squares fit with a semi-annual cycle.

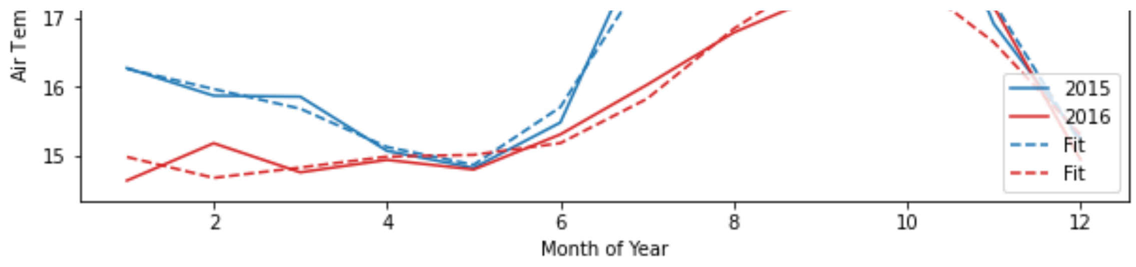
What is the amplitude of the semi-annual cycle? Does the augmented fit give you a different annual cycle?

In [22]:

```
# higher frequency curve fitting

num_months = 12
time = np.arange(1,13) # months

variables = ['Wave Height','Wind Speed','Water Temp','Air Temp']
ffit15 = np.zeros((4,12)); ffit15[:,:] = np.nan
param15 = np.zeros((4,6)); param15[:,:] = np.nan
ffit16 = np.zeros((4,12)); ffit16[:,:] = np.nan
param16 = np.zeros((4,6)); param16[:,:] = np.nan

tot_amp15 = np.zeros((4)); tot_amp15[:] = np.nan
tot_amp16 = np.zeros((4)); tot_amp16[:] = np.nan

for vv,variable in enumerate(variables):
    # calculate least squares fit for 2015 data
    data = means15[vv,:]
    A2 = np.array([np.ones(num_months), time, np.sin(2*np.pi*time/num_mo
    x = np.dot(np.linalg.inv(np.dot(A2.T, A2)), np.dot(A2.T, data))
    fit = np.dot(A2, x)

    tot_amp = np.sqrt(x[2]**2 + x[3]**2 + x[4]**2 + x[5]**2)

    # save
    ffit15[vv,:] = fit
    param15[vv,:] = x
    tot_amp15[vv] = tot_amp

    print(f'2015 Mean from {variable} fit: {x[0]} \nAmplitude of sine se
    print('\n')
    # calculate least squares fit for 2016 data
    data = means16[vv,:]
    A2 = np.array([np.ones(num_months), time, np.sin(2*np.pi*time/num_mo
    x = np.dot(np.linalg.inv(np.dot(A2.T, A2)), np.dot(A2.T, data))
    fit = np.dot(A2, x)

    tot_amp = np.sqrt(x[2]**2 + x[3]**2 + x[4]**2 + x[5]**2)

    # save
    ffit16[vv,:] = fit
    param16[vv,:] = x
    tot_amp16[vv] = tot_amp

    print(f'2016 Mean from {variable} fit: {x[0]} \nAmplitude of sine se
    print('\n')


ylabels = ['Wave Height [m]','Wind Speed [m/s]',r'Water Temperature [$^{
titles = ['Wave Height','Wind Speed','Water Temperature','Air Temperatur

fig,axes = plt.subplots(4,1,figsize=(10,20))
fig.suptitle('Least-Squares Fit for Monthly Data',fontweight='bold',y=0.
for vv,ax in enumerate(axes):
    ax.plot(months,means15[vv,:],color='tab:blue',label='2015')
    ax.plot(months,means16[vv,:],color='tab:red',label='2016')
    ax.plot(months,ffit15[vv,:],color='tab:blue',linestyle='--',label='F
    ax.plot(months,ffit16[vv,:],color='tab:red',linestyle='--',label='Fi
    if vv==3:
        ax.set_xlabel('Month of Year')

    ax.set_ylabel(ylabels[vv])
    ax.set_title(titles[vv])
    ax.legend(loc='lower right')

plt.show()
```

```
2015 Mean from Wave Height fit: 1.1006212996951859
Amplitude of sine semi-annual cycle:-0.04673598034487525
Amplitude of cosine semi-annual cycle: 0.05135380213654095


2016 Mean from Wave Height fit: 2.6511548045307487
Amplitude of sine semi-annual cycle:-0.1860814697250241
Amplitude of cosine semi-annual cycle: 0.04587303160663915


2015 Mean from Wind Speed fit: 3.1712022274962965
Amplitude of sine semi-annual cycle:0.07858858828315696
Amplitude of cosine semi-annual cycle: -0.09766098957008018


2016 Mean from Wind Speed fit: 5.023597516386346
Amplitude of sine semi-annual cycle:0.13622917789830216
Amplitude of cosine semi-annual cycle: -0.3417776911923421


2015 Mean from Water Temp fit: 18.63802803730167
Amplitude of sine semi-annual cycle:0.07924127755979193
Amplitude of cosine semi-annual cycle: -1.0328981233677816


2016 Mean from Water Temp fit: 16.565627639542573
Amplitude of sine semi-annual cycle:-0.3312387247787377
Amplitude of cosine semi-annual cycle: -0.2818091380766373


2015 Mean from Air Temp fit: 18.09563166224626
Amplitude of sine semi-annual cycle:0.12379058090004014
Amplitude of cosine semi-annual cycle: -1.0581381529403067


2016 Mean from Air Temp fit: 16.156760470456003
Amplitude of sine semi-annual cycle:-0.37328533349812626
Amplitude of cosine semi-annual cycle: -0.41049222844914657
```


Least-Squares Fit for Monthly Data
Wave Height
Wave Height [m]
3.00
2.75
2.50
2.25
2.00
1.75
1.50
2
4
6
8
10
12
2015
2016
Fit
Fit
Wind Speed
Wind Speed [m/s]
8
7
6
5
4
Water Temperature
Water Temperature [$^{\circ}$C]
21
20
19
18
17
16
15
Air Temperature
perature [$^{\circ}$C]
20
19
18



The amplitudes of the semi-annual fit are printed above.

In [23]:
```python
# compare annual cycle between low and high frequency fits:

for vv,var in enumerate(variables):
    print('2015 Fits: ')
    print(f'{var}: Low Frequency Total Amplitude = {total_amp15[vv]} \n 
    print('2016 Fits: ')
    print(f'{var}: Low Frequency Total Amplitude = {total_amp16[vv]} \n 
    print('\n')
```

```
2015 Fits:
Wave Height: Low Frequency Total Amplitude = 0.7147509156809175
 High Frequency Total Amplitude = 0.6700314319623479
2016 Fits:
Wave Height: Low Frequency Total Amplitude = 0.5573033805203481
 High Frequency Total Amplitude = 0.565961912111839


2015 Fits:
Wind Speed: Low Frequency Total Amplitude = 1.6686214684469112
 High Frequency Total Amplitude = 1.7721752380499447
2016 Fits:
Wind Speed: Low Frequency Total Amplitude = 0.9559609309252632
 High Frequency Total Amplitude = 1.2606796382258065


2015 Fits:
Water Temp: Low Frequency Total Amplitude = 3.2985721811903583
 High Frequency Total Amplitude = 2.980867801623151
2016 Fits:
Water Temp: Low Frequency Total Amplitude = 1.404934441237933
 High Frequency Total Amplitude = 1.5910666328673928


2015 Fits:
Air Temp: Low Frequency Total Amplitude = 3.4973819638411987
 High Frequency Total Amplitude = 3.13831156240496
2016 Fits:
Air Temp: Low Frequency Total Amplitude = 1.4836782110829647
 High Frequency Total Amplitude = 1.6796295686576301
```

The augmented fit roughly gives a similar total amplitude in the annual cycle.

Q5: $ \chi^2 $ and the misfit

What is the squared misfit of your least-squares fits?

The misfit for each point should be roughly equal to the uncertainty. We lose a degree of freedom for each function that we use to fit.

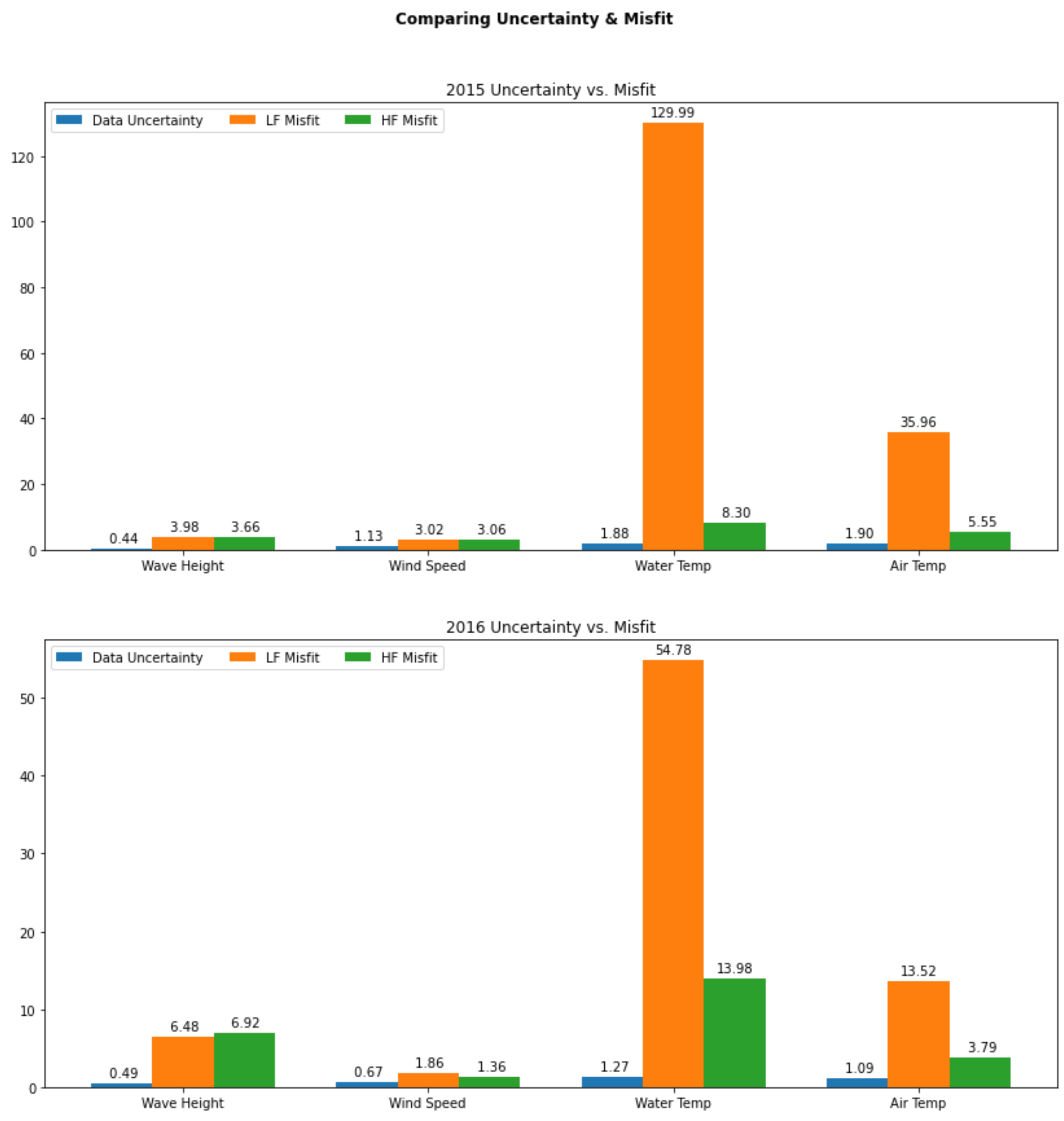
Formula: $$\chi^2 = \sum_{i=1}^N \frac{(y_i - \sum_{j=1}^M a_{ij} x_{j})^2}{\sigma_i^2}$$

Where we will use the standard error of the mean for the uncertainty (or $\sigma^2 $) for each of the data points.

In [24]:
```
# compute the misfit at each point

# low frequency
# chi = sum of the square of (data - fit) / sem^2
chi15 = np.zeros(4)
chi16 = np.zeros(4)

for vv,var in enumerate(variables):
    # cycle through variables
    for tt,mm in enumerate(months):
        # loop through each "observation" time point
        # in year 2015:
        sq = (means15[vv,tt] - fit15[vv,tt])**2
        sdev = error15[vv,tt]**2
        summing = sq/sdev
        chi15[vv] = chi15[vv]+summing

        # in year 2016:
        sq = (means16[vv,tt] - fit16[vv,tt])**2
        sdev = error16[vv,tt]**2
        summing = sq/sdev
        chi16[vv] = chi16[vv]+summing



# high frequency
# chi = sum of the square of (data - fit) / sem^2
cchi15 = np.zeros(4)
cchi16 = np.zeros(4)

for vv,var in enumerate(variables):
    # cycle through variables
    for tt,mm in enumerate(months):
        # loop through each "observation" time point
        # in year 2015:
        sq = (means15[vv,tt] - ffit15[vv,tt])**2
        sdev = error15[vv,tt]**2
        summing = sq/sdev
        cchi15[vv] = cchi15[vv]+summing

        # in year 2016:
        sq = (means16[vv,tt] - ffit16[vv,tt])**2
        sdev = error16[vv,tt]**2
        summing = sq/sdev
        cchi16[vv] = cchi16[vv]+summing
```

In [26]:

```
# compare misfit (for high and low frequency) and uncertainty:

# where the uncertainty is standard deviation aka sigma2 across data, mi

stdev15 = np.std(means15,axis=1)
stdev16 = np.std(means16,axis=1)

fig,axes = plt.subplots(2,1,figsize=(14,14))
fig.suptitle('Comparing Uncertainty & Misfit',fontweight='bold',y=0.95)
titles = ['2015 Uncertainty vs. Misfit', '2016 Uncertainty vs. Misfit']

# 2015
ax = axes[0]
xticks = np.arange(4)
width = 0.25 # the width of the bars
bars1 = ax.bar(xticks-1/4,stdev15,width,label='Data Uncertainty')
ax.bar_label(bars1, padding=3,fmt='%5.2f')
bars2 = ax.bar(xticks,chi15,width,label='LF Misfit')
ax.bar_label(bars2, padding=3,fmt='%5.2f')
bars3 = ax.bar(xticks+1/4,cchi15,width,label='HF Misfit')
ax.bar_label(bars3, padding=3, fmt='%5.2f')

# Add info and stuff to plot features
ax.set_title(titles[0]); ax.set_xticks(xticks, variables); ax.legend(loc

# 2016
ax = axes[1]
xticks = np.arange(4)
width = 0.25 # the width of the bars
bars1 = ax.bar(xticks-1/4,stdev16,width,label='Data Uncertainty')
ax.bar_label(bars1, padding=3,fmt='%5.2f')
bars2 = ax.bar(xticks,chi16,width,label='LF Misfit')
ax.bar_label(bars2, padding=3,fmt='%5.2f')
bars3 = ax.bar(xticks+1/4,cchi16,width,label='HF Misfit')
ax.bar_label(bars3, padding=3,fmt='%5.2f')

# Add info and stuff to plot features
ax.set_title(titles[1]); ax.set_xticks(xticks, variables); ax.legend(loc

plt.show()
```



For wave height, the misfit remains more or less the same between the model with just the annual cycle and the model with the annual + semi-annual cycle. For the 2016 fits, the misfit actually worsens (increases) with the annual and semi-annual cycle.

For wind speed, the misfit remains more or less the same between the model with just the annual cycle and the model with the annual + semi-annual cycle.

For both water & air temperature, the misfit is much improved (decreases) by adding a semi-annual cycle - especially for water temperature. Looking at the fits themselves, the annual+semi-annual model can much better capture the steeper changes in temp throughout the year, whereas the annual model can only gradually increase temp into the summer and then gradually decrease to the winter, unable to capture the large temperature "deviations" (aka high summer temperatures and low winter temps) in the seasonal trends. I guess this "seasonal" variation is a sub-annual trend that the higher frequency fit can represent which could not be represented in the lower frequency fit.

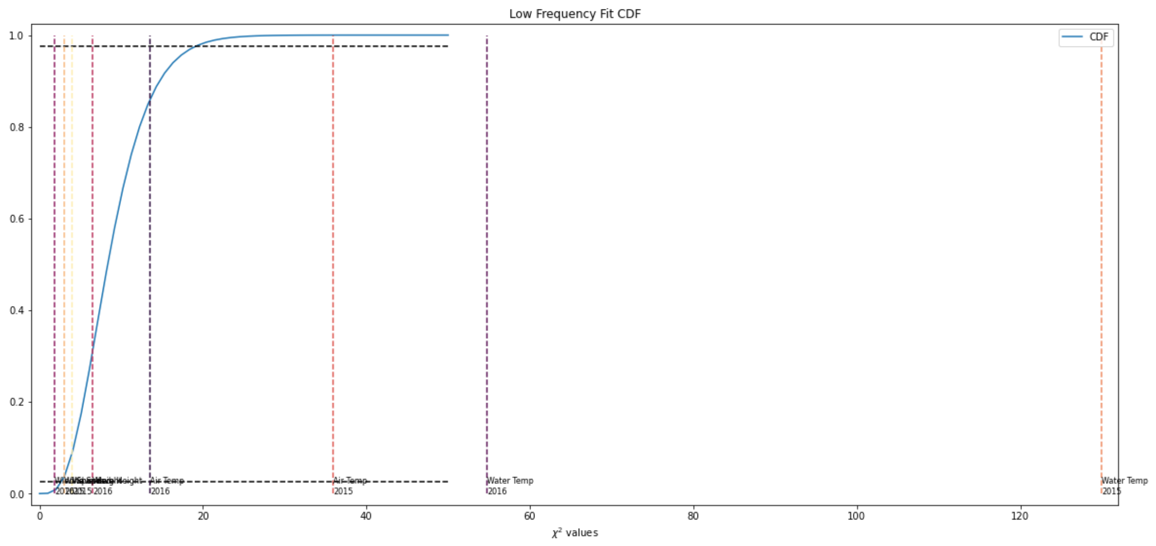
In [27]:
```
# We lose a degree of freedom for each function that we use to fit.
# degrees of freedom = N-M
# N = length of dataset / number of observations
N = 12
# M = number of model parameters (or function) used to fit (our case: me
M_lf = 3 # low frequency
M_hf = 5 # high frequency


deg_lf = N-M_lf
deg_hf = N-M_hf

print(f'Degrees of freedom: LF = {deg_lf}; HF = {deg_hf}')
```

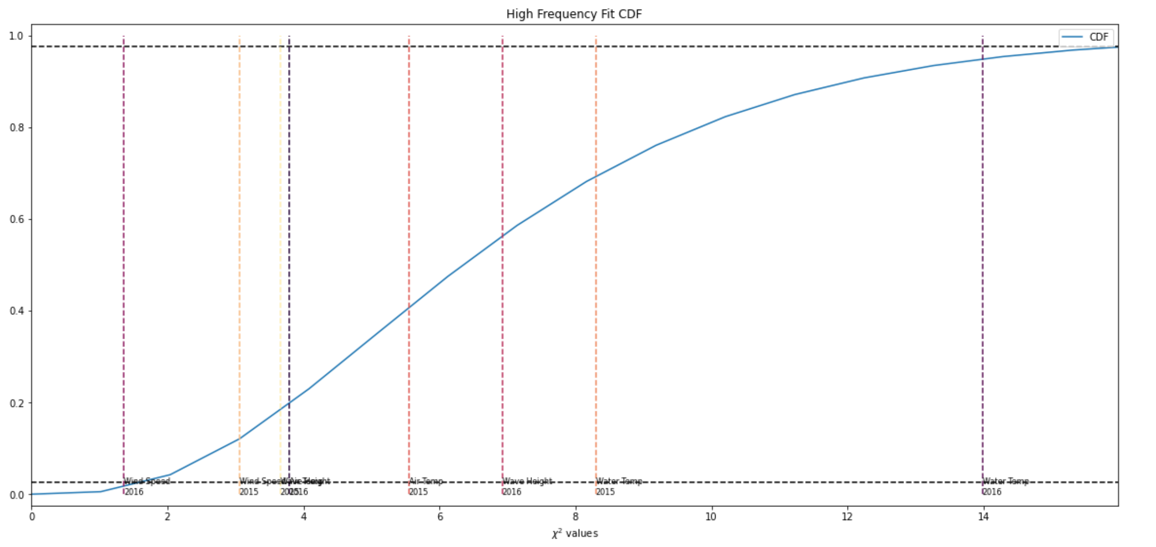
```
Degrees of freedom: LF = 9; HF = 7
```

Am I overfitting the data? If my misfit is less than my data uncertainty, then I am overfitting the data. All of my misfits are greater than my data uncertainty, so all of my model fits are underfitting the data. We can further investigate this by seeing whether the chi2 misfit values fall within the 2.5% and 97.5% of a chi2 distribution for the appropriate degrees of freedom (in our case: LF = 9; HF = 7).

In [36]:
```
def ind_nearest(array, value):
    array = np.asarray(array)
    idx = (np.abs(array - value)).argmin()
    return idx

chi = np.linspace(0,50)

lfcdf = scipy.stats.chi2.cdf(chi, df=deg_lf)
hfcdf = scipy.stats.chi2.cdf(chi, df=deg_hf)


ulf_idx = ind_nearest(lfcdf,0.975); u_lf = chi[ulf_idx];
blf_idx = ind_nearest(lfcdf,0.025); l_lf = chi[blf_idx];

print(f'For annual cycle (only low frequency fit), misfit should be above

print(f'Reminding myself of LF misfit values: 2015 - {chi15}  and 2016 va

uhf_idx = ind_nearest(hfcdf,0.975); u_hf = chi[uhf_idx];
bhf_idx = ind_nearest(hfcdf,0.025); l_hf = chi[bhf_idx];

print(f'For annual+semi-annual cycle (high frequency fit), misfit should

print(f'Reminding myself of HF misfit values: 2015 - {cchi15}  and 2016 v

cpos = np.linspace(0,1,8)
colors = [cmocean.cm.matter(c) for c in cpos]

text = ['Wave Height \n2015', 'Wind Speed \n2015', 'Water Temp \n2015',
lf_vals = np.append(chi15,chi16)
hf_vals = np.append(cchi15,cchi16)

fig,(ax1,ax2)=plt.subplots(2,1,figsize=(20,20))
ax1.set_title('Low Frequency Fit CDF')
ax1.plot(chi, lfcdf, label='CDF')
ax1.hlines(y=[0.975,0.025],xmin=0,xmax=50,color='k',linestyle='--')
ax1.vlines(x=lf_vals,ymin=0,ymax=1,color=colors,linestyle='--')
for ii,txt in enumerate(text):
    ax1.annotate(txt,xy=(lf_vals[ii],0),fontsize=8)
ax1.set_ylim(-0.025,1.025); ax1.set_xlim(-1,np.nanmax(lf_vals)+2)
ax1.set_xlabel(r'$\chi^2$ values')
ax1.legend()

ax2.set_title('High Frequency Fit CDF')
ax2.plot(chi, hfcdf, label='CDF')
ax2.hlines(y=[0.975,0.025],xmin=0,xmax=50,color='k',linestyle='--')
ax2.vlines(x=hf_vals,ymin=0,ymax=1,color=colors,linestyle='--')
for ii,txt in enumerate(text):
    ax2.annotate(txt,xy=(hf_vals[ii],0),fontsize=8)
ax2.set_ylim(-0.025,1.025); ax2.set_xlim(0,np.nanmax(hf_vals)+2)
ax2.set_xlabel(r'$\chi^2$ values')
ax2.legend()

plt.show()
```

```
For annual cycle (only low frequency fit), misfit should be above 3.06122
44897959187 and below 19.387755102040817
Reminding myself of LF misfit values: 2015 - [  3.98199415   3.01701316 1
29.98845861  35.95991563]  and 2016 values [ 6.48317649  1.8639303  54.77
69323  13.52110713]
For annual+semi-annual cycle (high frequency fit), misfit should be above
2.0408163265306123 and below 16.3265306122449
Reminding myself of HF misfit values: 2015 - [3.65687992 3.05636262 8.295
38574 5.54736595]  and 2016 values [ 6.92354815  1.36266764 13.98052057
3.78804762]
```





We can see that the 2015&2016 water temperature and 2015 air temp annual cycle misfits all fell above a normal chi2 distribution (over the 97.5th percentile). The wind speed for 2016 on the other hand was below the 2.5th percentile, which means that the wind speed low frequency fit for 2016 was overfit to the data.

In terms of the annual+semi-annual fits, almost all of high frequency cycle misfits fall within of the normal chi2 distribution - with the exception of wind speed, which falls below the 2.5 percentile - which means it is overfit to the data.

If I assumed there was 1 independent sample per day, then my uncertainty (standard error of the mean) would go down much lower (from N = ~4 to N = 30.5), which would make my misfits much higher.

Just some office hours notes (for me!):

underfitting - would be a smaller value for the uncertainty than the misfit

sigma2 = uncertainty = sem - each should be off by sigma if the misfit is less, then it means that you have a model that is more sophisticated than expected --> overfitted

overfitting -

low misfit compared to sigma --> just doesn't follow the assumptions you've made

compare to plotting the CDF: misfit is probably the

N = 12 reduced the misfit because made a model --> the CDF will look different then the misfit should be consistent with a chi2 variable: so the CDF of a chi2 variable would match meaning that you would be in the middle somewhere with your uncertainty