Physics 474 - Spring 2023
Homework 1

Author: Dillon Walton

In this homework we will practice fitting a function with parameters to some data.

skills we will excercise:

- reading in data
- plotting data
- writing user defined functions
- fitting a function to data with 'curve_fit'
- calculating χ^2 and χ^2 probaility
- plotting residuals
- analyzing data and making observations

The data we will be using is global temperature data compiled from The current citation for this dataset is:

```
Rohde, R. A. and Hausfather, Z.: The Berkeley Earth Land/Ocean
Temperature
Record, Earth Syst. Sci. Data, 12, 3469�3479,
https://doi.org/10.5194/essd-12-3469-2020, 2020.
```

Data is: year, month, delta_T(C), T_error(C)

where delta_T = Temp - (Jan 1951-Dec 1980 global mean temperature)

Estimated Jan 1951-Dec 1980 global mean temperature (C) Using air temperature above sea ice: 14.105 +/- 0.022

The data is provided in a comma-separated-value file named 'global_temps_datafile.csv'

Note that the temperature data is provided as a ΔT from the global mean 1951-1980 temperature of

$T^{51-80}_{mean} = 14.105 \pm 0.022°C$

Part 1) (1 pt)

Read in the data file and print the shape of the file

In [ ]:
```
#Your code here...
import pandas as pd
```

```
temp_data = pd.read_csv('./global_temps_datafile_worker.csv', sep=',', engine='
print(temp_data.shape)
```

```
(2074, 4)
```

Part 2) (4 pts)

Plot

- the data points with errorbars vs year.
- a horizontal line at $T_{mean}^{51-80} = 14.105°C$
- suggestion: use "alpha=0.2" in plotting data points

About errors: There are three errors that need to be added in quadrature

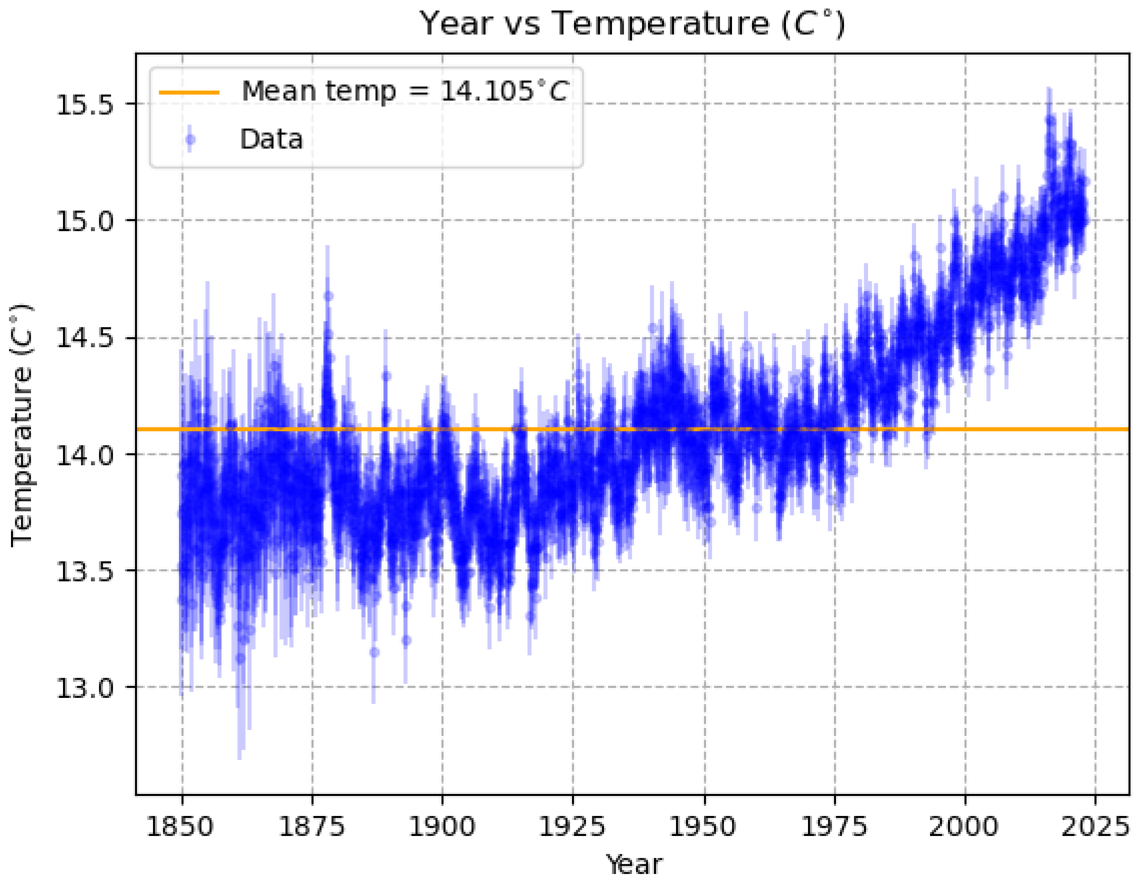
- the error on each of the measurements σ_T from the file
- error on the T_{mean}^{51-80} of $\sigma_{T_{mean}} = 0.022$
- a random systematic error of $\sigma_{sys} = 0.13$

Recall from lecture then

$$\sigma_{tot}^2 = \sigma_T^2 + \sigma_{T_{mean}}^2 + \sigma_{sys}^2$$

In [ ]:

```
#Your code here...
import matplotlib.pyplot as plt

temp_data_cp = temp_data.copy()
temp_data_cp['date'] = temp_data_cp['year'] + (temp_data_cp['month'] * (1/12))
temp_data_cp['error'] = ((temp_data_cp['error_t'])**2 + (0.022)**2 + (0.13)**2)
temp_data_cp['temp'] = 14.105 + temp_data_cp['delta_t']

plt.errorbar(temp_data_cp['date'],temp_data_cp['temp'],yerr=temp_data_cp['error
plt.axhline(y=14.105, color='orange', label=r'Mean temp = $14.105^{\circ} C$')
plt.grid(True,linestyle='--')
plt.title(r'Year vs Temperature ($C^{\circ}$)')
plt.xlabel('Year')
plt.ylabel(r'Temperature ($C^{\circ}$)')
plt.legend()
plt.show()
```



Part 2 Observations:

Part 3a) (2 pts)

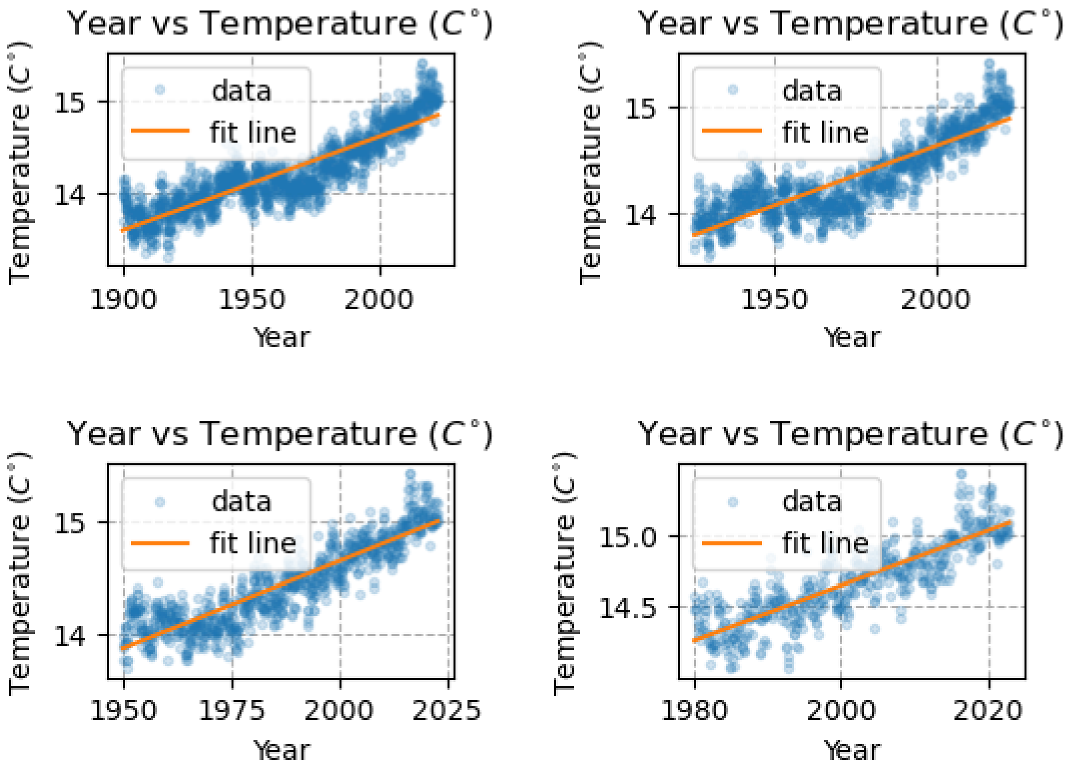
Fit a straight-line (i.e. first-order polynomial) to the data using 'curve_fit' from some point starting after the pre-industrial era (i.e. sometime after 1900). You might try after 1900, 1925, 1950, 1980...

sometime after 1900 but before the visible rise around 1980.

In [ ]:
```
#Your code here...
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
import math

def straightLine(m, x, b):
    return (m*x) + b

temp_data_cp = temp_data.copy()
temp_data_cp['date'] = temp_data_cp['year'] + (temp_data_cp['month'] * (1/12))
temp_data_cp['error'] = ((temp_data_cp['error_t'])**2 + (0.022)**2 + (0.13)**2)
temp_data_cp['temp'] = 14.105 + temp_data_cp['delta_t']

fig, axs = plt.subplots(2,2)
```

```
fig.tight_layout(pad=5.0)

nineteen_hundred = temp_data_cp[600:]
param1900, param1900_cov = curve_fit(straightLine, nineteen_hundred['date'], ni
nineteen_hundred_fit = straightLine(nineteen_hundred['date'], param1900[0], par

nineteen_twentyfive = temp_data_cp[900:]
param1925, param1925_cov = curve_fit(straightLine, nineteen_twentyfive['date'],
nineteen_twentyfive_fit = straightLine(nineteen_twentyfive['date'], param1925[0

nineteen_fifty = temp_data_cp[1200:]
param1950, param1950_cov = curve_fit(straightLine, nineteen_fifty['date'], nine
nineteen_fifty_fit = straightLine(nineteen_fifty['date'], param1950[0], param19

nineteen_eighty = temp_data_cp[1560:]
param1980, param1980_cov = curve_fit(straightLine, nineteen_eighty['date'], nir
nineteen_eighty_fit = straightLine(nineteen_eighty['date'], param1980[0], param


axs[0,0].plot(nineteen_hundred['date'], nineteen_hundred['temp'], '.', alpha=0.
axs[0,0].plot(nineteen_hundred['date'], nineteen_hundred_fit, '-', label='fit l
axs[0,0].grid(True,linestyle='--')
axs[0,0].set_xlabel('Year')
axs[0,0].set_ylabel(r'Temperature ($C^{\circ}$)')
axs[0,0].legend()
axs[0,0].set_title(r'Year vs Temperature ($C^{\circ}$)')


axs[0,1].plot(nineteen_twentyfive['date'], nineteen_twentyfive['temp'], '.', al
axs[0,1].plot(nineteen_twentyfive['date'], nineteen_twentyfive_fit, '-', label=
axs[0,1].grid(True,linestyle='--')
axs[0,1].set_xlabel('Year')
axs[0,1].set_ylabel(r'Temperature ($C^{\circ}$)')
axs[0,1].legend()
axs[0,1].set_title(r'Year vs Temperature ($C^{\circ}$)')

axs[1,0].plot(nineteen_fifty['date'], nineteen_fifty['temp'], '.', alpha=0.2, l
axs[1,0].plot(nineteen_fifty['date'], nineteen_fifty_fit, '-', label='fit line'
axs[1,0].grid(True,linestyle='--')
axs[1,0].set_xlabel('Year')
axs[1,0].set_ylabel(r'Temperature ($C^{\circ}$)')
axs[1,0].legend()
axs[1,0].set_title(r'Year vs Temperature ($C^{\circ}$)')

axs[1,1].plot(nineteen_eighty['date'], nineteen_eighty['temp'], '.', alpha=0.2,
axs[1,1].plot(nineteen_eighty['date'], nineteen_eighty_fit, '-', label='fit lir
axs[1,1].grid(True,linestyle='--')
axs[1,1].set_xlabel('Year')
axs[1,1].set_ylabel(r'Temperature ($C^{\circ}$)')
axs[1,1].legend()
axs[1,1].set_title(r'Year vs Temperature ($C^{\circ}$)')

# print(f'slope: {round(param1900[0],4)} +- {round(math.sqrt(param1900_cov[0,0]
# print(f'intercept: {round(param1900[1],1)} +- {round(math.sqrt(param1900_cov[
```

Out[ ]:
```
Text(0.5, 1.0, 'Year vs Temperature ($C^{\\circ}$)')
```



Part 3b) (3 pts)

1) Calculate and output the χ^2, dof, and χ^2 probability for the fit

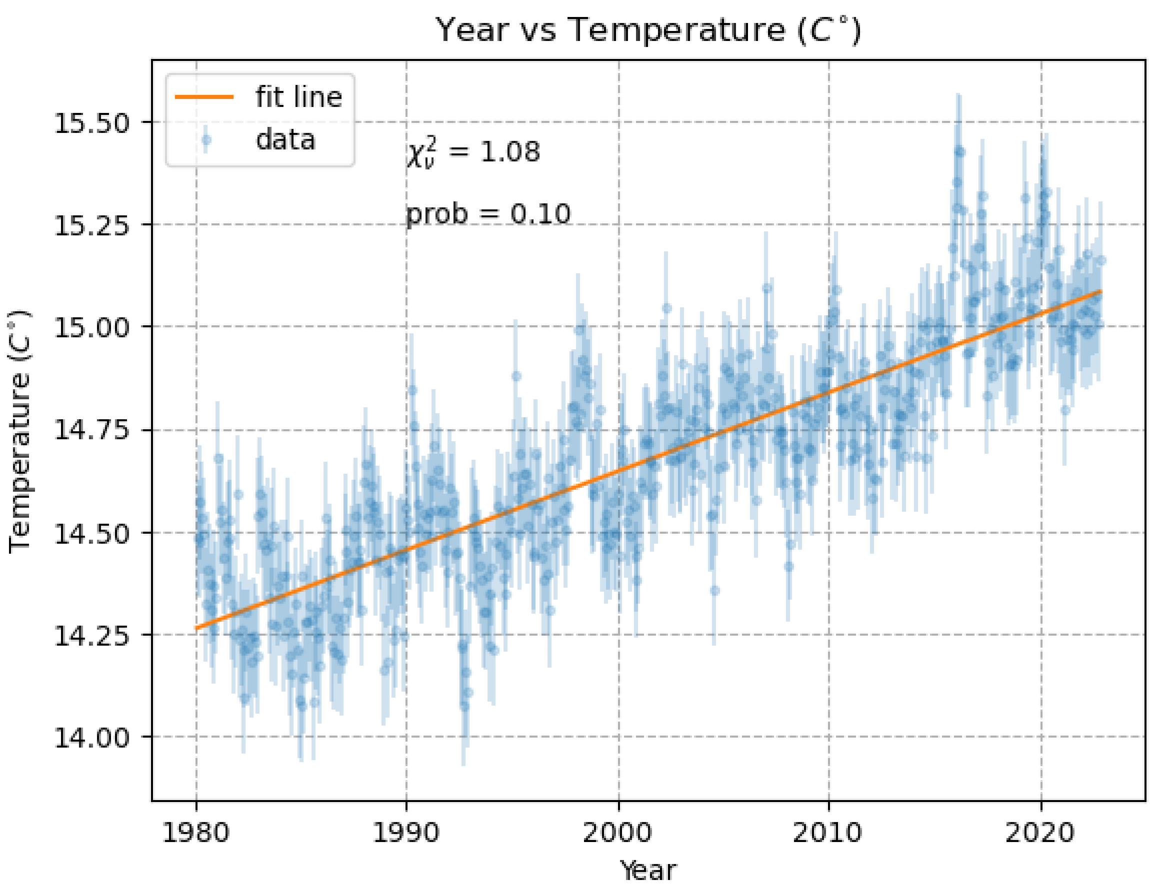
2) Plot

- the data with errorbars for the entire range of years
- the best fit line over the years for the fit
- put labels on the data, fit, axes, etc...

In [ ]:
```
#Your code here...
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
import scipy.stats as st
import math
import numpy as np

def straightLine(m, x, b):
    return (m*x) + b

def chi_squared(Theory,Data,sigma):
    if np.size(Theory)==np.size(Data) and np.size(Data)==np.size(sigma):
        chi2=np.sum((Theory-Data)**2/sigma**2)
        return chi2
    else:
        print('error - arrays of unequal size')
        return -1.

temp_data_cp = temp_data.copy()
temp_data_cp['date'] = temp_data_cp['year'] + (temp_data_cp['month'] * (1/12))
```

```
temp_data_cp['error'] = ((temp_data_cp['error_t'])**2 + (0.022)**2 + (0.13)**2)
temp_data_cp['temp'] = 14.105 + temp_data_cp['delta_t']

nineteen_eighty = temp_data_cp[1560:]
param1980, param1980_cov = curve_fit(straightLine, nineteen_eighty['date'], nin
nineteen_eighty_fit = straightLine(nineteen_eighty['date'], param1980[0], param

chi2 = chi_squared(nineteen_eighty_fit, nineteen_eighty['temp'], nineteen_eight
dof = nineteen_eighty.shape[0] - 2
prob = st.chi2.sf(chi2,dof)

plt.errorbar(nineteen_eighty['date'], nineteen_eighty['temp'], yerr=nineteen_ei
plt.plot(nineteen_eighty['date'], nineteen_eighty_fit, '-', label='fit line')
plt.grid(True,linestyle='--')
plt.xlabel('Year')
plt.ylabel(r'Temperature ($C^{\circ}$)')
plt.legend()
plt.text(1990,15.4,r'$\chi_\nu^2$ = {:.2f}'.format(chi2/dof))
plt.text(1990,15.25,r'prob = {:.2f}'.format(prob))
plt.title(r'Year vs Temperature ($C^{\circ}$)')
```

Out[ ]: `Text(0.5, 1.0, 'Year vs Temperature ($C^{\\circ}$)')`

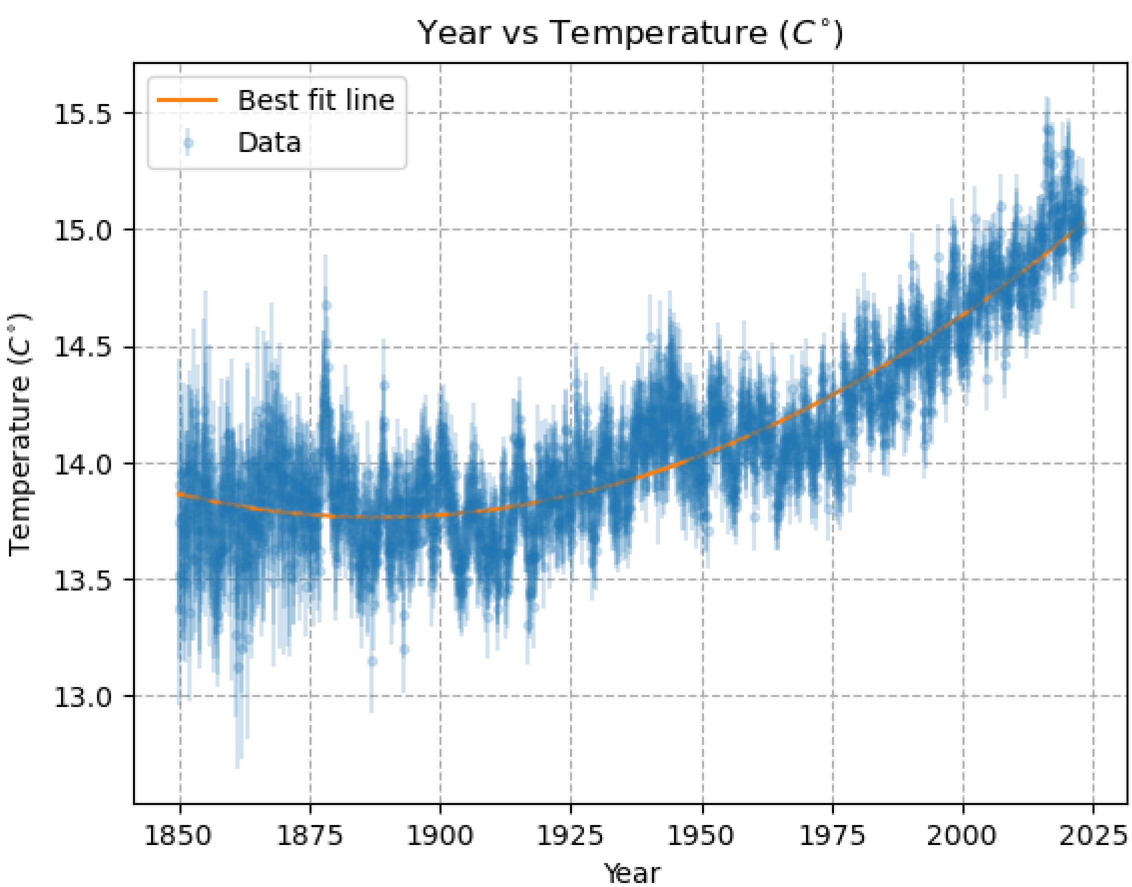


Part 3 Observations: (will depend on range used for linear fit)

Part 4a) (2 pts)

Fit a Quadratic (i.e. second-order polynomial) to the data using 'curve_fit' ffor the entire range of years

In [ ]:
```
#Your code here...
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

def objective(x, a, b, c):
    return a * x + b * x**2 + c

temp_data_cp = temp_data.copy()
temp_data_cp['date'] = temp_data_cp['year'] + (temp_data_cp['month'] * (1/12))
temp_data_cp['error'] = ((temp_data_cp['error_t'])**2 + (0.022)**2 + (0.13)**2)
temp_data_cp['temp'] = 14.105 + temp_data_cp['delta_t']

param, param_cov = curve_fit(objective, temp_data_cp['date'], temp_data_cp['ten
fit = objective(temp_data_cp['date'], param[0], param[1], param[2])

plt.errorbar(temp_data_cp['date'],temp_data_cp['temp'],yerr=temp_data_cp['error
plt.plot(temp_data_cp['date'], fit, '-', label='Best fit line')
plt.grid(True,linestyle='--')
plt.title(r'Year vs Temperature ($C^{\circ}$)')
plt.xlabel('Year')
plt.ylabel(r'Temperature ($C^{\circ}$)')
plt.legend()
plt.show()
plt.show()
```



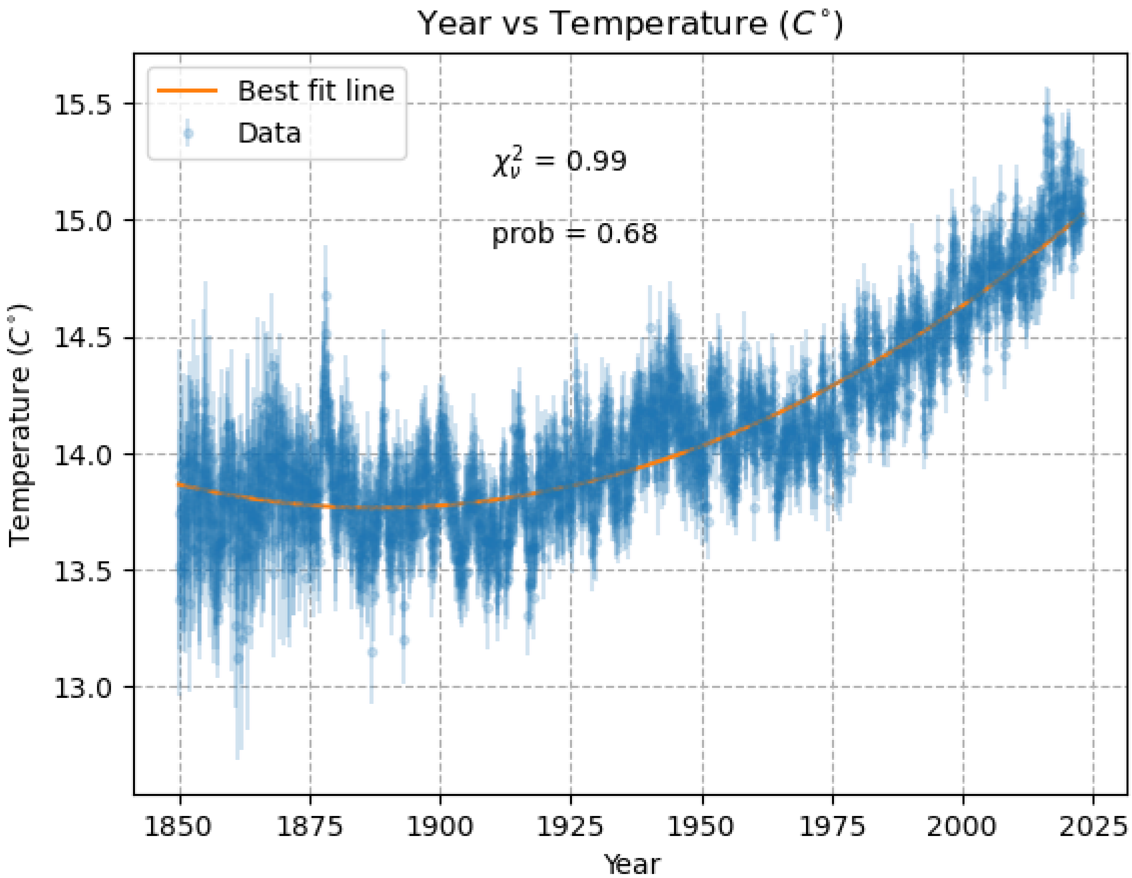
Part 4b) (3 pts)

1) Calculate and output the χ^2, dof, and χ^2 probability for the fit

2) Plot

- the data with errorbars for the entire range of years
- the best fit quadratic over the entire range of years
- put labels on the data, fit, axes, etc...

In [ ]:
```
#Your code here...
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as st    #for chi-squared probability
from scipy.optimize import curve_fit   #for curve_fit routine

def chi_squared(Theory,Data,sigma):
    if np.size(Theory)==np.size(Data) and np.size(Data)==np.size(sigma):
        chi2=np.sum((Theory-Data)**2/sigma**2)
        return chi2
    else:
        print('error - arrays of unequal size')
        return -1.

def objective(x, a, b, c):
    return a * x + b * x**2 + c

temp_data_cp = temp_data.copy()
temp_data_cp['date'] = temp_data_cp['year'] + (temp_data_cp['month'] * (1/12))
temp_data_cp['error'] = ((temp_data_cp['error_t'])**2 + (0.022)**2 + (0.13)**2)
temp_data_cp['temp'] = 14.105 + temp_data_cp['delta_t']

param, param_cov = curve_fit(objective, temp_data_cp['date'], temp_data_cp['tem
fit = objective(temp_data_cp['date'], param[0], param[1], param[2])

chi2 = chi_squared(objective(temp_data_cp['date'], param[0], param[1], param[2]
dof = temp_data_cp.shape[0] - 3
prob = st.chi2.sf(chi2,dof)

plt.figure()
plt.errorbar(temp_data_cp['date'],temp_data_cp['temp'],yerr=temp_data_cp['error
plt.plot(temp_data_cp['date'], fit, '-', label='Best fit line')
plt.grid(True,linestyle='--')
plt.title(r'Year vs Temperature ($C^{\circ}$)')
plt.xlabel('Year')
plt.ylabel(r'Temperature ($C^{\circ}$)')
plt.legend()
plt.text(1910,15.2,r'$\chi_\nu^2$ = {:.2f}'.format(chi2/dof))
plt.text(1910,14.9,r'prob = {:.2f}'.format(prob))
plt.show()
```



Part 4 Observations:

Part 5: (2 pts)

Make 2 subplots (2 rows x 1 column)

- top: residuals with errorbars for the straight-line fit for years used in the fit
- bottom: residuals with errorbars for quadratic fit for years used in the fit

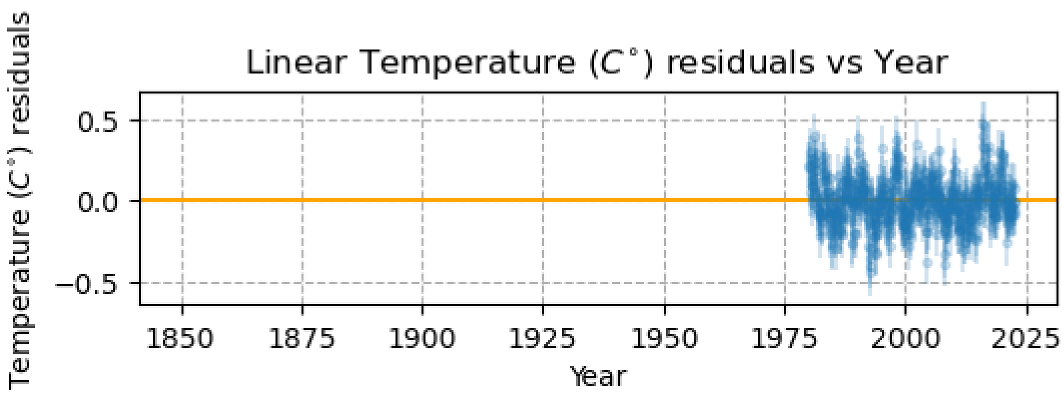
Use the same limits on the x-axis for both 1850-2025

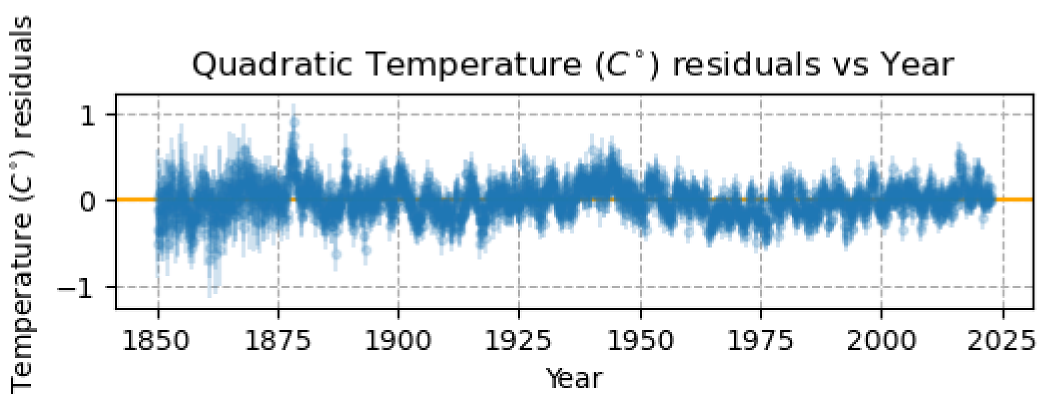
In [ ]:

```
#Your code here...
from numpy import random, mean, std
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit   #for curve_fit routine

def objectiveStraight(m, x, b):
    return (m*x) + b

def objectiveCurve(x, a, b, c):
    return a * x + b * x**2 + c

temp_data_cp = temp_data.copy()
temp_data_cp['date'] = temp_data_cp['year'] + (temp_data_cp['month'] * (1/12))
temp_data_cp['error'] = ((temp_data_cp['error_t'])**2 + (0.022)**2 + (0.13)**2)
```

```
temp_data_cp['temp'] = 14.105 + temp_data_cp['delta_t']

nineteen_eighty = temp_data_cp[1560:]

paramStraight, paramStraight_cov = curve_fit(straightLine, nineteen_eighty['dat
paramCurve, paramCurve_cov = curve_fit(objectiveCurve, temp_data_cp['date'], te

fig, axs = plt.subplots(2)
fig.tight_layout(pad=5.0)

axs[0].axhline(y=0, color='orange')
axs[0].errorbar(temp_data_cp['date'], nineteen_eighty['temp']-objectiveStraight
axs[0].grid(True,linestyle='--')
axs[0].set_xlabel('Year')
axs[0].set_ylabel(r'Temperature ($C^{\circ}$) residuals')
axs[0].set_title(r'Linear Temperature ($C^{\circ}$) residuals vs Year')

axs[1].axhline(y=0, color='orange')
axs[1].errorbar(temp_data_cp['date'], temp_data_cp['temp']-objectiveCurve(temp_
axs[1].grid(True,linestyle='--')
axs[1].set_xlabel('Year')
axs[1].set_ylabel(r'Temperature ($C^{\circ}$) residuals')
axs[1].set_title(r'Quadratic Temperature ($C^{\circ}$) residuals vs Year')
```

Out[ ]:
```
Text(0.5, 1.0, 'Quadratic Temperature ($C^{\\circ}$) residuals vs Year')
```





Part 5 observations: (will depend on range used for linear fit)

Part 6: (3 pts)

On a single plot show

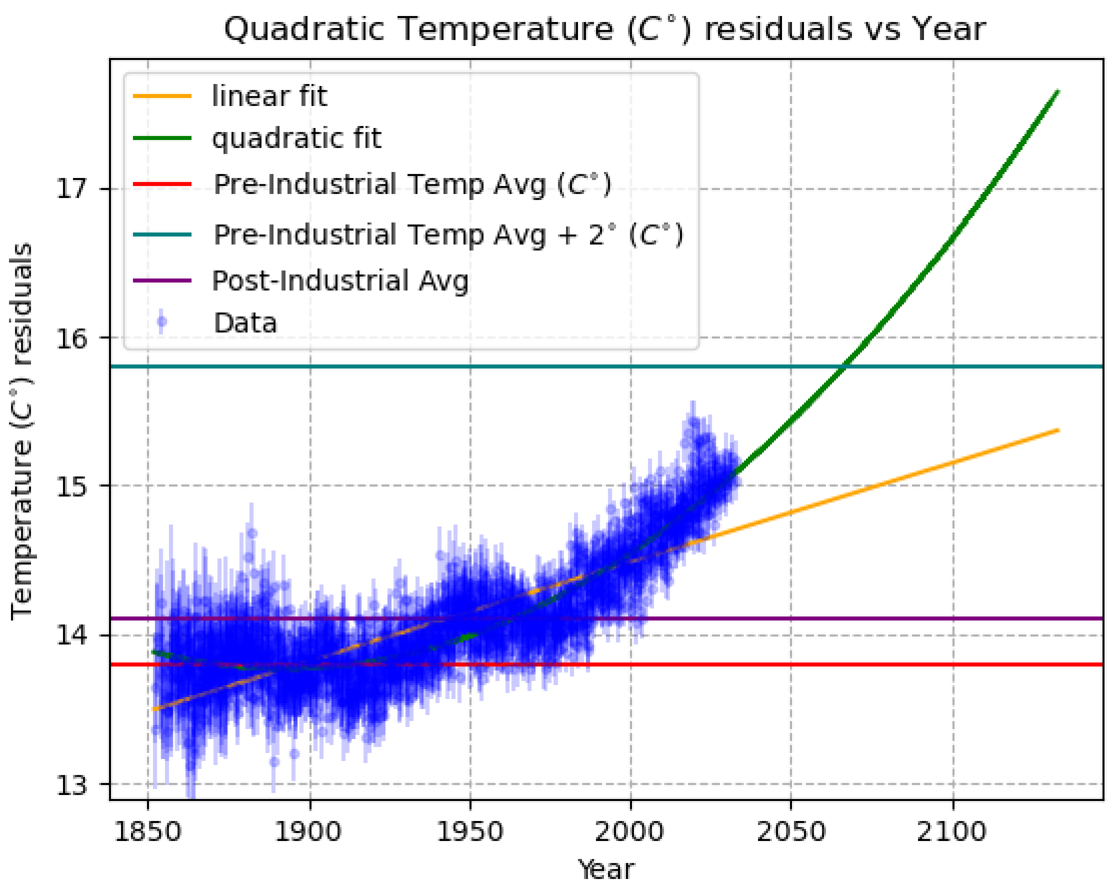
- data with errorbars
- both fits projected out 100 years
- a horizontal line from 1850-1900 with the mean pre-industrial temperature
- a horizontal line at the 1951-1980 T_{mean}
- a horizontal line at $\Delta T = 2^{\circ}C$ above the pre-industrial temperature

put all appropriate labels, etc... on plot

In [ ]:

```
#Your code here...
import matplotlib.pyplot as plt
import scipy.stats as st
from scipy.optimize import curve_fit
from numpy import mean
import pandas as pd


def objectiveStraight(m, x, b):
    return (m*x) + b

def objectiveCurve(x, a, b, c):
    return a * x + b * x**2 + c

temp_data_cp = temp_data.copy()
temp_data_cp['date'] = temp_data_cp['year'] + temp_data_cp['month']
temp_data_cp['error'] = ((temp_data_cp['error_t'])**2 + (0.022)**2 + (0.13)**2)
temp_data_cp['temp'] = 14.105 + temp_data_cp['delta_t']

projection_data = []
for i in range(1850,2122,1):
    for j in range(1,13,1):
        if i == 1850 and j == 1:
            continue
        else:
            data = [i,j]
            projection_data.append(data)

projection = pd.DataFrame(projection_data, columns=['year', 'month'])
projection['date'] = projection['year'] + projection['month']
projection['error'] = ((temp_data_cp['error_t'])**2 + (0.022)**2 + (0.13)**2)**
projection['temp'] = 14.105 + temp_data_cp['delta_t']

paramStraight, paramStraight_cov = curve_fit(objectiveStraight, temp_data_cp['d
straight_fit = objectiveStraight(projection['date'], paramStraight[0], paramStr

paramCurve, paramCurve_cov = curve_fit(objectiveCurve, temp_data_cp['date'], te
quadratic_fit = objectiveCurve(projection['date'], paramCurve[0], paramCurve[1]

projection['linear_fit'] = straight_fit
projection['quadratic_fit'] = quadratic_fit

pre_industrial_data = temp_data_cp[:600] # 1850 - 1900
pre_industrial_mean = mean(pre_industrial_data['temp'])

post_industrial_data = temp_data_cp[1200:1560] # 1950 - 1980
```

```
post_industrial_mean = mean(post_industrial_data['temp'])

# print(projection)

plt.errorbar(projection['date'],projection['temp'],yerr=projection['error'],fmt
plt.plot(projection['date'], projection['linear_fit'], '-', color='orange', lab
plt.plot(projection['date'], projection['quadratic_fit'], '-', color='green', l
plt.axhline(y=pre_industrial_mean, color='red', label=r'Pre-Industrial Temp Avg
plt.axhline(y=pre_industrial_mean+2, color='teal', label=r'Pre-Industrial Temp
plt.axhline(y=post_industrial_mean, color='purple', label=r'Post-Industrial Avg
plt.grid(True,linestyle='--')
plt.ylabel(r'Temperature ($C^{\circ}$) residuals')
plt.xlabel('Year')
plt.title(r'Quadratic Temperature ($C^{\circ}$) residuals vs Year')
plt.legend()
plt.show()
```



Final Observations and Summary:

Pretty interesting to see that the linear fit line does not cross above $2^{\circ}C$ before the hundred year mark. Unfortunately it seems we have experienced exponential growth since 1950. It would be interesting to see the population data in combination with this data, and any projections there are on what will happen when populations decrease.
