index

April 25, 2022

1 Visualizing Time Series Data - Lab

1.1 Introduction

As mentioned in an earlier lesson, time series visualizations play an important role in the analysis of time series data. Time series are often plotted to allow data diagnostics to identify temporal structures.

In this lab, we'll cover main techniques for visualizing time series data in Python using the minimum daily temperatures over 10 years (1981-1990) in the city of Melbourne, Australia. The units are in degrees Celsius and there are 3,650 observations. The source of the data is credited to the Australian Bureau of Meteorology.

1.2 Objectives

You will be able to:

- Explore the temporal structure of time series with line plots
- Construct and interpret time series histogram and density plots
- Create a time series heat map

1.3 Let's get started!

Run the cell below to import the necessary classes and libraries:

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

import pandas as pd
from pandas import Series
import matplotlib.pyplot as plt
```

- Import the dataset which is available in `'min_temp.csv'`
- Print the first five rows of the data

[15]:
```
# Load the data from 'min_temp.csv'
temp_data = pd.read_csv('min_temp.csv')

# Print the first five rows
temp_data.head()
```

[15]:
```
      Date  Daily_min
0   1/1/81       20.7
1   2/1/81       17.9
2   3/1/81       18.8
3   4/1/81       14.6
4   5/1/81       15.8
```

- Make sure the `'Date'` column is treated as an actual date by Python (notice how the date is formatted before attempting to changing the data type)
- Set the index of `temp_data` to this `'Date'` column

[16]:
```
# Change the data type of the 'Date' column
temp_data["Date"] = pd.to_datetime(temp_data["Date"], format='%d/%m/%y')

# Set the index to the 'Date' column
temp_data.set_index("Date", inplace = True)
```

Print the index of `temp_data`.

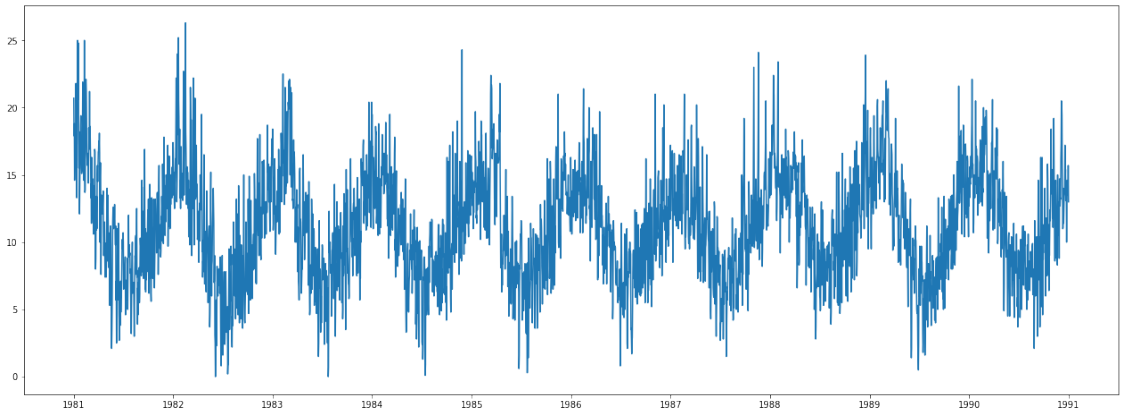
[17]:
```
# Print the index of the data
temp_data.index
```

[17]:
```
DatetimeIndex(['1981-01-01', '1981-01-02', '1981-01-03', '1981-01-04',
               '1981-01-05', '1981-01-06', '1981-01-07', '1981-01-08',
               '1981-01-09', '1981-01-10',
               …
               '1990-12-22', '1990-12-23', '1990-12-24', '1990-12-25',
               '1990-12-26', '1990-12-27', '1990-12-28', '1990-12-29',
               '1990-12-30', '1990-12-31'],
              dtype='datetime64[ns]', name='Date', length=3650, freq=None)
```

1.4 Time Series line plot

Create a time series line plot for `temp_data`

[22]:
```
# Draw a line plot using temp_data
fig, ax = plt.subplots(figsize = (22,8))
plt.plot(temp_data);
```

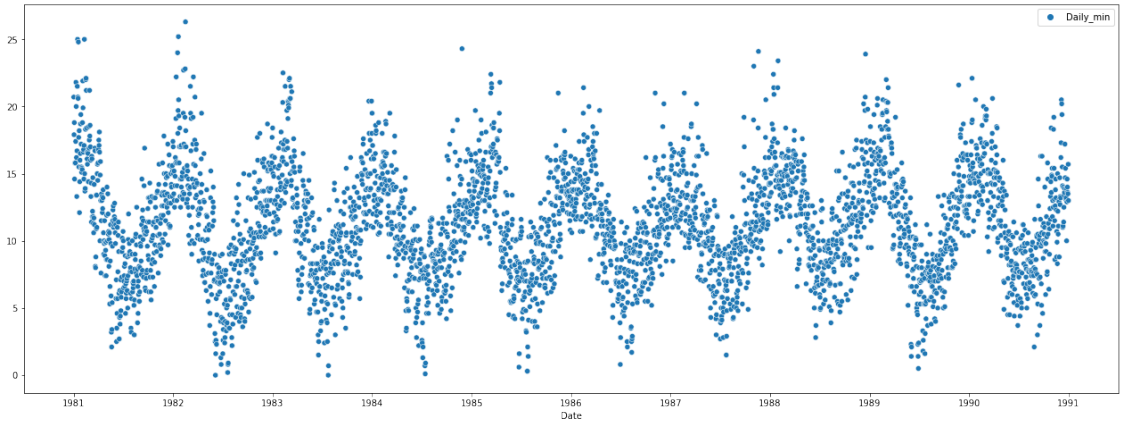


Some distinguishable patterns appear when we plot the data. Here we can see a pattern in our time series, i.e., temperature values are maximum at the beginning of each year and minimum at around the 6th month. Yes, we are talking about Australia here so this is normal. This cyclical pattern is known as seasonality and will be covered in later labs.

1.5 Time Series dot plot

For a dense time series, as seen above, you may want to change the style of a line plot for a more refined visualization with a higher resolution of events. One way could be to change the continuous line to dots, each representing one entry in the time series.

[44]:
```
# Use dots instead on a continuous line and redraw the time series
import seaborn as sns
fig, ax = plt.subplots(figsize = (22,8))
sns.scatterplot(data = temp_data);

# G
# temp_data.plot(figsize = (22,8), style = 'b.')
# plt.show()
```



This plot helps us identify clear outliers in certain years!

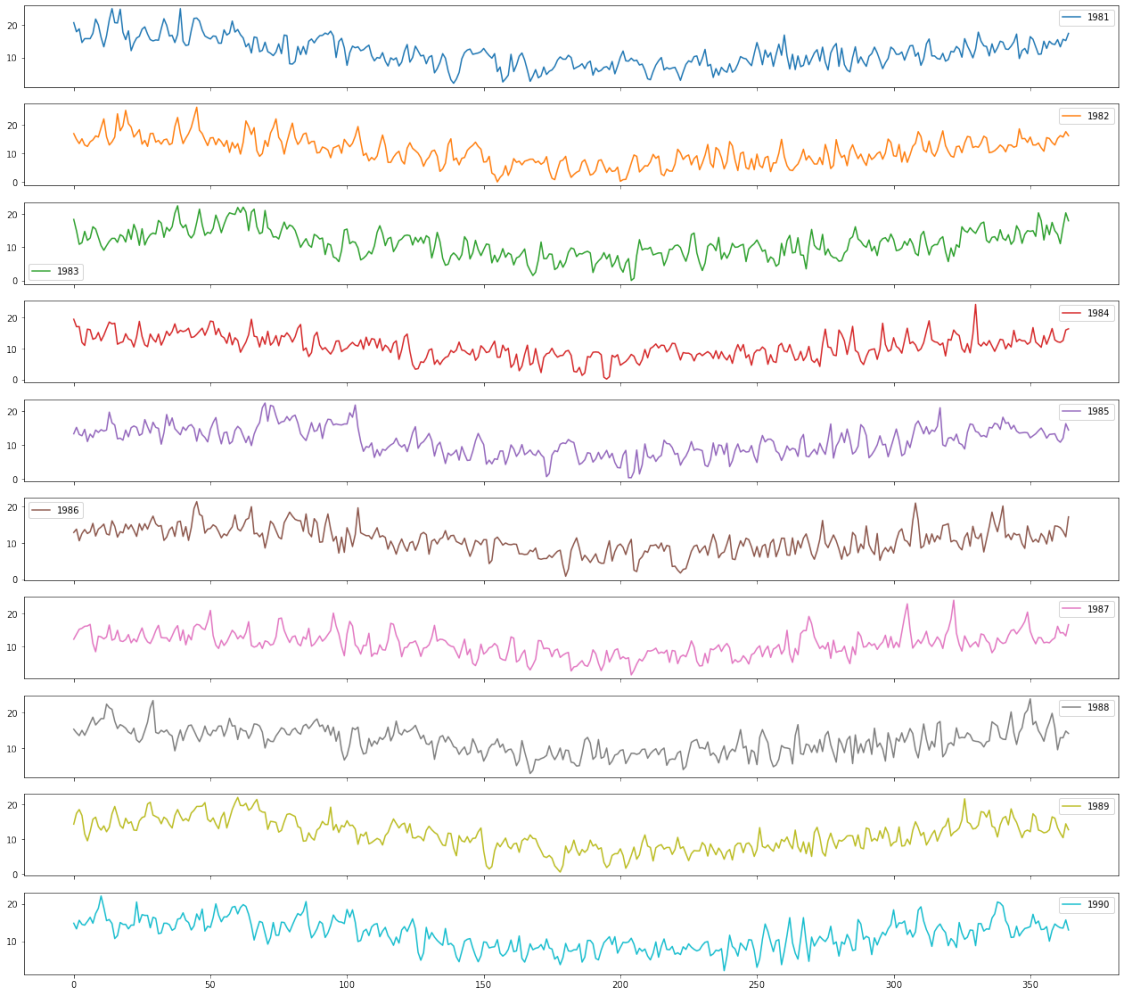
1.6 Grouping and Visualizing time series data

Now, let's group the data by year and create a line plot for each *year* for direct comparison. You can regroup data per year using Pandas' `grouper()` function in conjunction with the `.groupby()` method.

[63]:
```
# Use pandas grouper to group values using annual frequency
year_groups = temp_data.groupby(pd.Grouper(freq = "A"))
```

Rearrange the data so you can create subplots for each year.

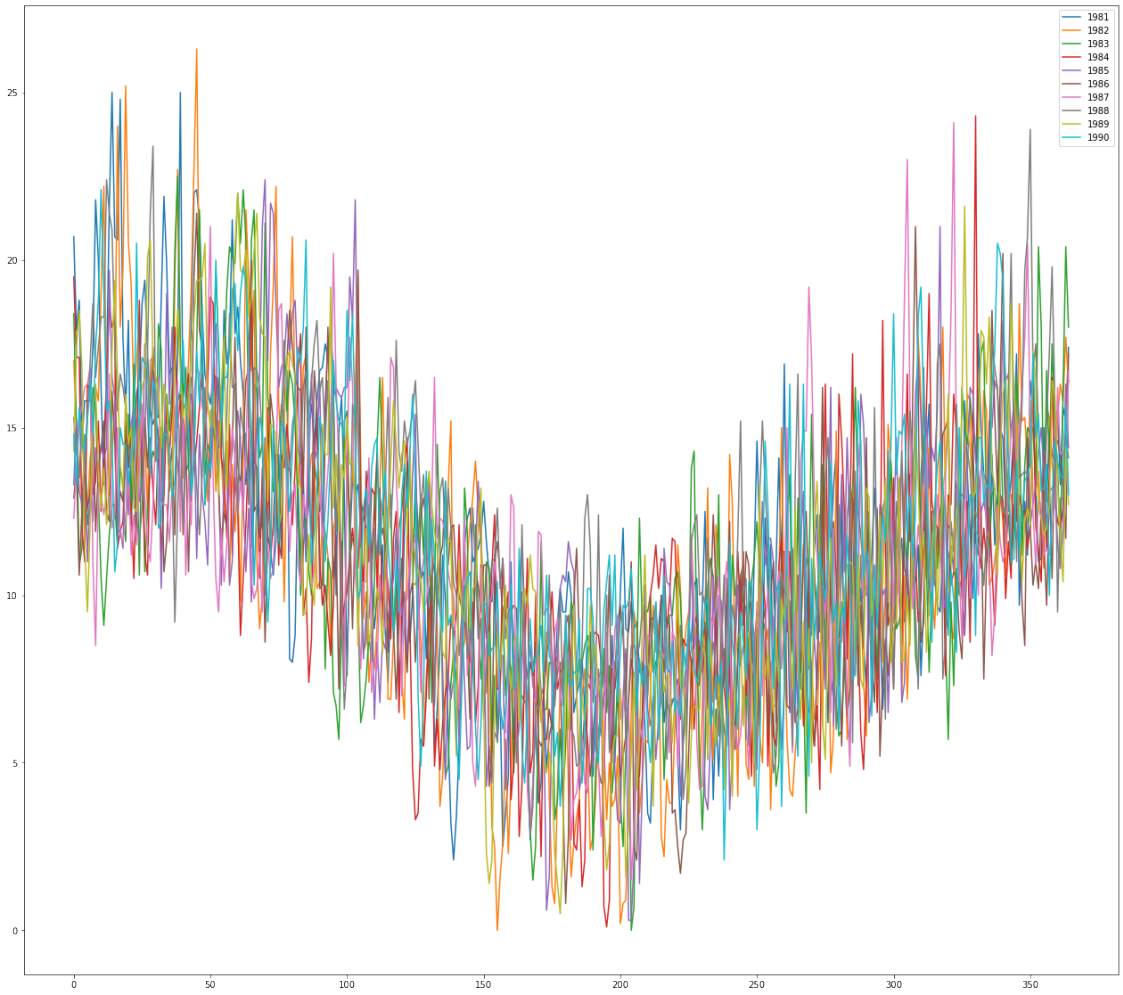
[64]:
```
# Create a new DataFrame and store yearly values in columns
temp_annual = temp_data.groupby(pd.Grouper(freq = "A"))


# Plot the yearly groups as subplots
to_plot = pd.DataFrame()
for yr, val in temp_annual:
    to_plot[yr.year] = val.values.ravel()

to_plot.plot(subplots = True, figsize = (22,20), legend = True);
```



You can see 10 subplots corresponding to the number of columns in your new DataFrame. Each plot is 365 days in length following the annual frequency.

Now, plot all the years on the same graph instead of different subplots.

[65]:
```
# Plot all years on the same graph
to_plot.plot(figsize = (22,20), legend = True);
```

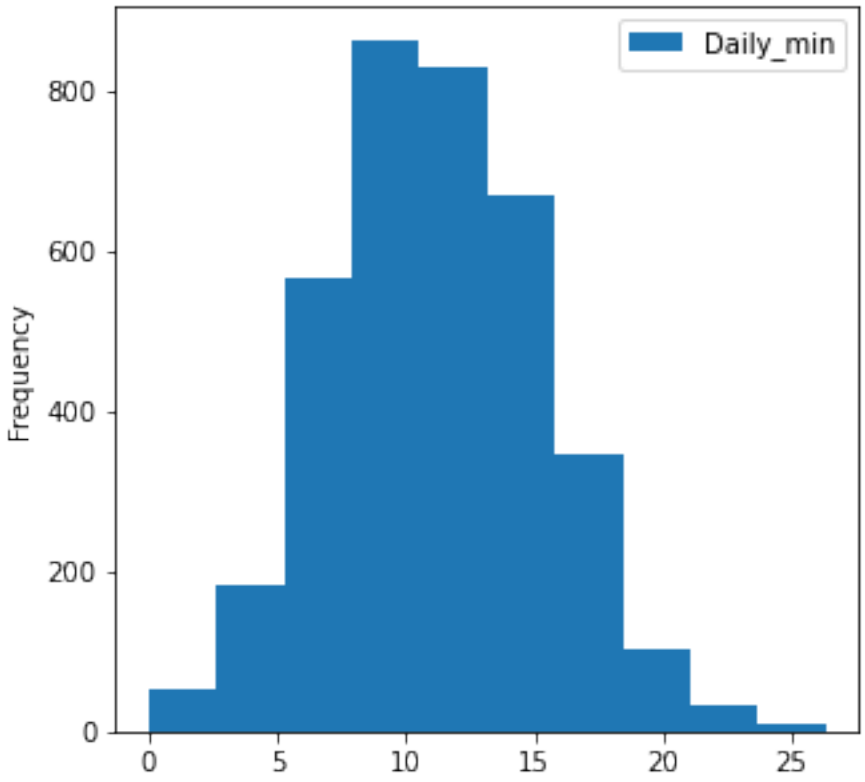


We can see in both plots above that due to the dense nature of time-series (365 values) and a high correlation between the values in different years (i.e. similar temperature values for each year), we can not clearly identify any differences in these groups. However, if you try this on the CO2 dataset used in the last lab, you should be able to see a clear trend showing an increase every year.

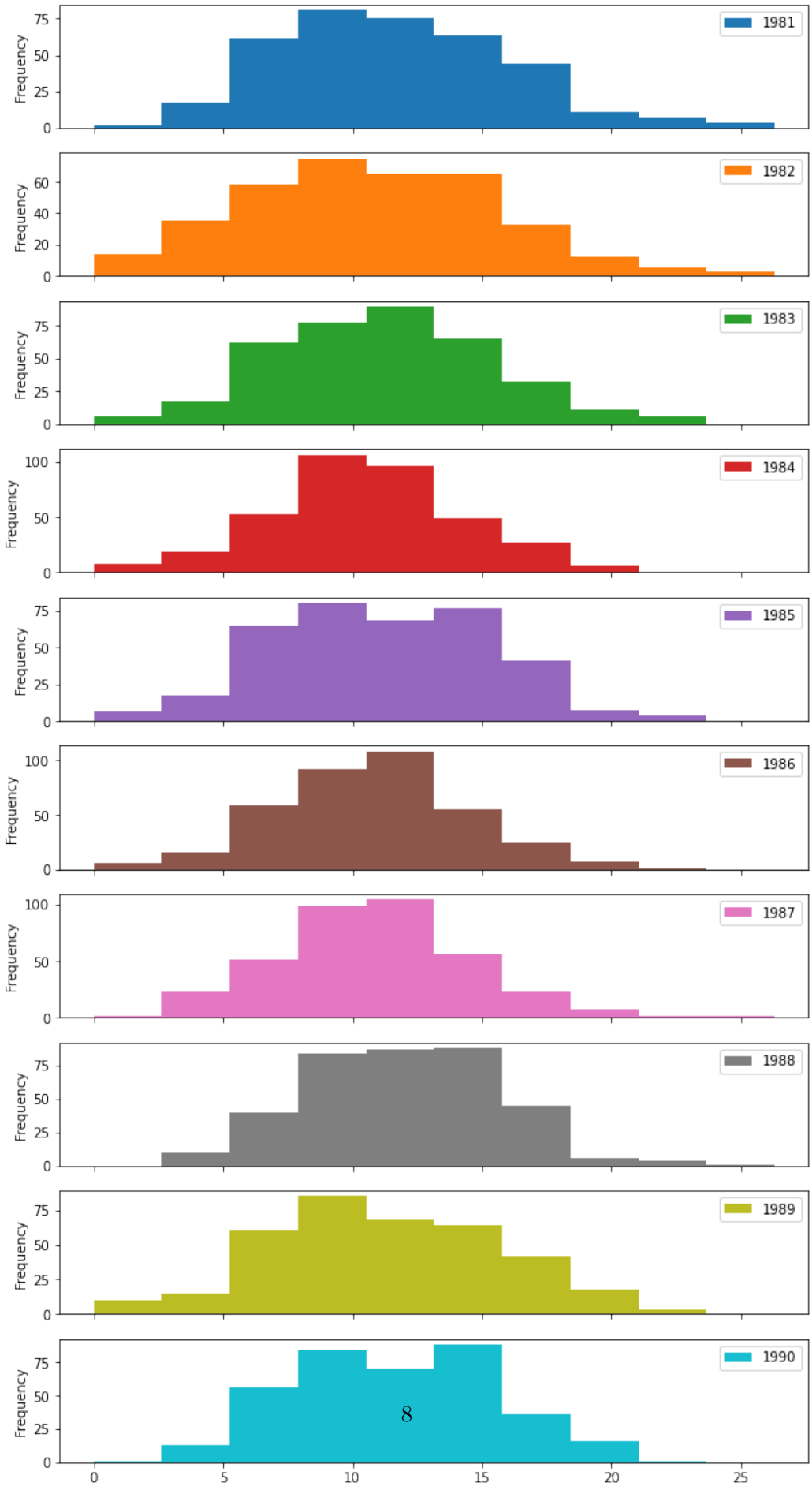
1.7 Time Series Histogram

Create a histogram for your data.

[68]:
```
# Plot a histogram of the temperature dataset
temp_data.plot(kind = "hist", figsize = (5,5));
```



[60]:
```
# Plot a histogram of the temperature dataset
to_plot.plot(kind = "hist", subplots = True, figsize = (10,20));
```

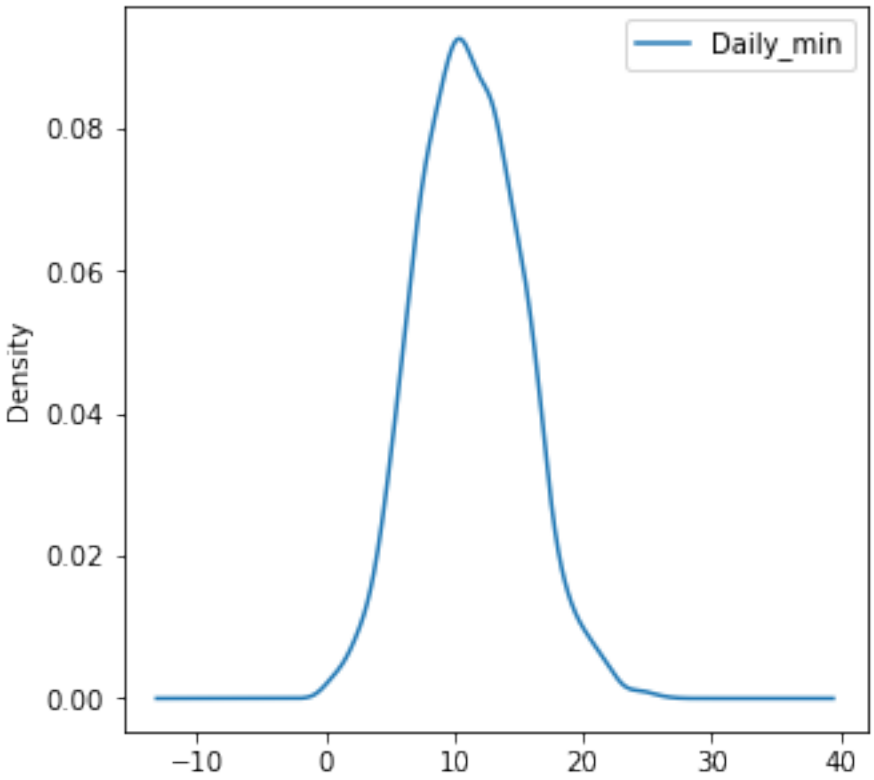

1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
Frequency
0
5
10
15
20
25

The plot shows a distribution that looks strongly Gaussian/Normal.

1.8 Time Series Density Plots

Create a time series density plot

[70]:
```
# Plot a density plot for temperature dataset
temp_data.plot(kind = "kde", figsize = (5,5));
```

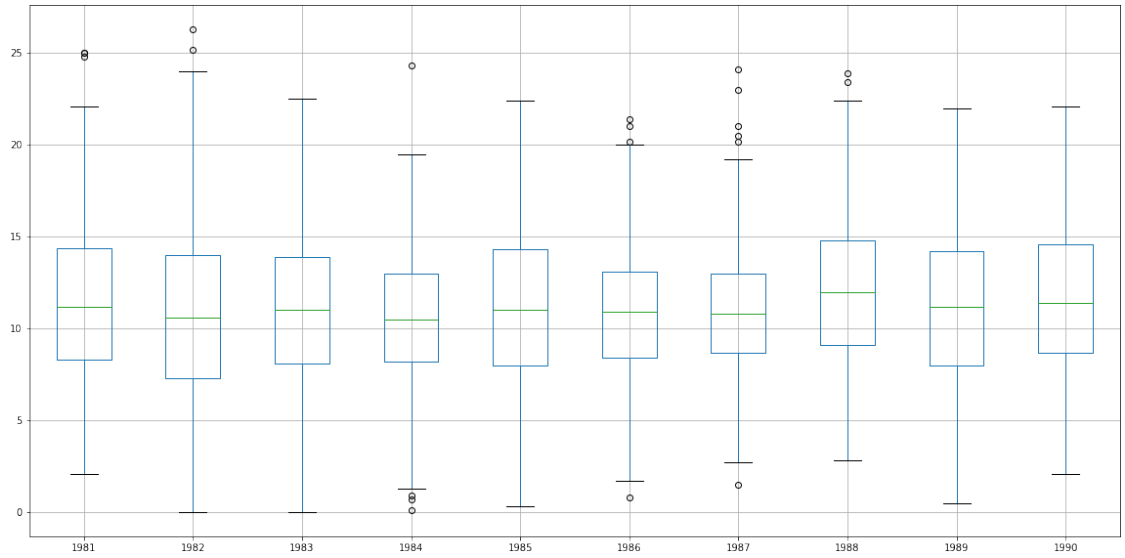


The density plot provides a clearer summary of the distribution of observations. We can see that perhaps the distribution is a little asymmetrical and perhaps a little pointy to be Gaussian.

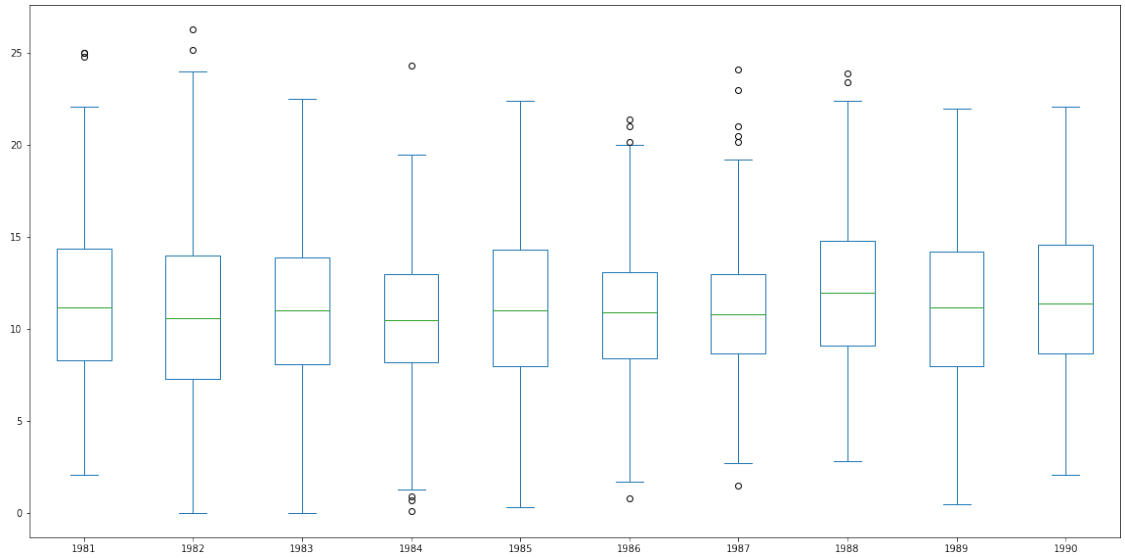
1.9 Time Series Box and Whisker Plots by Interval

Let's use our groups by years to plot a box and whisker plot for each year for direct comparison using the `.boxplot()` method.

[78]:
```
# Generate a box and whiskers plot for temp_annual
to_plot.boxplot(figsize = (20,10));
```



[74]:
```
# Generate a box and whiskers plot for temp_annual
to_plot.plot(kind = "box", figsize = (20,10));
```



In our plot above, we don't see much difference in the mean temperature over years, however, we can spot some outliers showing extremely cold or hot days.

We can also plot distribution across months within each year. Perform the following tasks to achieve this:

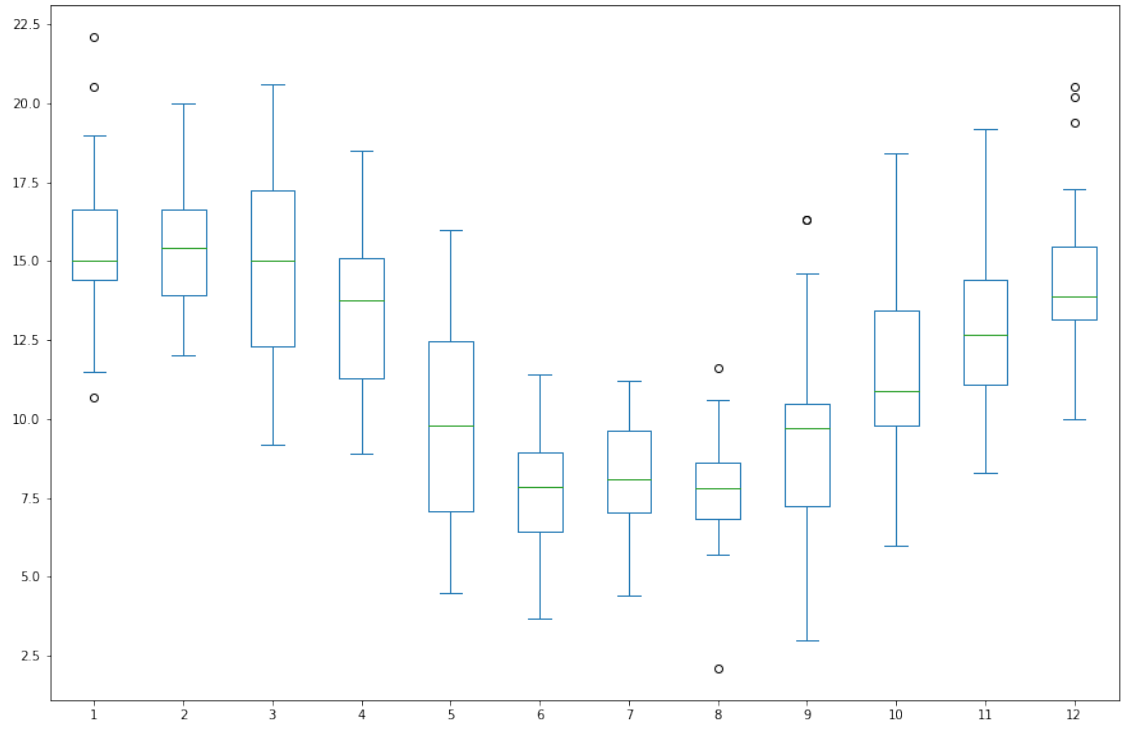
- Extract observations for the year 1990 only, the last year in the dataset
- Group observations by month, and add each month to a new DataFrame as a column

- Create 12 box and whisker plots, one for each month of 1990

[160]:
```
type(groups_monthly)
```

[160]:
```
pandas.core.groupby.generic.DataFrameGroupBy
```

[148]:
```
# Use temp_data to extract values for 1990
yr_1990 = temp_data["1990"]

# Group observations by month
groups_monthly = yr_1990.groupby(pd.Grouper(freq = "M"))

# # Add each month to DataFrame as a column



months_1990 = pd.concat([pd.DataFrame(x[1].values) for x in groups_monthly],␣
 ↪axis = 1)

months_df = pd.DataFrame(months_1990)

# # # Set the column names for each month i.e. 1,2,3, .., 12
months_df.columns = list(range(1, 13))

# # # Plot the box and whiskers plot for each month
months_df.plot(kind = "box", figsize = (15,10));
```



We see 12 box and whisker plots, showing the significant change in the distribution of minimum temperatures across the months of the year from the Southern Hemisphere summer in January to the Southern Hemisphere winter in the middle of the year, and back to summer again.

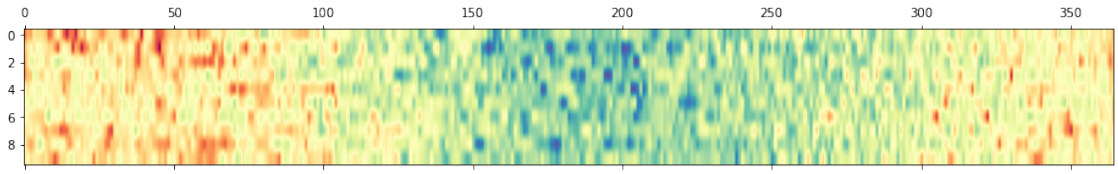
1.10 Time Series Heat Maps

Let's create a heat map of the minimum daily temperatures data.

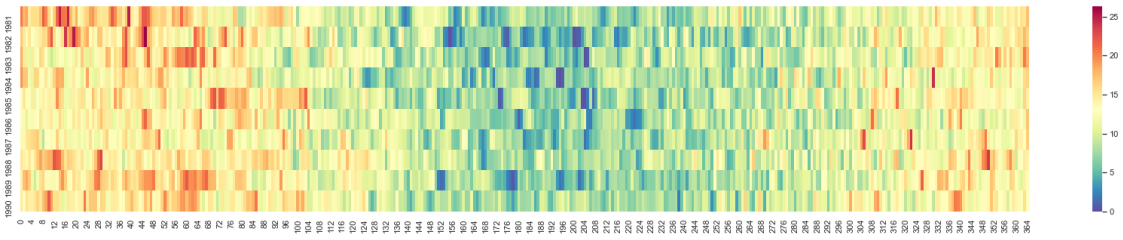
- Rotate (transpose) the `temp_annual` DataFrame so that each row represents one year and each column one day

- Use the `matshow()` function to draw a heat map for transposed yearly matrix

[169]:
```
# Transpose the yearly group DataFrame
year_matrix = to_plot.T

# Draw a heatmap with matshow()
plt.matshow(year_matrix, cmap=plt.cm.Spectral_r, aspect = "auto"
            , interpolation = None);
```



[179]:
```
import seaborn as sns
sns.set(rc={"figure.figsize":(30, 5)})
sns.heatmap(to_plot.T, cmap = plt.cm.Spectral_r);
```



We can now see that the plot shows the cooler minimum temperatures in the middle days of the years and the warmer minimum temperatures in the start and ends of the years, and all the fading and complexity in between.

Following this intuition, let's draw another heatmap comparing the months of the year in 1990. Each column represents one month, with rows representing the days of the month from 1 to 31.

[ ]:
```
# Draw a heatmap comparing the months of the year in 1990
```

The plot shows the same macro trend seen for each year on the zoomed level of month-to-month. We can also see some white patches at the bottom of the plot. This is missing data for those months that have fewer than 31 days, with February being quite an outlier with 28 days in 1990.

1.11 Summary

In this lab, you learned how to explore and better understand a time-series dataset using Pandas. You also learned how to explore the temporal relationships with line, scatter, box and whisper plots, histograms, density plots, and heat maps.