PCA Application (canadian cities dataset)

Data set / Download link

The data set of canadian cities was downloaded from this location:

https://simplemaps.com/data/canada-cities

In [77]:

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

In [78]:

```
# data set
fname = './data/maps/simplemaps_canadacities_basicv1.8/canadacities.csv'

# load data set into data frame
dframe = pd.read_csv(fname)
dframe.head(5)
```

Out[78]:

	city	city_ascii	province_id	province_name	lat	lng	population	dens
0	Toronto	Toronto	ON	Ontario	43.7417	-79.3733	5647656.0	442
1	Montréal	Montreal	QC	Quebec	45.5089	-73.5617	3675219.0	483
2	Vancouver	Vancouver	BC	British Columbia	49.2500	-123.1000	2426160.0	574
3	Calgary	Calgary	AB	Alberta	51.0500	-114.0667	1306784.0	159
4	Edmonton	Edmonton	AB	Alberta	53.5344	-113.4903	1151635.0	132

	city	city_ascii	province_id	province_name	lat	lng	population	dens

For a demonstration of how to use `PCA` for data analysis only the columns on `latitude (lat)` and `longitude (lng)` are required.

These are extracted first.

In [79]:
```
df_lon_lat = dframe.loc[:, ['lng', 'lat']]
df_lon_lat.head(3)
```

Out[79]:

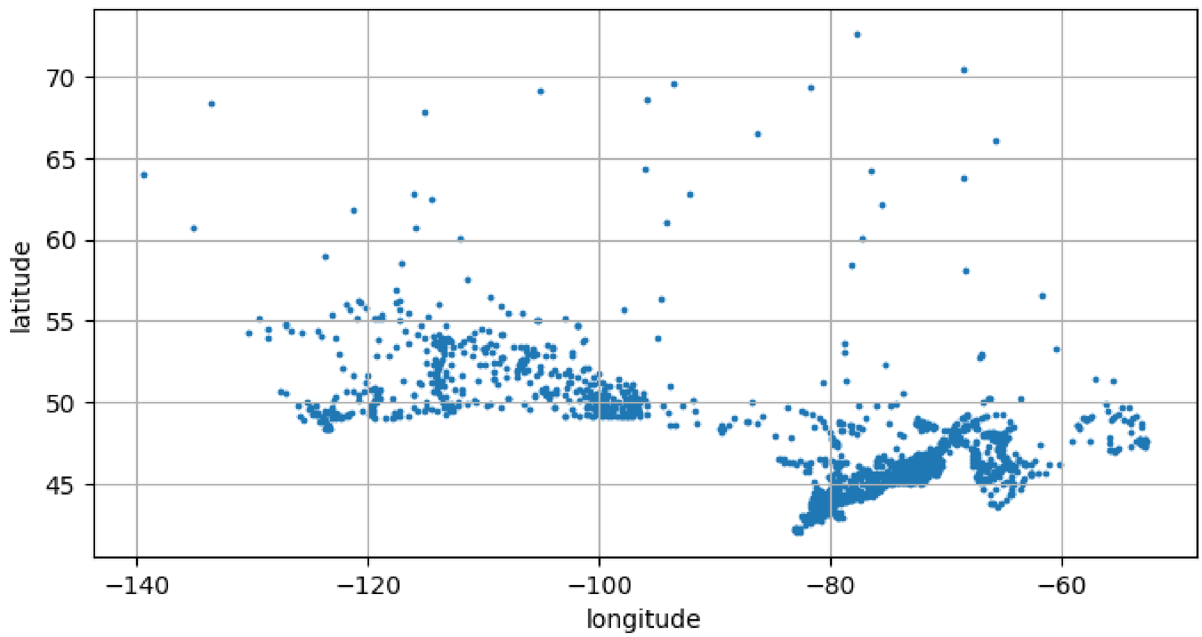
	lng	lat
0	-79.3733	43.7417
1	-73.5617	45.5089
2	-123.1000	49.2500

In [80]:
```
# the same data but this time as a ndarray
lon_lat = df_lon_lat.values[:, :]
print(f"lon_lat.shape  : {lon_lat.shape}")
```

```
lon_lat.shape  : (1737, 2)
```

In [81]:
```
fig1 = plt.figure(1, figsize=[8, 4])
ax_f1 = fig1.add_subplot(1, 1, 1)
ax_f1.scatter(lon_lat[:,0], lon_lat[:,1], s=3)
ax_f1.grid(True)
ax_f1.set_xlabel('longitude')
ax_f1.set_ylabel('latitude');
```

Out[81]:
```
Text(0, 0.5, 'latitude')
```



In [82]:
```
lon = lon_lat[:,0]
lat = lon_lat[:,1]

mean_lon = np.mean(lon)
mean_lat = np.mean(lat)
mean_lon_lat = np.mean(lon * lat)
mean_lon_lon = np.mean(lon * lon)
mean_lat_lat = np.mean(lat * lat)

x11 = mean_lon_lon - mean_lon**2
x22 = mean_lat_lat - mean_lat**2
x12 = mean_lon_lat - mean_lon * mean_lat
x21 = x12

print(f"entries of covariance matrix")
print(f"x11 : {x11}")
print(f"x22 : {x22}")
print(f"x12 : {x12}")
print(f"x21 : {x21}")
```

```
entries of covariance matrix
x11 : 358.09345445689814
x22 : 13.044417240758321
x12 : -39.52986920519061
x21 : -39.52986920519061
```

In [83]:
```
# centered data set
lon_c = lon - mean_lon
lat_c = lat - mean_lat

# build a centered data matrix (will be used later in this  notebook)
lonlat_c = np.column_stack((lon_c, lat_c))

x11 = np.mean(lon_c * lon_c)
x22 = np.mean(lat_c * lat_c)
x12 = np.mean(lon_c * lat_c)
```

```
print(f"x11 : {x11}")
print(f"x22 : {x22}")
print(f"x12 : {x12}")
```

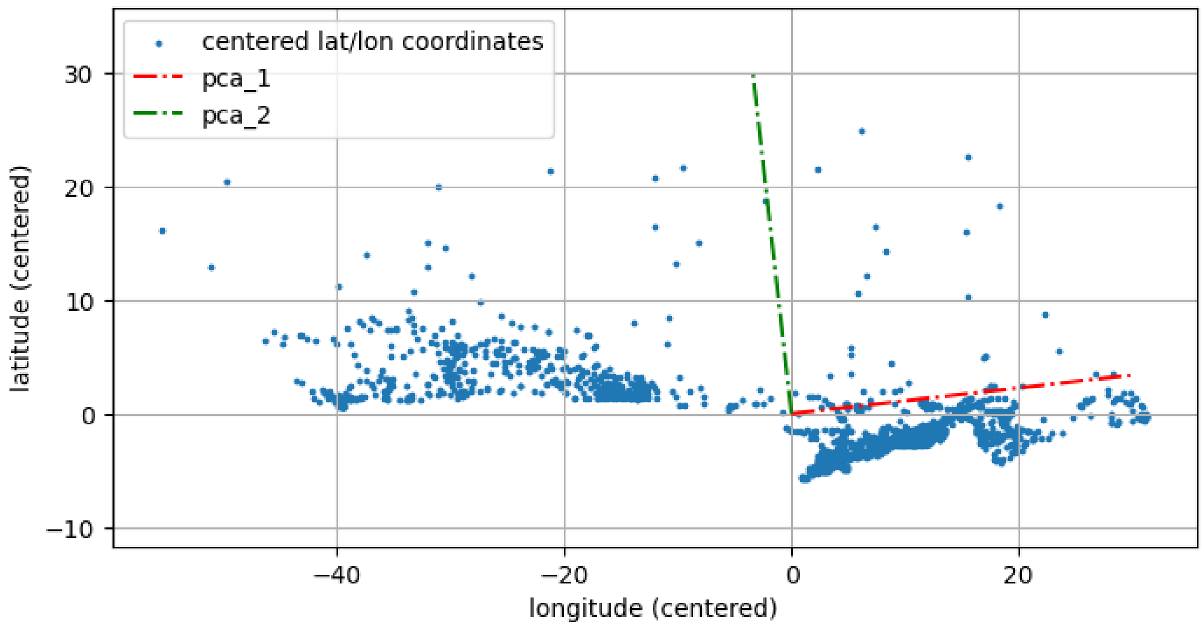
```
x11 : 358.0934544568978
x22 : 13.044417240759058
x12 : -39.529869205191034
```

In [84]:
```
# create covariance matrix and compute the eigen decomposition
Xmat = np.array([[x11, x12], [x12, x22]])
eigen_values, Vmat_t = np.linalg.eig(Xmat)

print(f"eigen_values  :\n{eigen_values}\n")
print(f"Vmat_t        :\n{Vmat_t}\n")
```

```
eigen_values  :
[362.56419025   8.57368145]

Vmat_t        :
[[ 0.99366517  0.11238121]
 [-0.11238121  0.99366517]]
```

In [85]:
```
fig2 = plt.figure(2, figsize=[8, 4])
ax_f2 = fig2.add_subplot(1, 1, 1)

# must preserve aspect ratio if orthogonal eigenvectors are displayed correctly ...
ax_f2.axis('equal')

ax_f2.scatter(lon_c, lat_c, s=3, label='centered lat/lon coordinates')
ax_f2.plot([0, 30*Vmat_t[0, 0]], [0, 30*Vmat_t[0, 1]], c='r', linestyle='-.', label
ax_f2.plot([0, 30*Vmat_t[1, 0]], [0, 30*Vmat_t[1, 1]], c='g', linestyle='-.', label
ax_f2.legend()
ax_f2.grid(True)
ax_f2.set_xlabel('longitude (centered)')
ax_f2.set_ylabel('latitude (centered)');
```

Out[85]:
```
Text(0, 0.5, 'latitude (centered)')
```

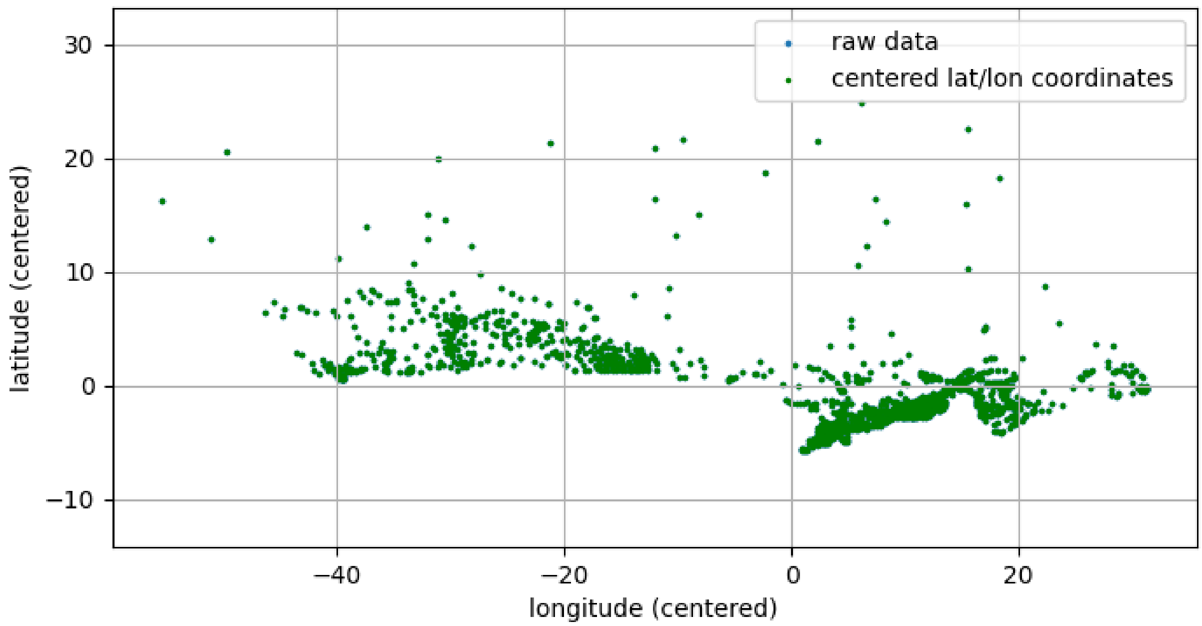


In [86]:
```
# projections on pca1

proj_pca1 = lonlat_c @ (Vmat_t[0, :]).T
pca1 = np.column_stack((proj_pca1, proj_pca1)) * Vmat_t[0, :]
proj_pca2 = lonlat_c @ (Vmat_t[1, :]).T
pca2 = np.column_stack((proj_pca2, proj_pca2)) * Vmat_t[1, :]
```

In [87]:
```
result = pca1 + pca2
```

In [88]:
```
fig3 = plt.figure(3, figsize=[8, 4])
ax_f3 = fig3.add_subplot(1, 1, 1)

# must preserve aspect ratio if orthogonal eigenvectors are displayed correctly ...
ax_f3.axis('equal')

ax_f3.scatter(lon_c, lat_c, s=3, label='raw data')
ax_f3.scatter(result[:, 0], result[:, 1], s=2, label='centered lat/lon coordinates'

ax_f3.legend()
ax_f3.grid(True)
ax_f3.set_xlabel('longitude (centered)')
ax_f3.set_ylabel('latitude (centered)');
```

Out[88]:
```
Text(0, 0.5, 'latitude (centered)')
```


raw data
centered lat/lon coordinates
latitude (centered)
longitude (centered)
30
20
10
0
−10
−40
−20
0
20

In [ ]: