visualization_with_Matplotlib_L2

October 18, 2023

1 Visualization with matplotlib II

1.1 Content:

- Introduction to part II
- Read data from CSV file
- Add a legend
- Read data from netCDF file
- Create a contour plot
- Create a filled contour plot
- Add a colorbar
- Create a vector plot
- Zoom into a map
- Map projections

1.2 Introduction to part II

In the second part of the introduction to visualization with Matplotlib we will show you how to read and display different types of data sets. We start with the 'simple' CSV datasets and then move on to 2D georeferenced data in netCDF format.

[1]:
```
%matplotlib inline
```

1.3 Read data from CSV file

From DWD we downloaded the CSV data set of yearly average temperature for Germany. It contains the data for Germany and each German state in the time range 1951 to 2020.

```
Zeitreihen fuer Gebietsmittel fuer Bundeslaender und Kombinationen von Bundeslaender, erstellt
Jahr;Jahr;Brandenburg/Berlin;Brandenburg;Baden-Wuerttemberg;Bayern;Hessen;Mecklenburg-Vorpomme
Niedersachsen;Niedersachsen/Hamburg/Bremen;Nordrhein-Westfalen;Rheinland-Pfalz;Schleswig-Holst
Sachsen;Sachsen-Anhalt;Thueringen/Sachsen-Anhalt;Thueringen;Deutschland;
1951;year;  5.15;  5.13;  3.52;  3.47;  3.42;  2.21;  1.71;  1.71;  1.73;  1.97;  0.38; ... 3.
1952;year;  5.46;  5.48; 14.31; 12.27; 10.37;  1.64;  2.55;  2.52;  6.71; 10.95;  0.82; ... 7.
1953;year;  0.51; 10.43;  4.93;  3.34;  6.44;  3.91;  4.08;  4.09;  3.73;  6.21;  1.88; ... 5.
1954;year;  5.16;  5.14;  2.43;  1.83;  2.98;  1.88;  1.52;  1.51;  1.37;  3.02;  0.69; ... 2.
```

```
1955;year;  1.14;  1.12;  1.19;  0.72;  1.54;  0.24;  0.48;  0.47;  1.21;  1.50;  0.02;  ... 0.9
1956;year;  0.81;  0.82;  1.13;  0.39;  0.66;  0.26;  0.25;  0.25;  0.41;  0.87;  0.02;  ... 0.5
1957;year;  8.10;  8.10;  7.13;  7.01;  8.32;  3.48;  5.39;  5.37;  8.14;  7.84;  2.07;  ... 6.8
1958;year;  2.61;  2.60;  2.91;  2.51;  0.97;  0.26;  0.25;  0.24;  0.73;  1.05;  0.00;  ... 1.5
1959;year;  6.76;  6.77;  5.89;  3.95;  7.67;  4.31;  5.59;  5.59;  6.78;  9.03;  2.66;  ... 5.6
1960;year;  3.06;  3.04;  1.25;  1.47;  0.96;  0.46;  0.61;  0.61;  0.65;  0.77;  0.08;  ... 1.3
...
```

To import the CSV data we use the Pandas package. Pandas provide a CSV reader function which makes it easy to read the data from file. The column names can be used directly to select the favored data. In this case, we want to display the data for Hamburg and Bavaria over all years.

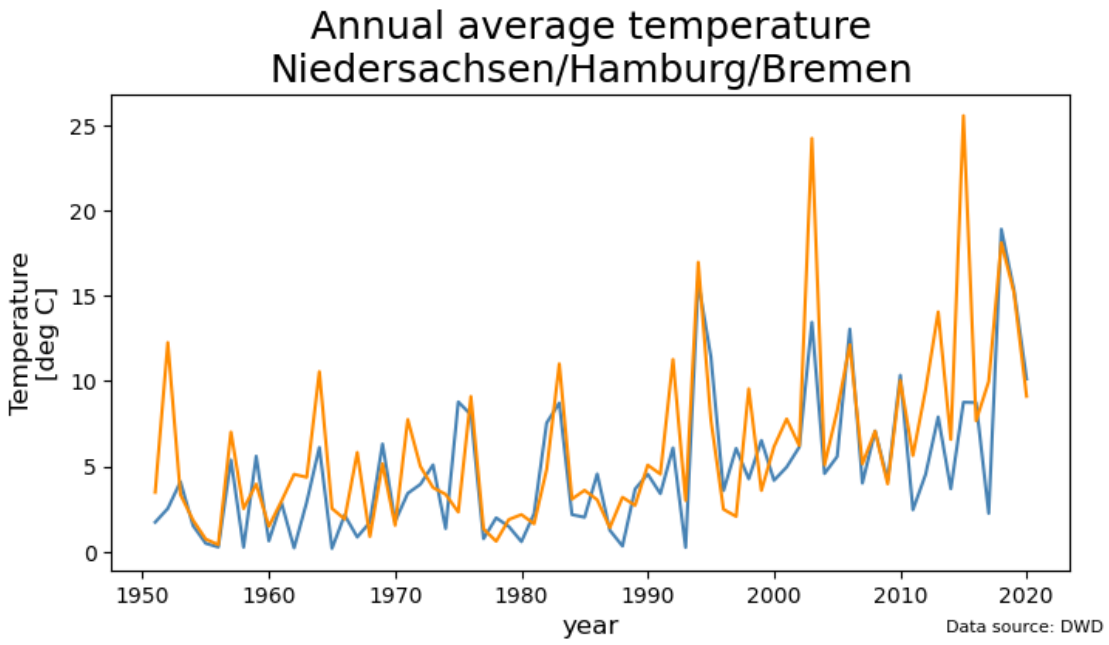
> **Note:**
>
> Make sure that the column name is correct, because in this case Hamburg does not exist alone, but in connection with Niedersachsen and Bremen (Niedersachsen/Hamburg/Bremen).

A problem is the '/' character in the column name, because in this case it would lead to an error in Python. To avoid this we define a variable named *state*, which gets the string 'Niedersachsen/Hamburg/Bremen'.

[2]:
```
import matplotlib.pyplot as plt
import pandas as pd
```

[3]:
```
data_file = '../data/DWD_regional_averages_txbs_year_hot_days.txt'

state = 'Niedersachsen/Hamburg/Bremen'

data = pd.read_csv(data_file, header=1, sep=';')
```

[4]:
```
fig, ax = plt.subplots(figsize=(8,4))

ax.set_title('Annual average temperature\n'+state, fontsize=18)
ax.set_ylabel('Temperature\n[deg C]', fontsize=12)
ax.set_xlabel('year', fontsize=12)
ax.tick_params(labelsize=10)

plot = ax.plot(data.Jahr, data[state], color='steelblue')

plt.text(0.8, 0.01, 'Data source: DWD', fontsize=8, transform=plt.gcf().
 ↪transFigure)

plot = ax.plot(data.Jahr, data['Bayern'], color='darkorange')
```

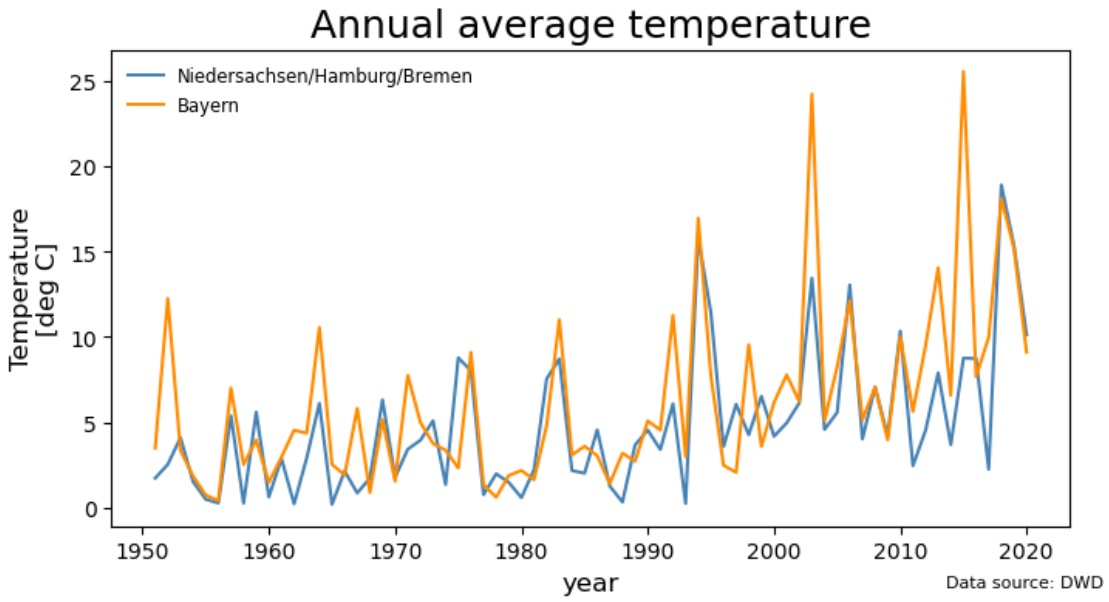


1.4 Add a legend

This looks quite nice, but a legend is still missing. Therefore, a **label** has to be added to each plot call and the `legend` fuction from pyplot is used to generate a legend box. By default, the legend is placed in a location where it does not overlap the plot, that can be changed by specifying the `loc` parameter.

[5]:
```
fig, ax = plt.subplots(figsize=(8,4))

ax.set_title('Annual average temperature', fontsize=18)
ax.set_ylabel('Temperature\n[deg C]', fontsize=12)
ax.set_xlabel('year', fontsize=12)
ax.tick_params(labelsize=10)

ax.plot(data.Jahr, data[state], color='steelblue', label=state)

ax.plot(data.Jahr, data['Bayern'], color='darkorange', label='Bayern')

#legend = ax.legend()

#legend = ax.legend(loc='lower right', shadow=True, fontsize='small',
#                   facecolor='lightgray', edgecolor='black')

legend = ax.legend(fontsize='small',
                   edgecolor='None')
```

```
tx = plt.text(0.8, 0.01, 'Data source: DWD', fontsize=8, transform=plt.gcf().
 ↪transFigure)
```

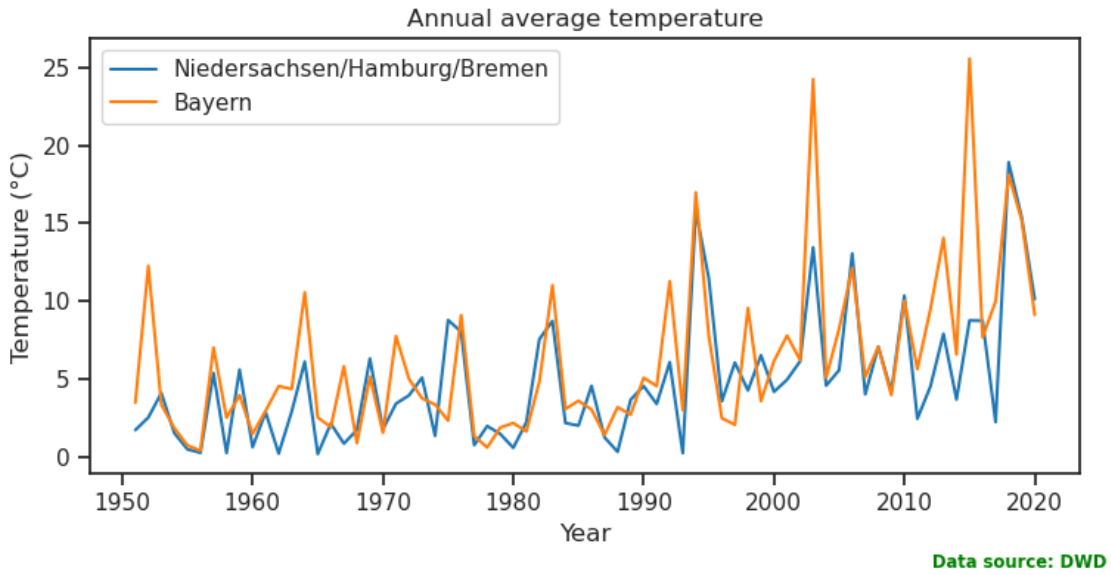


1.5 Same plot with seaborn

We can repeat the same plot using the `seaborn` library. Seaborn works well together with the Pandas library that we used to load the csv file. Notice that the seaborn lineplot function takes as arguments the data, as well as the columns to use for x/y. The legend is automatically generated when labels are provided. We can also control the parameters of the annotated text using keywords for alignment, font weight, and font size.

[6]:
```
import seaborn as sns

sns.set_style('ticks')
sns.set_context('notebook')

fig, ax = plt.subplots(figsize=(8,4))

sns.lineplot(data=data, x='Jahr', y='Niedersachsen/Hamburg/Bremen',␣
 ↪label='Niedersachsen/Hamburg/Bremen', ax=ax)
sns.lineplot(data=data, x='Jahr', y='Bayern', label='Bayern', ax=ax)
ax.set_title('Annual average temperature')
ax.set_ylabel('Temperature (°C)')
ax.set_xlabel('Year')
```

```
plt.text(0.95, 0.0, 'Data source: DWD', transform=plt.gcf().transFigure,␣
 ↪horizontalalignment='right', verticalalignment='bottom', fontsize='x-small',␣
 ↪fontweight='bold', color='green')

fig.tight_layout()
```



Exercise

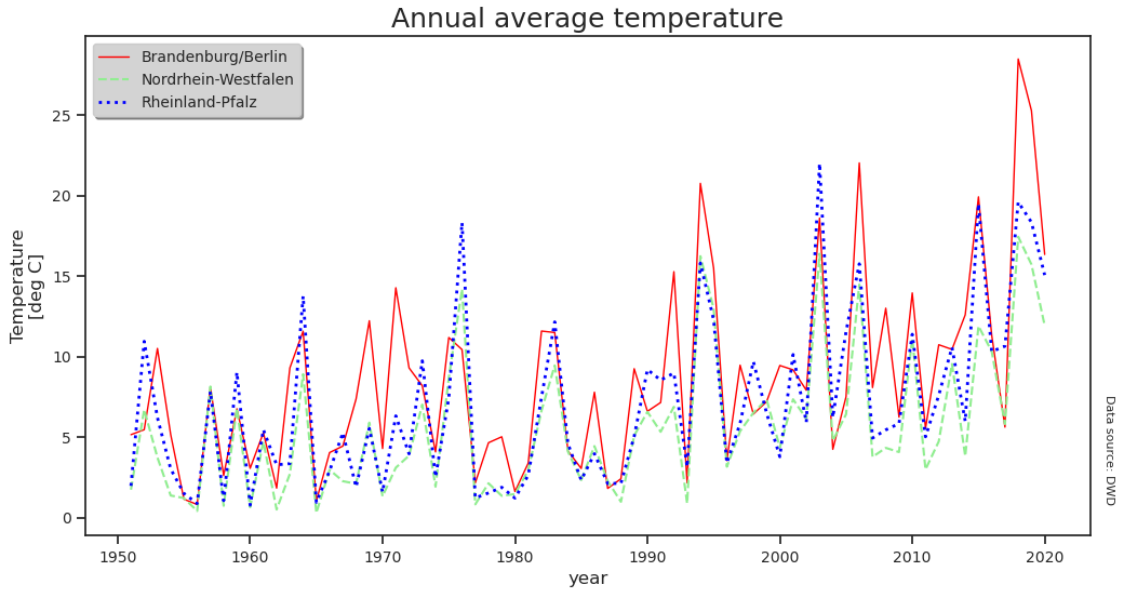
Plot the data for the states - 'Brandenburg/Berlin' (color red, line style solid, line width 1) - 'Nordrhein-Westfalen' (color light green, line style dashed, line width 1.5) and - 'Rheinland-Pfalz' (color blue, line style dotted, line width 2)

Add a legend to the plot and don't forget to add the data source but this time on the vertical right.

[ ]:

[7]:
```
fig, ax = plt.subplots(figsize=(12,6))

ax.set_title('Annual average temperature', fontsize=18)
ax.set_ylabel('Temperature\n[deg C]', fontsize=12)
ax.set_xlabel('year', fontsize=12)
ax.tick_params(labelsize=10)

ax.plot(data.Jahr, data['Brandenburg/Berlin'],
        color='red',
        linestyle='-',
        linewidth=1,
        label='Brandenburg/Berlin')
```

```
ax.plot(data.Jahr, data['Nordrhein-Westfalen'],
        color='lightgreen',
        linestyle='--',
        linewidth=1.5,
        label='Nordrhein-Westfalen')

ax.plot(data.Jahr, data['Rheinland-Pfalz'],
        color='blue',
        linestyle=':',
        linewidth=2,
        label='Rheinland-Pfalz')

legend = ax.legend(loc='upper left', shadow=True, fontsize='small')
legend.get_frame().set_facecolor('lightgray')

tx = plt.text(0.91, 0.16, 'Data source: DWD', rotation=270,
              fontsize=8, transform=plt.gcf().transFigure)
```



1.6 Read data from a netCDF file

Often the data is stored in a netCDF file, which can be read using the Xarray package.

`Input file: ../data/rectilinear_grid_2D.nc`

[8]:
```
import xarray as xr

data_file = '../data/rectilinear_grid_2D.nc'

ds = xr.open_dataset(data_file)

print(ds)
```

```
<xarray.Dataset>
Dimensions:  (lon: 192, lat: 96, time: 40)
Coordinates:
  * lon      (lon) float64 -180.0 -178.1 -176.2 -174.4 … 174.4 176.2 178.1
  * lat      (lat) float64 88.57 86.72 84.86 83.0 … -83.0 -84.86 -86.72 -88.57
  * time     (time) datetime64[ns] 2001-01-01 … 2001-01-10T18:00:00
Data variables:
    tsurf    (time, lat, lon) float32 …
    precip   (time, lat, lon) float32 …
    u10      (time, lat, lon) float32 …
    v10      (time, lat, lon) float32 …
    qvi      (time, lat, lon) float32 …
    slp      (time, lat, lon) float32 …
```

[9]:
```
print(ds.lon)
```

```
<xarray.DataArray 'lon' (lon: 192)>
array([-180.   , -178.125, -176.25 , -174.375, -172.5  , -170.625, -168.75 ,
       -166.875, -165.   , -163.125, -161.25 , -159.375, -157.5  , -155.625,
       -153.75 , -151.875, -150.   , -148.125, -146.25 , -144.375, -142.5  ,
       -140.625, -138.75 , -136.875, -135.   , -133.125, -131.25 , -129.375,
       -127.5  , -125.625, -123.75 , -121.875, -120.   , -118.125, -116.25 ,
       -114.375, -112.5  , -110.625, -108.75 , -106.875, -105.   , -103.125,
       -101.25 ,  -99.375,  -97.5  ,  -95.625,  -93.75 ,  -91.875,  -90.   ,
        -88.125,  -86.25 ,  -84.375,  -82.5  ,  -80.625,  -78.75 ,  -76.875,
        -75.   ,  -73.125,  -71.25 ,  -69.375,  -67.5  ,  -65.625,  -63.75 ,
        -61.875,  -60.   ,  -58.125,  -56.25 ,  -54.375,  -52.5  ,  -50.625,
        -48.75 ,  -46.875,  -45.   ,  -43.125,  -41.25 ,  -39.375,  -37.5  ,
        -35.625,  -33.75 ,  -31.875,  -30.   ,  -28.125,  -26.25 ,  -24.375,
        -22.5  ,  -20.625,  -18.75 ,  -16.875,  -15.   ,  -13.125,  -11.25 ,
         -9.375,   -7.5  ,   -5.625,   -3.75 ,   -1.875,    0.   ,    1.875,
          3.75 ,    5.625,    7.5  ,    9.375,   11.25 ,   13.125,   15.   ,
         16.875,   18.75 ,   20.625,   22.5  ,   24.375,   26.25 ,   28.125,
         30.   ,   31.875,   33.75 ,   35.625,   37.5  ,   39.375,   41.25 ,
         43.125,   45.   ,   46.875,   48.75 ,   50.625,   52.5  ,   54.375,
         56.25 ,   58.125,   60.   ,   61.875,   63.75 ,   65.625,   67.5  ,
         69.375,   71.25 ,   73.125,   75.   ,   76.875,   78.75 ,   80.625,
         82.5  ,   84.375,   86.25 ,   88.125,   90.   ,   91.875,   93.75 ,
         95.625,   97.5  ,   99.375,  101.25 ,  103.125,  105.   ,  106.875,
        108.75 ,  110.625,  112.5  ,  114.375,  116.25 ,  118.125,  120.   ,
```

```
        121.875,  123.75 ,  125.625,  127.5  ,  129.375,  131.25 ,  133.125,
        135.   ,  136.875,  138.75 ,  140.625,  142.5  ,  144.375,  146.25 ,
        148.125,  150.   ,  151.875,  153.75 ,  155.625,  157.5  ,  159.375,
        161.25 ,  163.125,  165.   ,  166.875,  168.75 ,  170.625,  172.5  ,
        174.375,  176.25 ,  178.125])
Coordinates:
  * lon      (lon) float64 -180.0 -178.1 -176.2 -174.4 … 174.4 176.2 178.1
Attributes:
    standard_name:  longitude
    long_name:      longitude
    units:          degrees_east
    axis:           X
```

[10]:
```
print(ds.lat)
```

```
<xarray.DataArray 'lat' (lat: 96)>
array([ 88.572169,  86.722531,  84.86197 ,  82.998942,  81.134977,  79.270559,
        77.405888,  75.541061,  73.676132,  71.811132,  69.946081,  68.080991,
        66.215872,  64.35073 ,  62.485571,  60.620396,  58.755209,  56.890013,
        55.024808,  53.159595,  51.294377,  49.429154,  47.563926,  45.698694,
        43.833459,  41.96822 ,  40.102979,  38.237736,  36.372491,  34.507243,
        32.641994,  30.776744,  28.911492,  27.046239,  25.180986,  23.315731,
        21.450475,  19.585219,  17.719962,  15.854704,  13.989446,  12.124187,
        10.258928,   8.393669,   6.528409,   4.66315 ,   2.79789 ,   0.93263 ,
        -0.93263 ,  -2.79789 ,  -4.66315 ,  -6.528409,  -8.393669, -10.258928,
       -12.124187, -13.989446, -15.854704, -17.719962, -19.585219, -21.450475,
       -23.315731, -25.180986, -27.046239, -28.911492, -30.776744, -32.641994,
       -34.507243, -36.372491, -38.237736, -40.102979, -41.96822 , -43.833459,
       -45.698694, -47.563926, -49.429154, -51.294377, -53.159595, -55.024808,
       -56.890013, -58.755209, -60.620396, -62.485571, -64.35073 , -66.215872,
       -68.080991, -69.946081, -71.811132, -73.676132, -75.541061, -77.405888,
       -79.270559, -81.134977, -82.998942, -84.86197 , -86.722531, -88.572169])
Coordinates:
  * lat      (lat) float64 88.57 86.72 84.86 83.0 … -83.0 -84.86 -86.72 -88.57
Attributes:
    standard_name:  latitude
    long_name:      latitude
    units:          degrees_north
    axis:           Y
```

[11]:
```
print(ds.time)
```

```
<xarray.DataArray 'time' (time: 40)>
array(['2001-01-01T00:00:00.000000000', '2001-01-01T06:00:00.000000000',
       '2001-01-01T12:00:00.000000000', '2001-01-01T18:00:00.000000000',
       '2001-01-02T00:00:00.000000000', '2001-01-02T06:00:00.000000000',
       '2001-01-02T12:00:00.000000000', '2001-01-02T18:00:00.000000000',
       '2001-01-03T00:00:00.000000000', '2001-01-03T06:00:00.000000000',
```

```
       '2001-01-03T12:00:00.000000000', '2001-01-03T18:00:00.000000000',
       '2001-01-04T00:00:00.000000000', '2001-01-04T06:00:00.000000000',
       '2001-01-04T12:00:00.000000000', '2001-01-04T18:00:00.000000000',
       '2001-01-05T00:00:00.000000000', '2001-01-05T06:00:00.000000000',
       '2001-01-05T12:00:00.000000000', '2001-01-05T18:00:00.000000000',
       '2001-01-06T00:00:00.000000000', '2001-01-06T06:00:00.000000000',
       '2001-01-06T12:00:00.000000000', '2001-01-06T18:00:00.000000000',
       '2001-01-07T00:00:00.000000000', '2001-01-07T06:00:00.000000000',
       '2001-01-07T12:00:00.000000000', '2001-01-07T18:00:00.000000000',
       '2001-01-08T00:00:00.000000000', '2001-01-08T06:00:00.000000000',
       '2001-01-08T12:00:00.000000000', '2001-01-08T18:00:00.000000000',
       '2001-01-09T00:00:00.000000000', '2001-01-09T06:00:00.000000000',
       '2001-01-09T12:00:00.000000000', '2001-01-09T18:00:00.000000000',
       '2001-01-10T00:00:00.000000000', '2001-01-10T06:00:00.000000000',
       '2001-01-10T12:00:00.000000000', '2001-01-10T18:00:00.000000000'],
      dtype='datetime64[ns]')
Coordinates:
  * time     (time) datetime64[ns] 2001-01-01 … 2001-01-10T18:00:00
Attributes:
    standard_name:  time
```

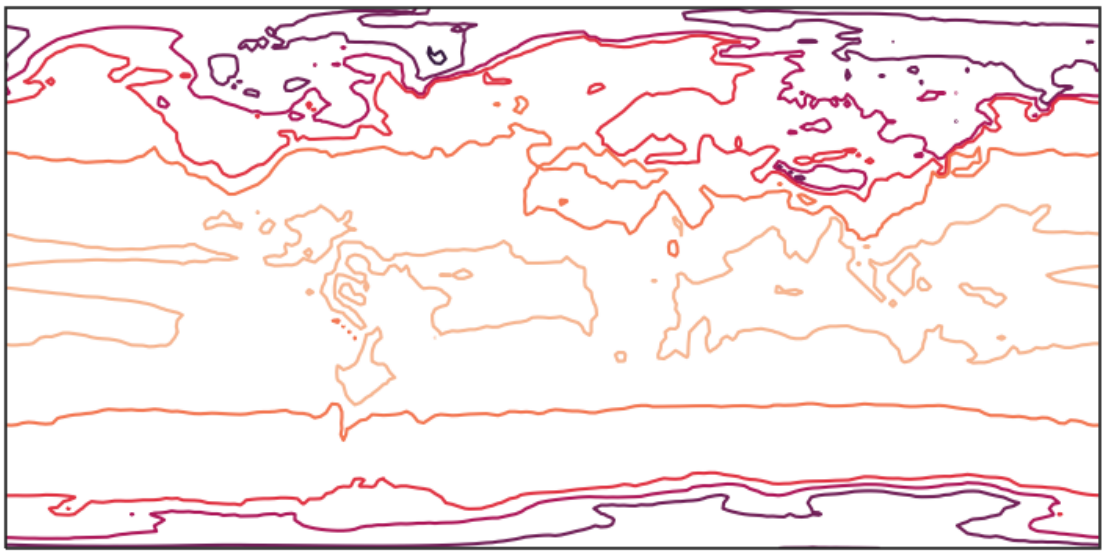
1.7 Create a contour plot

Once we have read the data we can create a contour plot for the variable tas using the plot function contour. Since we want to use maps with different projections in the following examples, we import the Cartopy package (https://scitools.org.uk/cartopy/docs/latest/).

[12]:
```
import cartopy.crs as ccrs
```

The first example uses only the default settings and a cylindrical equidistant projection called PlateCarree ().

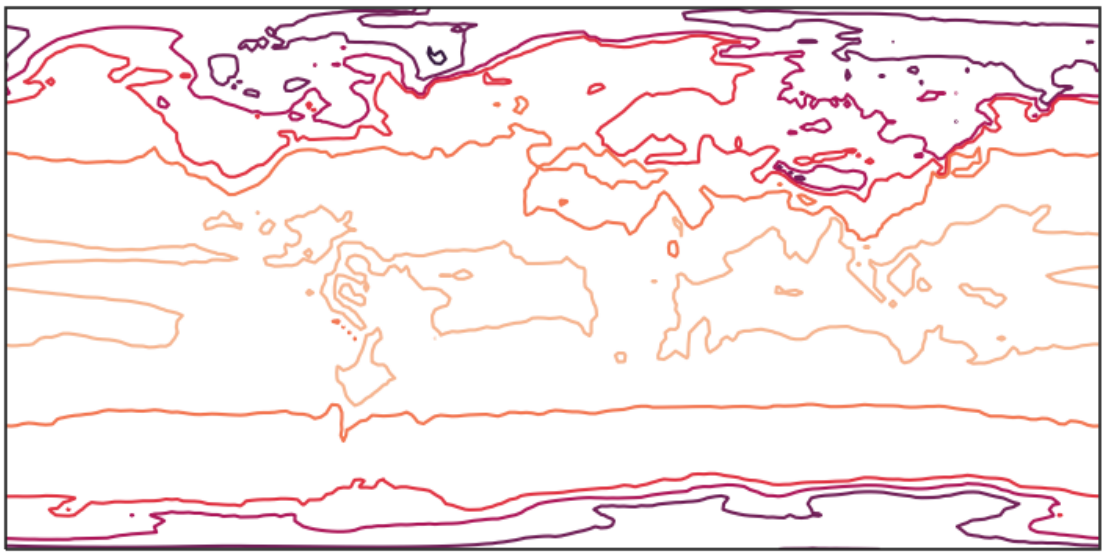
[13]:
```
fig = plt.figure(figsize=(9,6))
ax = plt.axes(projection=ccrs.PlateCarree())

cn = ax.contour(ds.lon, ds.lat, ds.tsurf[0,:,:], transform=ccrs.PlateCarree())
```



Another way to create the axis in one line of code:

[14]:
```
fig, ax = plt.subplots(figsize=(9,6), subplot_kw={"projection": ccrs.
↪PlateCarree()})

ax.contour(ds.lon, ds.lat, ds.tsurf[0,:,:], transform=ccrs.PlateCarree())
```

[14]:
```
<cartopy.mpl.contour.GeoContourSet at 0x7fff913a44f0>
```

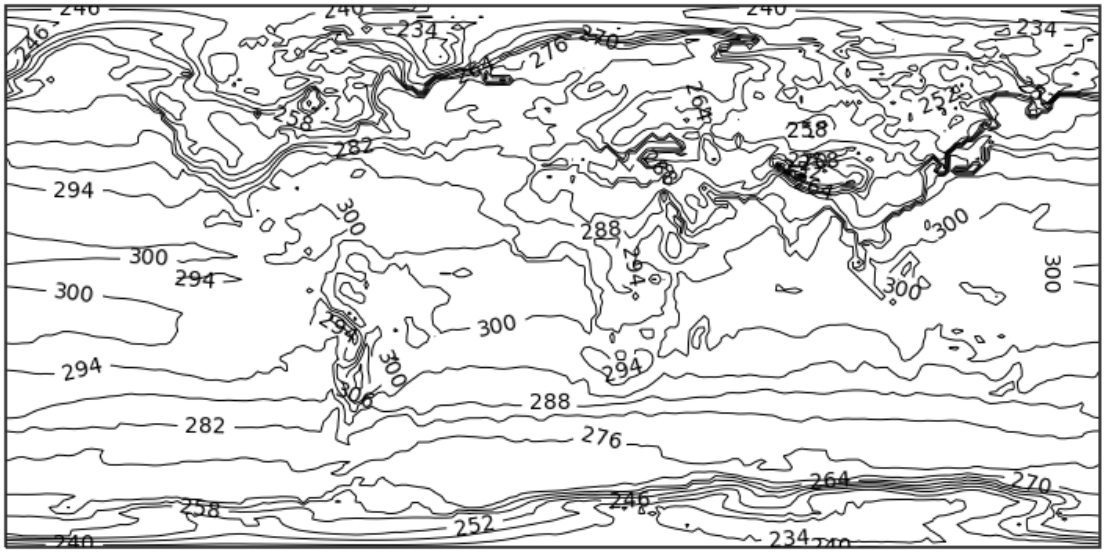


Draw the contour lines with a single color, increase the number of contour lines, and write the data

values inline.

[15]:
```
fig = plt.figure(figsize=(9,6))
ax = plt.axes(projection=ccrs.PlateCarree())

cn = ax.contour(ds.lon, ds.lat, ds.tsurf[0,:,:],
                levels=15,
                colors='black',
                linewidths=0.7,
                transform=ccrs.PlateCarree())

ax.clabel(cn, cn.levels, inline=True, fontsize=10)
```

[15]:
```
<a list of 47 text.Text objects>
```

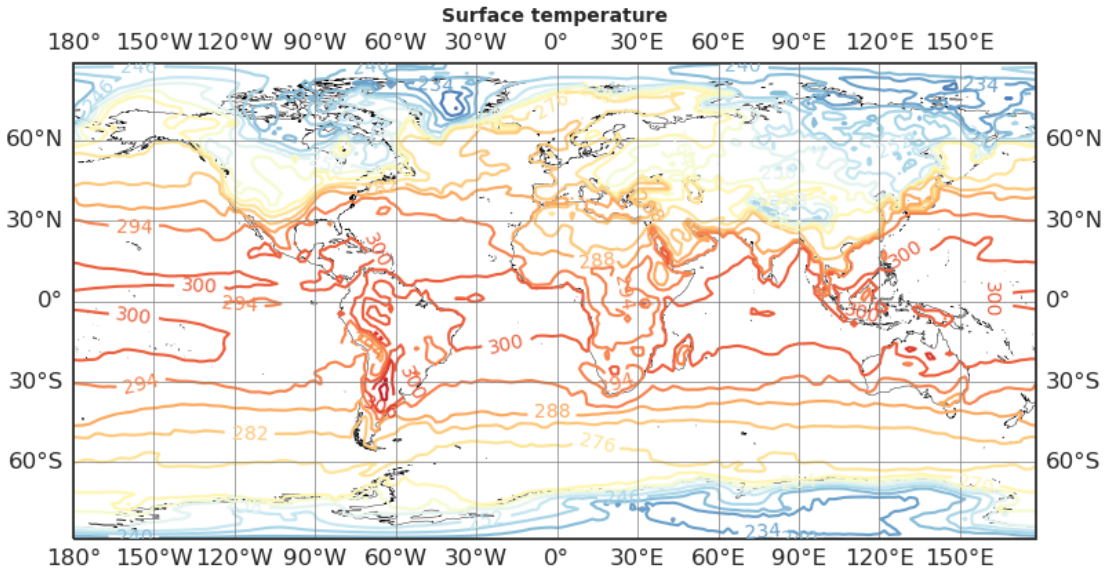


Next, we add the coast- and grid lines as well as a title. The colormap is changed to a 'blue to yellow to red' colormap by the reversed colormap `RdYlBu_r`.

[16]:
```
fig, ax = plt.subplots(figsize=(9,6), subplot_kw={"projection": ccrs.
↪PlateCarree()})

colormap = "RdYlBu_r"

ax.coastlines(resolution='50m', linewidth=0.3, color='black')

ax.gridlines(draw_labels=True, linewidth=0.5, color='gray',
             xlocs=range(-180,180,30), ylocs=range(-90,90,30))

ax.set_title('Surface temperature', fontsize=10, fontweight='bold')
```

```
cn = ax.contour(ds.lon, ds.lat, ds.tsurf[0,:,:],
                cmap=colormap,
                levels=15,
                transform=ccrs.PlateCarree())
ax.clabel(cn, cn.levels, inline=True, fontsize=10)
```

[16]: <a list of 47 text.Text objects>



Note:

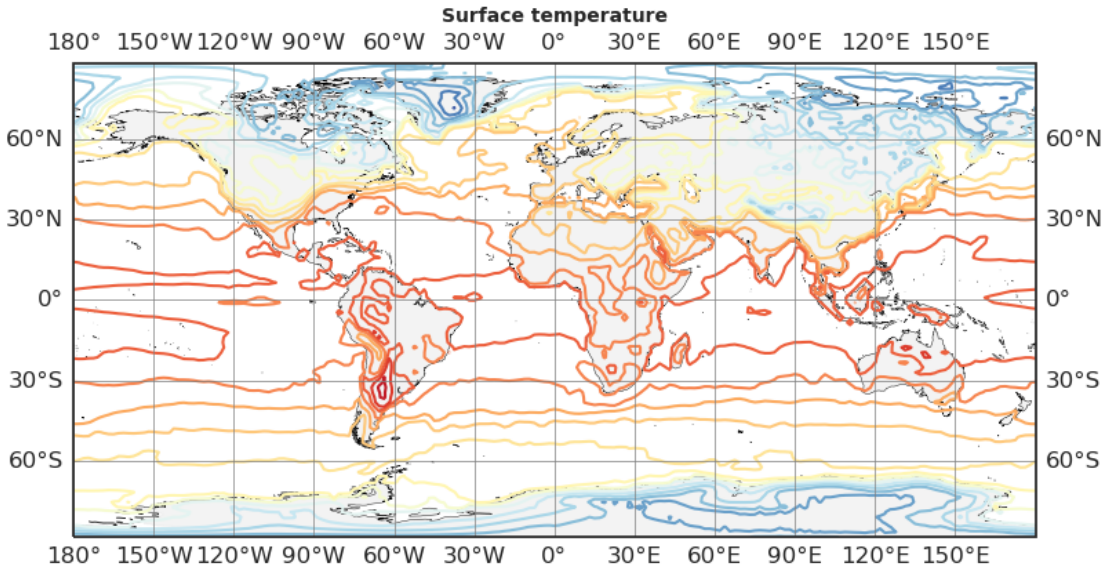
If there is a gap around the zero longitude Cartopy provides the function `add_cyclic_point`to append the first column after the last column.

[17]:
```
from cartopy.util import add_cyclic_point

data_wrap, lon_wrap = add_cyclic_point(ds.tsurf, coord=ds.lon, axis=2)
```

Finally, the fill the land masses with a gray color that is slightly transparent. Therefore we use `cfeature.LAND` from the cartopy features set.

[18]:
```
import cartopy.feature as cfeature

fig, ax = plt.subplots(figsize=(9,6), subplot_kw={"projection": ccrs.
 ↪PlateCarree()})

ax.coastlines(resolution='50m', linewidth=0.3, color='black')
ax.gridlines(draw_labels=True, linewidth=0.5, color='gray',
             xlocs=range(-180,180,30), ylocs=range(-90,90,30))
```

```
ax.add_feature(cfeature.LAND, color='lightgray', zorder=0,
               linewidth=0.5, edgecolor='black', alpha=0.25)
ax.set_title('Surface temperature', fontsize=10, fontweight='bold')

cn = ax.contour(lon_wrap, ds.lat, data_wrap[0,:,:], cmap=colormap,
                levels=15, transform=ccrs.PlateCarree())
```



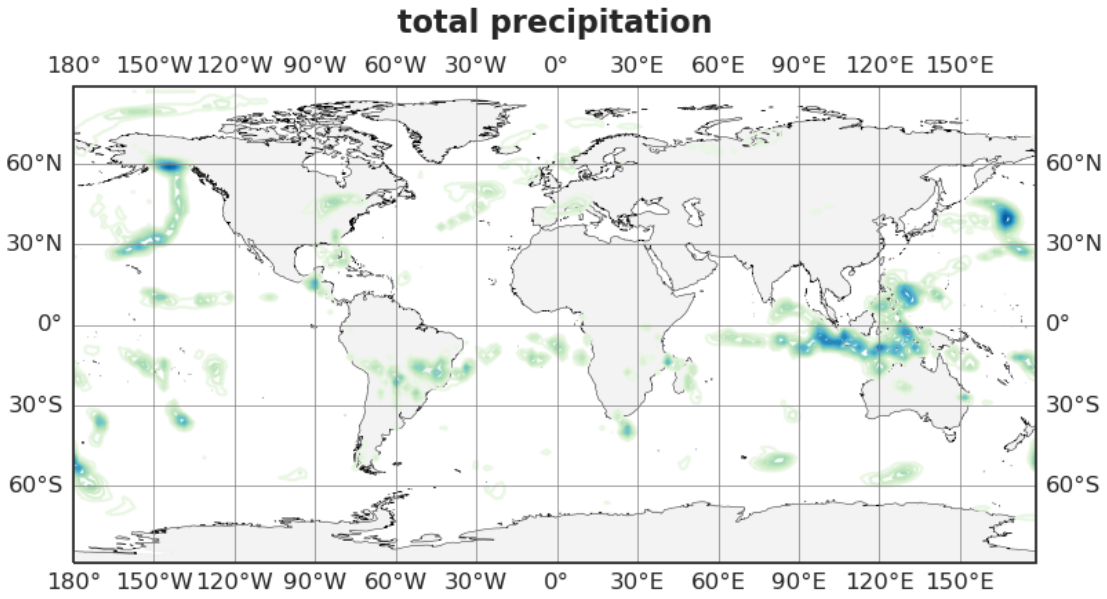
Exercise

- create a global contour plot of the variable 'precip'
- choose a different colormap like 'GnBu'
- change the title string

[ ]:

[19]:
```
fig, ax = plt.subplots(figsize=(9,6), subplot_kw={"projection": ccrs.
 ↪PlateCarree()})

ax.coastlines(resolution='50m', linewidth=0.3, color='black')
ax.gridlines(draw_labels=True, linewidth=0.5, color='gray',
             xlocs=range(-180,180,30), ylocs=range(-90,90,30))

ax.add_feature(cfeature.LAND, color='lightgray', zorder=0,
               linewidth=0.5, edgecolor='black', alpha=0.25)

title = ds.precip.long_name

ax.set_title(title, fontsize=16, y=1.09, fontweight='bold')
```

```
colormap = 'GnBu'

cn = ax.contour(ds.lon, ds.lat, ds.precip[0,:,:],
                cmap=colormap,
                levels=15,
                transform=ccrs.PlateCarree())
```



1.8 Cartopy features

https://scitools.org.uk/cartopy/docs/latest/matplotlib/feature_interface.html

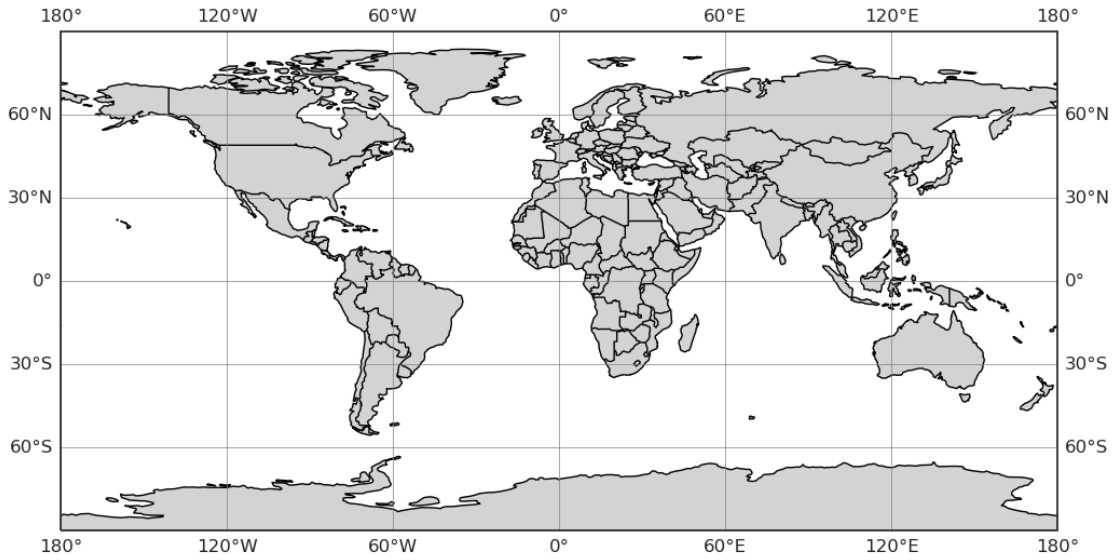
- BORDERS
- COASTLINE
- LAKES
- LAND
- OCEAN
- RIVERS
- STATES

  import cartopy.feature as cfeature

[20]:
```
fig, ax = plt.subplots(figsize=(12,6), subplot_kw={"projection": ccrs.
↪PlateCarree()})

ax.gridlines(draw_labels=True, linewidth=0.5, color='gray')
```

```
ax.add_feature(cfeature.LAND, color='lightgray',
               linewidth=0.5, edgecolor='black')
ax.add_feature(cfeature.COASTLINE)
ax.add_feature(cfeature.BORDERS)
```

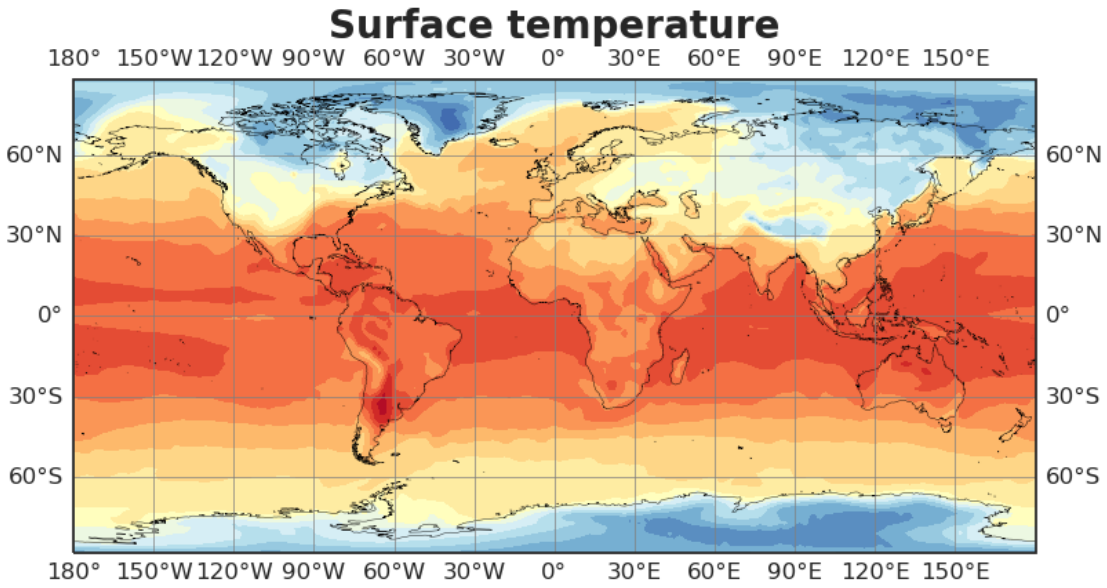
[20]: <cartopy.mpl.feature_artist.FeatureArtist at 0x7fff90da7280>



1.9 Create a filled contour plot

To switch from contour lines to filled contours just change the plot command from `plt.contour` to `plt.contourf`.

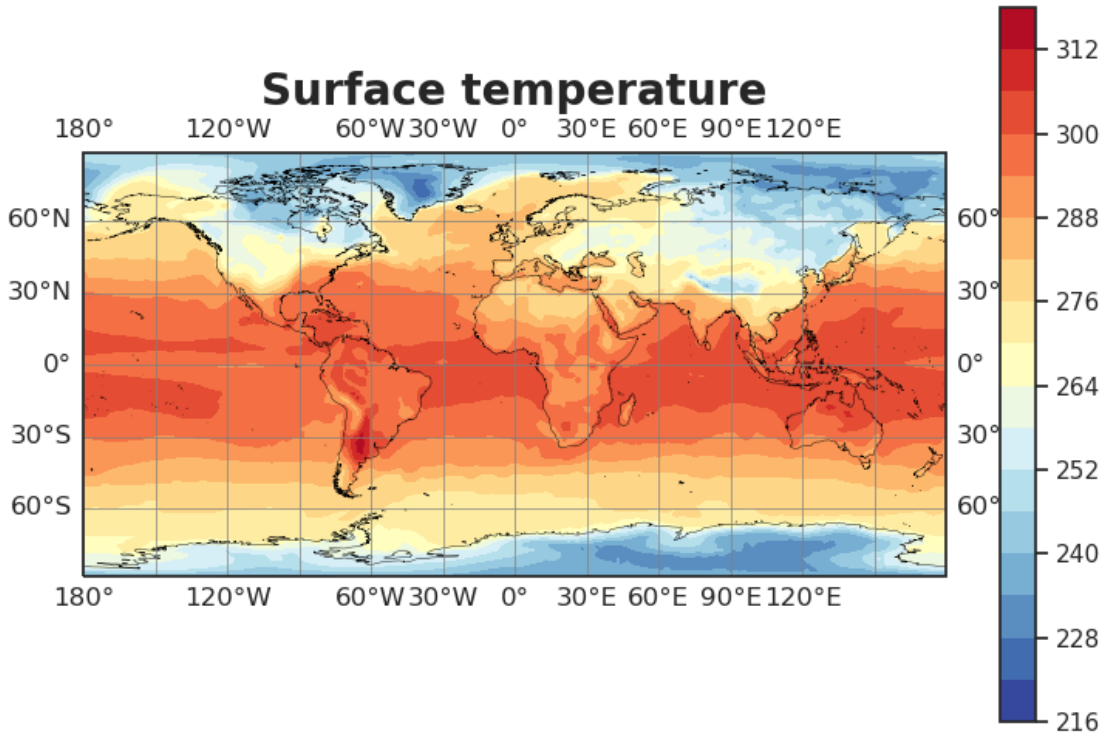
[21]:
```
fig, ax = plt.subplots(figsize=(9,6), subplot_kw={"projection": ccrs.
  ↪PlateCarree()})

colormap = "RdYlBu_r"

ax.coastlines(resolution='50m', linewidth=0.3, color='black')
ax.gridlines(draw_labels=True, linewidth=0.5, color='gray',
             xlocs=range(-180,180,30), ylocs=range(-90,90,30))
ax.set_title('Surface temperature', fontsize=20, fontweight='bold')

cnf = ax.contourf(lon_wrap, ds.lat, data_wrap[0,:,:], cmap=colormap,
                  levels=15, transform=ccrs.PlateCarree())
```



1.10 Add a colorbar

To better understanding we want to add a colorbar to the plot. To do this we have to create a mappable image which is done by assigning a variable to the contour plot and call `plt.colorbar()`.

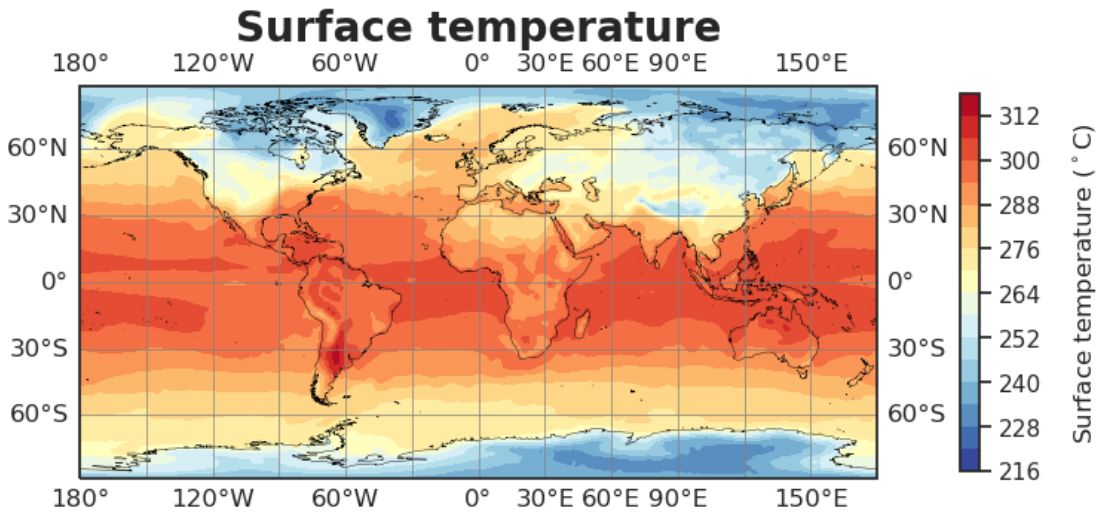
[22]:
```
fig, ax = plt.subplots(figsize=(9,6), subplot_kw={"projection": ccrs.
↪PlateCarree()})

ax.coastlines(resolution='50m', linewidth=0.3, color='black')
ax.gridlines(draw_labels=True, linewidth=0.5, color='gray',
             xlocs=range(-180,180,30), ylocs=range(-90,90,30))
ax.set_title('Surface temperature', fontsize=20, fontweight='bold')

plt_cn = ax.contourf(lon_wrap, ds.lat, data_wrap[0,:,:], cmap=colormap,
                     levels=15, transform=ccrs.PlateCarree())

cbar = plt.colorbar(plt_cn)
```



Well, that is not what we want, the colorbar is to large and the annotation of the right y-axis is not readable anymore.

We want a smaller colorbar shifted to the right with a vertical label string.

[23]:
```
fig, ax = plt.subplots(figsize=(9,6), subplot_kw={"projection": ccrs.
 ↪PlateCarree()})

ax.coastlines(resolution='50m', linewidth=0.3, color='black')
ax.gridlines(draw_labels=True, linewidth=0.5, color='gray',
             xlocs=range(-180,180,30), ylocs=range(-90,90,30))
ax.set_title('Surface temperature', fontsize=20, fontweight='bold')

plt_cn = ax.contourf(lon_wrap, ds.lat, data_wrap[0,:,:], cmap=colormap,
                     levels=15, transform=ccrs.PlateCarree())

cbar = plt.colorbar(plt_cn, pad=0.08, shrink=0.55)
cbar.set_label('Surface temperature $(^\circ\mathrm{C})$', labelpad=15, y=.5,␣
 ↪rotation=90)
```



Exercise

Create the plot above but change the following

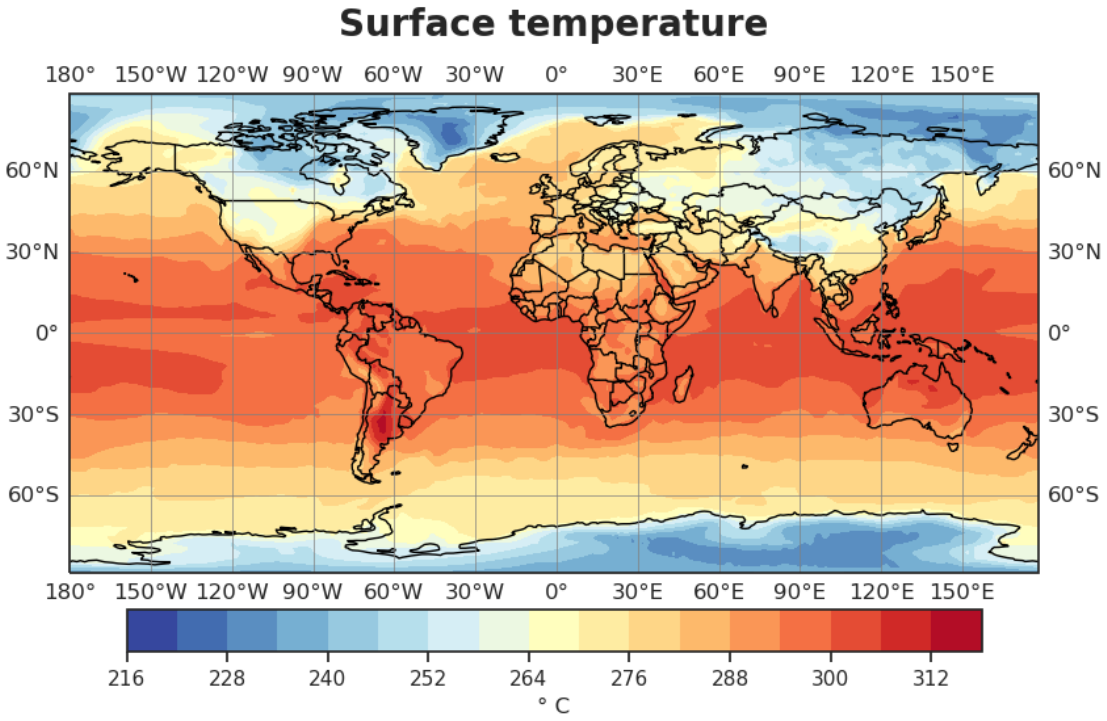
- increase the plot size
- add the country border lines
- draw a horizontal colorbar below the plot

Tipp: google pyplot.colorbar or

https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.colorbar.html

[ ]:

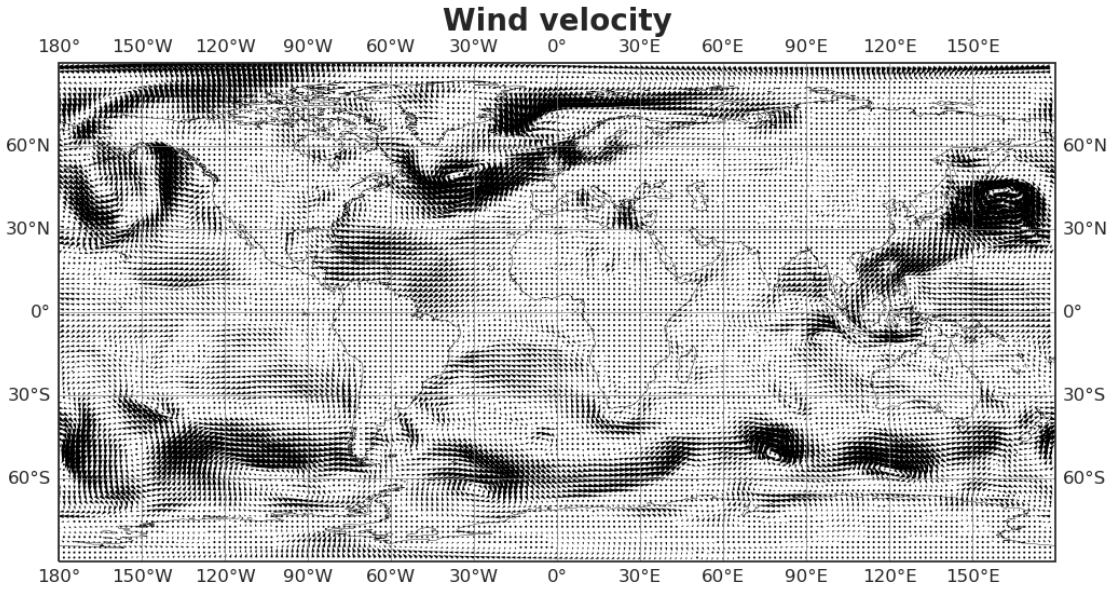
[24]:
```
fig, ax = plt.subplots(figsize=(12,6), subplot_kw={"projection": ccrs.
 ↪PlateCarree()})

ax.add_feature(cfeature.COASTLINE)
ax.add_feature(cfeature.BORDERS)
#ax.coastlines(resolution='50m', linewidth=0.3, color='black')
ax.gridlines(draw_labels=True, linewidth=0.5, color='gray',
             xlocs=range(-180,180,30), ylocs=range(-90,90,30))
ax.set_title('Surface temperature', y=1.1, fontsize=20, fontweight='bold')

plt_cn = ax.contourf(ds.lon, ds.lat, ds.tsurf[0,:,:], cmap=colormap,
                     levels=15, transform=ccrs.PlateCarree())

#cbar = plt.colorbar(plt_cn, location='bottom')
cbar = plt.colorbar(plt_cn, orientation='horizontal', pad=0.06, shrink=0.7)
#cbar.set_label('$^o$ C')
cbar.set_label('° C')
```



1.11 Create a vector plot

In the next section we will explain how to create a vector plot from the wind components u10 and v10. The name of the plot function is not vector as expected but `quiver`.

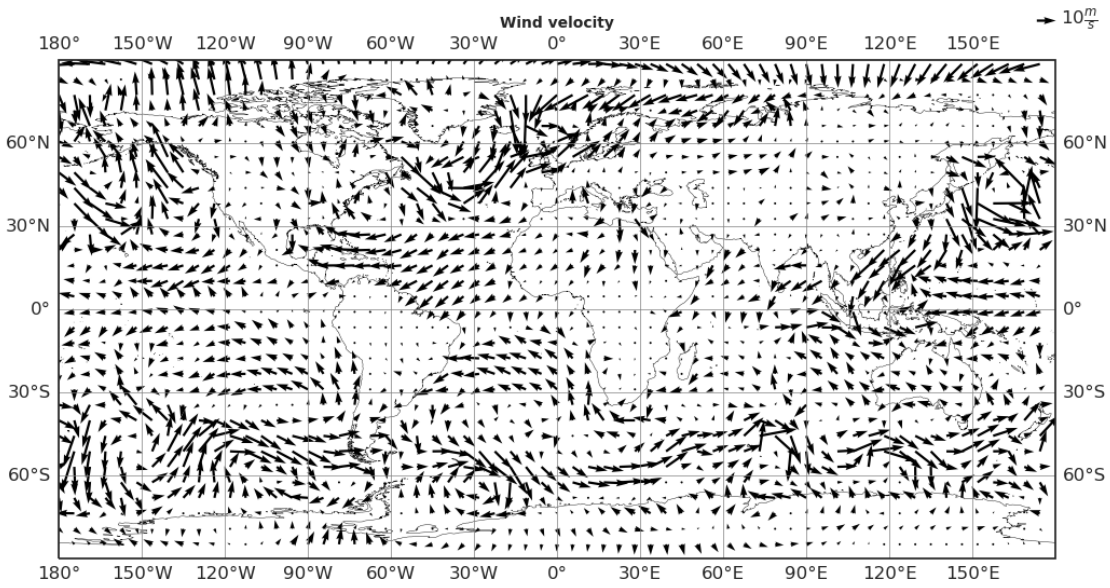
[25]:
```
u10 = ds.u10
v10 = ds.v10
lat = ds.lat[::-1]
lon = ds.lon[:]

fig, ax = plt.subplots(figsize=(12,6), subplot_kw={"projection": ccrs.
 ↪PlateCarree()})

ax.coastlines(resolution='50m', linewidth=0.3, color='black')
ax.gridlines(draw_labels=True, linewidth=0.5, color='gray',
             xlocs=range(-180,180,30), ylocs=range(-90,90,30))
ax.set_title('Wind velocity', fontsize=20, fontweight='bold')

vec = ax.quiver(ds.lon, ds.lat, u10[0,:,:], v10[0,:,:], transform=ccrs.
 ↪PlateCarree())
```



Well, that are too many arrows and we should thin out the field of vectors and add a reference vector to the upper right corner. To add a reference vector we can use `quiverkey`.

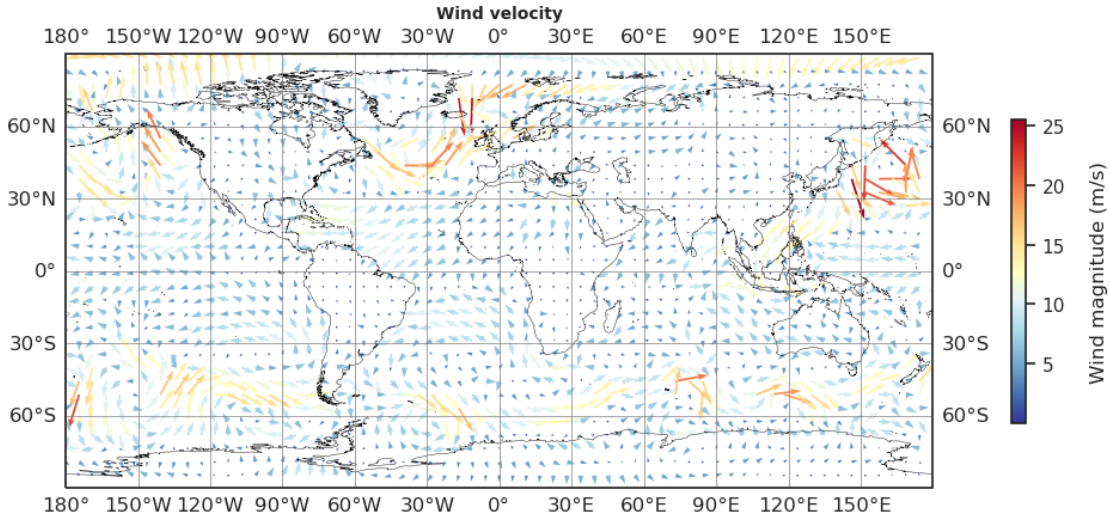
[26]:
```
fig, ax = plt.subplots(figsize=(12,6), subplot_kw={"projection": ccrs.
 ↪PlateCarree()})

ax.coastlines(resolution='50m', linewidth=0.3, color='black')
ax.gridlines(draw_labels=True, linewidth=0.5, color='gray',
             xlocs=range(-180,180,30), ylocs=range(-90,90,30))
ax.set_title('Wind velocity', fontsize=10, fontweight='bold')

inc = 3
vec = ax.quiver(ds.lon[::inc], ds.lat[::inc],
                u10[0,::inc,::inc],
                v10[0,::inc,::inc],
                transform=ccrs.PlateCarree())

ax.quiverkey(vec, X=1.0, Y=1.08, U=10, label=r'$10 \frac{m}{s}$',
             labelpos='E', labelsep=0.05)
```

[26]:
```
<matplotlib.quiver.QuiverKey at 0x7fffb0884b80>
```



A nice visualization type is the coloring of the vectors of the wind components by the wind magnitude. After calculating the magnitude we can use it in the quiver function call. We also add a colorbar for the magnitude.

[27]:
```
import numpy as np

fig, ax = plt.subplots(figsize=(12,6), subplot_kw={"projection": ccrs.
 ↪PlateCarree()})

ax.coastlines(resolution='50m', linewidth=0.3, color='black')
ax.gridlines(draw_labels=True, linewidth=0.5, color='gray',
             xlocs=range(-180,180,30), ylocs=range(-90,90,30))
ax.set_title('Wind velocity', fontsize=10, fontweight='bold')

magnitude = np.sqrt(u10**2 + v10**2)

inc = 3
vec = ax.quiver(ds.lon[::inc], ds.lat[::inc],
                u10[0,::inc,::inc],
                v10[0,::inc,::inc],
                magnitude[0,::inc,::inc],
                cmap=colormap,
                transform=ccrs.PlateCarree())

cbar = plt.colorbar(vec, pad=0.07, shrink=0.55)
cbar.set_label('Wind magnitude (m/s)', labelpad=15, y=.5, rotation=90)
```

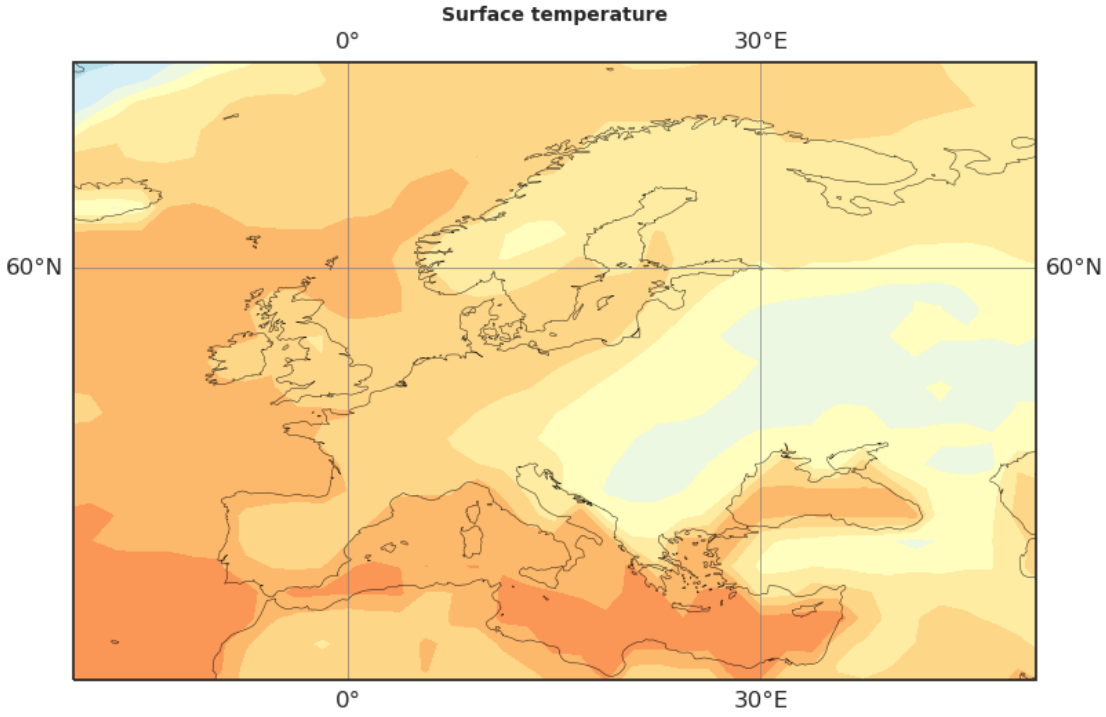


1.12 Zoom into a map

The cutting or zooming of a certain area can be easily implemented with `set_extent`. In the following example we want to display only the region of Europe.

In addition we draw the country borderlines which are available as Cartopy feature `cfeature.BORDERS`. Due to the fact that the coastlines are more grayish and thinner the line width and color are changed.

[28]:
```
fig, ax = plt.subplots(figsize=(9,6), subplot_kw={"projection": ccrs.
 ↪PlateCarree()})

ax.add_feature(cfeature.BORDERS, linewidth=0.6, color='None')
ax.coastlines(resolution='50m', linewidth=0.3, color='black')
ax.gridlines(draw_labels=True, linewidth=0.5, color='gray',
             xlocs=range(-180,180,30), ylocs=range(-90,90,30))
ax.set_title('Surface temperature', fontsize=10, fontweight='bold')
ax.set_extent([-20, 50, 30, 75], crs=ccrs.PlateCarree())

cnf = ax.contourf(lon_wrap, ds.lat, data_wrap[0,:,:], cmap=colormap,
                  levels=15, transform=ccrs.PlateCarree())
```



1.13 Map projections

To understand how cartopy and matplotlib uses projections and transformations we will have a closer look at the wording.

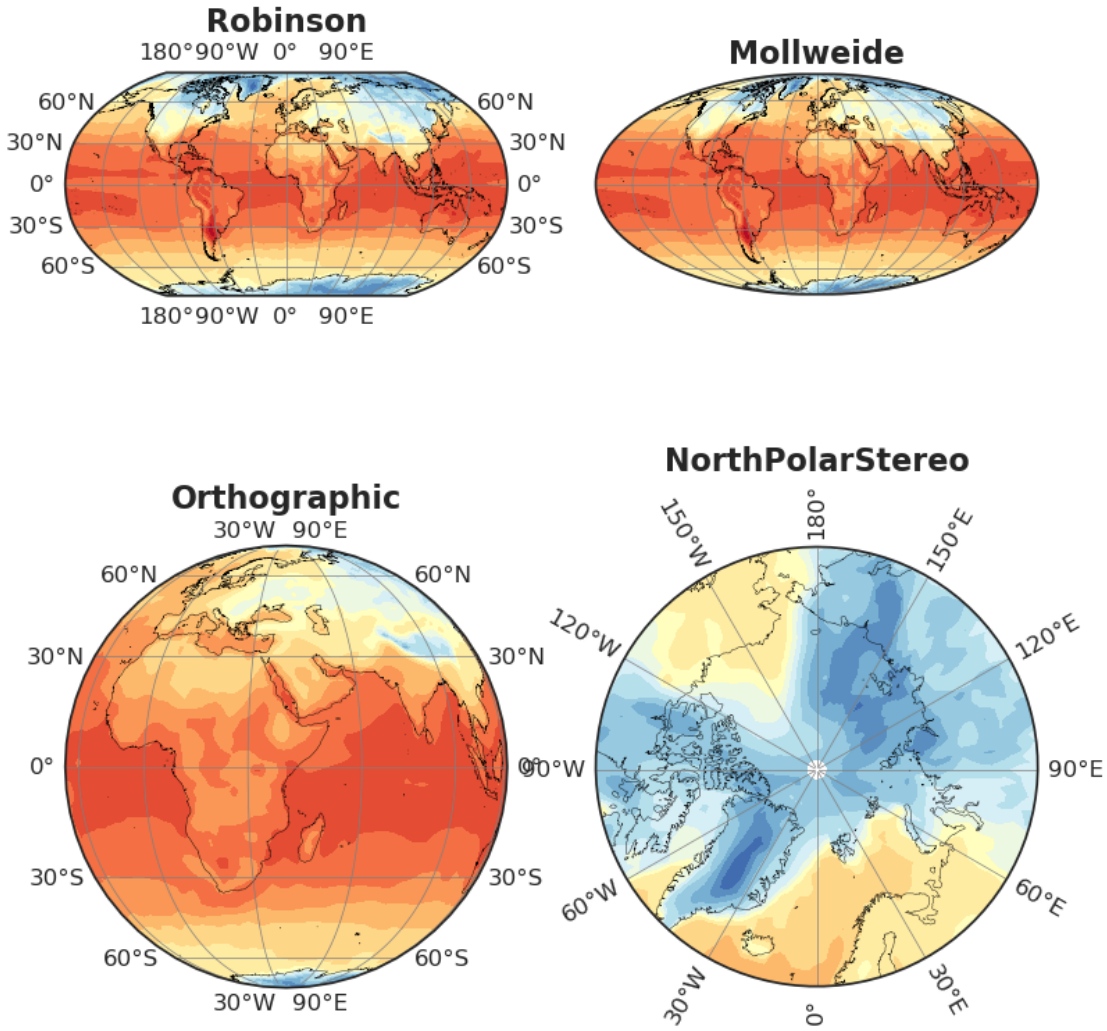
There are more than 30 projections available (https://scitools.org.uk/cartopy/docs/latest/reference/projections.ht

In the next example we will generate 4 plots showing Mollweide, Robinson, Orthographic, and North Polar Stereographic projection.

[29]:
```
import cartopy.crs as ccrs
from cartopy.util import add_cyclic_point

import matplotlib.pyplot as plt
import matplotlib.path as mpath

import numpy as np
import xarray as xr

data_file = '../data/rectilinear_grid_2D.nc'

ds = xr.open_dataset(data_file)

data_wrap, lon_wrap = add_cyclic_point(ds.tsurf, coord=ds.lon, axis=2)
```

```
data_crs = ccrs.PlateCarree()

colormap = "RdYlBu_r"

fig = plt.figure(figsize=[10, 10])

ax1 = plt.subplot(2, 2, 1, projection=ccrs.Robinson())
ax2 = plt.subplot(2, 2, 2, projection=ccrs.Mollweide())
ax3 = plt.subplot(2, 2, 3, projection=ccrs.Orthographic(central_latitude=0.0,
                                                        central_longitude=40.0))
ax4 = plt.subplot(2, 2, 4, projection=ccrs.NorthPolarStereo())


ax1.coastlines(resolution='50m', linewidth=0.3, color='black')
ax1.gridlines(draw_labels=True, linewidth=0.5, color='gray',
              xlocs=range(-180,180,30), ylocs=range(-90,90,30))
ax1.set_title('Robinson', fontsize=16, fontweight='bold')
ax1.contourf(lon_wrap, ds.lat, data_wrap[0,:,:], cmap=colormap,
             levels=15, transform=data_crs)


ax2.coastlines(resolution='50m', linewidth=0.3, color='black')
ax2.gridlines(linewidth=0.5, color='gray',
              xlocs=range(-180,180,30), ylocs=range(-90,90,30))
ax2.set_title('Mollweide', fontsize=16, fontweight='bold')
ax2.contourf(lon_wrap, ds.lat, data_wrap[0,:,:], cmap=colormap,
             levels=15, transform=data_crs)


ax3.coastlines(resolution='50m', linewidth=0.3, color='black')
ax3.gridlines(draw_labels=True, linewidth=0.5, color='gray',
              xlocs=range(-180,180,30), ylocs=range(-90,90,30))
ax3.set_title('Orthographic', fontsize=16, fontweight='bold')
ax3.set_global()
ax3.contourf(lon_wrap, ds.lat, data_wrap[0,:,:], cmap=colormap,
             levels=15, transform=data_crs)


ax4.coastlines(resolution='50m', linewidth=0.3, color='black')
ax4.gridlines(draw_labels=True, linewidth=0.5, color='gray',
              xlocs=range(-180,180,30), ylocs=range(-90,90,30))
ax4.set_title('NorthPolarStereo', fontsize=16, fontweight='bold')
ax4.set_global()
ax4.set_extent([-180, 180, 60, 90], ccrs.PlateCarree())
```

```
# Compute a circle in axes coordinates (boundary for the map)
theta = np.linspace(0, 2*np.pi, 100)
center, radius = [0.5, 0.5], 0.5
verts = np.vstack([np.sin(theta), np.cos(theta)]).T
circle = mpath.Path(verts * radius + center)

ax4.set_boundary(circle, transform=ax4.transAxes)
ax4.contourf(lon_wrap, ds.lat, data_wrap[0,:,:], cmap=colormap,
             levels=15, transform=data_crs)

fig.subplots_adjust(top=0.90, bottom=0.10, left=0.15, right=0.85,
                    hspace=0.1, wspace=0.2)
```



Exercise

Create the following map:

- projection 'Orthographic' that we can see Australia at the center of the map
- draw the land areas colored
- draw the ocean areas colored
- draw coastlines
- draw gridlines every 10°

[ ]:

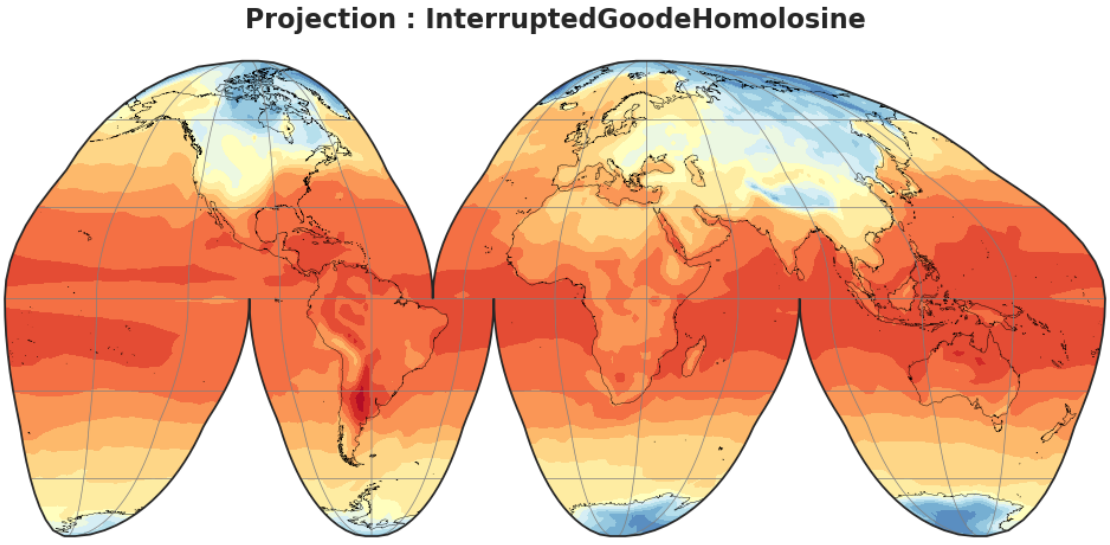
[30]:
```
projection = ccrs.Orthographic(central_longitude=130, central_latitude=-35)

fig, ax = plt.subplots(figsize=(8,8), subplot_kw={"projection": projection})

ax.add_feature(cfeature.LAND)
ax.add_feature(cfeature.OCEAN)
ax.coastlines(resolution='50m', linewidth=0.3, color='black')
ax.gridlines(linewidth=0.5, color='gray',
             xlocs=range(-180,180,10), ylocs=range(-90,90,10))
ax.set_title('Oceania', y = 1.05, fontsize=20, fontweight='bold')
```

[30]:
```
Text(0.5, 1.05, 'Oceania')
```

Just for fun :-)

[31]:
```
projection = ccrs.InterruptedGoodeHomolosine()
data_crs = ccrs.PlateCarree()

fig, ax = plt.subplots(figsize=(12,8), subplot_kw={"projection": projection})

ax.coastlines(resolution='50m', linewidth=0.3, color='black')
ax.gridlines(linewidth=0.5, color='gray',
             xlocs=range(-180,180,30), ylocs=range(-90,90,30))
```

```
ax.set_title('Projection : InterruptedGoodeHomolosine', y = 1.05,
             fontsize=16, fontweight='bold')

cnf = ax.contourf(lon_wrap, ds.lat, data_wrap[0,:,:], cmap=colormap,
            levels=15, transform=data_crs)
```



2 Tipp

DKRZ provides some visualization examples for Python

https://docs.dkrz.de/doc/visualization/sw/python/index.html

[ ]: