Untitled

November 11, 2020

1 Plotting Tutorial - compare plot

In this example we ‘compare’ runs between (usually) two different runs. However as I do not yet have two different files, the example plots will look at changes between timesteps and layers instead.

[This is a work in progress!]

[1]:
```
import aervis.plotting as ap
```

```
AerVis 0.0.2
:: AerVis plotting imports complete ::
```

1.1 Overlapping Variables

As I do not have a base and perturbed run, I begin by finding the overlap between two runs. To do this we use the `AerComp` class providing it with a base and a perturbed file

[2]:
```
base = '../../ACURE_PPE_001_bs714a.nc'
peturbed = '../../bk417a.nc'

compare = ap.AerComp(base, peturbed)
```

[3]:
```
#### Having loaded the files we can compare variables
```

[4]:
```
inboth = compare.diff_var()
# Print the first 10 variables which appear in both.
print('The first 10 matches are:\n',list(inboth)[:10])
```

```
The first 10 matches are:
 ['m01s00i507', 'm01s30i114', 'm01s50i217', 'm01s03i232', 'm01s01i271',
'm01s03i226', 'm01s05i205', 'm01s05i231', 'm01s34i109', 'm01s02i252']
```

[5]:
```
compare.stashinfo('m01s02i422')
```

```
--- m01s02i422 ---
        BASE (ACURE_PPE_001_bs714a.nc)
                description: atmosphere optical thickness due to dust ambient
aerosol
```

```
                dims: ('pseudo_level', 'latitude', 'longitude')


        PETURBED (bk417a.nc)
                description: atmosphere optical thickness due to dust ambient
aerosol
                dims: ('pseudo_level', 'time', 'latitude', 'longitude')
```

1.2 variable contents

We can expolre each dimension using the get_vars function

[6]:
```
compare.get_vars('pseudo_level')
```

[6]:
```
{'base': masked_array(data=[  3,   4,   6,   7,   8,   9, 101, 102, 103, 201,
202,
                    301, 302, 401, 402, 501, 502, 901, 902, 903, 904, 905,
                    906, 907, 908, 909, 910],
              mask=False,
        fill_value=999999,
             dtype=int32),
 'peturbed': masked_array(data=[  3,   4,   6,   7,   8,   9, 101, 102, 103,
201, 202,
                    301, 302, 401, 402, 501, 502, 901, 902, 903, 904, 905,
                    906, 907, 908, 909, 910],
              mask=False,
        fill_value=999999,
             dtype=int32)}
```

However we can also see that only one of the files contains a time dimension Using this we can see that trying to view it for both will create an error:

[7]:
```
compare.get_vars('time')
```

```
Stashcode %s does not exist in both files
```

To overcome get its values we need to read it manually

[8]:
```
compare.peturbed.variables['time']
```

[8]:
```
<class 'netCDF4._netCDF4.Variable'>
float64 time(time)
    _FillValue: nan
    standard_name: time
    units: hours since 1970-01-01
    calendar: gregorian
unlimited dimensions:
```

```
current shape = (4,)
filling on
```

If we want a human readable dictionary we can use `gettime` with the run we wish to use

[9]:
```
compare.gettime('peturbed')
```

[9]:
```
{'16-01-2008 00:00': 333456.0,
 '15-02-2008 00:00': 334176.0,
 '16-03-2008 00:00': 334896.0,
 '15-04-2008 00:01': 335616.0}
```

1.3 Selecting comparison parameters

We need to make a selection of items to plot from each dataset and give it to the plotting function

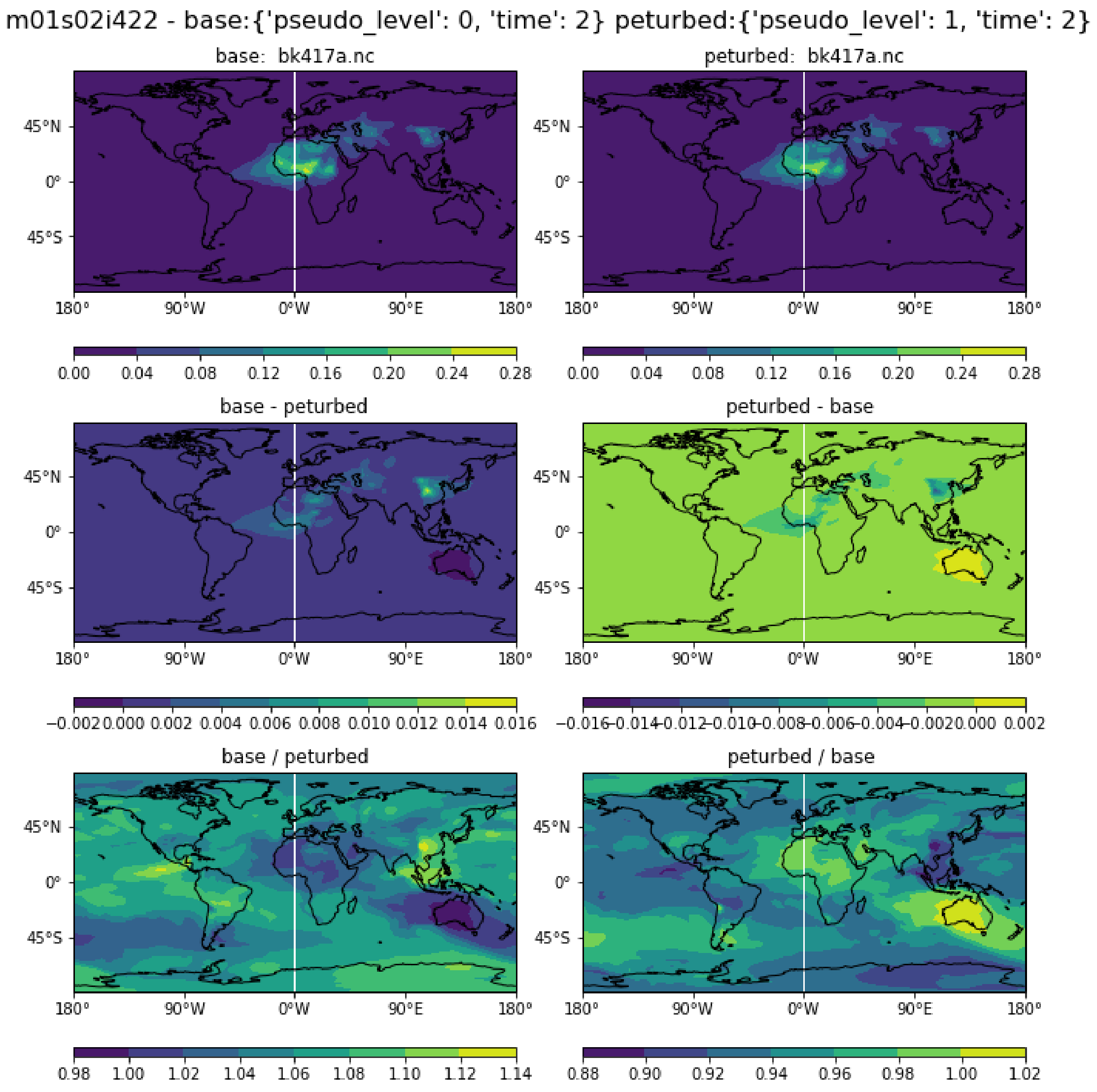
1.3.1 Unfortunately as we only have one working file (containing Latitude coordinates we will use the same file with different timesteps

[10]:
```
base = '../../bk417a.nc'
peturbed = '../../bk417a.nc'

compare = ap.AerComp(base, peturbed)
```

[11]:
```
stashcode = 'm01s02i422'
base_plot = {'pseudo_level':0,'time':0}
peturbed_plot = {'pseudo_level':0,'time':1}
```

[ ]:
```

```

[14]:
```
compare.compareplot(stashcode,[base_plot,peturbed_plot])
```



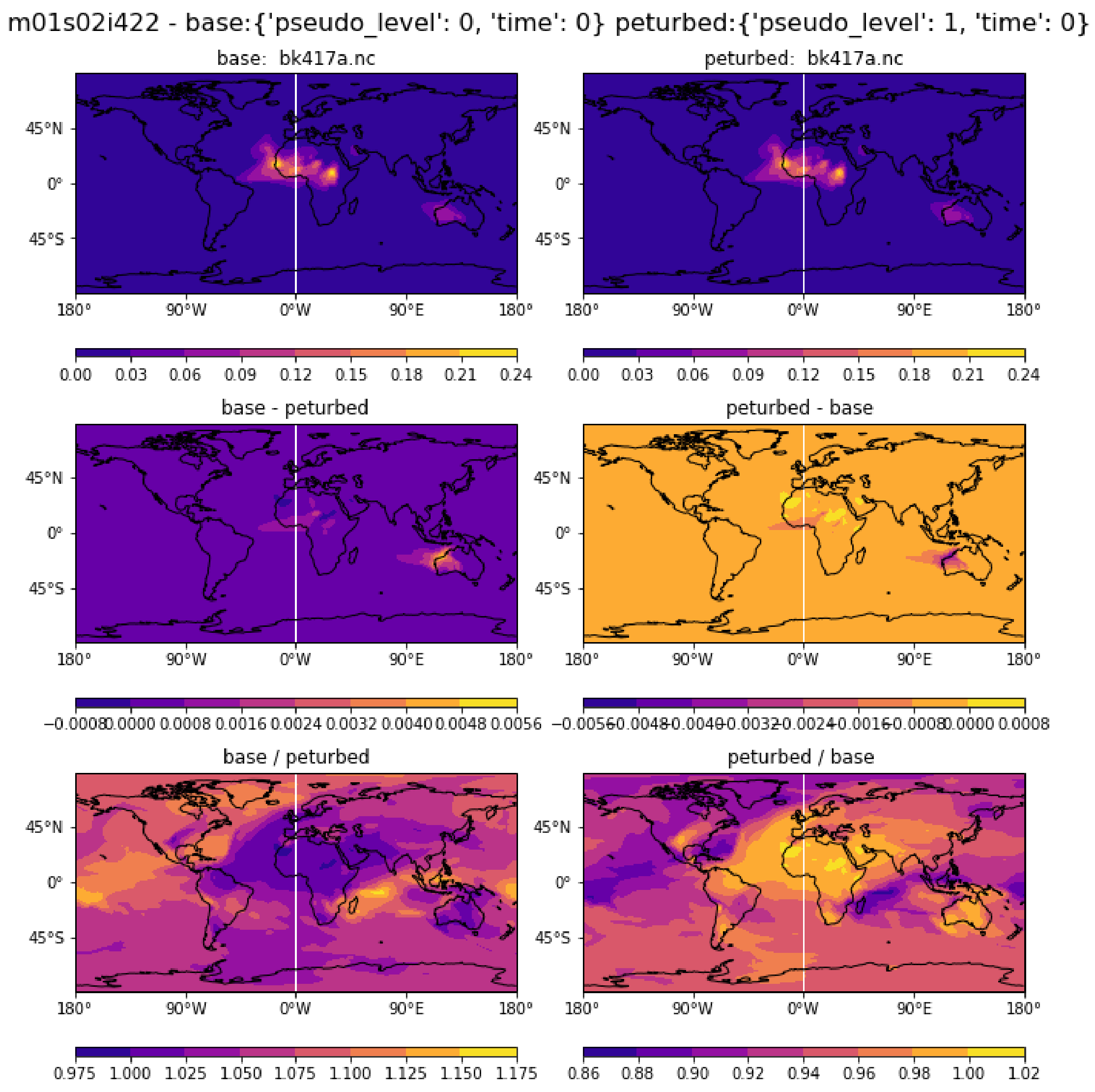
1.4 Multiple plots (cycling through comparison inputs)

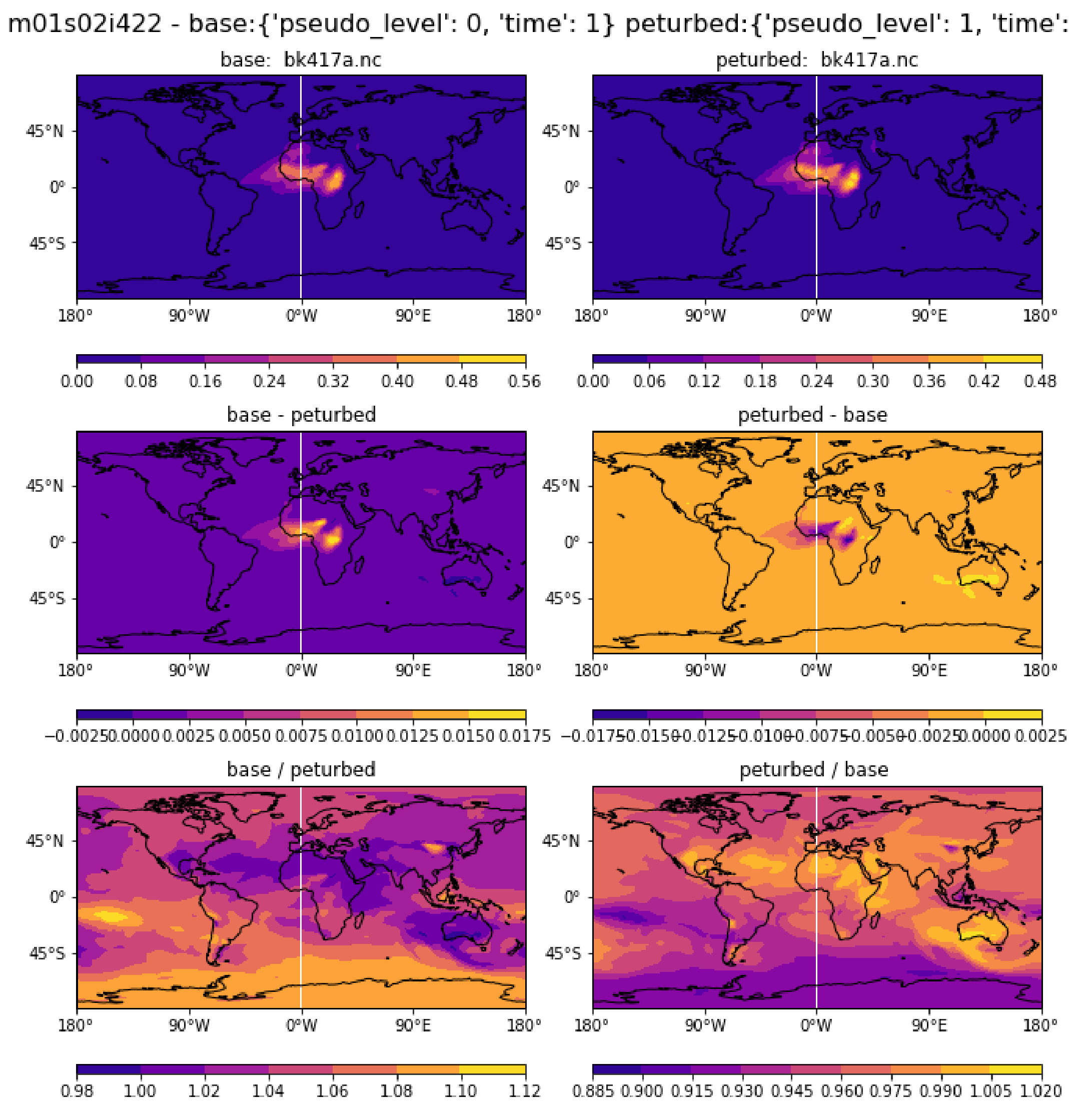
In the case where we are saving plots, or creating multiples it is useful to put the plotting function within a loop. This can potentially be run in parallel… Scripts can be added once we know what this code is to be used as?

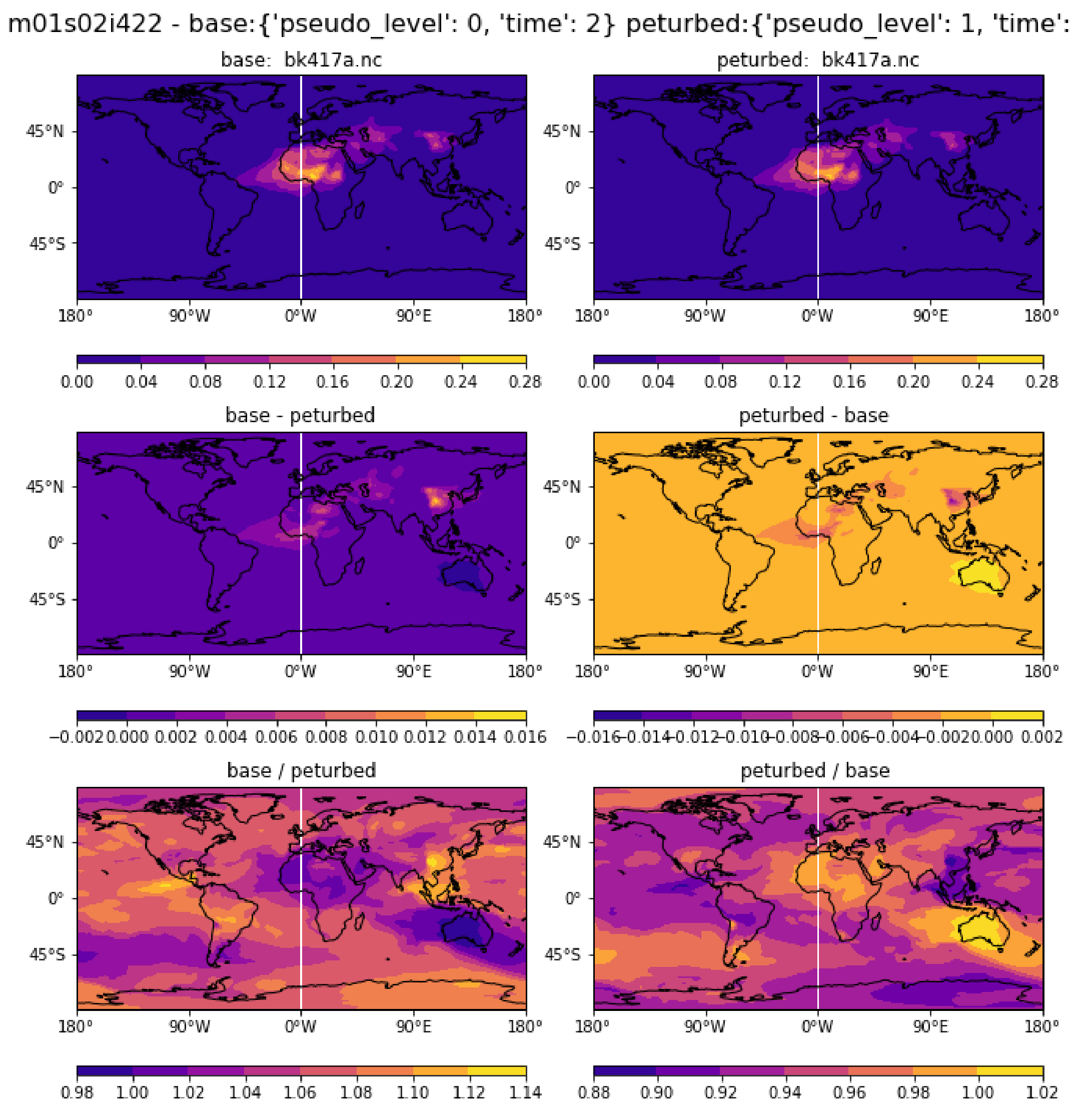
The followign example cycles through timesteps. For each set of plots I will compare psuedolevel 0 with pseudolevel 1 - I need some example runs.

[13]:
```
stashcode = 'm01s02i422'

for timestep in range(len(compare.gettime('peturbed').values())-1):
    # insert loop variables in parameters
```

```
base_plot = {'pseudo_level':0,'time':timestep}
peturbed_plot = {'pseudo_level':1,'time':timestep}

#plotcode
compare.compareplot(stashcode,[base_plot,peturbed_plot],cmap='plasma')
```




m01s02i422 - base:{'pseudo_level': 0, 'time': 1} peturbed:{'pseudo_level': 1, 'time': 1}
base: bk417a.nc
peturbed: bk417a.nc
base - peturbed
peturbed - base
base / peturbed
peturbed / base
45°N
0°
45°S
180°
90°W
0°W
90°E
180°
0.00 0.08 0.16 0.24 0.32 0.40 0.48 0.56
0.00 0.06 0.12 0.18 0.24 0.30 0.36 0.42 0.48
−0.0025 0.0000 0.0025 0.0050 0.0075 0.0100 0.0125 0.0150 0.0175
−0.0175 −0.0150 −0.0125 −0.0100 −0.0075 −0.0050 −0.0025 0.0000 0.0025
0.98 1.00 1.02 1.04 1.06 1.08 1.10 1.12
0.885 0.900 0.915 0.930 0.945 0.960 0.975 0.990 1.005 1.020


m01s02i422 - base:{'pseudo_level': 0, 'time': 2} peturbed:{'pseudo_level': 1, 'time': 2}
base: bk417a.nc
peturbed: bk417a.nc
base - peturbed
peturbed - base
base / peturbed
peturbed / base
45°N
0°
45°S
180°
90°W
0°W
90°E
180°
0.00 0.04 0.08 0.12 0.16 0.20 0.24 0.28
−0.002 0.000 0.002 0.004 0.006 0.008 0.010 0.012 0.014 0.016
−0.016 −0.014 −0.012 −0.010 −0.008 −0.006 −0.004 −0.002 0.000 0.002
0.98 1.00 1.02 1.04 1.06 1.08 1.10 1.12 1.14
0.88 0.90 0.92 0.94 0.96 0.98 1.00 1.02

```
[ ]:
```