
FAIR Documentation

Release 1.2.1

FAIR development team

Mar 12, 2018

CONTENTS:

1	Installation	1
1.1	From the Python Package Index (PyPI)	1
2	FAIR	3
2.1	Basic run	3
2.2	RCPs	4
2.3	Concentrations of well-mixed greenhouse gases	6
2.4	Radiative forcing	7
2.5	Ensemble generation	9
3	Indices and tables	13
	Python Module Index	15
	Index	17

INSTALLATION

1.1 From the Python Package Index (PyPI)

Probably the easiest way: simply type `pip install fair` from a terminal.

This notebook gives some simple examples of how to run and use the Finite Amplitude Impulse Response (FAIR) model.

The Finite Amplitude Impulse Response (FAIR) model is a simple emissions-based climate model. It allows the user to input emissions of greenhouse gases and short lived climate forcers in order to estimate global mean atmospheric GHG concentrations, radiative forcing and temperature anomalies.

The original FAIR model (v1.0) was developed to simulate the earth system response to CO emissions, with all non-CO forcing implemented as an “external” source. It was developed by Richard Millar, Zebedee Nicholls, Pierre Friedlingstein and Myles Allen. The motivation for developing it and its formulation is documented in a paper published in Atmospheric Chemistry and Physics in 2017 (doi:10.5194/acp-2016-405).

The emissions-based model extends FAIR by replacing all sources of non-CO forcing with relationships that are based on the source emissions, with the exception of natural forcings (viz. variations in solar irradiance and volcanic eruptions). It is useful for assessing future policy commitments to anthropogenic emissions (something which we can control) than to radiative forcing (something which is less certain and which we can only partially control).

The emissions based model was developed by Chris Smith with input from Piers Forster, Leighton Regayre and Giovanni Passerello, in parallel with Nicolas Leach, Richard Millar and Myles Allen.

```
%matplotlib inline
```

```
import fair
fair.__version__
```

```
'1.1.3+39.g35f5e83.dirty'
```

```
import numpy as np

from matplotlib import pyplot as plt
plt.style.use('seaborn-darkgrid')
plt.rcParams['figure.figsize'] = (16, 9)
```

2.1 Basic run

Here we show how FAIR can be run with step change CO emissions and sinusoidal non-CO forcing timeseries.

```
emissions = np.zeros(250)
emissions[125:] = 10.0
other_rf = np.zeros(emissions.size)
for x in range(0, emissions.size):
```

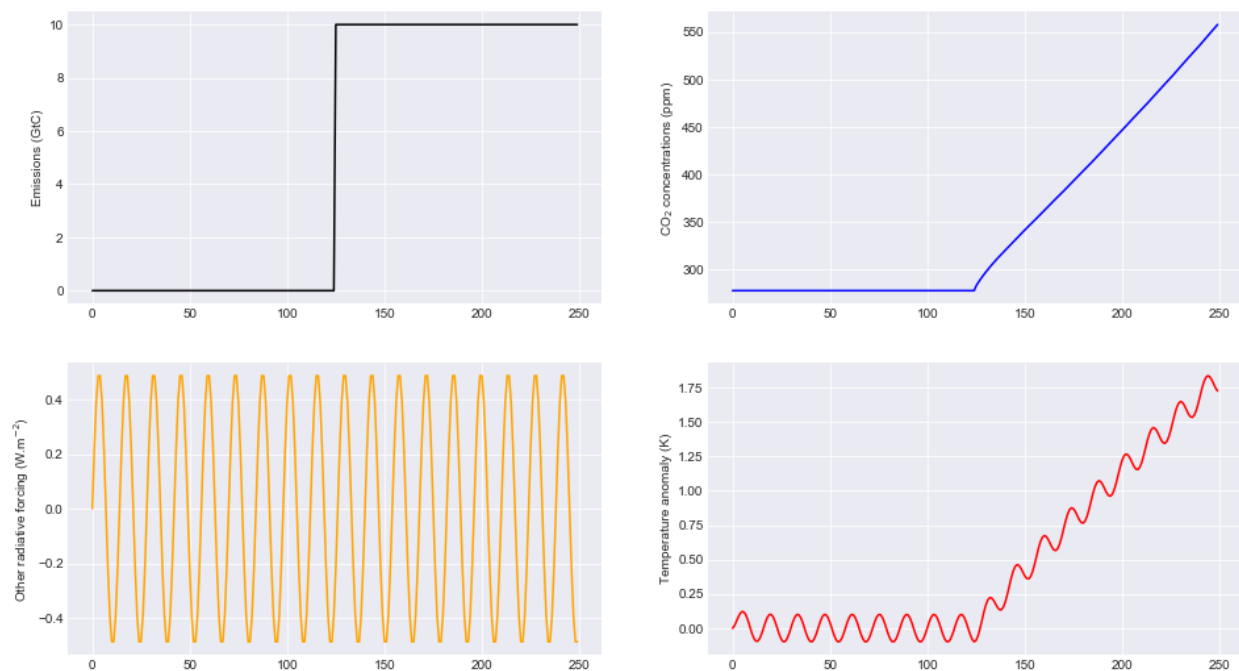
```

other_rf[x] = 0.5 * np.sin(2 * np.pi * (x) / 14.0)

C,F,T = fair.forward.fair_scm(
    emissions=emissions,
    other_rf=other_rf,
    useMultigas=False
)

fig = plt.figure()
ax1 = fig.add_subplot(221)
ax1.plot(range(0, emissions.size), emissions, color='black')
ax1.set_ylabel('Emissions (GtC)')
ax2 = fig.add_subplot(222)
ax2.plot(range(0, emissions.size), C, color='blue')
ax2.set_ylabel('CO2 concentrations (ppm)')
ax3 = fig.add_subplot(223)
ax3.plot(range(0, emissions.size), other_rf, color='orange')
ax3.set_ylabel('Other radiative forcing (W.m-2)')
ax4 = fig.add_subplot(224)
ax4.plot(range(0, emissions.size), T, color='red')
ax4.set_ylabel('Temperature anomaly (K)');

```



2.2 RCPs

We can run FAIR with the CO emissions and non-CO forcing from the four representative concentration pathway scenarios. To use the emissions-based version specify `useMultigas=True` in the call to `fair_scm()`.

By default in multi-gas mode, volcanic and solar forcing plus natural emissions of methane and nitrous oxide are switched on.

```

from fair.RCPs import rcp3pd, rcp45, rcp6, rcp85

```



```

fig = plt.figure()
ax1 = fig.add_subplot(221)
ax2 = fig.add_subplot(222)
ax3 = fig.add_subplot(223)
ax4 = fig.add_subplot(224)

C26, F26, T26 = fair.forward.fair_scm(emissions=rcp3pd.Emissions.emissions)
ax1.plot(rcp3pd.Emissions.year, rcp3pd.Emissions.co2_fossil, color='green', label=
↪ 'RCP3PD')
ax2.plot(rcp3pd.Emissions.year, C26[:, 0], color='green')
ax3.plot(rcp3pd.Emissions.year, np.sum(F26, axis=1), color='green')
ax4.plot(rcp3pd.Emissions.year, T26, color='green')

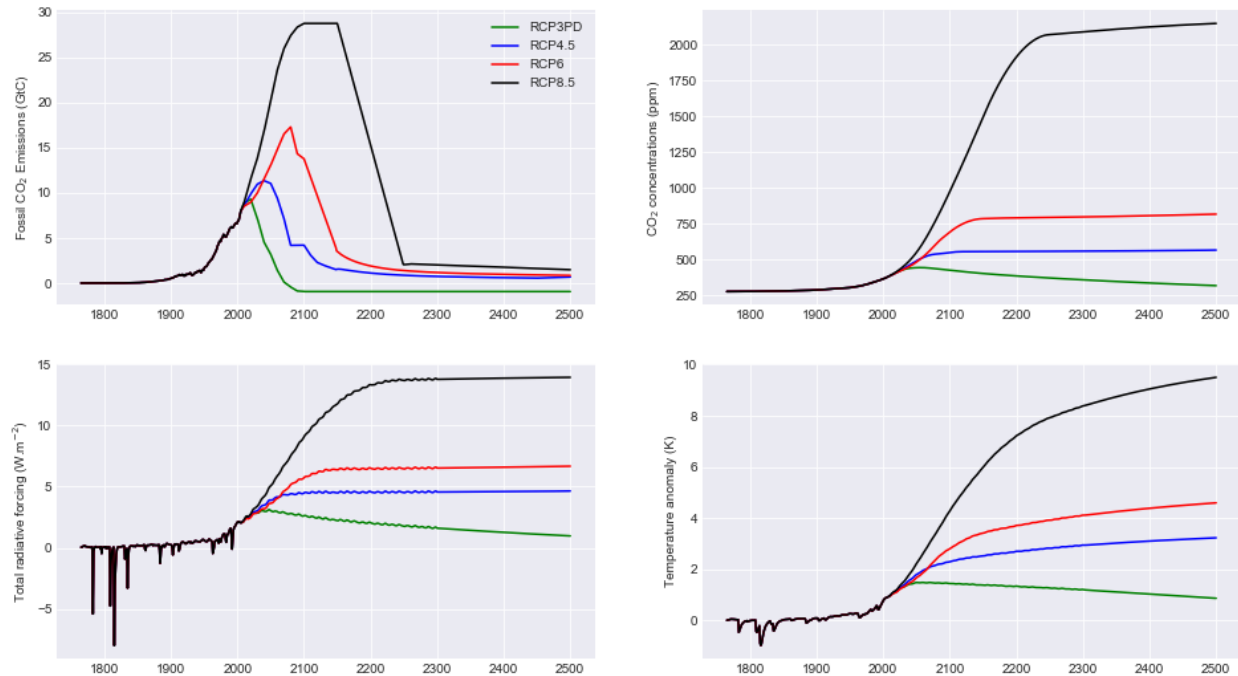
C45, F45, T45 = fair.forward.fair_scm(emissions=rcp45.Emissions.emissions)
ax1.plot(rcp45.Emissions.year, rcp45.Emissions.co2_fossil, color='blue', label='RCP4.5
↪ ')
ax2.plot(rcp45.Emissions.year, C45[:, 0], color='blue')
ax3.plot(rcp45.Emissions.year, np.sum(F45, axis=1), color='blue')
ax4.plot(rcp45.Emissions.year, T45, color='blue')

C60, F60, T60 = fair.forward.fair_scm(emissions=rcp6.Emissions.emissions)
ax1.plot(rcp6.Emissions.year, rcp6.Emissions.co2_fossil, color='red', label='RCP6')
ax2.plot(rcp6.Emissions.year, C60[:, 0], color='red')
ax3.plot(rcp6.Emissions.year, np.sum(F60, axis=1), color='red')
ax4.plot(rcp6.Emissions.year, T60, color='red')

C85, F85, T85 = fair.forward.fair_scm(emissions=rcp85.Emissions.emissions)
ax1.plot(rcp85.Emissions.year, rcp85.Emissions.co2_fossil, color='black', label='RCP8.
↪ 5')
ax2.plot(rcp85.Emissions.year, C85[:, 0], color='black')
ax3.plot(rcp85.Emissions.year, np.sum(F85, axis=1), color='black')
ax4.plot(rcp85.Emissions.year, T85, color='black')

ax1.set_ylabel('Fossil CO2 Emissions (GtC)')
ax1.legend()
ax2.set_ylabel('CO2 concentrations (ppm)')
ax3.set_ylabel('Total radiative forcing (W.m-2)')
ax4.set_ylabel('Temperature anomaly (K)');

```



2.3 Concentrations of well-mixed greenhouse gases

The output of FAIR (in most cases) is a 3-element tuple of concentrations, effective radiative forcing and temperature change since pre-industrial. Concentrations are a 31-column array of greenhouse gases. The indices correspond to the order given in the RCP concentration datasets (table 2 in Smith et al., <https://www.geosci-model-dev-discuss.net/gmd-2017-266/>). We can investigate the GHG concentrations coming out of the model:

```
fig = plt.figure()
ax1 = fig.add_subplot(221)
ax2 = fig.add_subplot(222)
ax3 = fig.add_subplot(223)
ax4 = fig.add_subplot(224)

ax1.plot(rcp3pd.Emissions.year, C26[:,1], color='green', label='RCP3PD')
ax1.plot(rcp45.Emissions.year, C45[:,1], color='blue', label='RCP4.5')
ax1.plot(rcp6.Emissions.year, C60[:,1], color='red', label='RCP6')
ax1.plot(rcp85.Emissions.year, C85[:,1], color='black', label='RCP8.5')
ax1.set_title("Methane concentrations, ppb")

ax2.plot(rcp3pd.Emissions.year, C26[:,2], color='green', label='RCP3PD')
ax2.plot(rcp45.Emissions.year, C45[:,2], color='blue', label='RCP4.5')
ax2.plot(rcp6.Emissions.year, C60[:,2], color='red', label='RCP6')
ax2.plot(rcp85.Emissions.year, C85[:,2], color='black', label='RCP8.5')
ax2.set_title("Nitrous oxide concentrations, ppb")

# How to convert the H and F gases to single-species equivalents? Weight by radiative_
# efficiency.
from fair.constants import radeff
C26_hfc134a_eq = np.sum(C26[:,3:15]*radeff.aslist[3:15],axis=1)/radeff.HFC134A #
# indices 3:15 are HFCs and PFCs
C45_hfc134a_eq = np.sum(C45[:,3:15]*radeff.aslist[3:15],axis=1)/radeff.HFC134A
C60_hfc134a_eq = np.sum(C60[:,3:15]*radeff.aslist[3:15],axis=1)/radeff.HFC134A
```

```

C85_hfc134a_eq = np.sum(C85[:,3:15]*radeff.aslist[3:15],axis=1)/radeff.HFC134A

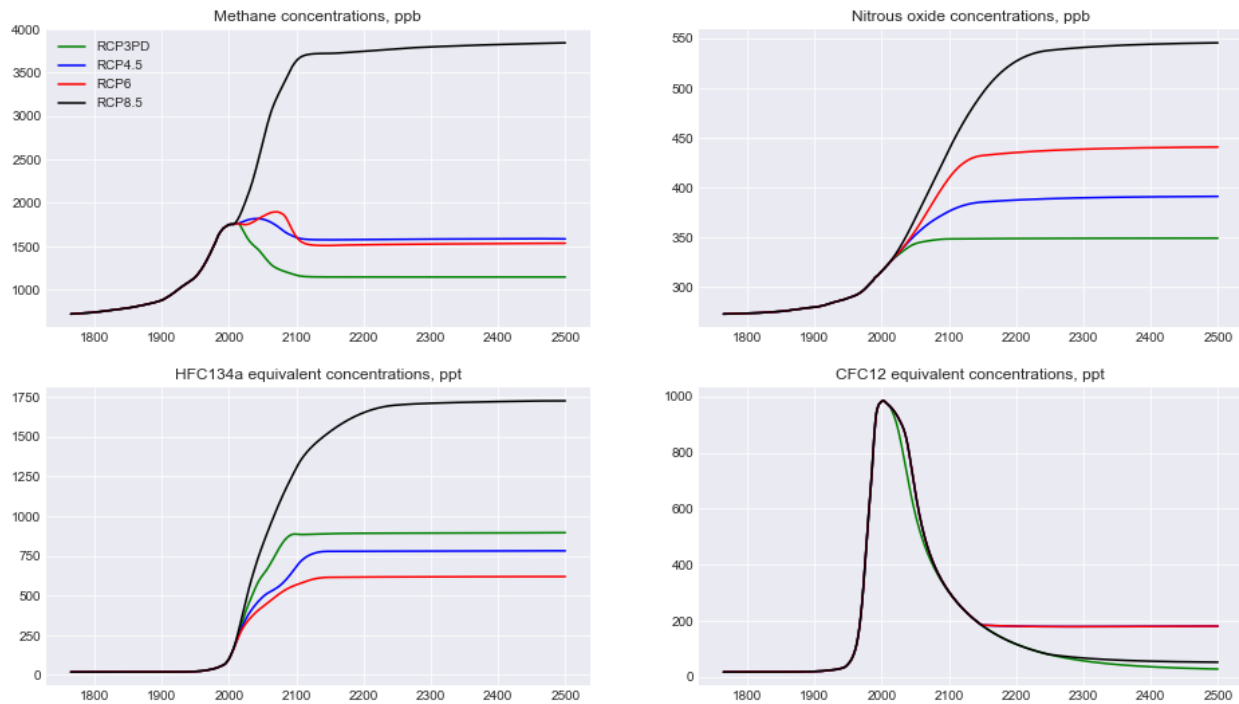
C26_cfc12_eq = np.sum(C26[:,15:31]*radeff.aslist[15:31],axis=1)/radeff.CFC12  #_
↪indices 15:31 are ozone depleters
C45_cfc12_eq = np.sum(C45[:,15:31]*radeff.aslist[15:31],axis=1)/radeff.CFC12
C60_cfc12_eq = np.sum(C60[:,15:31]*radeff.aslist[15:31],axis=1)/radeff.CFC12
C85_cfc12_eq = np.sum(C85[:,15:31]*radeff.aslist[15:31],axis=1)/radeff.CFC12

ax3.plot(rcp3pd.Emissions.year, C26_hfc134a_eq, color='green', label='RCP3PD')
ax3.plot(rcp45.Emissions.year, C45_hfc134a_eq, color='blue', label='RCP4.5')
ax3.plot(rcp6.Emissions.year, C60_hfc134a_eq, color='red', label='RCP6')
ax3.plot(rcp85.Emissions.year, C85_hfc134a_eq, color='black', label='RCP8.5')
ax3.set_title("HFC134a equivalent concentrations, ppt")

ax4.plot(rcp3pd.Emissions.year, C26_cfc12_eq, color='green', label='RCP3PD')
ax4.plot(rcp45.Emissions.year, C45_cfc12_eq, color='blue', label='RCP4.5')
ax4.plot(rcp6.Emissions.year, C60_cfc12_eq, color='red', label='RCP6')
ax4.plot(rcp85.Emissions.year, C85_cfc12_eq, color='black', label='RCP8.5')
ax4.set_title("CFC12 equivalent concentrations, ppt")
ax1.legend()

```

```
<matplotlib.legend.Legend at 0x7f7ef86e6850>
```



2.4 Radiative forcing

We consider 13 separate species of radiative forcing: CO, CH, NO, minor GHGs, tropospheric ozone, stratospheric ozone, stratospheric water vapour from methane oxidation, contrails, aerosols, black carbon on snow, land use change, volcanic and solar (table 1 in Smith et al., <https://www.geosci-model-dev-discuss.net/gmd-2017-266/>). Here we show some of the more interesting examples.

```

fig = plt.figure()
ax1 = fig.add_subplot(221)
ax2 = fig.add_subplot(222)
ax3 = fig.add_subplot(223)
ax4 = fig.add_subplot(224)

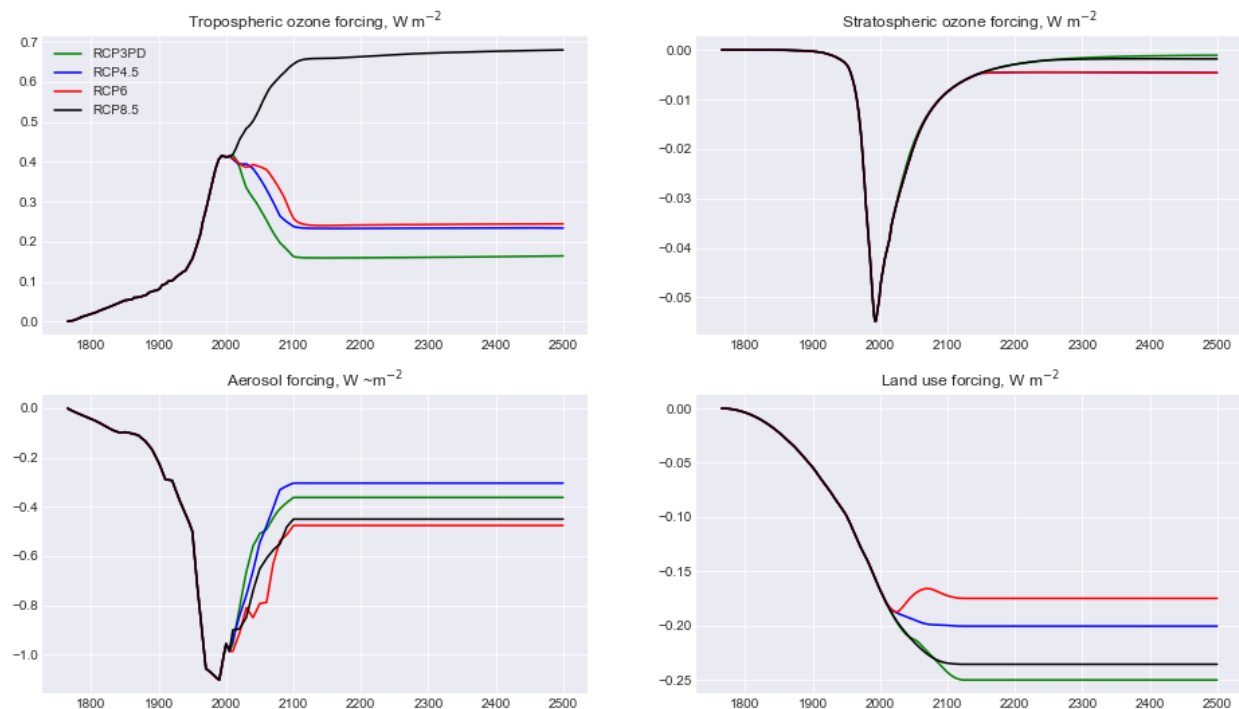
ax1.plot(rcp3pd.Emissions.year, F26[:,4], color='green', label='RCP3PD')
ax1.plot(rcp45.Emissions.year, F45[:,4], color='blue', label='RCP4.5')
ax1.plot(rcp6.Emissions.year, F60[:,4], color='red', label='RCP6')
ax1.plot(rcp85.Emissions.year, F85[:,4], color='black', label='RCP8.5')
ax1.set_title("Tropospheric ozone forcing, W m$^{-2}$")

ax2.plot(rcp3pd.Emissions.year, F26[:,5], color='green', label='RCP3PD')
ax2.plot(rcp45.Emissions.year, F45[:,5], color='blue', label='RCP4.5')
ax2.plot(rcp6.Emissions.year, F60[:,5], color='red', label='RCP6')
ax2.plot(rcp85.Emissions.year, F85[:,5], color='black', label='RCP8.5')
ax2.set_title("Stratospheric ozone forcing, W m$^{-2}$")

ax3.plot(rcp3pd.Emissions.year, F26[:,8], color='green', label='RCP3PD')
ax3.plot(rcp45.Emissions.year, F45[:,8], color='blue', label='RCP4.5')
ax3.plot(rcp6.Emissions.year, F60[:,8], color='red', label='RCP6')
ax3.plot(rcp85.Emissions.year, F85[:,8], color='black', label='RCP8.5')
ax3.set_title("Aerosol forcing, W ~m$^{-2}$")

ax4.plot(rcp3pd.Emissions.year, F26[:,10], color='green', label='RCP3PD')
ax4.plot(rcp45.Emissions.year, F45[:,10], color='blue', label='RCP4.5')
ax4.plot(rcp6.Emissions.year, F60[:,10], color='red', label='RCP6')
ax4.plot(rcp85.Emissions.year, F85[:,10], color='black', label='RCP8.5')
ax4.set_title("Land use forcing, W m$^{-2}$")
ax1.legend();

```



2.5 Ensemble generation

An advantage of FAIR is that it is very quick to run (much less than a second on an average machine). Therefore it can be used to generate probabilistic future ensembles. We'll show a 100-member ensemble.

```
from scipy import stats

# generate some (bad) TCR and ECS pairs
tcrcs = stats.norm.rvs(size=(100,2), loc=[1.75,3], scale=[0.4,0.8], random_
    ↪state=38571)

# generate some forcing scale factors with SD of 10% of the best estimate
F_scale = stats.norm.rvs(size=(100,13), loc=1, scale=0.1, random_state=40000)

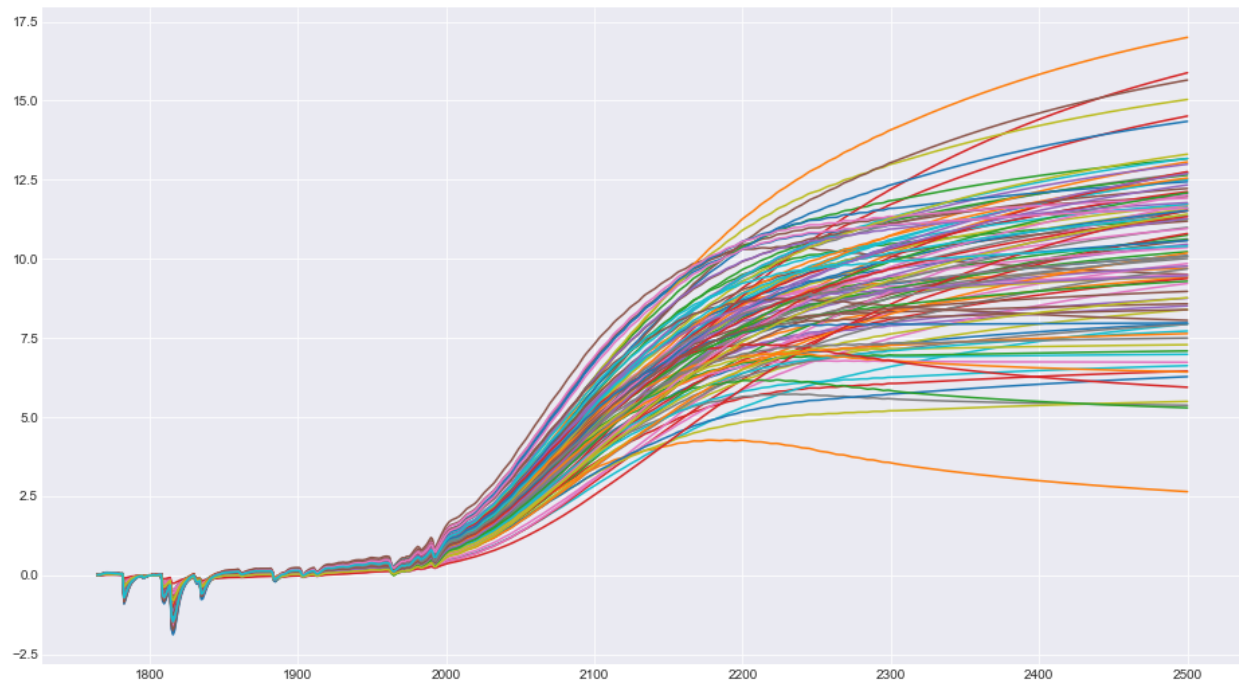
# do the same for the carbon cycle parameters
r0 = stats.norm.rvs(size=100, loc=35, scale=3.5, random_state=41000)
rc = stats.norm.rvs(size=100, loc=0.019, scale=0.0019, random_state=42000)
rt = stats.norm.rvs(size=100, loc=4.165, scale=0.4165, random_state=45000)

T = np.zeros((736,100))

%time
for i in range(100):
    _, _, T[:,i] = fair.forward.fair_scm(emissions=rcp85.Emissions.emissions,
        r0 = r0[i],
        rc = rc[i],
        rt = rt[i],
        tcrcs = tcrcs[i,:],
        scale = F_scale[i,:],
        F2x = 3.74*F_scale[i,0])    # scale F2x with the CO2 scaling_
    ↪factor for consistency
```

```
CPU times: user 0 ns, sys: 0 ns, total: 0 ns
Wall time: 7.15 µs
```

```
fig = plt.figure()
ax1 = fig.add_subplot(111)
ax1.plot(rcp85.Emissions.year, T);
```



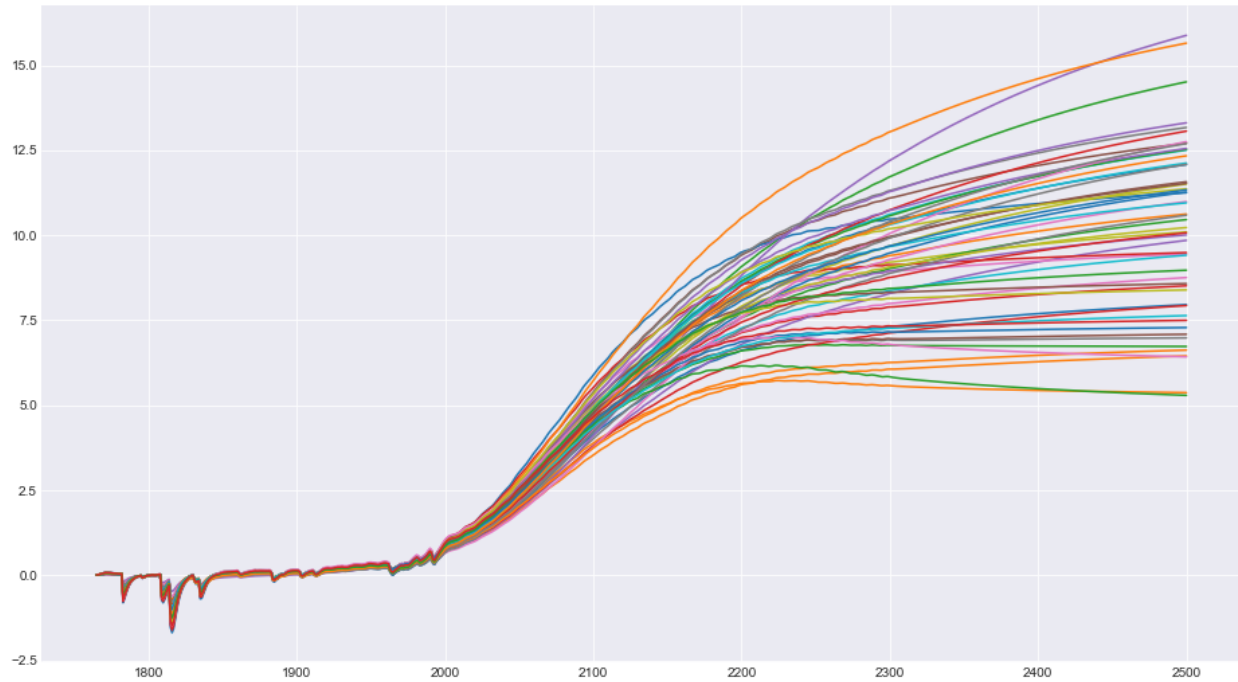
The resulting projections show a large spread. Some of these ensemble members are unrealistic, ranging from around 0.4 to 2.0 K temperature change in the present day, whereas we know in reality it is more like 0.9 (plus or minus 0.2). Therefore we can constrain this ensemble to observations.

```
from fair.tools.constrain import hist_temp
CW = np.loadtxt('fair/tools/tempobs/had4_krig_annual_v2_0_0.txt') # Cowtan & Way in-
    filled dataset of global temperatures
constrained = np.zeros(100, dtype=bool)
for i in range(100):
    # we use observed trend from 1880 to 2016
    constrained[i] = hist_temp(CW[30:,1], T[1880-1765:2017-1765,i], CW[30:,0])
```

```
# How many ensemble members passed the constraint?
print np.sum(constrained)
```

```
54
```

```
# What does this do to the ensemble?
fig = plt.figure()
ax1 = fig.add_subplot(111)
ax1.plot(rcp85.Emissions.year, T[:,constrained]);
```



Some, but not all, of the higher end scenarios have been constrained out, but there is still quite a large range of total temperature change projected for 2500 even under this constraint.

From these constraints it is possible to obtain posterior distributions on effective radiative forcing, ECS, TCR, TCRE and other metrics.

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

f

fair, [11](#)

INDEX

F

`fair` (module), [11](#)