Assessing Negative Carbon Dioxide Emissions from the Perspective of a National 'Fair Share' of the Remaining Global Carbon Budget

Barry McMULLIN, Paul PRICE, Michael B. JONES, Alwynne H. McGEEVER

Mitigation and Adaptation Strategies for Global Change (https://link.springer.com/journal/11027)

Article DOI: 10.1007/s11027-019-09881-6 (https://doi.org/10.1007/s11027-019-09881-6)

Accepted for publication: 01 July 2019

Corresponding Author: Barry McMullin (http://www.eeng.dcu.ie/~mcmullin)

Supplementary material

Acknowledgements: A data product of the IE-NETs research project (http://ienets.eeng.dcu.ie/), supported by the Irish Environmental Projection Agency (EPA) (http://epa.ie/) Research Programme 2014-2020 (grant number 2016-CCRP-MS.36).

Background: "Exponential" pathways

"Exponential mitigation" is a plausible "default" CO_2 emissions mitigation pathway: it is characterised by a constant year-on-year fractional change in the annual emissions rate, typically expressed as a percentage (here denoted R, and with a negative value indicating exponential *decline*). It is plausible because earlier emissions reductions, relative to a large base, are likely to be easier to achieve than later reductions relative to an already much contracted base (the "low hanging fruit" concept). So, as a first, crude, approximation, constant fractional reduction corresponds to something like "constant mitigation effort", with some concession to "inter-generational justice" (as opposed to, for example, *linear* pathways that would "backload" greater year-on-year mitigation effort on future generations).

Exponentially *declining* pathways are also characterised by an asymptotic approach to zero and having finite cumulative emissions, or cumulative "quota" ("area under the curve") in the limit extended to infinite time. For any given quota, relative to a given starting emissions level, any alternative pathway (having varying mitigation rate) would, at some point, require a mitigation rate *higher* than the constant rate of the exponential pathway (at least for strictly *positive* pathways: the situation is different if nett negative emissions rates are allowed, but we do not explore that case here). That is, for given quota, and starting emissions level, the (unique) corresponding exponential pathway represents the pathway with the "least maximum" mitigation rate (in fact, constant mitigation rate), for all possible strictly positive pathways.

This cumulative quota consideration is of particular relevance to CO_2: as a very long lived gas, its warming effect is best understood by looking not at the emissions rate in any particular year, but the accumulation in time; and, to stabilize climate, its emissions rate must indeed fall to (nett) zero (at least), which is not necessarily true of shorter-lived GHGs.

Exponentially *growing* pathways are also a common modelling default in various contexts: for example, in a situation of projected stable economic growth, and in the absence of stringent decoupling of emissions from such growth, then the default emissions rate pathway would also be growth at a (fixed) year-on-year fractional increase, or exponential growth. Such pathways are characterised by the same mathematical form, but now R is positive rather than negative, and the cumulative emissions are now *not* constrained by any finite limit. Cumulative quotas can still be calculated up to some fixed time horizon: but because they would implicitly continue growing beyond that time horizon, they are not associated with any meaningful global temperature limit.

As a special case, $R = 0$% represents constant or "flat-lined" emissions.

Mathematical background

Geometric sequence

The discussion so far has been in terms of "exponential" functions which, in a technical mathematical sense would be functions of a continuous independent variable (time). In practice, GHG inventory reporting is in terms of a discrete time variable (normally *annual* emissions). Accordingly, the formal mathematical treatment is not an exponential function of a continuous variable but rather takes the form of a *geometric sequence* where each succeeding element of the sequence is in a fixed proportion to the previous element.

A geometric sequence is characterised in general as:

$$x(k) = x_0 r^k$$

where: k represents the time (year) relative to some arbitrary base year (0); x_0 is the emissions level in year 0; r is the (fixed) geometric ratio between the emissions in any given year and the previous year. Mitigation is represented by $r < 1$, and growth by $r > 1$ (and flatlining by $r = 1$). The year on year fractional *change*, introduced previously, is given by $R = (r - 1)$. (Aside: r can be taken as vaguely mnemonic for geometric *ratio*…)

Deriving $Q(n)$ and Q_∞ from x_0, R, n

The cumulative emissions over any given number of steps (years), n, is given by the sum of the series:

$$Q(n) = \sum_{k=0}^{k=(n-1)} x(k) = x_0 \left(\frac{1 - r^n}{1 - r} \right)$$

or, expressed directly in terms of R:

$$Q(n) = x_0 \left(\frac{1 - (R + 1)^n}{-R} \right) = x_0 \left(\frac{(R + 1)^n - 1}{R} \right)$$

Provided $r < 1$ (i.e., $R < 0$) this converges to a finite limit as $n \to \infty$:

$$Q_\infty = \frac{x_0}{1 - r} = \frac{x_0}{-R}$$

Deriving R from Q_∞

Conversely, we might ask: given a "current" emissions rate (x_0 in the base year, $k = 0$) and quota (Q_∞) what is the corresponding exponential mitigation rate (R)?

$$
\begin{aligned}
Q_\infty &= \frac{x_0}{1-r} \\
Q_\infty(1-r) &= x_0 \\
Q_\infty - rQ_\infty &= x_0 \\
rQ_\infty &= Q_\infty - x_0 \\
r &= \frac{Q_\infty - x_0}{Q_\infty} \\
r &= 1 - \left(\frac{x_0}{Q_\infty}\right)
\end{aligned}
$$

and:

$$
R = (r-1) = -\frac{x_0}{Q_\infty}
$$

Note that the convention here is that the year 0 emissions are *included* in the quota.

Deriving R from x_0, n, x_n

In this case we suppose we have two elements of the pathway (geometric series), x_0 and x_n (for some specified n) and we want to know the (unique) pathway, characterised by the corresponding R. The pathway could correspond to growth, flatlining or mitigation respectively, just depending on the two points.

$$
\begin{aligned}
x_n &= x_0 r^n \\
r^n &= \frac{x_n}{x_0} \\
n\log(r) &= \log\left(\frac{x_n}{x_0}\right) \\
\log(r) &= \frac{1}{n}\log\left(\frac{x_n}{x_0}\right) \\
r &= \exp\left(\frac{1}{n}\log\left(\frac{x_n}{x_0}\right)\right) \\
R = r - 1 &= \exp\left(\frac{1}{n}\log\left(\frac{x_n}{x_0}\right)\right) - 1
\end{aligned}
$$

In [1]:

```
# Some generic utility functions, based on the mathematical framework...
import math

def rate_from_quota(x_0, Q):
    R = -(x_0/Q)
    return R

def quota_from_rate(x_0, R):
    Q = -(x_0/R)
    return Q

def fixed_term_quota_from_rate(x_0, R, start_year, end_year):
    n = end_year - start_year + 1 # n is inclusive of both start and end year
    if (R == 0.0):
        Q_n = x_0 * n # Flatline
    else:
        Q_n = x_0 * (((R + 1.0)**n - 1.0)/R)
    return Q_n

def rate_from_two_points(year_0, x_0, year_n, x_n):
    n = year_n - year_0 + 1 # n is inclusive of both year_0 and year_n
    R = math.exp((1.0/n)*math.log(x_n/x_0)) - 1.0
    return R

# Test
print "Test 1: rate_from_quota()"
x_0 = 40.0
Q = 1000.0
print "In: x_0 = %5.3f" % x_0
print "In: Q = %5.2f" % Q
R = rate_from_quota(x_0, Q)
print "Out: R = %4.3f%%" % (R*100.0)

print "Test 2: quota_from_rate()"
print "In: x_0 = %5.3f" % x_0
print "In: R = %4.3f%%" % (R*100.0)
Q = quota_from_rate(x_0, R)
print "Out: Q = %5.2f" % Q

print "Test 3: fixed_term_quota_from_rate()"
print "In: x_0 = %5.3f" % x_0
print "In: R = %4.3f%%" % (R*100.0)
start_year = 2015
end_year = 2035
print "In: start_year = %4d" % start_year
print "In: end_year = %4d" % end_year
Q_n = fixed_term_quota_from_rate(x_0, R, start_year, end_year)
print "Out: Q_n = %5.2f" % Q_n

print "Test 4: rate_from_two_points()"
year_0 = 2015
year_n = 2050
x_n = x_0 * 0.2 # -80% over the pathway
print "In: year_0 = %4d" % year_0
print "In: x_0 = %5.3f" % x_0
print "In: year_n = %4d" % year_n
print "In: x_n = %5.3f" % x_n
R = rate_from_two_points(year_0, x_0, year_n, x_n)
```

```
print "Out: R = %4.3f%%" % (R*100.0)


```

```
Test 1: rate_from_quota()
In: x_0 = 40.000
In: Q = 1000.00
Out: R = -4.000%
Test 2: quota_from_rate()
In: x_0 = 40.000
In: R = -4.000%
Out: Q = 1000.00
Test 3: fixed_term_quota_from_rate()
In: x_0 = 40.000
In: R = -4.000%
In: start_year = 2015
In: end_year = 2035
Out: Q_n = 575.68
Test 4: rate_from_two_points()
In: year_0 = 2015
In: x_0 = 40.000
In: year_n = 2050
In: x_n = 8.000
Out: R = -4.372%
```

Input Data/Parameters

Sources:

- GCB estimates: IPCC Special Report on Global warming of 1.5°C (https://www.ipcc.ch/sr15/): Chapter 2, Mitigation Pathways Compatible with 1.5°C in the Context of Sustainable Development (https://www.ipcc.ch/site/assets/uploads/sites/2/2019/02/SR15_Chapter2_Low_Res.pdf), Section 2.2.2 (pp. 104-108).
- Global population: UN World Population Prospects 2017 (https://esa.un.org/unpd/wpp/DataQuery/)
- Global Emissions (2015, 2016, 2017): Global Carbon Project (https://doi.org/10.18160/GCP-2018) (dataset, .xlsx format)
- Irish population: Central Statistics Office of Ireland (CSO) (https://www.cso.ie/en/media/csoie/releasespublications/documents/population/2017/Chapter_1_Populatic
- Irish historical emissions: UNFCCC National Inventory Report (NIR) (https://unfccc.int/process/transparency-and-reporting/reporting-and-review-under-the-convention/greenhouse-gas-inventories-annex-i-parties/submissions/national-inventory-submissions-2017)
- Irish emissions projections: Environmental Protection Agency of Ireland (EPA) Ireland's Greenhouse Gas Emissions Projections 2016-2035 (http://www.epa.ie/pubs/reports/air/airemissions/ghgprojections/Ireland_2017_GHG_Emission_Projections 2035.xlsx)
- Irish National Mitigation Objective (NMO): Government of Ireland (2014) National Policy Position on Climate Action and Low Carbon Development (https://www.dccae.gov.ie/en-ie/climate-action/publications/Pages/National-Policy-Position.aspx)

In [2]:

```
GCB_from_2018_to_NettZero = {}
GCB_from_2018_to_NettZero['mid'] = 1.170E+06
    # SR15, Table 2.2 (p. 108), <+2C @66%, central estimate, MtCO2
    # Cumulative from 2018 to time of nett zero emission rate

GCB_range_SR15 = 0.5
    # SR15 (p. 107) aggregated GCB fractional uncertainty (±)

GCB_from_2018_to_NettZero['low'] = (GCB_from_2018_to_NettZero['mid']
    * (1.0 - GCB_range_SR15))
GCB_from_2018_to_NettZero['high'] = (GCB_from_2018_to_NettZero['mid']
    * (1.0 + GCB_range_SR15))

GCB_NettZero_to_2100 = -100.0E+03
    # SR15, p. 107, additional order-of-magnitude cumulative removals (negati
    # from time of nett zero emission rate to 2100 (to stabilize temperature
    # continuing Earth system feedbacks)

GCP_emissions_global_FFI_Gt = {'2015': 9.68,'2016': 9.74, '2017': 9.87 }
    # Global Carbon Project, GtC/yr
GCP_emissions_global_LU_Gt = {'2015': 1.62,'2016': 1.30, '2017': 1.39 }
    # Global Carbon Project, GtC/yr
GtC_MtCO2_multiplier = 3.664e3

GCP_emissions_global = {} # MtCO2/yr
for key in ['2015', '2016', '2017'] :
    GCP_emissions_global[key] = (
        GCP_emissions_global_FFI_Gt[key]
        + GCP_emissions_global_LU_Gt[key]) * GtC_MtCO2_multiplier

GCP_emissions_global_2015_to_2017 = sum(GCP_emissions_global.values())

GCB = {}
    # Global Carbon Budget, 2015-2100 (MtCO2, nett FFI+LU)
    # Combine components for 2015-2017 (historical) + 2018 to time of nett ze
    # emissions + time of nett zero emissions to 2100. Given relatively high
    # uncertainty, we round to 1e4 (MtCO2)
for key in ['low', 'mid', 'high'] :
    GCB_raw = (GCB_from_2018_to_NettZero[key] +
        GCP_emissions_global_2015_to_2017 + GCB_NettZero_to_2100)
    GCB[key] = round(GCB_raw / 1e4) * 1e4

emissions_IE = {
    '1990': 38.286,
    '2015': 42.776,
    '2035_WAM': 46.705,
    '2035_WEM': 49.508} # All MtCO2/yr, nett FFI+LU

emissions_IE_share = {'2015': emissions_IE['2015']/GCP_emissions_global['2015

pop_global = {'2015': 7.38E+09} # UNEP

pop_IE = {}
pop_IE['2011'] = 4.5883E+06
pop_IE['2016'] = 4.7619E+06
pop_IE['2015'] = pop_IE['2011'] + \
    (pop_IE['2016'] - pop_IE['2011'])*((2015.0-2011.0)/(2016.0-2011.0))
    # Linear interpolation
```

```
pop_IE_share = {'2015': pop_IE['2015']/pop_global['2015']}

pathway_start_year = 2015
NMO_target_year = 2050
NMO_pathway_n = NMO_target_year - pathway_start_year + 1
BAU_target_year = 2035
BAU_pathway_n = BAU_target_year - pathway_start_year + 1

print GCB

```

```
{'high': 1780000.0, 'low': 610000.0, 'mid': 1190000.0}
```

Output/result dataset container

Basically we creat a single, global, "scenarios" list in which to collect all results in a (somewhat) systematic format. Each element should a dataset (python dictionary) collecting all the data for one distinct "scenario". While there is a fixed set of possible keys for all scenarios, all scenarios will not have values for all keys - according to what makes sense for the particular scenario method; but all scenarios with the same 'method' value must provide values for the same set of keys. It can be compared to a single table in an SQL database, allowing that some rows will have null values for some columns (fields). (It would be relatively straightforward to dump it out to a database and/or a spreadsheet format if one wished...)

Main "extra" here is adopting a class structure so that we can introduce tailored display methods. See: [Custom Display Logic (for iPython notebooks) (https://github.com/ipython/ipython/blob/master/examples/IPython%20Kernel/Custom%20Display%20Logic.ipy]
This is currently tailored to display only a subset of the keys that may be present...

In [0]:

```
from IPython.display import (
    display, display_html, display_png, display_svg
)

class Scenario:
    def __init__(self):
        self.dict = {}

    def __repr__(self):
        slist = []
        #slist.append('[repr] ')
        dict = self.dict
        keys = dict.keys()
        if 'name' in keys:
            slist.append(dict['name'])
        if 'Q_2035' in keys:
            slist.append(', Q_2035: %.f' % dict['Q_2035'])
        if 'Q' in keys:
            slist.append(', Q: %.f' % dict['Q'])
        if 'Q_per_capita' in keys:
            slist.append(', Q_per_capita: %.f' % dict['Q_per_capita'])
        if 'R' in keys:
            slist.append(', R: %+3.2f%%' % (dict['R']*100.0))
        return ''.join(slist)

    def _repr_html_(self):
        slist = ['<tr>']
        #slist.append('<p><strong>[repr_html]</strong> ')
        dict = self.dict
        keys = dict.keys()
        value = ''
        if 'name' in keys:
            value = dict['name']
        slist.append('<td>%s</td>' % value)
        value = ''
        if 'Q_2035' in keys:
            value = '%.f' % dict['Q_2035']
        slist.append('<td>%s</td>' % value)
        value = ''
        if 'Q' in keys:
            value = '%.f' % dict['Q']
        slist.append('<td>%s</td>' % value)
        value = ''
        if 'Q_per_capita' in keys:
            value = '%.f' % dict['Q_per_capita']
        slist.append('<td>%s</td>' % value)
        value = ''
        if 'R' in keys:
            value = '%+3.2f%%' % (dict['R']*100.0)
        slist.append('<td>%s</td>' % value)
        slist.append('</tr>')
        return ''.join(slist)

class ScenarioSet:
    def __init__(self):
        self.list = []

    def __repr__(self):
```

```
        slist = []
        for s in self.list:
            slist.append(s.__repr__())
        #return ''.join(slist)
        return 'hello world'

    def _repr_html_(self):
        slist = []
        slist.append('<table>')
        slist.append('<tr>')
        slist.append('<th>Scenario</th>')
        slist.append('<th>Quota [2015,2035]</th>')
        slist.append('<th>Quota</th>')
        slist.append('<th>Quota per capita</th>')
        slist.append('<th>R</th>')
        slist.append('</tr>')
        for s in self.list:
            slist.append(s._repr_html_())
        slist.append('</table>')
        return ''.join(slist)


all_scenarios = ScenarioSet() # Global container

def clear_all_scenarios():
    all_scenarios.list = []

```

M1: Raupach

National CO_2 Quota derived from Global Carbon Budget (GCB)

Given the GCB range (from 2015) of Rogeli et al (2016) we calculate the national "CO_2 Quota" for Ireland, based on the allocation methods of Raupach et al (2014); this is also expressed in terms of equivalent per capita quota, and exponential mitigation pathway rate (R). The fixed term quota for 2015-2035 is also calculated for later use. Such a dataset (dictionary of key-value pairs) together represents the full results for one "scenario" for this method. We generate 9 Raupach scenarios in total (3 GCB variants by 3 values for the "sharing index").

In [4]:

```
def raupach_quotas_from_GCB(GCB_value):
    quotas = {}
    quotas['pop'] = pop_IE_share['2015'] * GCB_value
    quotas['inertia'] = emissions_IE_share['2015'] * GCB_value
    blend_w = 0.5
    quotas['blend'] = (quotas['pop'] * blend_w) + (quotas['inertia'] * (1-ble
    return quotas

def raupach_scenarios():
    scenarios = ScenarioSet()
    for GCB_name in ('low', 'mid', 'high'):
        GCB_value = GCB[GCB_name]
        quotas = raupach_quotas_from_GCB(GCB_value)
        for sharing in ('pop', 'blend', 'inertia'):
            s=Scenario()
            d=s.dict
            d['method'] = 'raupach'
            d['GCB_name'] = GCB_name
            d['GCB_value'] = GCB_value
            d['sharing'] = sharing
            d['name'] = d['method'] + '-' + GCB_name + 'GCB-' + sharing
            d['Q'] = quotas[sharing]
            d['Q_per_capita'] = (d['Q']/pop_IE['2015']) * 1e6 # Convert from
            d['R'] = rate_from_quota(emissions_IE['2015'], d['Q'])
            d['Q_2035'] = fixed_term_quota_from_rate(
                emissions_IE['2015'], d['R'], 2015, 2035)
            scenarios.list.append(s)
    return scenarios

clear_all_scenarios()
r_scenarios = raupach_scenarios()
display_html(r_scenarios)

all_scenarios.list.extend(r_scenarios.list)
#display_html(all_scenarios)
```

Scenario	Quota [2015,2035]	Quota	Quota per capita	R
raupach-lowGCB-pop	356	391	83	-10.95%
raupach-lowGCB-blend	429	510	108	-8.38%
raupach-lowGCB-inertia	486	630	133	-6.79%
raupach-midGCB-pop	536	762	161	-5.61%
raupach-midGCB-blend	600	996	211	-4.30%
raupach-midGCB-inertia	645	1229	260	-3.48%
raupach-highGCB-pop	629	1140	241	-3.75%
raupach-highGCB-blend	682	1490	315	-2.87%
raupach-highGCB-inertia	717	1839	389	-2.33%

M2: "Policy"

Based on "interpretations"/"extrapolations" of the NMO for EGBET CO2, "reduce by at least 80% relative to 1990 by 2050".

- NMO 2050 target applied to *all* CO_2 (FFI+LU: not just EGBET)
- Geometric pathway, extrapolated indefinitely (beyond 2050 NMO target point)
- Start in 2015 (Paris!)
- Two "ambition" variants:
  - low: -80% by 2050
  - high: -95% by 2050
- Given two points (2015, 2050), fit a pathway (calculate R)
- From there calc:
  - Q
  - Q_2035
  - Q_per_capita

In [5]:

```
def policy_scenarios():
    scenarios = ScenarioSet()
    year_0 = 2015
    x_0 = emissions_IE['2015']
    year_n = 2050
    target = {'low': -0.8, 'high': -0.95}
    for ambition_name in ('high', 'low'):
        s=Scenario()
        d=s.dict
        d['method'] = 'policy'
        d['ambition_name'] = ambition_name
        d['name'] = d['method'] + '-' + ambition_name + '-ambition'
        d['ambition_target'] = target[ambition_name]
        x_n = emissions_IE['1990'] * (1.0 + d['ambition_target'])
        d['R'] = rate_from_two_points(year_0, x_0, year_n, x_n)
        d['Q'] = quota_from_rate(x_0, d['R'])
        d['Q_per_capita'] = (d['Q']/pop_IE['2015']) * 1e6 # Convert from MtC
        d['Q_2035'] = fixed_term_quota_from_rate(
            emissions_IE['2015'], d['R'], 2015, 2035)
        scenarios.list.append(s)
    return scenarios

#clear_all_scenarios()
pol_scenarios = policy_scenarios()
display_html(pol_scenarios)

all_scenarios.list.extend(pol_scenarios.list)
#display_html(all_scenarios)

```

Scenario	Quota [2015,2035]	Quota	Quota per capita	R
policy-high-ambition	433	517	109	-8.27%
policy-low-ambition	581	917	194	-4.67%

M3: "Projections" (BAU)

Based on most recent (2017) EPA projections, extending to 2035:

- Two EPA-defined variants (recorded under "ambition" key, though none are "ambitious"!):
  - WEM: With Existing Measures

- WAM: With Additional Measures
- One added "baseline" variant:
  - FLAT: flatlining emissions at 2015 level
- We co-erce into geometric pathways, based on 2035 projected emissions. This discards additional pathway detail in the projections; but allows extraction of "comparable" R-values.
- As R ≥ 0 in all three cases, none have a finite Q_∞, and thus no Q_per_capita value either.
- We do not extrapolate between 2035 at all, but, for comparison purposes, we do calculate Q_2035.
- Calculated Q_2035 values will not precisely match cumulative EPA projections because the EPA pathways are not simply geometric; but they would not match anyway because we are looking at FFI+LU, whereas EPA is, at best, FFI.

In [6]:

```
def projections_scenarios():
    scenarios = ScenarioSet()
    year_0 = 2015
    x_0 = emissions_IE['2015']
    year_n = 2035
    x_projected = {'FLAT': emissions_IE['2015'], 'WAM': emissions_IE['2035_WA
    for ambition_name in ('FLAT', 'WAM', 'WEM'):
        s=Scenario()
        d=s.dict
        d['method'] = 'projections'
        d['ambition_name'] = ambition_name
        d['name'] = d['method'] + '-' + ambition_name
        d['emissions_2035'] = x_projected[ambition_name]
        x_n = d['emissions_2035']
        d['R'] = rate_from_two_points(year_0, x_0, year_n, x_n)
        d['Q_2035'] = fixed_term_quota_from_rate(
            emissions_IE['2015'], d['R'], 2015, 2035)
        scenarios.list.append(s)
    return scenarios

#clear_all_scenarios()
prj_scenarios = projections_scenarios()
display_html(prj_scenarios)

all_scenarios.list.extend(prj_scenarios.list)
#display_html(all_scenarios)

```

Scenario	Quota [2015,2035]	Quota	Quota per capita	R
projections-FLAT	898			+0.00%
projections-WAM	937			+0.42%
projections-WEM	964			+0.70%

Display all accumulated scenarios

In [7]:

```
display_html(all_scenarios)
```

Scenario	Quota [2015,2035]	Quota	Quota per capita	R
raupach-lowGCB-pop	356	391	83	-10.95%
raupach-lowGCB-blend	429	510	108	-8.38%
raupach-lowGCB-inertia	486	630	133	-6.79%
raupach-midGCB-pop	536	762	161	-5.61%
raupach-midGCB-blend	600	996	211	-4.30%
raupach-midGCB-inertia	645	1229	260	-3.48%
raupach-highGCB-pop	629	1140	241	-3.75%
raupach-highGCB-blend	682	1490	315	-2.87%
raupach-highGCB-inertia	717	1839	389	-2.33%
policy-high-ambition	433	517	109	-8.27%
policy-low-ambition	581	917	194	-4.67%
projections-FLAT	898			+0.00%
projections-WAM	937			+0.42%
projections-WEM	964			+0.70%

Extract Selected Summary Data

Extract a subset of the key-value entries for a subset of the scenarios. (In fact, the subsetting of the keys here is redundant for the moment, as `Scenario.__repr__()` is already limiting the displayed output...)

(To copy/paste the table output, see bookmarklet technique in [How can I copy to the clipboard the output of a cell in a Jupyter notebook? (https://stackoverflow.com/questions/44229820/how-can-i-copy-to-the-clipboard-the-output-of-a-cell-in-a-jupyter-notebook)](https://stackoverflow.com/questions/44229820/how-can-i-copy-to-the-clipboard-the-output-of-a-cell-in-a-jupyter-notebook))

In [8]:

```
def select_key_values(s_in):
    selected_keys = ['name', 'Q_2035', 'Q', 'Q_per_capita', 'R']
    s_out = Scenario()
    s_out.dict = {k : v for (k, v) in s_in.dict.items() if (k in selected_key
    return s_out

selected_names = ['raupach-lowGCB-pop', 'raupach-midGCB-blend',
                   'raupach-highGCB-inertia',
                  'policy-high-ambition', 'policy-low-ambition',
                  'projections-FLAT', 'projections-WAM','projections-WEM']

selected_scenarios = ScenarioSet()
selected_scenarios.list = [select_key_values(s) for s in all_scenarios.list
                           if (s.dict['name'] in selected_names)]

display(selected_scenarios)

```

Scenario	Quota [2015,2035]	Quota	Quota per capita	R
raupach-lowGCB-pop	356	391	83	-10.95%
raupach-midGCB-blend	600	996	211	-4.30%
raupach-highGCB-inertia	717	1839	389	-2.33%
policy-high-ambition	433	517	109	-8.27%
policy-low-ambition	581	917	194	-4.67%
projections-FLAT	898			+0.00%
projections-WAM	937			+0.42%
projections-WEM	964			+0.70%

In [0]:

```

```