Environ Health Perspect

DOI: 10.1289/EHP12080

Note to readers with disabilities: *EHP* strives to ensure that all journal content is accessible to all readers. However, some figures and Supplemental Material published in *EHP* articles may not conform to 508 standards due to the complexity of the information being presented. If you need assistance accessing journal content, please contact ehp508@niehs.nih.gov. Our staff will work with you to assess and meet your accessibility needs within 3 working days.

Supplemental Material

Climate Change, Summer Temperature, and Heat-Related Mortality in Finland: Multicohort Study with Projections for a Sustainable vs. Fossil-Fueled Future to 2050

Mika Kivimäki, G. David Batty, Jaana Pentti, Juuso Suomi, Solja T. Nyberg, Joonas Merikanto, Kalle Nordling, Jenni Ervasti, Sakari B. Suominen, Antti-Ilari Partanen, Sari Stenholm, Jukka Käyhkö, and Jussi Vahtera

Table of Contents

Statistical code

Supplement

Statistical code

Analysis of raw cmip6 warming trends

```
"""
Original file is located at
    https://colab.research.google.com/drive/1ZF-2d9Fbs0Uzh5_jNtiwRH6tOfzNDxMN

This Python code was developed on run on https://colab.research.google.com/. Written by
Kalle Nordling.
"""

import xarray as xr
import numpy as np
import pandas as pd
import fsspec
import gcsfs

df = pd.read_csv('https://storage.googleapis.com/cmip6/cmip6-zarr-consolidated-
stores.csv')
df_tas = df[(df.table_id.isin(['Amon'])) & (df.variable_id.isin(['tas'])) &
df.experiment_id.isin(['historical','ssp119','ssp585','ssp126']) & (df.member_id == 'r1i1p1f1')
]
#df_tas = df[(df.table_id.isin(['Amon'])) & (df.variable_id.isin(['tasmax','tas','tasmin']))]
df_tas

a=df_tas[df_tas.experiment_id == "ssp126"].source_id.values
b=df_tas[df_tas.experiment_id == "ssp585"].source_id.values
models=np.intersect1d(a,b)
models=list(models)
models.remove('ACCESS-CM2')
models.remove('ACCESS-ESM1-5')
df_tas = df[(df.table_id.isin(['Amon'])) & (df.variable_id.isin(['tas'])) &
df.experiment_id.isin(['historical','ssp119','ssp585','ssp126']) & (df.member_id == 'r1i1p1f1')
& df.source_id.isin(models) ]
df_tas[df_tas.experiment_id == "ssp119"].source_id.values

def load_data(source_id, expt_id,var):
    #get list of urls
    uri = df_tas[(df_tas.source_id == source_id) & (df_tas.experiment_id == expt_id) &
(df_tas.variable_id == var)].zstore
    #get list of member_id's
    member_id = df_tas[(df_tas.source_id == source_id) &
                    (df_tas.experiment_id == expt_id) & (df_tas.variable_id ==
var)].member_id#.values[0]
```

```
    #concentrate all members to single xarray dataframe
    dss = []
    for ur in uri.values:
        print(ur)
        mapper=gcs.get_mapper(ur)
        dss.append(xr.open_zarr(mapper, consolidated=True))
    try:
      ds= xr.concat(dss,dim='member_id').assign_coords(source_id=list(member_id))
      return ds
    except:
      return 0
source_ids = df_tas.source_id.unique()
results={}
gcs = gcsfs.GCSFileSystem(token='anon')
for exp in ['historical','ssp126','ssp119','ssp585']:
  results[exp]={}
  for model in source_ids:
    print(model,exp)
    data = load_data(model,exp,'tas')
    if data != 0:
      results[exp][model]=data#load_data(model,exp,'hurs')

results.items()

import matplotlib.pyplot as plt

def calcMean(data):
    if data is None:
        return None
    if ('longitude' in data.dims) and ('latitude' in data.dims):
        data = data.rename({'longitude':'lon', 'latitude': 'lat'})
    print(data)
    data = data.sel(lon=slice(20,31),lat=slice(60,71))
    weights = np.cos(np.deg2rad(data.lat))
    weights.name = "weights"
    data_weighted = data.weighted(weights)
    data_mean= data_weighted.mean(("lon", "lat"))
    return data_mean

gmst_dicts = {}

for exp, dic in results.items():
    tmp = {}
    for model, data in dic.items():
        if model == "ICON-ESM-LR":
          continue
        tmp[model] = calcMean(data)
```

```
    gmst_dicts[exp] = tmp

gmst_dicts

gmst_anoms={}
for exp, data in gmst_dicts.items():
    gmst_anoms[exp] = {}
    for model,data in data.items():
      gmst_anoms[exp][model] = (data -
(gmst_dicts['historical'][model].groupby('time.year').mean().sel(year=slice('1850','1900')).m
ean(dim=['year','member_id'])))

gmst_dicts.keys()

def is_amj(month):
    return (month >= 5) & (month <= 9)
glob =  gmst_dicts['historical']
dss = [ds.tas.sel(time=is_amj(ds['time.month'])).groupby('time.year').mean(dim='time') for
key, ds in glob.items()]
historical_cmip6=
xr.concat(dss,dim='source_id',coords='minimal').assign_coords(source_id=list(glob.keys()))


cmip_data={}

for exp in ['ssp119','ssp585','ssp126']:
    print(exp)
    glob = gmst_dicts[exp]
    dss = [ds.tas.sel(time=is_amj(ds['time.month'])).groupby('time.year').mean(dim='time') for
key, ds in glob.items()]
    tmp2=
xr.concat(dss,dim='source_id',coords='minimal').assign_coords(source_id=list(glob.keys()))


    models=list(gmst_dicts[exp].keys())
    hist_models=list(gmst_dicts['historical'].keys())
    sel_models=list(set(models).intersection(hist_models))

    hist_sel=historical_cmip6.sel(dict(source_id=sel_models))

    tmp2=tmp2.sel(dict(source_id=sel_models))
    hist_sel=hist_sel.sel(year=slice(1850,2014))
    sce= tmp2.sel(year=slice(2015,2100))
    sce_file=xr.concat([hist_sel,sce],dim='time').sum(dim='time')
    sce_file = sce_file.where(sce_file > 0)
```

```
    #sce_file-=273.15
    cmip_data[exp] = sce_file.load()

cmip_data[exp]

fig, ax = plt.subplots(1,1,figsize=[20,10])
colors = ['lime','tab:cyan', 'tab:blue', 'darkorange', 'tab:red']
for i,exp in enumerate(['ssp119','ssp126','ssp585']):
    tmp = cmip_data[exp]
    tmp=  tmp-(tmp.sel(year=slice('1850','1900')).mean(dim=['year','member_id']))

    data=tmp.isel(member_id=0).mean(dim='source_id')
    data.plot(label=exp,color=colors[i],linewidth=4)

    #for model in list(tmp.source_id):
    #
tmp.isel(member_id=0).sel(year=slice(1900,2020)).sel(source_id=model).plot(color=colors[i]
,linewidth=1,alpha=0.4)


plt.legend(fontsize=18)
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
plt.ylabel('Temperature  anomaly')
plt.show()

from sklearn.linear_model import LinearRegression


def calcTrend(data_in,year,step):
    linear_regressor = LinearRegression()
    stop=year+step
    if stop>2100:
        stop=2100
    X = data_in.sel(year=slice(str(year),str(stop))).year.to_masked_array().reshape(-1, 1)
    Y = data_in.sel(year=slice(str(year),str(stop))).to_masked_array().reshape(-1, 1)
    trend=np.nan
    try:
        linear_regressor.fit(X, Y)  # perform linear regression
        trend=linear_regressor.coef_[0]*10
    except:
        print('trend error')
    return trend

cmip6_trend_ssp={}
```

```
for ssp in ['ssp119','ssp126','ssp585']:
   cmip6_trend_ssp[ssp] = {}
   for m in cmip_data[ssp].isel(member_id=0).source_id:
      #try:
         name = str(m.to_masked_array())
         print(name)
         data =cmip_data[ssp].isel(member_id=0).sel(source_id=m)
         cmip6_trend_ssp[ssp][name] = {}
         for y_start in range(2020,2100,30):
            cmip6_trend_ssp[ssp][name][str(y_start)] = calcTrend(data,y_start,30)

#hist trend
cmip6_trend_hist = {}
for ssp in ['ssp585']:
   cmip6_trend_hist[ssp] = {}
   for m in cmip_data[ssp].isel(member_id=0).source_id:
      #try:
         name = str(m.to_masked_array())
         print(name)
         data =cmip_data[ssp].isel(member_id=0).sel(source_id=m)
         cmip6_trend_hist[ssp][name] = calcTrend(data,1980,40)

import seaborn as sns

def calcWmean(a,obs):
  a = np. squeeze(a)
  w=1/(np.abs(np.array(a)-obs))
  return np.average(a,weights=w), w

#e=['FGOALS-g3','EC-Earth3-Veg-LR','CAMS-CSM1-0']
sns.set()

plot_x={'2020':1,'2050':2,'2080':3}
plot_x2={'2020':1.25,'2050':2.25,'2080':3.25}
plot_x2={'hist':0.35,'2020':1.0,'2050':2.25,'2080':3.25}
fig,ax = plt.subplots(1,1,figsize=[20,15])

colors = ['lime','tab:cyan', 'tab:blue', 'darkorange', 'tab:red']

cmip6_mean_f={'2020':{'ssp119':[],'ssp126':[],'ssp245':[],'ssp370':[],'ssp585':[]},'2050':{'ssp1
19':[],'ssp126':[],'ssp245':[],'ssp370':[],'ssp585':[]},'2080':{'ssp119':[],'ssp126':[],'ssp245':[],'ss
p370':[],'ssp585':[]}}
cmip6_mean_hist=[]
for i,exp in enumerate(['ssp585']):
   for j,(model,trend) in enumerate(cmip6_trend_hist[exp].items()):
      # if model in exclude:
      #    continue
```

```
        if (j == 0):
            ax.plot(plot_x2['hist']+0.15,trend,'*',color='gray',label='Hist',markersize=15)
        else:
            ax.plot(plot_x2['hist']+0.15,trend,'*',color='gray',markersize=15)
        #print(model,trend)
        cmip6_mean_hist.append(trend)

for i,exp in enumerate(['ssp119','ssp126','ssp585']):
    for j,(model,period) in enumerate(cmip6_trend_ssp[exp].items()):
        #print(period)
        #if model in exclude:
        #    continue
        #print(model)
        for k,(pre,trend) in enumerate(period.items()):
            if pre != "2020":
              continue
            if (pre == "2020") and (model=="EC-Earth3-Veg"):
              print('add legend')
              ax.plot(plot_x2[pre]+(i+0.005)*0.1,trend,'*',color=colors[i],label=exp,markersize=15)
            else:
              ax.plot(plot_x2[pre]+(i+0.005)*0.1,trend,'*',color=colors[i],markersize=15)

            cmip6_mean_f[pre][exp].append(trend)

#for pre,mean in cmip6_mean_f.items():
        print(np.squeeze(np.array(cmip6_mean_hist)))
        hist_mean_cmip6, weights=calcWmean(np.array(cmip6_mean_hist),0.486)

ax.plot(plot_x2['hist']+0.15,hist_mean_cmip6,marker='o',color='k',markersize=20,fillstyle=N
one)

print('historical trend',np.nanmean(cmip6_mean_hist))
for pre,mean in cmip6_mean_f.items():
    for j,exp in enumerate(['ssp119','ssp126','ssp585']):
        a=np.array(cmip6_mean_f[pre][exp])
        # print(a.shape,weights.shape)

ax.plot(plot_x2[pre]+(j+0.005)*0.1,np.average(a),marker='o',color='k',markersize=20,fillstyle
=None)
        #print(pre,exp,np.nanmean(cmip6_mean_f[pre][exp]))

#ax[0].set_ylim(-1,1)
ax.set_ylabel(' Trend C°/decade' ,fontsize=25)
ax.set_xticks([0.5,1.05])#,2.25,3.35])
ax.set_xticklabels(['1980-2019','2020-2050'],fontsize=25)#,'2050-2080','2080-
2100'],fontsize=25)
#ax.set_xticklabels(['2020-2050'],fontsize=25)#,'2050-2080','2080-2100'],fontsize=25)
```

```
for tick in ax.yaxis.get_major_ticks():
   tick.label.set_fontsize(25)
hist_trend=0.486
ax.plot([-1,2],[hist_trend,hist_trend],linewidth=3,label='OBS')
ax.set_xlim(0.3,1.5)
ax.legend(fontsize=20)
plt.savefig('figures/figure.eps',dpi=1800)
plt.savefig('figures/figure.png')
#ax.set_title('Future warming trends',fontsize=20)

cmip6_mean_hist
print(len(cmip6_trend_ssp['ssp119']))
print(len(cmip6_trend_ssp['ssp126']))
print(len(cmip6_trend_ssp['ssp585']))

output={'historical':[],'ssp119':[],'ssp126':[],'ssp585':[]}
print(len(models))
for model in models:
  output['historical'].append(float(cmip6_trend_hist['ssp585'][model]))
  output['ssp126'].append(float(cmip6_trend_ssp['ssp126'][model]['2020']))
  output['ssp585'].append(float(cmip6_trend_ssp['ssp585'][model]['2020']))

  if model in list(cmip6_trend_ssp['ssp119'].keys()):
     output['ssp119'].append(float(cmip6_trend_ssp['ssp119'][model]['2020']))
  else:
     output['ssp119'].append(np.nan)

df=pd.DataFrame(data=output,index=models)
print(df)
df.to_csv('data/cmip6_raw_trends.csv')
```

Analysis of weighted cmip6 warming trends

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Jun 12 10:00:51 2023

@author: Antti-Ilari Partanen (antti-ilari.partanen@fmi.fi)
"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pathlib import Path
from scipy.stats import norm, gaussian_kde

'''

IPCC AR6 TS.3.2 Climate Sensitivity and Earth System Feedbacks:
"Based on process understanding, warming over the instrumental
record, and emergent constraints, the best estimate of TCR is 1.8°C,
the likely range is 1.4°C to 2.2°C and the very likely range is 1.2°C to
2.4°C. There is a high level of agreement among the different lines of
evidence (Figure TS.16c) (high confidence). {7.5.5}""

'''


datadir=Path('data')
inputdir=datadir / 'input'
outputdir=datadir / 'output'
figdir=Path('figures')



#open CMIP6 data
df_cmip6=pd.read_csv(ouputdir / 'cmip6_raw_trends.csv', index_col=0)

def add_str(string_to_add, list_of_strings_in):
    '''
    Adds a string to each string of a list

    Parameters
    ----------
    string_to_add : str
        String to be appended at the end of each string in list_of_strings_in.
    list_of_strings_in : list
```

```
        A list of strings.

    Returns
    -------
    list_of_strings_out : list
        List of strings with string_to_add appended at the end of each string.

    '''
    list_of_strings_out=list_of_strings_in.copy()
    for i,string in enumerate(list_of_strings_out):
        list_of_strings_out[i]=string+string_to_add
    return list_of_strings_out

# Re-Calculate warming rates
ssps3=['ssp119','ssp126', 'ssp585']
ssps2=['ssp126', 'ssp585']
obs_warming_trend=0.486 # K / decade
obs_heat_trend=0.56 # K / decade

# Calculate scaled trends for SSPs
df_cmip6[add_str('-scaled',ssps3)]=df_cmip6[ssps3].div(df_cmip6['historical'],
axis=0)*obs_warming_trend


# Calculated heat index trends
df_cmip6[add_str('-heat_index_scaled',ssps3)]=df_cmip6[ssps3].div(df_cmip6['historical'],
axis=0)*obs_heat_trend


# Set TCR values for models based on data compiled by Hausfather et al. (2022)
# Hausfather Z, Marvel K, Schmidt GA, Nielsen-Gammon JW, Zelinka M. Climate simulations:
# recognize the 'hot model' problem. Nature 2022

models_list=['GFDL-ESM4', 'IPSL-CM6A-LR', 'BCC-CSM2-MR', 'MIROC6', 'MRI-ESM2-0',
      'CESM2-WACCM', 'CanESM5', 'AWI-CM-1-1-MR', 'INM-CM4-8',
      'INM-CM5-0', 'NESM3', 'CAMS-CSM1-0', 'MPI-ESM1-2-LR',
      'MPI-ESM1-2-HR', 'NorESM2-LM', 'FGOALS-g3', 'KACE-1-0-G',
      'FGOALS-f3-L', 'NorESM2-MM', 'FIO-ESM-2-0', 'IITM-ESM',
      'EC-Earth3-Veg-LR', 'EC-Earth3-Veg', 'EC-Earth3', 'CIESM',
      'CMCC-CM2-SR5', 'TaiESM1', 'CAS-ESM2-0', 'CMCC-ESM2', 'KIOST-ESM']

tcr_values=np.array([[1.63007205],
      [2.34884317],
      [1.54689164],
      [1.54890551],
      [1.66674546],
      [1.91474622],
```

```
        [2.71181275],
        [2.03181034],
        [1.29919935],
        [1.41020129],
        [2.72],
        [1.72800426],
        [1.82089151],
        [1.63765083],
        [1.49095073],
        [1.50335498],
        [2.04],
        [1.94],
        [1.22383802],
        [2.22],
        [1.65624099],
        [np.NaN],
        [2.65642223],
        [2.3 ],
        [2.24969205],
        [2.14218748],
        [2.2698537 ],
        [np.NaN],
        [1.9230164 ],
        [np.NaN]])

data_tcr=pd.DataFrame(tcr_values,index=models_list,columns=['TCR'])
df_cmip6['TCR']=data_tcr['TCR'].loc[df_cmip6.index]

#Remove models without TCR value
df_cmip6.dropna(subset=['TCR'], inplace=True)


#Formulate TCR distribution based on the IPCC report
NINETY_TO_ONESIGMA = norm.ppf(0.95)
tcr_norm=norm(loc=1.8, scale=0.6 / NINETY_TO_ONESIGMA)

# Calculate weights based on TCR
df_cmip6['weight']=tcr_norm.pdf(df_cmip6['TCR'])
df_cmip6['weight']=df_cmip6['weight']/df_cmip6['weight'].sum()



def calculate_means(df_in,ssps):
    df_out=df_in.copy()
    scens='historical|TCR|weight'
    for ssp in ssps:
        scens=scens+'|'+ssp
```

```
    #Select only certain ssps and drop rows with NaN values
    df_out=df_out.filter(regex=scens).dropna(axis=0)
    # ssps_scaled=['historical']+ssps+add_str('-scaled', ssps)
    # df_out=df_cmip6[ssps_scaled+['TCR','weight']]
    #Normalize weights so they sum to 1
    df_out.loc[:,'weight']=df_out['weight']/df_out['weight'].sum()

    df_out.loc['posterior_mean']=df_out.iloc[:,0:-1].mul(df_out['weight'], axis=0).sum(axis=0)
    df_out.loc['prior_mean']=df_out.mean(axis=0)

    return df_out

#Calculate posterior trends from models with data for all SSPs and TCR
df_all_ssps=calculate_means(df_cmip6, ssps3)


# Calculate means for SSP1-2.6 and SSP5-8.5
df_all_models=calculate_means(df_cmip6, ssps2)


def calculate_kdes(df):
    columns=df.columns[0:-1] #Select all except weights
    kde_posterior=dict()
    kde_prior=dict()
    for column in columns:
        kde_prior[column]=gaussian_kde(df[column][0:-2])
        kde_posterior[column]=gaussian_kde(df[column][0:-2],weights=df['weight'][0:-2])
    return kde_prior, kde_posterior

kde_prior_all_ssps, kde_posterior_all_ssps=calculate_kdes(df_all_ssps)
kde_prior_all_models, kde_posterior_all_models=calculate_kdes(df_all_models)


# Calculate credible intervals for the prior and posterior distributions
def calculate_intervals(df_in,ssps,kde, kde_posterior):
    df_out=df_in.copy()
    intervals=dict()
    intervals_posterior=dict()

    scens=df_out.columns[:-1]

    for scen in scens:
        n=1000000
        seed=1
        intervals[scen]=np.quantile(kde[scen].resample(n,seed=seed),q=[0.025,0.5, 0.975])
```

```
intervals_posterior[scen]=np.quantile(kde_posterior[scen].resample(n,seed=seed),q=[0.025,
0.5, 0.975])
        df_out.loc['ci95_prior_low', scen]=intervals[scen][0]
        df_out.loc['ci95_prior_high', scen]=intervals[scen][2]
        df_out.loc['prior_median', scen]=intervals[scen][1]
        df_out.loc['ci95_posterior_low', scen]=intervals_posterior[scen][0]
        df_out.loc['ci95_posterior_high', scen]=intervals_posterior[scen][2]
        df_out.loc['posterior_median', scen]=intervals_posterior[scen][1]

    return intervals, intervals_posterior, df_out


intervals_all_ssps, intervals_posterior_all_ssps,
df_all_ssps=calculate_intervals(df_all_ssps,ssps3,kde_prior_all_ssps,
kde_posterior_all_ssps)
intervals_all_models, intervals_posterior_all_models,
df_all_models=calculate_intervals(df_all_models,ssps2,kde_prior_all_models,
kde_posterior_all_models)

def visualize(df, intervals, intervals_posterior, title='', figname=None,
cut_warming_rate=False):

    #Do not plot intervals, means and medians along model data
    models=df.index[0:-8]

    fig,ax=plt.subplots(1,1, figsize=(7.5,6))

    #Select columns excluding TCR and weight
    xticklabels=df.columns[0:-2].sort_values()


    for i,column in enumerate(xticklabels):

        if i==len(xticklabels)-1:
            label_ci='95% credible interval (prior)'
            label_ci_posterior='95% credible interval (posterior)'
            label_scatter='Individual models'
            label_posterior_mean='Posterior mean'
            label_mean='Prior mean'
        else:
            label_ci='_no_legend'
            label_ci_posterior='_no_legend'
            label_scatter='_no_legend'
            label_posterior_mean='_no_legend'
            label_mean='_no_legend'
```

```
        #Plot individula models
        sc=ax.scatter(i*np.ones(len(df.loc[models, column])),
                df.loc[models, column],c=df.loc[models,'weight'],
                cmap='GnBu',edgecolors='black', label=label_scatter)



        #Plot parameters on how far away from the model scatter line draw
        # CIs and mean values
        prior_inc=0.3
        posterior_inc=0.15

        #Plot 95% CI for posterior results
        ax.plot((i+posterior_inc)*np.ones(3),intervals_posterior[column],marker='o',
color='gray', label=label_ci_posterior)

        #Plot 95% CI for prior results
        ax.plot((i+prior_inc)*np.ones(3),intervals[column],marker='o', color='black',
label=label_ci)

        #Plot posterior means
        ax.scatter([i+posterior_inc], df.loc['posterior_mean', column],marker='x',
color='gray',s=100, label=label_posterior_mean)

        #Plot unposterior means
        ax.scatter([i+prior_inc], df.loc['prior_mean', column],marker='x', color='black',s=100,
label=label_mean)

    #Plot observed trend
    ax.axhline(y=obs_warming_trend, label='Observed trend', color='gray', linestyle='dashed')


    ax.legend()
    ax.set_ylabel('May-September warming rate in Finland (°C/decade)')
    ax.set_xticks(np.arange(len(xticklabels)))
    ax.set_xticklabels(xticklabels, minor=False, rotation=45)
    cbar=fig.colorbar(sc)
    cbar.set_label('Weight')
    ax.set_title(title)
    if cut_warming_rate:
        ax.set_ylim([-1.25,3.3])
    fig.savefig(figname, dpi=150)

visualize(df_all_ssps, intervals_all_ssps, intervals_posterior_all_ssps, title='SSPs 1-1.9, 1-2.6
and 5-8.5 with 8 models', figname=figdir / 'scatter_all_ssps.png')
visualize(df_all_models, intervals_all_models, intervals_posterior_all_models, title='SSPs  1-
2.6 and 5-8.5 with 27 models', figname=figdir / 'scatter_all_models.png')
```

```
visualize(df_all_models, intervals_all_models, intervals_posterior_all_models, title='SSPs  1-
2.6 and 5-8.5 with 27 models', figname=figdir /
'scatter_all_models_cut_warming_rate.png',cut_warming_rate=True )
# visualize(df_all_models)

def plot_pdfs(df,ssps,kde_prior, kde_posterior, figname):
   fig,ax=plt.subplots(len(ssps),2)
   i=0
   j=0
   for ssp in ssps:
      j=0
      for method in ['','-scaled']:
         scen=ssp+method
         grid=np.linspace(-1,3, 10000)

         # Plot posterior histogram and kde
         df[scen][0:-2].hist(weights=df['weight'][0:-2], bins=8, density=True, ax=ax[i,j],
alpha=0.5, color='orange',label='posterior histogram')
         ax[i,j].plot(grid,kde_posterior[scen].pdf(grid), color='orange',label='Posterior  KDE')

         # Plot unposterior histogram and kde
         df[scen][0:-2].hist(bins=8, density=True, ax=ax[i,j], alpha=0.5, color='blue',label='Prior
histogram')
         ax[i,j].set_title(scen)
         j+=1


      i+=1
   ax[i-1,j-1].legend(bbox_to_anchor=(1.05, 1))
   fig.savefig(figname, dpi=150, bbox_inches='tight')

plot_pdfs(df_all_ssps, ssps3, kde_prior_all_ssps, kde_posterior_all_ssps, figdir /
'warming_rate_pdfs_all_ssps.png')
plot_pdfs(df_all_models, ssps2, kde_prior_all_models, kde_posterior_all_models, figdir /
'warming_rate_pdfs_all_models.png')

# Plot TCR distributions
fig, ax=plt.subplots(1,1)
tcr_grid=np.linspace(0,4,10000)
ax.plot(tcr_grid,tcr_norm.pdf(tcr_grid), label='IPCC AR6')
ax.plot(tcr_grid,kde_prior_all_ssps['TCR'].pdf(tcr_grid), label='Prior with all SSPs')
ax.plot(tcr_grid,kde_posterior_all_ssps['TCR'].pdf(tcr_grid), label='Posterior with all SSPs')
ax.plot(tcr_grid,kde_prior_all_models['TCR'].pdf(tcr_grid), label='Prior with all mdodels')
ax.plot(tcr_grid,kde_posterior_all_models['TCR'].pdf(tcr_grid), label='Posterior with all
models')
ax.set_xlabel('Transient climate response °C')
ax.legend()
```

```
fig.savefig(figdir / 'TCR_pdfs.png', dpi=150)

#Plot scatter plot with TCR against warming rates
def plot_scatter_TCR_warming_rate(df, title, figname, cut_warming_rate=False):
    models=df.index[0:-8]
    fig, ax=plt.subplots(1,1)
    for scen in df.columns[0:-2]:
        ax.scatter(df.loc[models, 'TCR'],df.loc[models, scen],label=scen, s=5)

    ax.legend()
    ax.set_xlabel('Transient climate response °C')
    ax.set_ylabel('May-September warming rate°C/decade')
    ax.set_title(title)
    if cut_warming_rate:
        ax.set_ylim([-0.8, 4.1])
    fig.savefig(figname, dpi=150)

plot_scatter_TCR_warming_rate(df_all_models, 'SSPs  1-2.6 and 5-8.5 with 27 models', figdir
/ 'scatter_TCR_warmig_rate_all_models.png')
plot_scatter_TCR_warming_rate(df_all_models, 'SSPs  1-2.6 and 5-8.5 with 27 models', figdir
/ 'scatter_TCR_warmig_rate_all_models_cut_warming_rate.png', cut_warming_rate=True)
plot_scatter_TCR_warming_rate(df_all_ssps, 'SSPs 1-1.9, 1-2.6 and 5-8.5 with 8 models',
figdir / 'scatter_TCR_warmig_rate_all_ssps.png')



#Save data into Excel
with pd.ExcelWriter(outputdir/'processsed_cmip6_data.xlsx') as writer:

    # use to_excel function and specify the sheet_name and index
    # to store the dataframe in specified sheet
    df_all_ssps.to_excel(writer, sheet_name="All SSPs")
    df_all_models.to_excel(writer, sheet_name="All models")
```

Analysis of heat – mortality associations

Statistical code (SAS, version 9.4)

```
*********************************************************;
***  Table 3-4, Figure 6-7 (population model)  ***;
*********************************************************;
data heat1;
set heat;
HI=round(HI7); *7-day heat index*;
*HI=round(HI2); *2-day heat index*;
*HI=round(Tmax7); *7-day Tmax*;
*HI=round(Tmax2); *2-day Tmax*;
if .<HI<=13 then HI_class6=1;
if 13<HI<=15 then HI_class6=2;
if 15<HI<=17 then HI_class6=3;
if 17<HI<=19 then HI_class6=4;
if HI=20 then HI_class6=5;
if HI>20 then HI_class6=6;
/*
if .<HI<=13 then HI_class6=1;
if 13<HI<=20 then HI_class6=2;
if HI>20 then HI_class6=6;
*/
run;
proc freq data=heat1 noprint;
*tables HI_class6*year*dgno / out=k5; *model 1*;
tables HI_class6*year*age*sex*dgno / out=k5; *model 2*;
*tables HI_class6*year*age*sex*educ*dgno / out=k5; *model 3*;
*tables HI_class6*year*age*sex*educ*building*ndvi*deprivation*pop*dgno / out=k5;
*model 4*;
*tables
HI_class6*year*age*sex*educ*building*ndvi*deprivation*pop*obesity*smoking*alco*inact
ive*dgno / out=k5; *model 5*;
data k5; set k5;
ln_ftime=log(1/1500000); ** per 10000 summers **;
run;
%macro heatpop (dg,resfile);
data k5; set k5;
vdg=&dg;
dead=0;
if dgno=vdg then dead=1;
proc means data=k5 nway noprint; where dead=1;
var count; class HI_class6; output out=freqe sum=Event;
proc means data=k5 nway noprint;
var count; class HI_class6; output out=freqp sum=pdays;
data freqp; set freqp; Py10000=pdays/1500000;
```

```
proc genmod data=k5;
    class HI_class6 year;
*    model dead = year HI_class6 / dist=poisson offset=ln_ftime; *model 1*;
    model dead = year age sex HI_class6 / dist=poisson offset=ln_ftime; *model 2*;
*    model dead = year age sex educ HI_class6 / dist=poisson offset=ln_ftime; *model 3*;
*    model dead = year age sex educ building ndvi deprivation pop HI_class6 / dist=poisson
offset=ln_ftime; *model 4*;
*    model dead = year age sex educ building ndvi deprivation pop obesity smoking alco
inactive HI_class6 / dist=poisson offset=ln_ftime; *model 5*;
    weight count;
    lsmeans HI_class6 / exp cl;
    estimate '1 vs 2' HI_class6 1 -1 0 0 0 0;
    estimate '3 vs 2' HI_class6 0 -1 1 0 0 0;
    estimate '4 vs 2' HI_class6 0 -1 0 1 0 0;
    estimate '5 vs 2' HI_class6 0 -1 0 0 1 0;
    estimate '6 vs 2' HI_class6 0 -1 0 0 0 1;

*     estimate '1 vs 2' HI_class6 1 -1 0;
*     estimate '6 vs 2' HI_class6 0 -1 1;
    ods output Estimates=rr LSMeans=rate;
data rate; set rate; rate_10000py=ExpEstimate; rate_low=LowerExp; rate_upp=UpperExp;
data rr; set rr; HI_class6=1*substr(Label,1,1); RR=MeanEstimate; RR_low=MeanLowerCL;
RR_upp=MeanUpperCL;
data res; merge freqe freqp rate rr; by HI_class6; dgno=&dg; if RR=. then RR=1;
data res; merge res(in=i) lib.legends; by dgno; if i;
run;
proc append base=&resfile data=res;
run;
%mend;
*******************************************************;
proc datasets lib=work memtype=data nolist; delete results; quit;
%heatpop(99,results); *Any*;
%heatpop(4,results);  *Cancer*;
%heatpop(32,results); *CVD*;
%heatpop(73,results); *External*;
%heatpop(792,results); *Other*;
proc print data=results;
    id dgno;
    var legend HI_class6 Event Py10000 rate_10000py rate_low rate_upp RR RR_low RR_upp
ProbChiSq;
run;

*******************************************************;
***  Figure 5-7 (case-crossover)  ***;
*******************************************************;
%macro heatcase (dg,resfile);
data d3; set cases;
```

```
    HI=round(HI7); *7-day heat index*;
   *HI=round(HI2); *2-day heat index*;
   *HI=round(Tmax7); *7-day Tmax*;
   *HI=round(Tmax2); *2-day Tmax*;
    if .<HI<=20 then HI_class2=0;
    if HI>20 then HI_class2=1;
    vdg=&dg;
    dead=0;
    if dgno=vdg then dead=1;
  IF dead=1 or controlday IN (-365,365); *1 year before and after death*;
* IF dead=1 or controlmonth=1; *same day of the weeks at the month*;
proc logistic data=d3 descending;
    class HI_class2(ref='0') / param=ref;
    model dead = HI_class2;
    strata id;
    ods output ParameterEstimates=pe OddsRatios=oe ResponseProfile=re;
data oe; set oe; OR=OddsRatioEst; OR_low=LowerCL; OR_upp=UpperCL; keep Effect OR
OR_low OR_upp;
data pe; set pe; keep ProbChiSq;
data re; set re; if Outcome=1; keep count;
data res; merge oe pe re; dgno=&dg; deaths=count;
data res; merge res(in=i) lib.legends; by dgno; if i;
keep legend dgno deaths OR OR_low OR_upp ProbChiSq;
run;
proc append base=&resfile data=res;
run;
%mend;
*******************************************************;
proc datasets lib=work memtype=data nolist; delete results; quit;
%heatcase(99,results); *Any*;
%heatcase(4,results);  *Cancer*;
%heatcase(32,results); *CVD*;
%heatcase(73,results); *External*;
%heatcase(792,results); *Other*;
proc print data=results;
    id dgno;
    var legend deaths OR OR_low OR_upp ProbChiSq;
run;

*********************************************************************;
***  Figure 8 (PAF, bootstrap 95% confidence interval)  ***;
*********************************************************************;
proc format;
value hif -10-13='1' 14-20='2' 21-35='6';
data heat1;
set heat;
HI=round(HI7); *Observed heat index*;
```

```
proc freq data=heat1 noprint;
tables id*HI*year*dgno / out=obs1;
format HI hif.;
run;
data person1;
set obs1; by id; if first.id;
keep id;
run;
*******************************************************;
%macro PAF (seed,resfile);
proc surveyselect data=person1 noprint seed=&seed
   out=BootSample method=urs samprate=1 reps=1;
run;
data Bobs2;
   merge obs1(in=i) BootSample(in=j);
   by id;
   if i and j;
data Bobs2;
   set Bobs2;
   count=NumberHits*count;
run;
proc freq data=obs1 noprint; *original sample*;
*proc freq data=Bobs2 noprint; *bootsample*;
   tables HI*year*dgno / out=Bobs2;
   weight count;
run;
*******************************************************;
data obs3; set Bobs2;
   dead=0;
   if dgno=32 then dead=1; ** CVD **;
   ln_ftime=log(1/1500000); ** per 10000 py **;
run;
proc genmod data=obs3;
   class HI year;
   model dead = year HI / dist=poisson offset=ln_ftime; weight count;
   estimate '1 vs 2' HI 1 -1 0;
   estimate '6 vs 2' HI 0 -1 1;
   ods output Estimates=rr;
   data rr1; set rr; if substr(Label,1,1)=1; RR1=MeanEstimate;
   data rr6; set rr; if substr(Label,1,1)=6; RR6=MeanEstimate;
run;
data res; merge rr1 rr6;
   keep RR1 RR6;
proc append base=&resfile data=res;
run;
%mend;
*******************************************************;
```

```
** original sample **;
proc datasets lib=work memtype=data nolist; delete OutStats0; quit;
%PAF(1,OutStats0);
** 500 bootsamples **;
proc datasets lib=work memtype=data nolist; delete OutStats1; quit;
%MACRO PAF2;
%DO I = 1 %TO 500;
   %PAF(&I,OutStats1);
%END;
%MEND PAF2;
%PAF2;
data outstats_all;
set OutStats0 OutStats1; ** original sample + bootsample **;
retain sample -1;
sample=sample+1;
run;
*******************************************************;
** PAF 95% confidence interval **;
data b1;
   set outstats_all;
   ************************************;
   ******* year 2010 *******;
   ** Observed **;
   prop1=0.42376; prop6=0.05259;
/*
   ******* year 2040 *******;
   ** Sustainable (SSP119) **;
   prop1=0.3643; prop6=0.0782;
   ** Current-warming trend **;
   prop1=0.30501; prop6=0.1280;
   ** Fossil-fueled (SSP585) **;
   prop1=0.2958; prop6=0.1376;
   ** 1C change per decade **;
   prop1=0.2084; prop6=0.2969;
*/
   ************************************;
   denom=1+prop1*(RR1-1)+prop6*(RR6-1);
   paf1=prop1*(RR1-1)/denom*100;
   paf6=prop6*(RR6-1)/denom*100;
   days1=prop1*100;
   days6=prop6*100;
run;
*******************************************************;
** original estimates **;
data b0;
   set b1;
  IF sample=0;
```

```
run;
** PAF, 95% confidence interval (2.5% & 97.5%) **;
data b2;
    set b1;
  IF sample>0;
run;
proc sort data=b2; by paf1;
data p21; set b2; if _n_=round(0.025*500); paf1_low=paf1; keep paf1_low;
data p22; set b2; if _n_=round(0.975*500); paf1_upp=paf1; keep paf1_upp;

proc sort data=b2; by paf6;
data p61; set b2; if _n_=round(0.025*500); paf6_low=paf6; keep paf6_low;
data p62; set b2; if _n_=round(0.975*500); paf6_upp=paf6; keep paf6_upp;
run;
data b6;
    merge b0 p21 p22 p61 p62;
run;
proc print data=b6;
    var days1 RR1 paf1 paf1_low paf1_upp
       days6 RR6 paf6 paf6_low paf6_upp;
run;

*********************************************************************;
***  Table 5 (WEIGHTED RR & PAF, bootstrap 95% confidence interval)  ***;
*********************************************************************;
proc format;
value hif -10-13='1' 14-20='2' 21-35='6';
data heat1;
set heat;
HI=round(HI7); *Observed heat index*;
if sex=1 and .<age<65 then sexage=1;
if sex=1 and age>=65 then sexage=2;
if sex=2 and .<age<65 then sexage=3;
if sex=2 and age>=65 then sexage=4;
proc freq data=heat1 noprint;
tables id*HI*year*sexage*dgno / out=obs1;
format HI hif.;
run;
data person1;
set obs1; by id; if first.id;
keep id;
run;
*******************************************************;
%macro PAF (seed,resfile);
proc surveyselect data=person1 noprint seed=&seed
    out=BootSample method=urs samprate=1 reps=1;
run;
```

```
data Bobs2;
   merge obs1(in=i) BootSample(in=j);
   by id;
   if i and j;
data Bobs2;
   set Bobs2;
   count=NumberHits*count;
run;
proc freq data=obs1 noprint; *original sample*;
*proc freq data=Bobs2 noprint; *bootsample*;
   tables sexage*HI*year*dgno / out=Bobs2;
   weight count;
run;
*******************************************************;
data obs3; set Bobs2;
   dead=0;
   if dgnro=32 then dead=1; ** CVD **;
   ln_ftime=log(1/1500000); ** per 10000 py **;
run;
proc genmod data=obs3;
   class HI year;
   model dead = year HI / dist=poisson offset=ln_ftime; weight count;
   estimate '1 vs 2' HI 1 -1 0;
   estimate '6 vs 2' HI 0 -1 1;
   by sexage;
   ods output Estimates=rr;
   data rr11; set rr; if sexage=1 and substr(Label,1,1)=1; RR11=MeanEstimate;
   data rr16; set rr; if sexage=1 and substr(Label,1,1)=6; RR16=MeanEstimate;
   data rr21; set rr; if sexage=2 and substr(Label,1,1)=1; RR21=MeanEstimate;
   data rr26; set rr; if sexage=2 and substr(Label,1,1)=6; RR26=MeanEstimate;
   data rr31; set rr; if sexage=3 and substr(Label,1,1)=1; RR31=MeanEstimate;
   data rr36; set rr; if sexage=3 and substr(Label,1,1)=6; RR36=MeanEstimate;
   data rr41; set rr; if sexage=4 and substr(Label,1,1)=1; RR41=MeanEstimate;
   data rr46; set rr; if sexage=4 and substr(Label,1,1)=6; RR46=MeanEstimate;
run;
data res; merge rr11 rr16 rr21 rr26 rr31 rr36 rr41 rr46;
   keep RR11 RR16 RR21 RR26 RR31 RR36 RR41 RR46;
proc append base=&resfile data=res;
run;
%mend;
*******************************************************;
** original sample **;
proc datasets lib=work memtype=data nolist; delete OutStats0; quit;
%PAF(1,OutStats0);
** 500 bootsamples **;
proc datasets lib=work memtype=data nolist; delete OutStats1; quit;
%MACRO PAF2;
```

```
%DO I = 1 %TO 500;
   %PAF(&I,OutStats1);
%END;
%MEND PAF2;
%PAF2;
data woutstats_all;
set OutStats0 OutStats1; ** original sample + bootsample **;
retain sample -1;
sample=sample+1;
run;
*******************************************************;
** weighted RR & PAF **;
data b1;
   set woutstats_all;
   ************************************;
   ******* year 2010 *******;
   RR1=0.40497*RR11+0.06896*RR21+0.41894*RR31+0.10713*RR41;
   RR6=0.40497*RR16+0.06896*RR26+0.41894*RR36+0.10713*RR46;
   cvd=2040;
   ** Observed **;
   prop1=0.42376; prop6=0.05259;
/*
   ******* year 2040 *******;
   RR1=0.38291*RR11+0.10482*RR21+0.37097*RR31+0.14130*RR41;
   RR6=0.38291*RR16+0.10482*RR26+0.37097*RR36+0.14130*RR46;
   cvd=3665;
   ** Sustainable (SSP119) **;
   prop1=0.3635; prop6=0.0787;
   ** Current-warming trend **;
   prop1=0.30501; prop6=0.12800;
   ** Fossil-fueled (SSP585) **;
   prop1=0.2971; prop6=0.1364;
   ** 0C change perdecade **;
   prop1=0.42376; prop6=0.05259;
   ** 1C change per decade **;
   prop1=0.2084; prop6=0.2969;
*/
   ************************************;
   denom=1+prop1*(RR1-1)+prop6*(RR6-1);
   paf1=prop1*(RR1-1)/denom*100;
   paf6=prop6*(RR6-1)/denom*100;
   pafn1=round(paf1*cvd/100);
   pafn6=round(paf6*cvd/100);
   days1=prop1*100;
   days6=prop6*100;
run;
*******************************************************;
```

```
** original estimates **;
data b0;
    set b1;
  IF sample=0;
run;
** PAF, 95% confidence interval (2.5% & 97.5%) **;
data b2;
    set b1;
  IF sample>0;
run;
proc sort data=b2; by RR1;
data r21; set b2; if _n_=round(0.025*500); RR1_low=RR1; keep RR1_low;
data r22; set b2; if _n_=round(0.975*500); RR1_upp=RR1; keep RR1_upp;

proc sort data=b2; by RR6;
data r61; set b2; if _n_=round(0.025*500); RR6_low=RR6; keep RR6_low;
data r62; set b2; if _n_=round(0.975*500); RR6_upp=RR6; keep RR6_upp;

proc sort data=b2; by paf1;
data p21; set b2; if _n_=round(0.025*500); paf1_low=paf1;
pafn1_low=round(paf1_low*cvd/100); keep paf1_low pafn1_low;
data p22; set b2; if _n_=round(0.975*500); paf1_upp=paf1;
pafn1_upp=round(paf1_upp*cvd/100); keep paf1_upp pafn1_upp;

proc sort data=b2; by paf6;
data p61; set b2; if _n_=round(0.025*500); paf6_low=paf6;
pafn6_low=round(paf6_low*cvd/100); keep paf6_low pafn6_low;
data p62; set b2; if _n_=round(0.975*500); paf6_upp=paf6;
pafn6_upp=round(paf6_upp*cvd/100); keep paf6_upp pafn6_upp;
run;
data b6;
    merge b0 r21 r22 r61 r62 p21 p22 p61 p62;
run;
proc print data=b6;
    var cvd days6 RR6 RR6_Low RR6_Upp paf6 paf6_low paf6_upp pafn6 pafn6_low
pafn6_upp;
run;
```