

Temperature Profile Classification - 2 Class system - r2

GMM classification of Southern Ocean Argo float temperature profile data. This notebook uses a previously created model, PCA and sample data.

Choices for data

In []:

```
#Experiment data for analysis
dataVariableId = 'thetao'
dataExperimentId = 'historical'
dataSourceId = 'UKESM1-0-LL'
dataInstitutionId = 'MOHC'
approvedIds = ["r1l1p1f2", "r2i1p1f2", "r3i1p1f2"] #insert start of approved member_ids

#File imports
maskName = "OceanMaskVolcello"
modelName = "GMM_UK_2Class_R2"

#Data definitions
startDate = '1980-01'
endDate = '2009-12'
timeRange = slice(startDate, endDate)
levSel = slice(0, 2000) #Selected levels to be investigated
maxLat = -30 #Selected latitude to be investigated
```

Libaries and Modules

Importing the necessary libaries and modules for the notebook.

In [2]:

```
#Import cell
import calendar
import cartopy.crs as ccrs
import cartopy.feature as cfeature
import dask.dataframe as dd
import fsspec
import matplotlib.dates as mdates
import matplotlib as mpl ####
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import matplotlib.ticker as ticker
import xarray as xr
import zarr

from dask import config
from dask import delayed
from joblib import dump, load
from matplotlib.pyplot import cm
from sklearn import mixture
from sklearn.decomposition import PCA
from sklearn import preprocessing
from scipy import signal
```

```
config.set(**{'array.slicing.split_large_chunks': True})
print("Imports complete")
```

Imports complete

Importing data sets

Importing the data for the models.

Import sample data set and corresponding time/geo data

In [3]:

```
importName = modelName + "_Meta_Full"
sampleMetaDFL = pd.read_csv(importName)
sampleMetaDFL = sampleMetaDFL.drop(columns=["Unnamed: 0"])
sampleMetaDFL["time"] = pd.to_datetime(sampleMetaDFL["time"])
print("Labeled sample meta data loaded from "+ importName +" and stored in sampleMetaDFL")
sampleMetaDFL.head()
```

Labeled sample meta data loaded from GMM_UK_2Class_R2_Meta_Full and stored in sampleMetaDFL. 7989840 samples loaded.

Out[3]:

	lat	lon	time	labelSorted	max posterior prob	classUncertainty
0	-66.111520	73.5	1980-01-01	0	1.0	3.899727e-09
1	-65.703316	73.5	1980-01-01	0	1.0	6.174536e-10
2	-65.288570	73.5	1980-01-01	0	1.0	1.859135e-10
3	-64.867195	73.5	1980-01-01	0	1.0	1.053628e-10
4	-64.439100	73.5	1980-01-01	0	1.0	9.551204e-11

In [4]:

```
#Meta data date processing cell
globalStartDate = sampleMetaDFL["time"][0]
globalDateInc = sampleMetaDFL["time"].unique()[1] - globalStartDate
globalEndDateIn = sampleMetaDFL["time"].iloc[-1]
globalEndDateOut = globalEndDateIn + globalDateInc

globalStartDateStr = str(globalStartDate) [:7]
globalEndDateInStr = str(globalEndDateIn) [:7]
globalEndDateOutStr = str(globalEndDateOut) [:7]
print("Sample meta data dates manipulated and stored in globalStartDate etc.")
```

Sample meta data dates manipulated and stored in globalStartDate etc.

Loading ocean Masks

In [5]:

```
#Ocean mask import cell
maskFile = xr.open_dataset(maskName)
oceanMask = maskFile.to_array()
maskFile = xr.open_dataset("OceanMaskUKESM1")
oceanMask2 = maskFile.to_array()
print("Mask Loaded and stored in oceanMask and oceanMask2 (volcello and UKESM).")
```

Mask Loaded and stored in oceanMask and oceanMask2 (volcello and UKESM).

Unpacking ocean masks

```
In [6]: #Mask unpacking cell
geoRange = oceanMask #copying mask
geoRange = geoRange.rename({"variable":"cleanMe"}) #Dimension removal
geoRange = geoRange.sel(cleanMe = geoRange.cleanMe.values[0]) #Dimension removal
geoRange = geoRange.reset_coords("cleanMe", drop=True) #Dimension removal
geoRangeS = geoRange.stack(ij =("i", "j")) #Stacking
geoRangeFilt = geoRangeS.dropna("ij")
print("Ocean mask unpacked into geoRangeFilt.")

geoRange2 = oceanMask2 #copying mask
geoRange2S = geoRange2.stack(ij =("i", "j")) #Stacking
geoRangeFilt2 = geoRange2S.dropna("ij")
print("UKESM Ocean mask unpacked into geoRangeFilt2.")
```

Ocean mask unpacked into geoRangeFilt.
 UKESM Ocean mask unpacked into geoRangeFilt2.

Date Calculations

```
In [7]: #Date calculation cell
startDateNp = np.datetime64(startDate, 'M')
endDateNp = np.datetime64(endDate, 'M')
timeDiff = endDateNp - startDateNp
timeDiff = timeDiff.astype(int) + 1
print("Calculated date range.")
```

Calculated date range.

Calculation functions

Functions:

- butter_lowpass - Returns a filtered version of the input array using a butterworth filter

```
In [8]: #Calculation functions cell
def butter_lowpass(data,cut,order=4,sample_freq=1) :
    nyq = 0.5*sample_freq
    pass_freq = 1./cut/nyq
    sos=signal.butter(order, pass_freq, 'low', output='sos')
    filt=signal.sosfiltfilt(sos,data)
    return filt

print("Calculation functions defined.")
```

Calculation functions defined.

Plotting functions

Functions:

- locationPlotGroup - plots location and classification of data points for an input numpy array.
- locationPlotGroupDFLab - plots location and classification of data points for a provided data frame.
- locationPlotGroupDFMonthly - plots location and classification of data points for an input data frame in monthly subplots.
- locationPlotGroupDFTime - plots mean classification of data points for an input data frame.

- locationPlotTime - plots locations of an input data array on a map with a colour scale for time.
- locationPlotUncertaintyDF - plots uncertainty in classification on a location plot.
- locationPlotUncertaintyDFMonthly - plots location and uncertainty of data points for an input data frame in monthly subplots.
- locationPlotXr - plots dataframe/xr data locations (used for masks)
- locationUncertaintyMean - plots mean latitude per longitude +-1 standard deviation on orthogonal plot
- locationUncertaintyMeanCart - plots mean latitude per longitude +-1 standard deviation on standard plot
- timeSingleLatMean - plots mean singular latitude over time on standard plot

In [9]:

```
#Plotting functions Cell
#sampleDepthAxis = dfESMLatLevT["lev"]
decadeColours = ["#CC6677", "#DDCC77", "#117733"]
timeList = sampleMetaDFL["time"].unique()

def locationPlotGroup(metadataArray, size, plotNo):
    '''Plots locations of numpy arrays with group colour scheme'''
    plt.figure(plotNo, figsize=size)
    ax = plt.axes(projection=ccrs.SouthPolarStereo())
    ax.add_feature(cfeature.OCEAN)
    ax.add_feature(cfeature.COASTLINE)
    ax.coastlines()
    ax.gridlines()
    im = ax.scatter(metadataArray[1], metaDataArray[0], transform=ccrs.PlateCarree(), c =
    cb = plt.colorbar(im)
    plt.plot(np.arange(0, 361, 1), np.ones(361)*-29.5, transform=ccrs.PlateCarree(), color="#E
    plt.title("Grouped Sample Locations ("+str(len(metadataArray[0]))+")")

def locationPlotGroupDFLab(dataFrame, title, size, plotNo):
    '''Plots locations of data frame points with group colour scheme'''
    plt.figure(plotNo, figsize=size)
    ax = plt.axes(projection=ccrs.SouthPolarStereo())
    ax.add_feature(cfeature.OCEAN)
    ax.add_feature(cfeature.COASTLINE)
    ax.coastlines()
    ax.gridlines()
    im = ax.scatter(dataFrame["lon"], dataFrame["lat"], transform=ccrs.PlateCarree(), c =
    cb = plt.colorbar(im)
    plt.plot(np.arange(0, 361, 1), np.ones(361)*-29.5, transform=ccrs.PlateCarree(), color="#E
    plt.title(str(title))

def locationPlotGroupDFMonthly(dataFrame, title, plotNo):
    '''Plots locations of dataframe points by monthly subplot with group colour scheme'''
    fig = plt.figure(plotNo, figsize=(30, 42))
    plt.title(str(title))
    for i in range(1, 13):
        #timeData = dataFrame.where(dataFrame["time"].dt.month==i) #normal dt info
        timeData = dataFrame.where(dataFrame["time"] == i) #groupby dt info
        ax = plt.subplot(4, 3, i, projection=ccrs.SouthPolarStereo())
        ax.add_feature(cfeature.OCEAN)
        ax.add_feature(cfeature.COASTLINE)
        ax.coastlines()
        ax.gridlines()
        im = ax.scatter(timeData["lon"], timeData["lat"], transform=ccrs.PlateCarree(), c =
        plt.plot(np.arange(0, 361, 1), np.ones(361)*-29.5, transform=ccrs.PlateCarree(), color="#E
        plt.title(calendar.month_abbr[i])
    plt.subplots_adjust(wspace=0, hspace=0.05)
    cb_ax = fig.add_axes([0.27, 0.1, 0.5, 0.02])
    cbar = fig.colorbar(im, cax=cb_ax, orientation="horizontal")
```

```

def locationPlotGroupDFTime(dataFrame, title, size, plotNo):
    '''Plots locations of data frame points with group colour scheme'''
    plt.figure(plotNo, figsize=size)
    ax = plt.axes(projection=ccrs.SouthPolarStereo())
    ax.add_feature(cfeature.OCEAN)
    ax.add_feature(cfeature.COASTLINE)
    ax.coastlines()
    ax.gridlines()
    im = ax.scatter(dataFrame["lon"], dataFrame["lat"], transform=ccrs.PlateCarree(), c =
    cb = plt.colorbar(im)
    loc = mdates.AutoDateLocator()
    cb.ax.yaxis.set_major_locator(loc)
    cb.ax.yaxis.set_major_formatter(mdates.ConciseDateFormatter(loc))
    plt.plot(np.arange(0,361,1),np.ones(361)*-29.5, transform=ccrs.PlateCarree(), color="E"
    plt.title(str(title))

def locationPlotTime(dataArray, size, plotNo):
    '''Plots locations of numpy arrays with date colour scheme'''
    plt.figure(plotNo, figsize=size)
    ax = plt.axes(projection=ccrs.SouthPolarStereo())
    ax.add_feature(cfeature.OCEAN)
    ax.add_feature(cfeature.COASTLINE)
    ax.coastlines()
    ax.gridlines()
    im = ax.scatter(dataArray[1], dataArray[0], transform=ccrs.PlateCarree(), c= mdates.da
    cb = plt.colorbar(im)
    loc = mdates.AutoDateLocator()
    cb.ax.yaxis.set_major_locator(loc)
    cb.ax.yaxis.set_major_formatter(mdates.ConciseDateFormatter(loc))
    plt.plot(np.arange(0,361,1),np.ones(361)*-29.5, transform=ccrs.PlateCarree(), color="E
    plt.title("Sample Locations ("+str(len(dataArray[0]))+")")

def locationPlotUncertaintyDF(dataFrame, title, size, plotNo):
    '''Plots input data array classification uncertainties'''
    plt.figure(plotNo, figsize=size)
    ax = plt.axes(projection=ccrs.SouthPolarStereo())
    ax.add_feature(cfeature.OCEAN)
    ax.add_feature(cfeature.COASTLINE)
    ax.coastlines()
    ax.gridlines()
    im = ax.scatter(dataFrame["lon"], dataFrame["lat"], transform=ccrs.PlateCarree(), c =
    cb = plt.colorbar(im)
    plt.plot(np.arange(0,361,1),np.ones(361)*-29.5, transform=ccrs.PlateCarree(), color="E
    plt.title(str(title))

def locationPlotUncertaintyDFMonthly(dataFrame, title, plotNo):
    '''Plots locations of dataframe points by monthly subplot with group colour scheme'''
    fig = plt.figure(plotNo, figsize=(30,42))
    plt.title(str(title))
    for i in range(1, 13):
        #timeData = dataFrame.where(dataFrame["time"].dt.month==i) #normal dt info
        timeData = dataFrame.where(dataFrame["time"] ==i) #groupby dt info
        ax = plt.subplot(4, 3, i, projection=ccrs.SouthPolarStereo())
        ax.add_feature(cfeature.OCEAN)
        ax.add_feature(cfeature.COASTLINE)
        ax.coastlines()
        ax.gridlines()
        im = ax.scatter(timeData["lon"], timeData["lat"], transform=ccrs.PlateCarree(), c
        #cb = plt.colorbar(im, fraction=0.046, pad=0.04)
        plt.plot(np.arange(0,361,1),np.ones(361)*-29.5, transform=ccrs.PlateCarree(), colo
        plt.title(calendar.month_abbr[i])
    plt.subplots_adjust(wspace=0, hspace=0.05)
    cb_ax = fig.add_axes([0.27, 0.1, 0.5, 0.02])

```

```

cbar = fig.colorbar(im, cax=cb_ax, orientation="horizontal")

def locationPlotXr(dataArray, size, plotNo):
    '''Plots locations of numpy arrays with date colour scheme'''
    plt.figure(plotNo, figsize=size)
    ax = plt.axes(projection=ccrs.SouthPolarStereo())
    ax.add_feature(cfeature.OCEAN)
    ax.add_feature(cfeature.COASTLINE)
    ax.coastlines()
    ax.gridlines()
    im = ax.scatter(dataArray["lon"], dataArray["lat"], transform=ccrs.PlateCarree())
    plt.plot(np.arange(0, 361, 1), np.ones(361)*-29.5, transform=ccrs.PlateCarree(), color="B")
    plt.title("Sample Locations ("+str(len(dataArray["lat"]))+")")

def locationUncertaintyMean(decadeMeanList, decadeStdList, uncertaintyThreshold, figSize,
    '''Plots mean latitude for classification uncertainty with +- 1 std on a map'''
    plt.figure(plotNo, figsize=figSize)
    ax = plt.axes(projection=ccrs.SouthPolarStereo())
    ax.add_feature(cfeature.OCEAN)
    ax.add_feature(cfeature.COASTLINE)
    ax.coastlines()
    ax.gridlines()
    for i in range(len(decadeMeanList)):
        ax.plot(decadeMeanList[i].index, decadeMeanList[i]["lat"], transform=ccrs.PlateCarree())
        ax.plot(decadeMeanList[i].index, decadeMeanList[i]["lat"] + decadeStdList[i]["lat"], transform=ccrs.PlateCarree())
        ax.plot(decadeMeanList[i].index, decadeMeanList[i]["lat"] - decadeStdList[i]["lat"], transform=ccrs.PlateCarree())
    plt.plot(np.arange(0, 361, 1), np.ones(361)*-29.5, transform=ccrs.PlateCarree(), color="B")
    plt.title("Classification Uncertainty above "+str(uncertaintyThreshold)+" Mean Latitude")
    plt.legend()

def locationUncertaintyMeanCart(decadeMeanList, decadeStdList, uncertaintyThreshold, figSize,
    '''Plots mean latitude for classification uncertainty with +- 1 std as a standard plot'''
    plt.figure(plotNo, figsize=figSize)
    for i in range(len(decadeMeanList)):
        plt.plot(decadeMeanList[i].index, decadeMeanList[i]["lat"], color=decadeColours[i])
        plt.plot(decadeMeanList[i].index, decadeMeanList[i]["lat"] + decadeStdList[i]["lat"], color=decadeColours[i])
        plt.plot(decadeMeanList[i].index, decadeMeanList[i]["lat"] - decadeStdList[i]["lat"], color=decadeColours[i])
    plt.title("Classification Uncertainty above "+str(uncertaintyThreshold)+" Mean Latitude")
    plt.legend()

def timeSingleLatMean(meanLatArr, lineStyle, arrThreshold, arrSmoothFact, plotNo):
    '''Plots singular Latitude mean on standard plot'''
    plt.figure(plotNo, figsize=(20,10))
    plt.plot(timeList, meanLatArr, lw=1, ls=lineStyle, label="Thresh_"+str(arrThreshold)+"_Smoothed")
    plt.xlabel("Date")
    plt.ylabel("Singular Mean Latitude")
    plt.title("Classification Uncertainty Mean Latitude over time")
    plt.legend()

print("Plotting functions defined.")

```

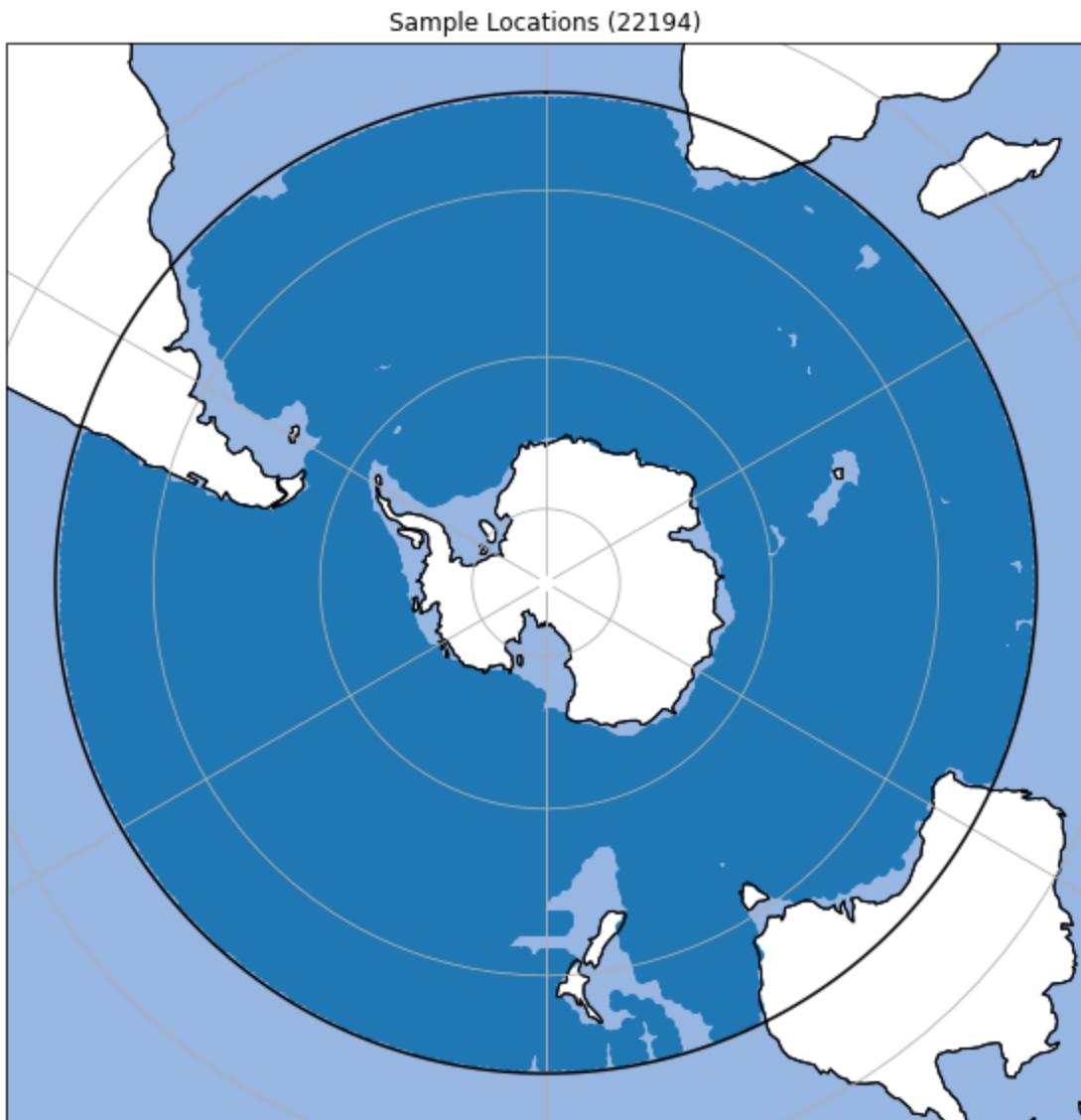
Plotting functions defined.

Plotting Ocean Mask

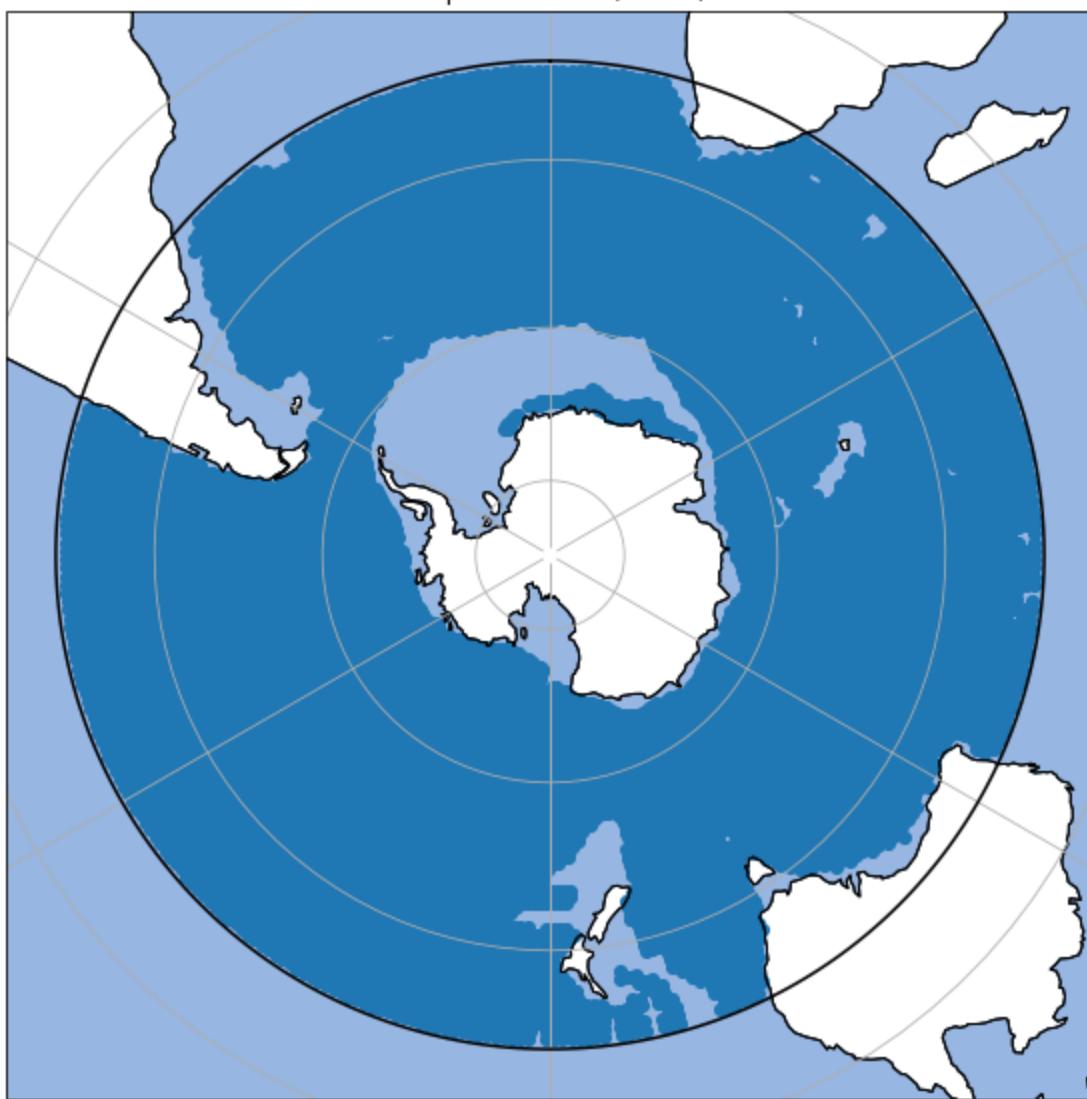
In [10]:

```
#Mask plotting cell
locationPlotXr(geoRangeFilt, (10,10), 1) #OceanMaskVolcello
```

```
locationPlotXr(geoRangeFilt2, (10,10), 2) #OceanMaskUKESM1  
plt.show()
```



Sample Locations (19548)



Anomalies

In [11]:

```
#Initial anomalous data point df creation cell
labelAnomDF = sampleMetaDFL[np.logical_or(np.logical_and(sampleMetaDFL["lat"]<-60, sampleMetaDFL["lat"]>-45), sampleMetaDFL["lat"]<0)]
print("Class 0 above -45 and Class 1 below -60 lat. Stored in labelAnomDF. "+str(len(labelAnomDF)))
```

Class 0 above -45 and Class 1 below -60 lat. Stored in labelAnomDF. 3843 anomalous points detected.

In [12]:

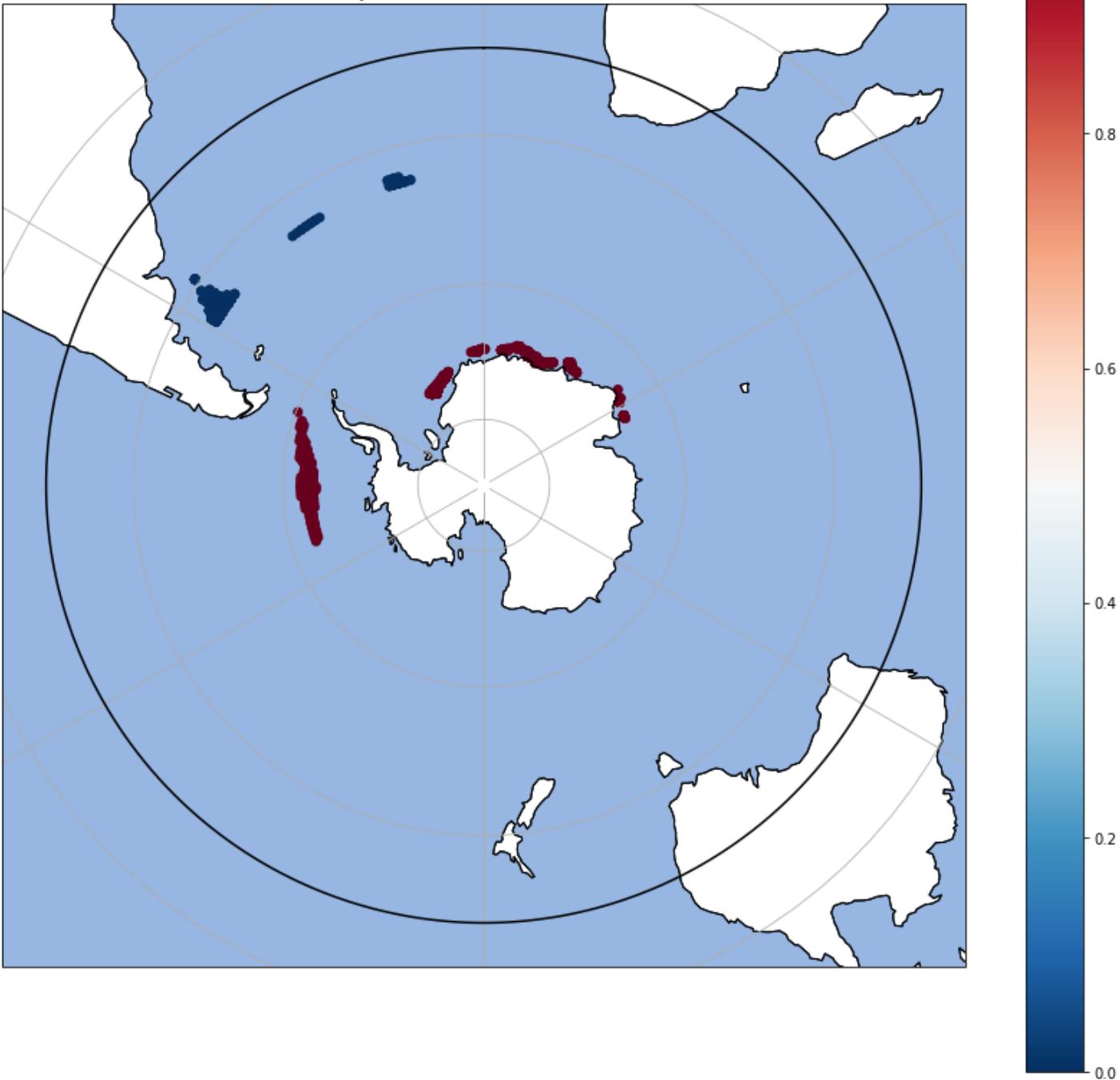
```
#High confidence anomalous data point df creation cell
labelAnomConDF = labelAnomDF[labelAnomDF["classUncertainty"]<0.25]
print("High Classification Confidence DF of labelAnomDF (<0.25). "+str(len(labelAnomConDF)))
```

High Classification Confidence DF of labelAnomDF (<0.25). 1245 anomalous points detected.

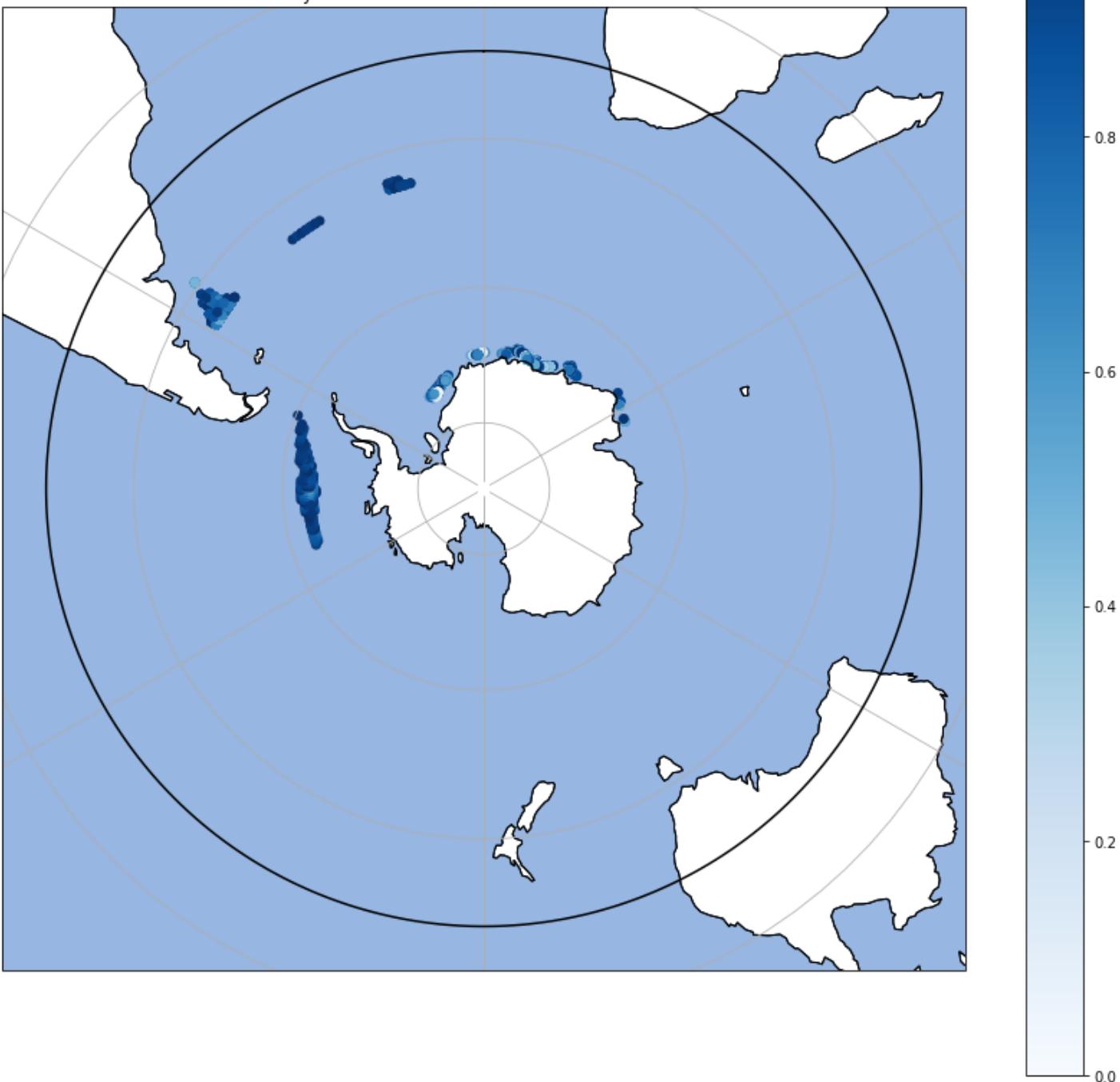
In [13]:

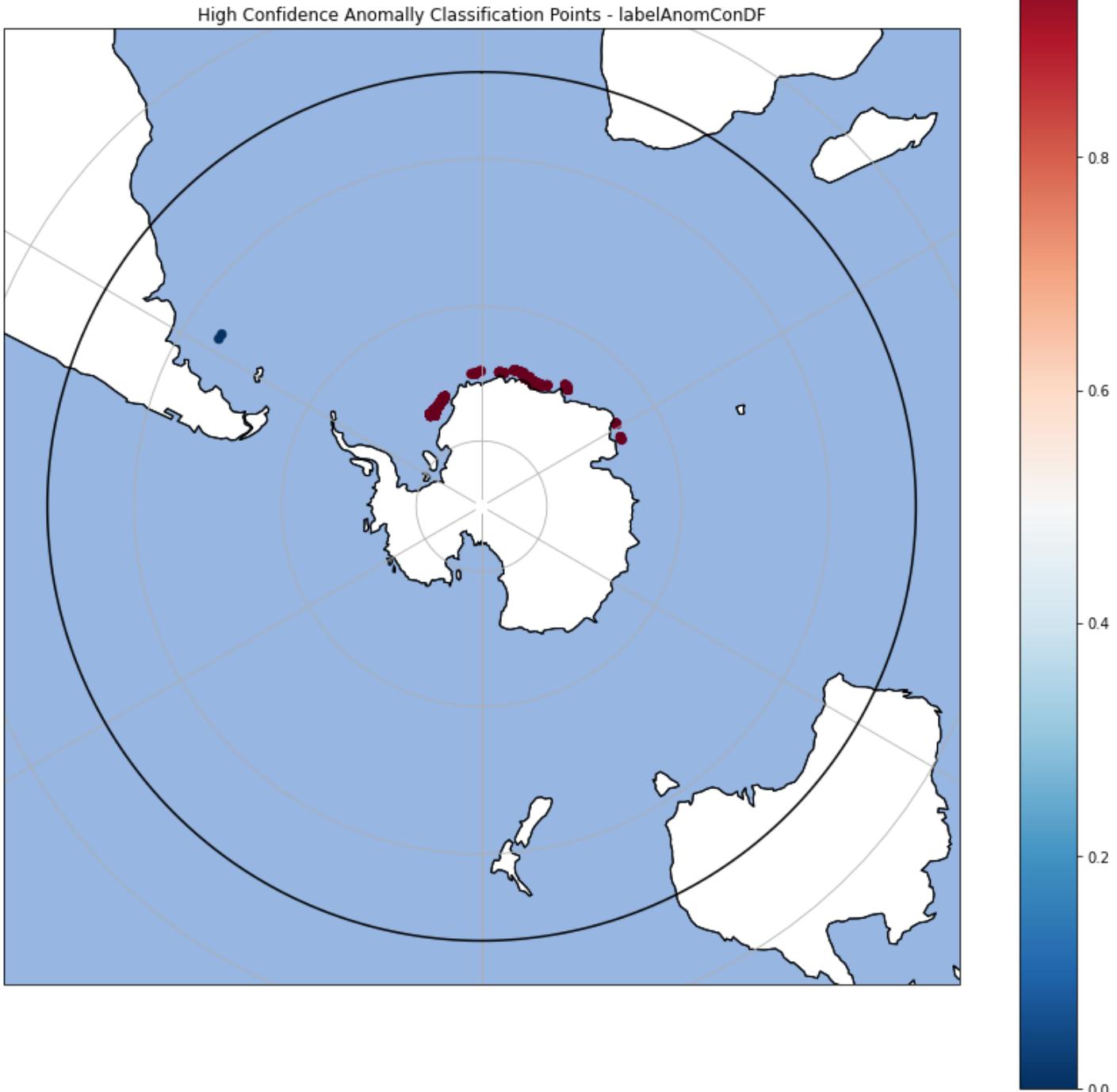
```
locationPlotGroupDFLab(labelAnomDF, "Anomaly Classification Points - labelAnomDF", (15,15))
locationPlotUncertaintyDF(labelAnomDF, "Anomaly Classification Point Uncertainties - labelAnomDF")
locationPlotGroupDFLab(labelAnomConDF, "High Confidence Anomaly Classification Points - labelAnomConDF", (15,15))
plt.show()
```

Anomaly Classification Points - labelAnomDF



Anomaly Classification Point Uncertainties - labelAnomDF



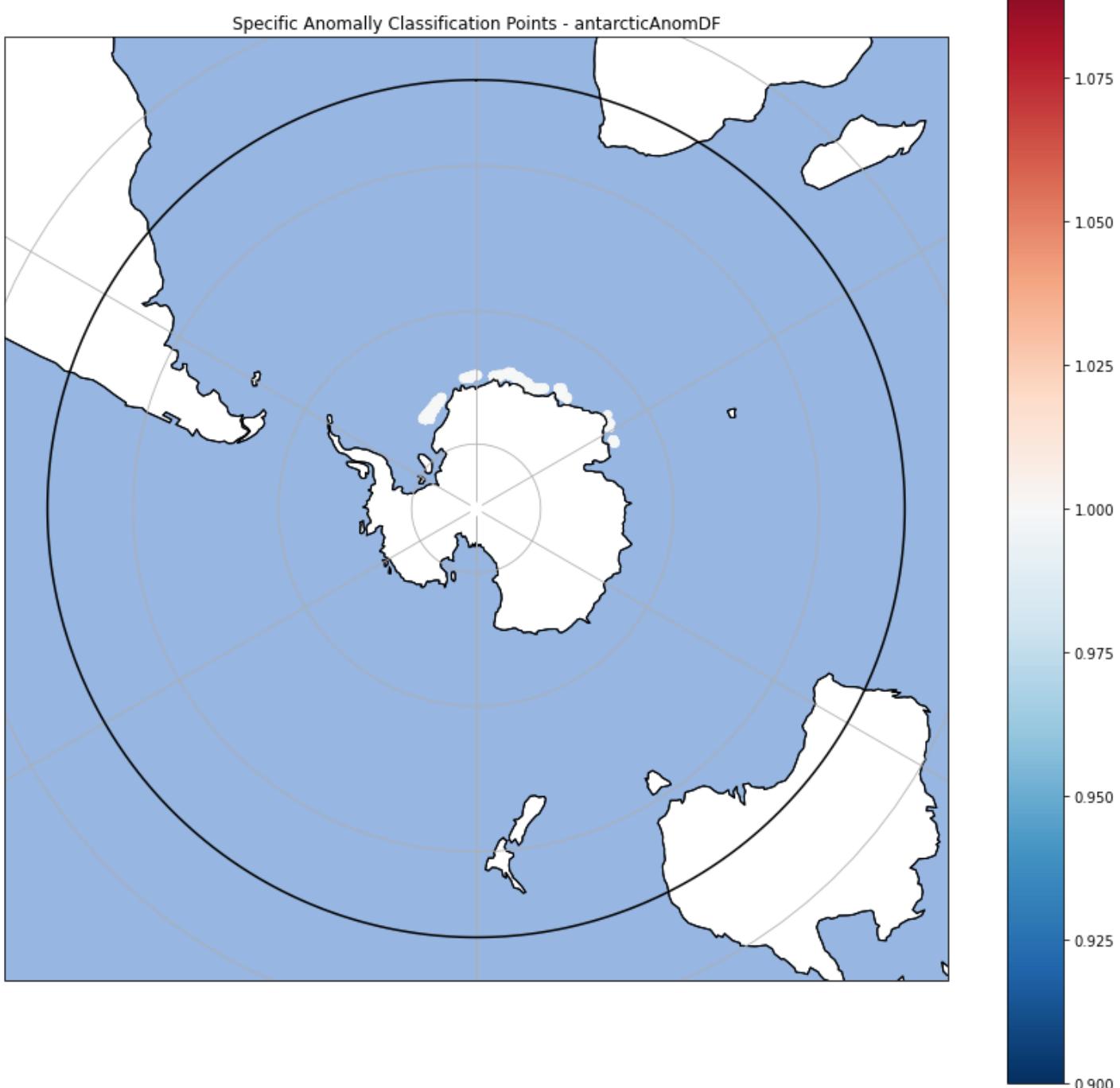


Antarctic Anomalous Classifications

In [14]:

```
antarcticAnomDF = sampleMetaDFL[np.logical_and(np.logical_and(sampleMetaDFL["labelSorted"]  
locationPlotGroupDFLab(antarcticAnomDF, "Specific Anomaly Classification Points - antarct  
print(str(len(antarcticAnomDF))+ " anomalous samples found. (Lat<-65, Lon<70, Lon>-60)"  
plt.show()
```

2809 anomalous samples found. (Lat<-65, Lon<70, Lon>-60)

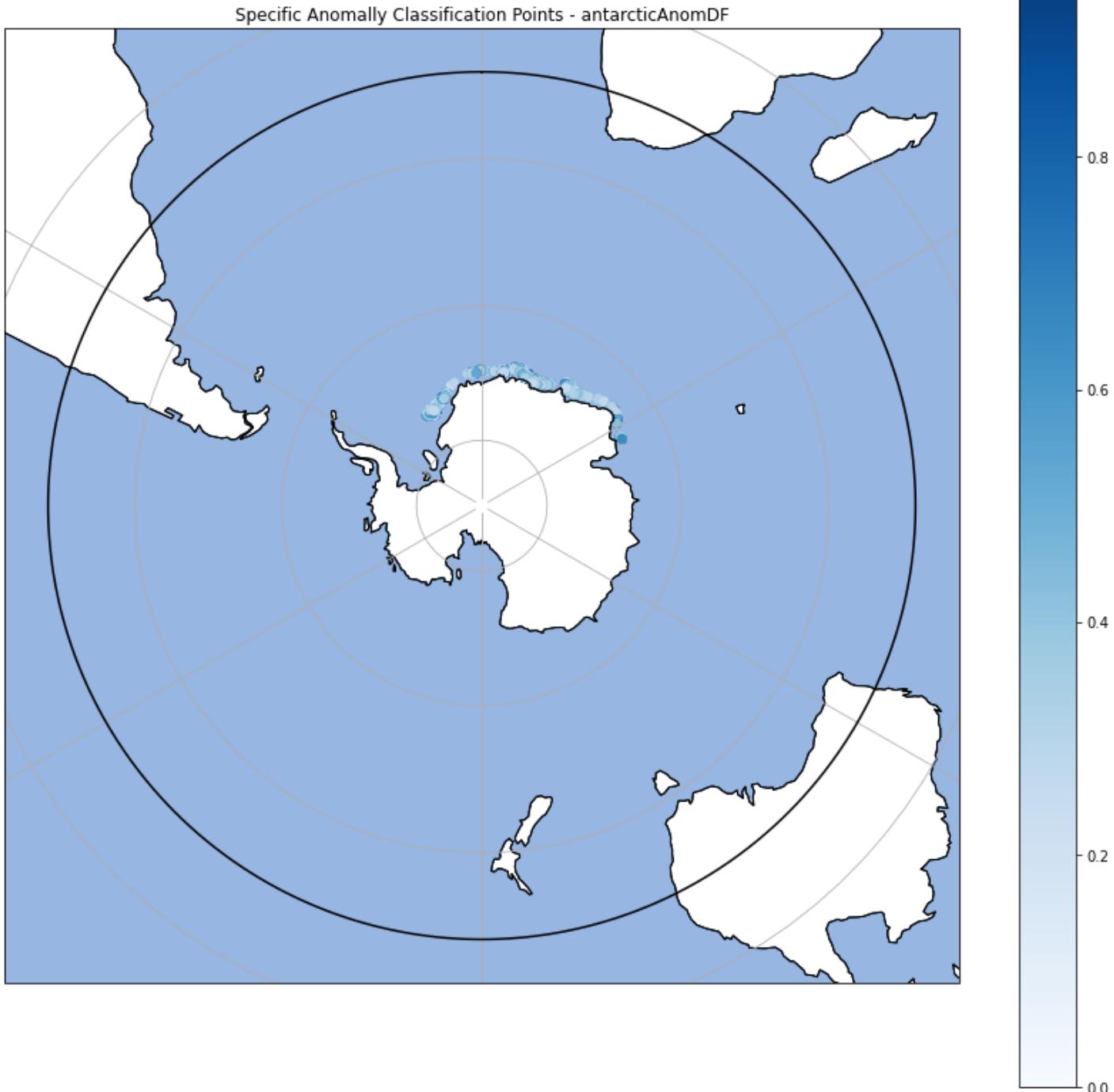


Antarctic Anomalous Classification Uncertainty

In [15]:

```
antarcticUncertDF = sampleMetaDFL[np.logical_and(np.logical_and(sampleMetaDFL["classUncert",  
locationPlotUncertaintyDF(antarcticUncertDF, "Specific Anomaly Classification Points - ar  
print(str(len(antarcticUncertDF))+" anomalous samples found. (Lat<-65, Lon<70, Lon>-60)")  
plt.show()
```

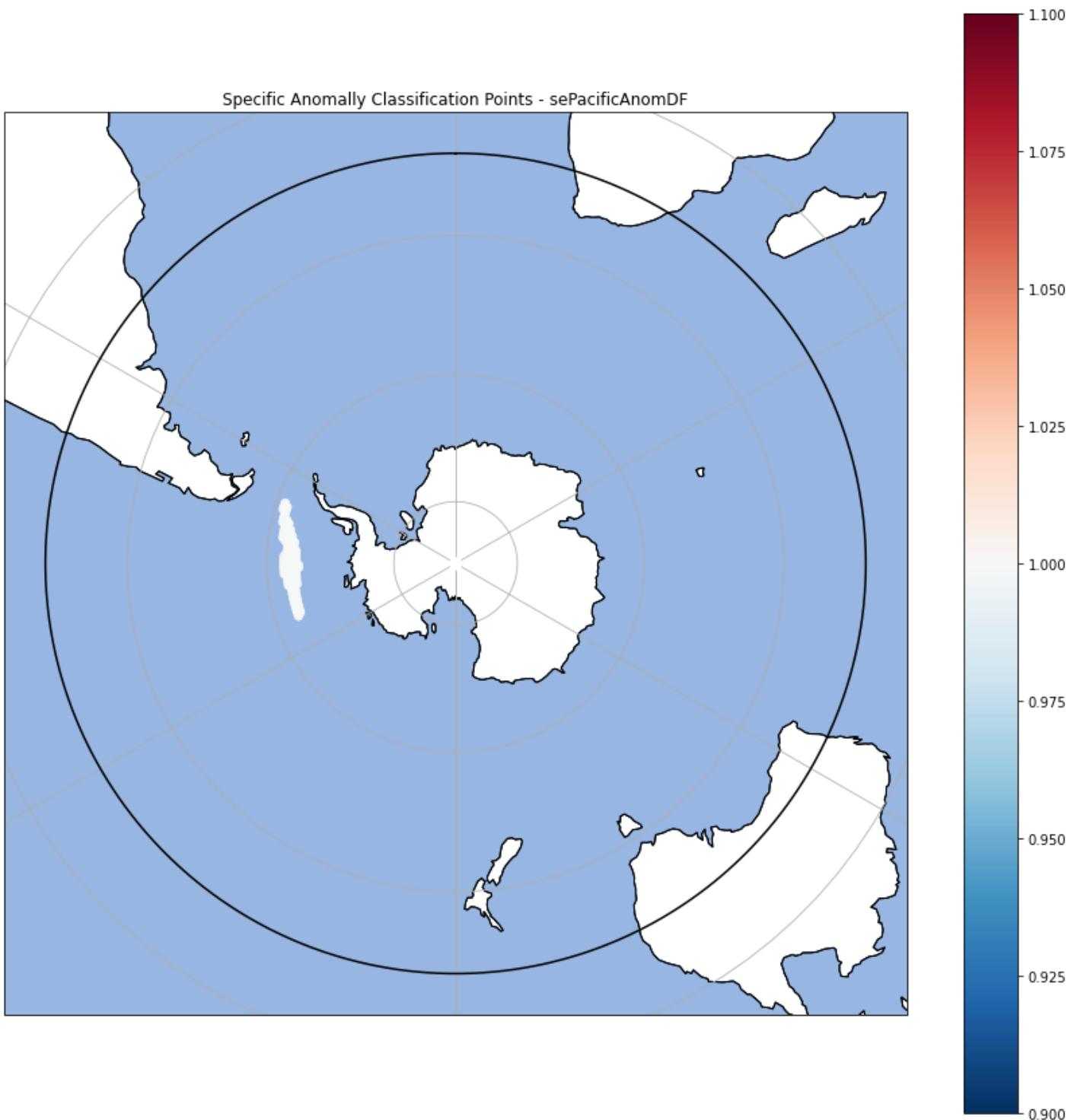
4564 anomalous samples found. (Lat<-65, Lon<70, Lon>-60)



South-East Pacific Classifications

```
In [16]: sePacificAnomDF = sampleMetaDFL[np.logical_and(np.logical_and(sampleMetaDFL["labelSorted"] == "sePacific", sampleMetaDFL["lat"] < -61), sampleMetaDFL["lon"] < -60, sampleMetaDFL["lon"] > -140)]
locationPlotGroupDFLab(sePacificAnomDF, "Specific Anomaly Classification Points - sePacific")
print(str(len(sePacificAnomDF))+" anomalous samples found. (Lat<-61, Lon<-60, Lon>-140)")
plt.show()
```

215 anomalous samples found. (Lat<-61, Lon<-60, Lon>-140)



```
In [17]: print("South-East Pacific Anomaly Count (Lat<-61, Lon<-60, Lon>-140)")  
sePacificAnomDF.groupby("time") ["labelSorted"].value_counts()
```

```
South-East Pacific Anomaly Count (Lat<-61, Lon<-60, Lon>-140)  
Out[17]: time      labelSorted  
1994-09-01  1          74  
1994-10-01  1          67  
1994-11-01  1          14  
1996-11-01  1           5  
2006-10-01  1          19  
2006-11-01  1          11  
2007-08-01  1           1  
2007-09-01  1          24  
Name: labelSorted, dtype: int64
```

In [18]:

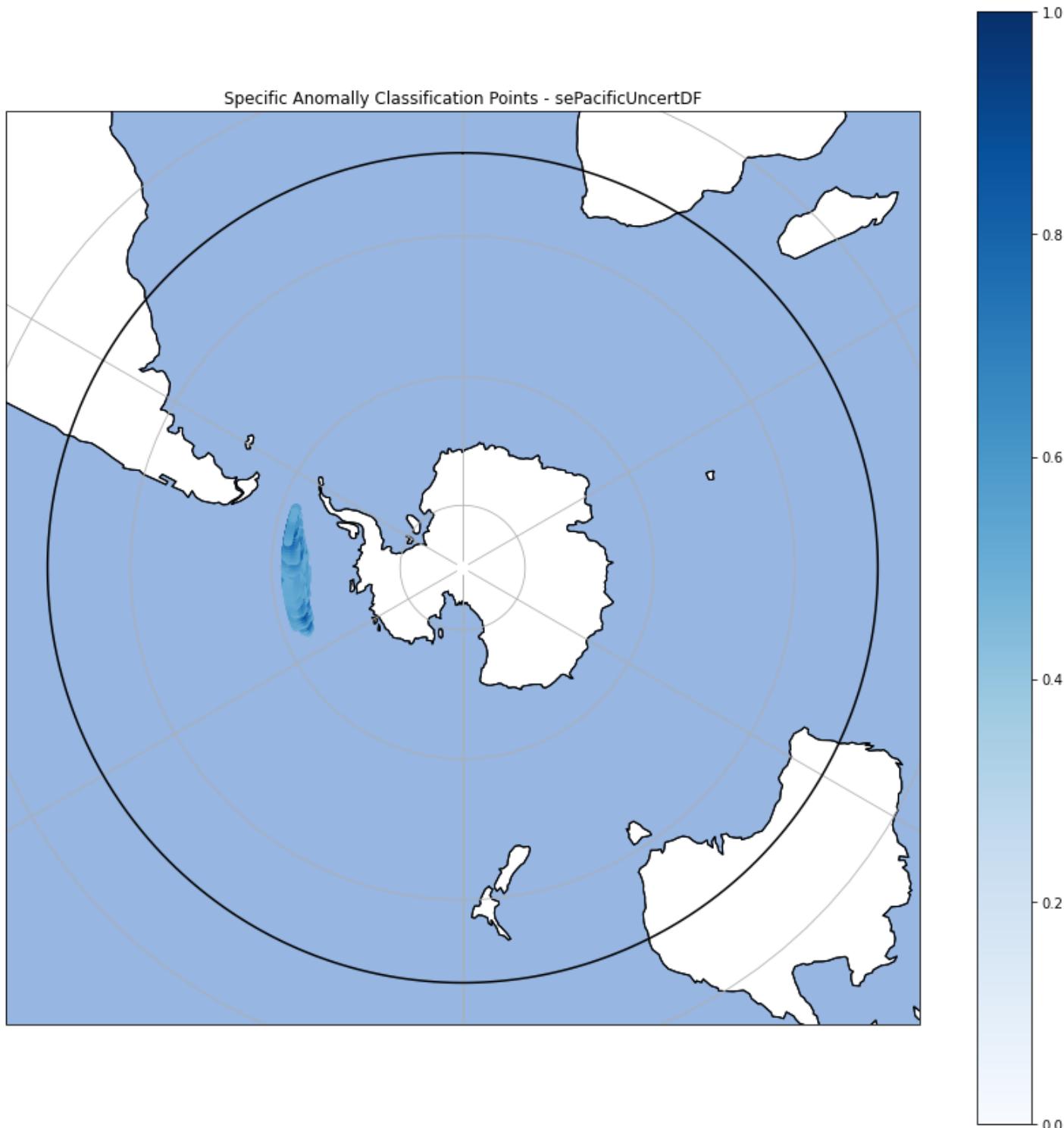
```

timeRange = np.arange(np.datetime64("1980-01"), np.datetime64("2000-01"), np.timedelta64(1, "D"))

sePacificUncertDF = sampleMetaDF[np.logical_and(np.logical_and(sampleMetaDF["classUncert"] == "Specific Anomaly", sampleMetaDF["locationPlotUncertainty"] > 0), sampleMetaDF["lat"] < -62, sampleMetaDF["lon"] < -60, sampleMetaDF["lon"] > -140)]
print(str(len(sePacificUncertDF))+" anomalous samples found. (Lat<-62, Lon<-60, Lon>-140)")
plt.show()

```

1998 anomalous samples found. (Lat<-62, Lon<-60, Lon>-140)



In [19]:

```
i = 0
sePacificUncertStack = sePacificUncertDF.groupby("time") ["time"].value_counts()
print("South-East Pacific Uncertainty Count (Lat<-62, Lon<-60, Lon>-140)")
print("Time\t\t", "Count")
for x in sePacificUncertStack:
    strIndex = str(sePacificUncertStack.index[i])
```

```
print(strIndex[12:22]+"\t", x)
i +=1
```

```
South-East Pacific Uncertainty Count (Lat<-62, Lon<-60, Lon>-140)
Time          Count
1980-01-01    3
1980-02-01    45
1988-10-01    20
1988-11-01    23
1988-12-01    6
1989-02-01    24
1990-10-01    86
1990-11-01    45
1994-01-01    43
1994-02-01    73
1994-08-01    92
1994-09-01    179
1994-10-01    175
1994-11-01    135
1994-12-01    72
1995-01-01    63
1995-02-01    48
1995-03-01    30
1996-10-01    53
1996-11-01    68
1996-12-01    50
1997-01-01    25
1997-02-01    24
1997-03-01    12
1997-07-01    15
1997-09-01    6
2006-10-01    119
2006-11-01    103
2006-12-01    60
2007-01-01    43
2007-02-01    31
2007-07-01    31
2007-08-01    68
2007-09-01    74
2007-10-01    45
2007-11-01    9
```

Data Readjustment

In [20]:

```
preAdLen = len(sampleMetaDFL)
if True:
    sampleMetaDFL = pd.concat([sampleMetaDFL, antarcticAnomDF, antarcticAnomDF]).drop_duplicates()
    print("Removing antarcticAnomDF.")
    sampleMetaDFL = pd.concat([sampleMetaDFL, antarcticUncertDF, antarcticUncertDF]).drop_duplicates()
    print("Removing antarcticUncertDF.")
    sampleMetaDFL = pd.concat([sampleMetaDFL, sePacificAnomDF, sePacificAnomDF]).drop_duplicates()
    print("Removing sePacificAnomDF.")
    sampleMetaDFL = pd.concat([sampleMetaDFL, sePacificUncertDF, sePacificUncertDF]).drop_duplicates()
    print("Removing sePacificUncertDF.\n")

postAdLen = len(sampleMetaDFL)
lenDiff = preAdLen - postAdLen
print("SampleMetaDFL adjusted from "+str(preAdLen)+" to "+str(postAdLen)+". "+str(lenDiff))
```

```
Removing antarcticAnomDF.
Removing antarcticUncertDF.
Removing sePacificAnomDF.
```

```
Removing sePacificUncertDF.
```

```
SampleMetaDFL adjusted from 7989840 to 7982024. 7816 samples dropped.
```

```
In [21]:
```

```
print("Last data point in set (expected 2009-12):")  
sampleMetaDFL.iloc[-1]
```

```
Last data point in set (expected 2009-12):
```

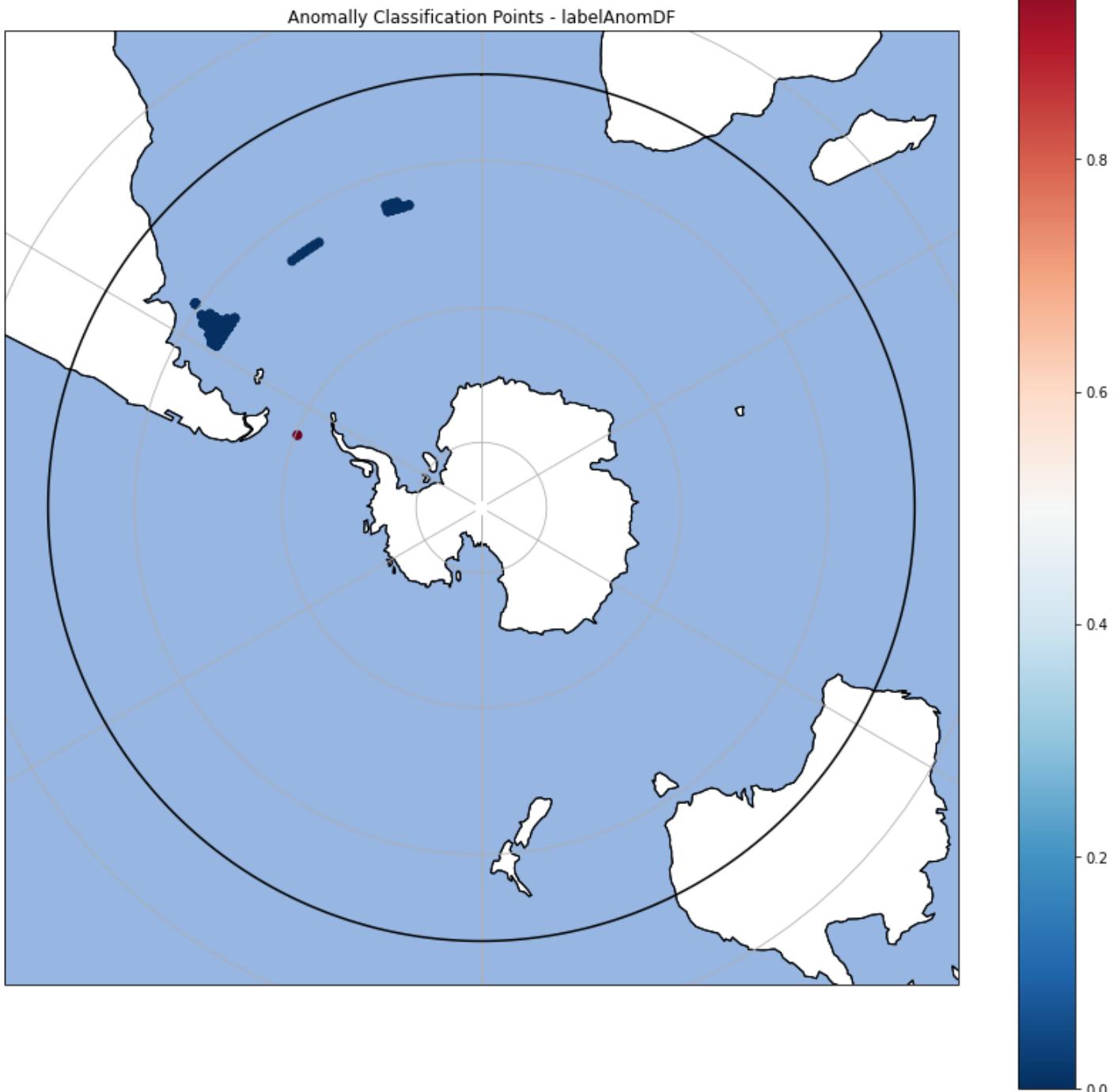
```
Out[21]:
```

```
lat           -30.455408  
lon            72.500000  
time    2009-12-01 00:00:00  
labelSorted          1  
max posterior prob 1.000000  
classUncertainty   0.000000  
Name: 7989839, dtype: object
```

```
In [22]:
```

```
labelAnomDF = sampleMetaDFL[np.logical_or(np.logical_and(sampleMetaDFL["lat"]<-60, sampleMetaDFL["time"]>="2009-12-01"), np.logical_and(sampleMetaDFL["lat"]>-45, sampleMetaDFL["time"]<="2009-12-01"))]  
print("Class 0 above -45 and Class 1 below -60 lat. Stored in labelAnomDF. "+str(len(labelAnomDF)))  
locationPlotGroupDFLab(labelAnomDF, "Anomaly Classification Points - labelAnomDF", (15,15))
```

```
Class 0 above -45 and Class 1 below -60 lat. Stored in labelAnomDF. 819 anomalous points detected.
```



In [23]:

```
antarcticAnomDF = sampleMetaDFL[np.logical_and(np.logical_and(sampleMetaDFL["labelSorted"])]
antarcticLen = len(antarcticAnomDF)
if antarcticLen:
    locationPlotGroupDFLab(antarcticAnomDF, "Specific Anomaly Classification Points - ant")
    print(str(antarcticLen)+" Antarctic anomalous samples found (Lat<-65, Lon<70, Lon>-60.)")
plt.show()
```

0 Antarctic anomalous samples found (Lat<-65, Lon<70, Lon>-60.).

Average location information

In [24]:

```
#Average classification per location cell
averageGeoG = sampleMetaDFL.groupby(["lat", "lon"])
averageGeo = averageGeoG.mean(["lat", "lon"])
```

```
averageGeo = averageGeo.reset_index()
print("Average location information calculated and stored in averageGeo.")
```

Average location information calculated and stored in averageGeo.

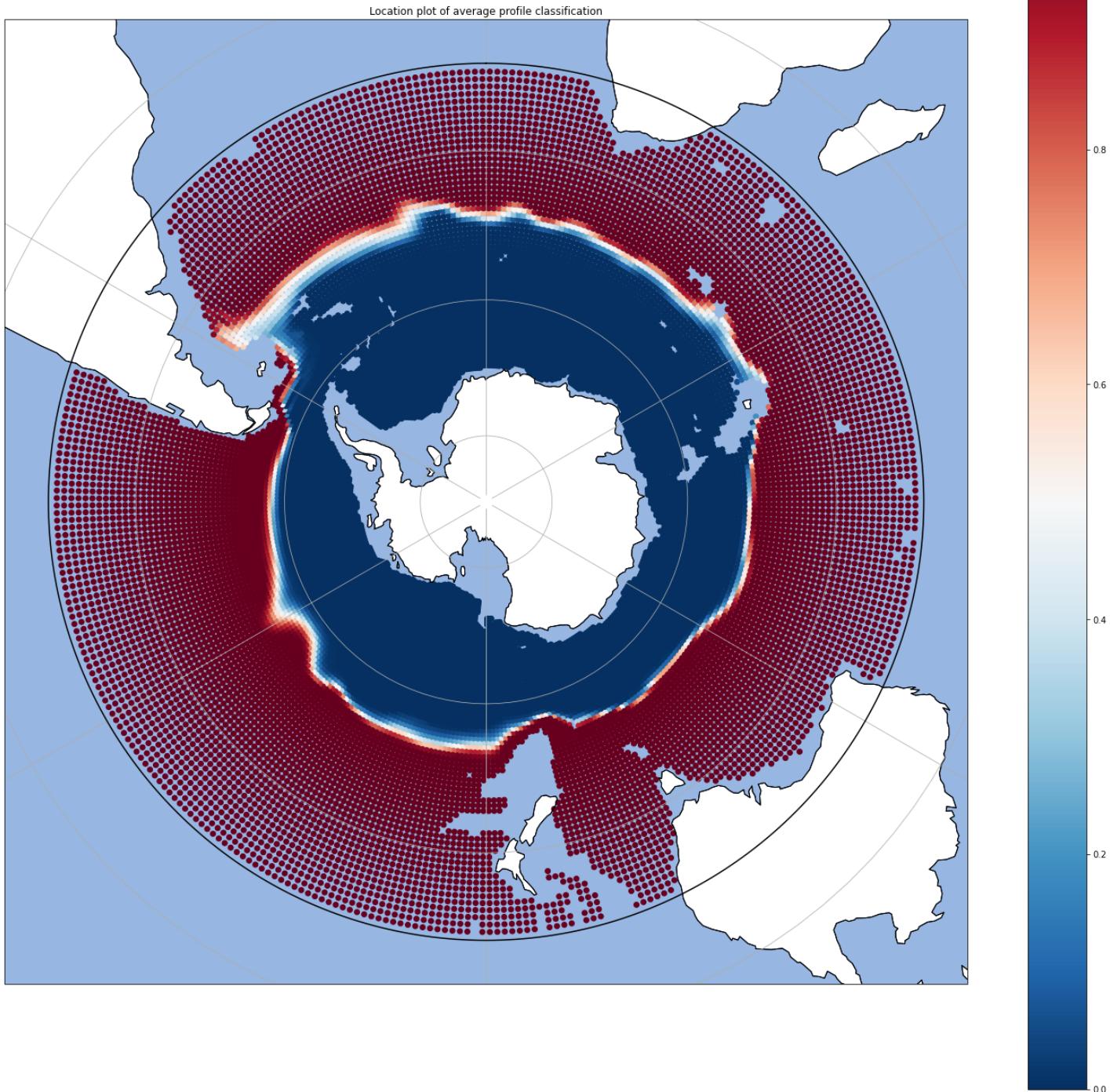
In [25]:

```
#Average classification per location and month cell
averageMonthGeoG = sampleMetaDFL.groupby([sampleMetaDFL.time.dt.month, "lat", "lon"])
averageMonthGeo = averageMonthGeoG.mean(["lat", "lon"])
averageMonthGeo = averageMonthGeo.reset_index()
print("Average monthly location information calculated and stored in averageMonthGeo.")
```

Average monthly location information calculated and stored in averageMonthGeo.

In [26]:

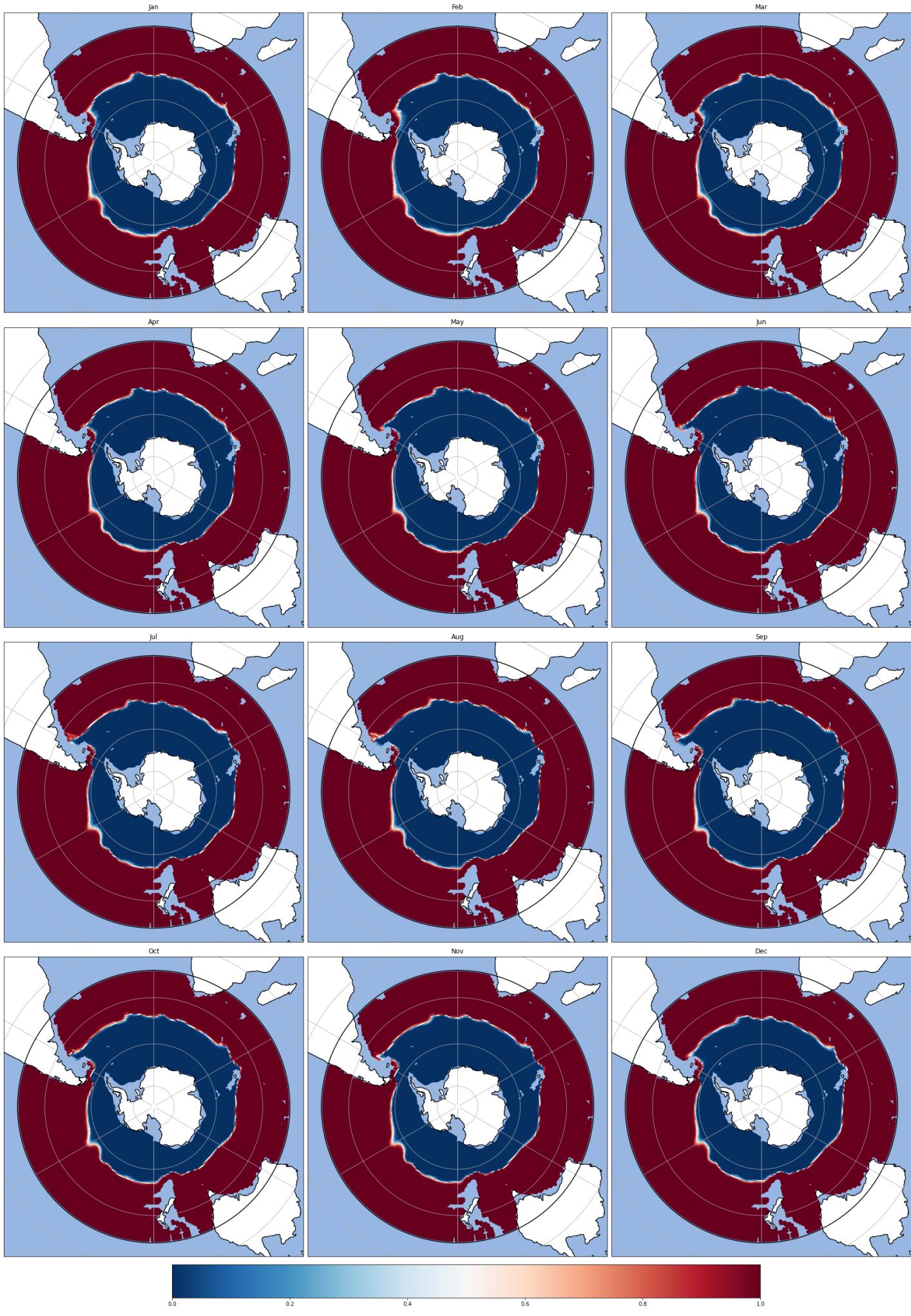
```
#Average classification location plot cell
locationPlotGroupDFLab(averageGeo, "Location plot of average profile classification", (25,
plt.show()
```



In [27]:

```
locationPlotGroupDFMonthly(averageMonthGeo, "Monthly summaries for training data set", 1)
print("Average Classifications, grouped by month.")
plt.show()
```

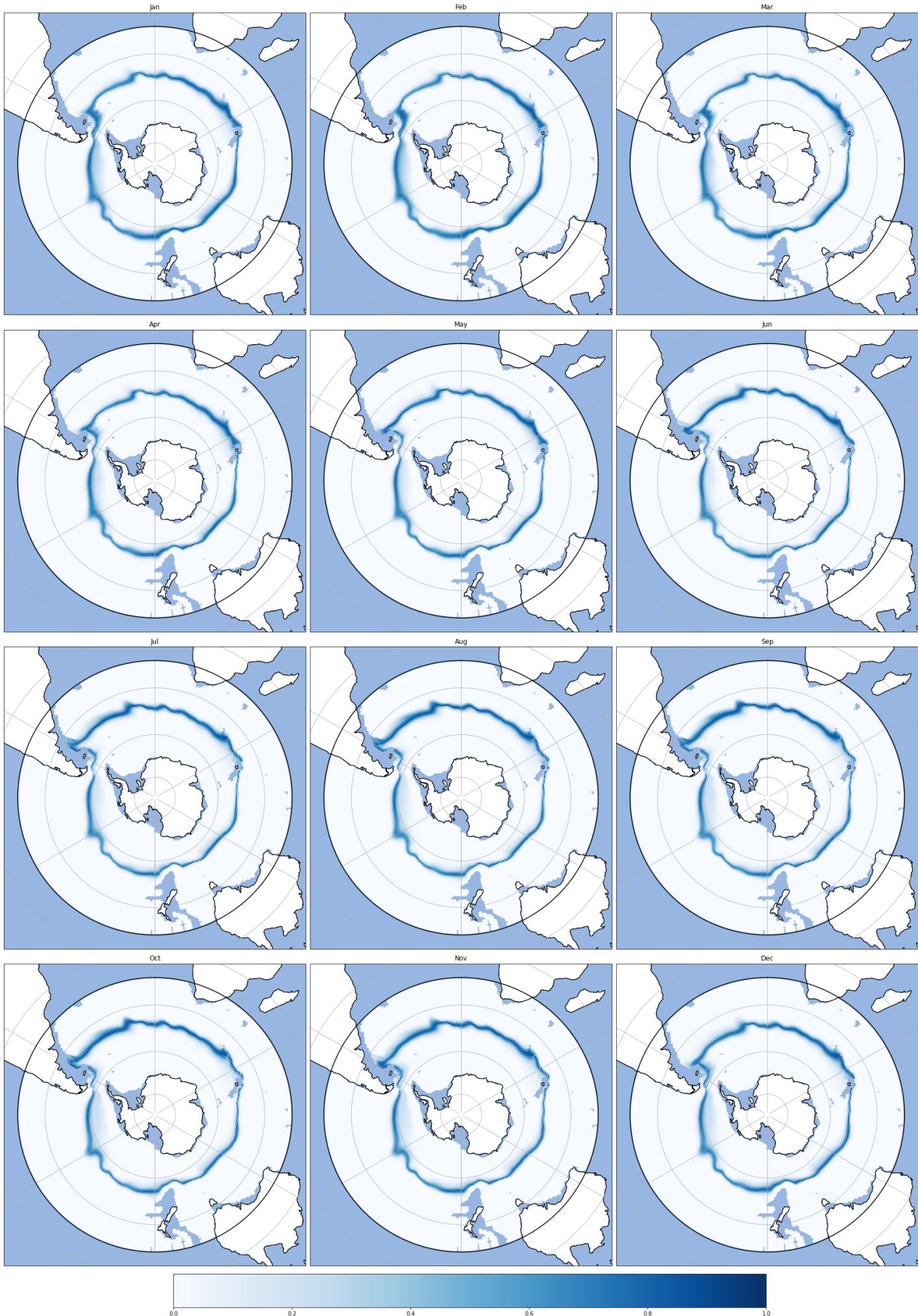
Average Classifications, grouped by month.



In [28]:

```
locationPlotUncertaintyDFMonthly(averageMonthGeo, "Monthly uncertainty", 1)
print("Average Uncertainty in classifications, grouped by month.")
plt.show()
```

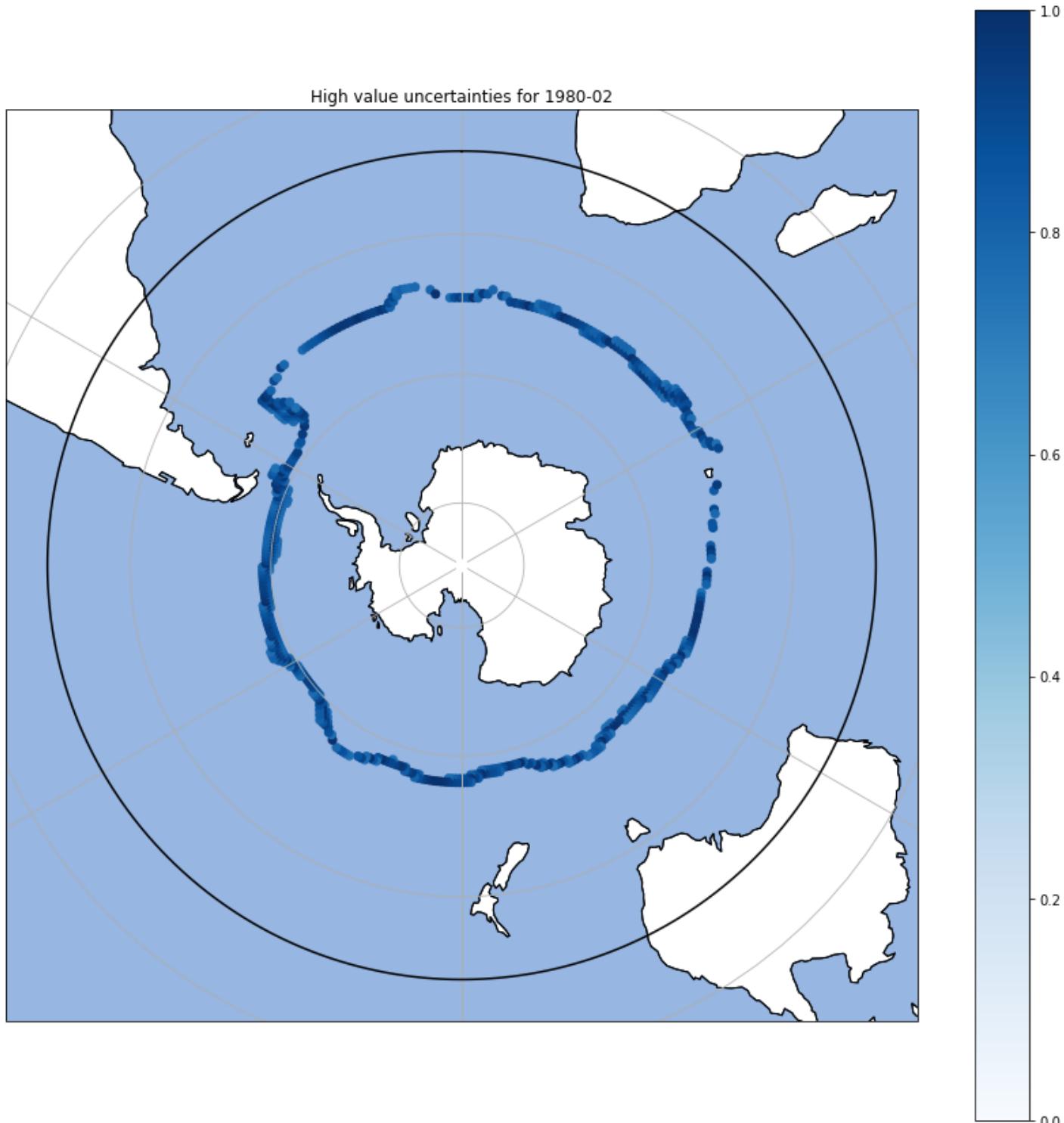
Average Uncertainty in classifications, grouped by month.



Single Month Properties

In [29]:

```
testTime = "1980-02"
uncertaintyDFL = sampleMetaDFL[sampleMetaDFL["classUncertainty"] > 0.75]
monthlyData = uncertaintyDFL[uncertaintyDFL["time"]== "1980-02"]#[ "lon"].unique()
locationPlotUncertaintyDF(monthlyData, "High value uncertainties for "+testTime, (15,15),
plt.show()
print("Single Month data identified and plotted and stored in monthlyData. "+str(len(monthlyData)))
```



Single Month data identified and plotted and stored in monthlyData. 607 high classification uncertainty points identified.

Mean Singular Latitudes over time

In [30]:

```

thresholds = [0.25, 0.50, 0.75, 0.85]
smoothFactor1 = 24
smoothFactor2 = 120
monthlyLatMMList = []
monthlyLatMMS1List = []
monthlyLatMMS2List = []

for i in range(len(thresholds)):
    uncertaintyDFL = sampleMetaDFL[sampleMetaDFL["classUncertainty"] > thresholds[i]]

    uncertaintyDFLLonTimeG = uncertaintyDFL.groupby(["lon", "time"])
    uncertaintyDFLLonM = uncertaintyDFLLonTimeG.mean("lon")
    uncertaintyDFLLonM = uncertaintyDFLLonM.reorder_levels(['time', 'lon']).sort_index()
    uncertaintyDFLLonMTIMEG = uncertaintyDFLLonM.groupby("time")

    monthlyLatMMList.append(uncertaintyDFLLonMTIMEG.mean() [["lat"]])
    monthlyLatMMS1List.append(butter_lowpass(monthlyLatMMList[i][ "lat"], smoothFactor1))
    monthlyLatMMS2List.append(butter_lowpass(monthlyLatMMList[i][ "lat"], smoothFactor2))

print("Monthly Singular Mean Latitudes calculated, with smoothing.")

```

Monthly Singular Mean Latitudes calculated, with smoothing.

In [31]:

```

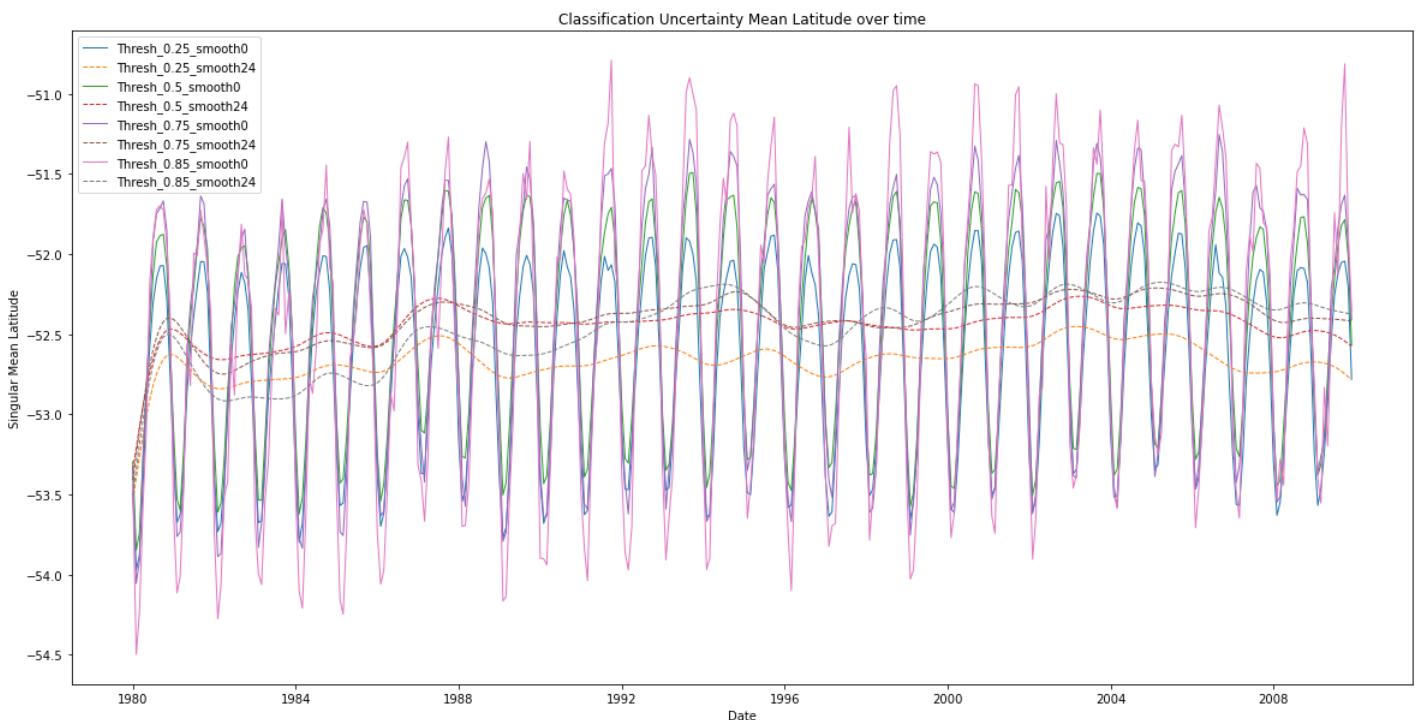
for i in range(len(thresholds)):
    timeSingleLatMean(monthlyLatMMList[i], "--", thresholds[i], 0, 1)
    timeSingleLatMean(monthlyLatMMS1List[i], "----", thresholds[i], smoothFactor1, 1)

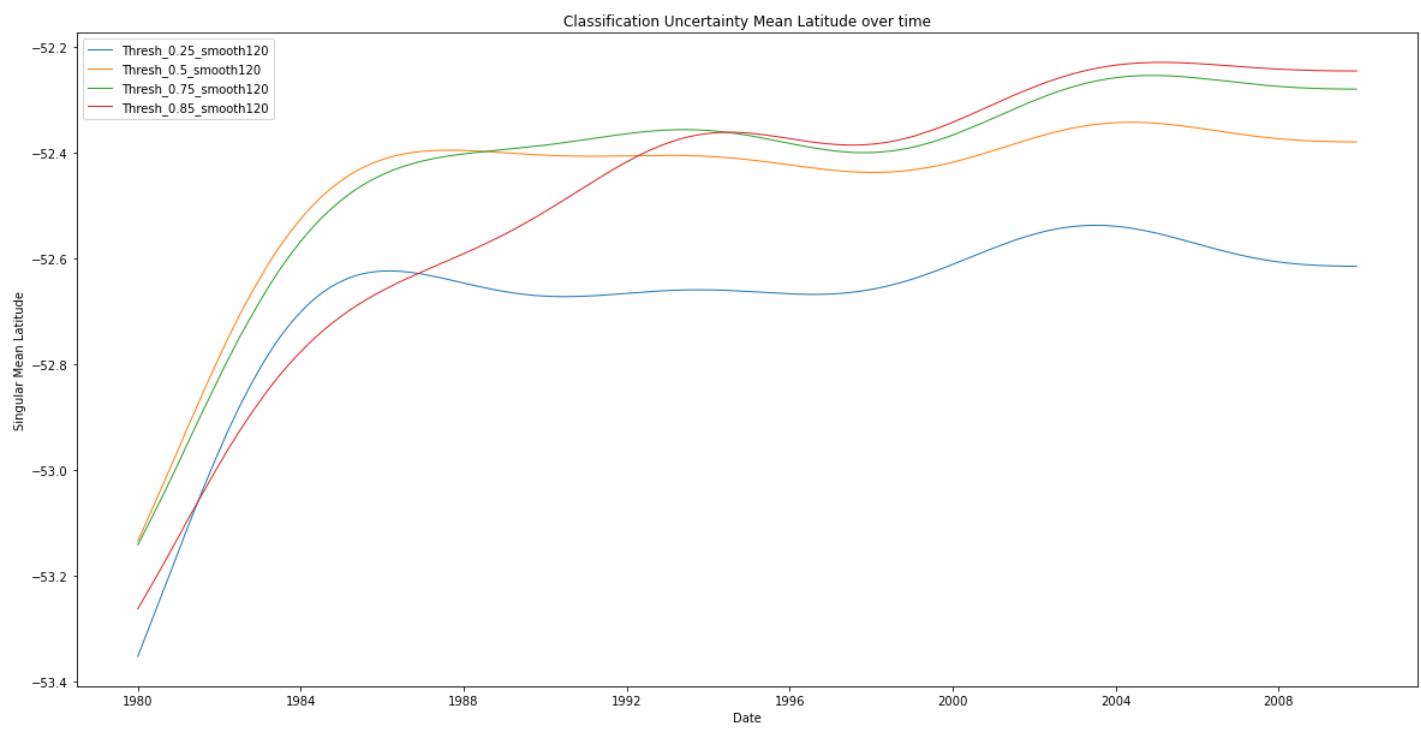
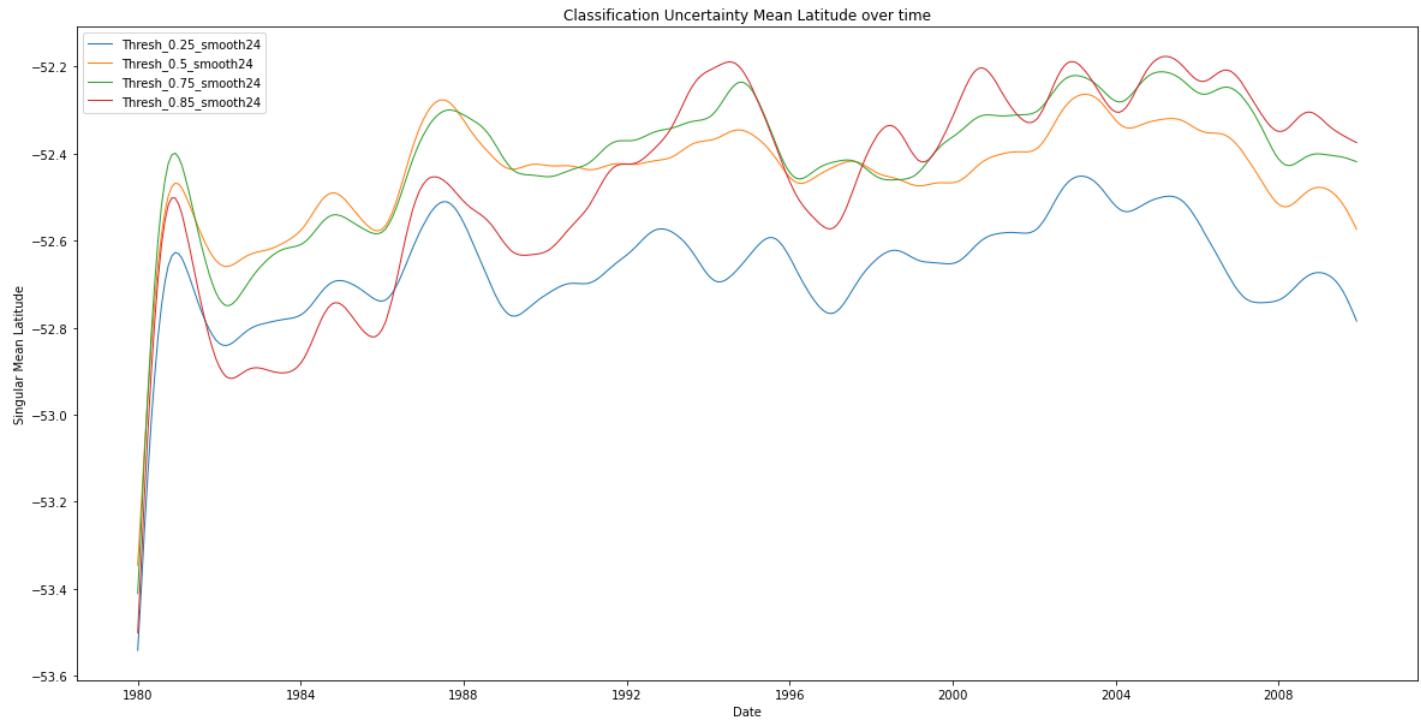
    timeSingleLatMean(monthlyLatMMS1List[i], "--", thresholds[i], smoothFactor1, 2)
    timeSingleLatMean(monthlyLatMMS2List[i], "--", thresholds[i], smoothFactor2, 3)

for i in range(1, 4):
    plt.figure(i)
    #plt.gca().invert_yaxis()

plt.show()

```



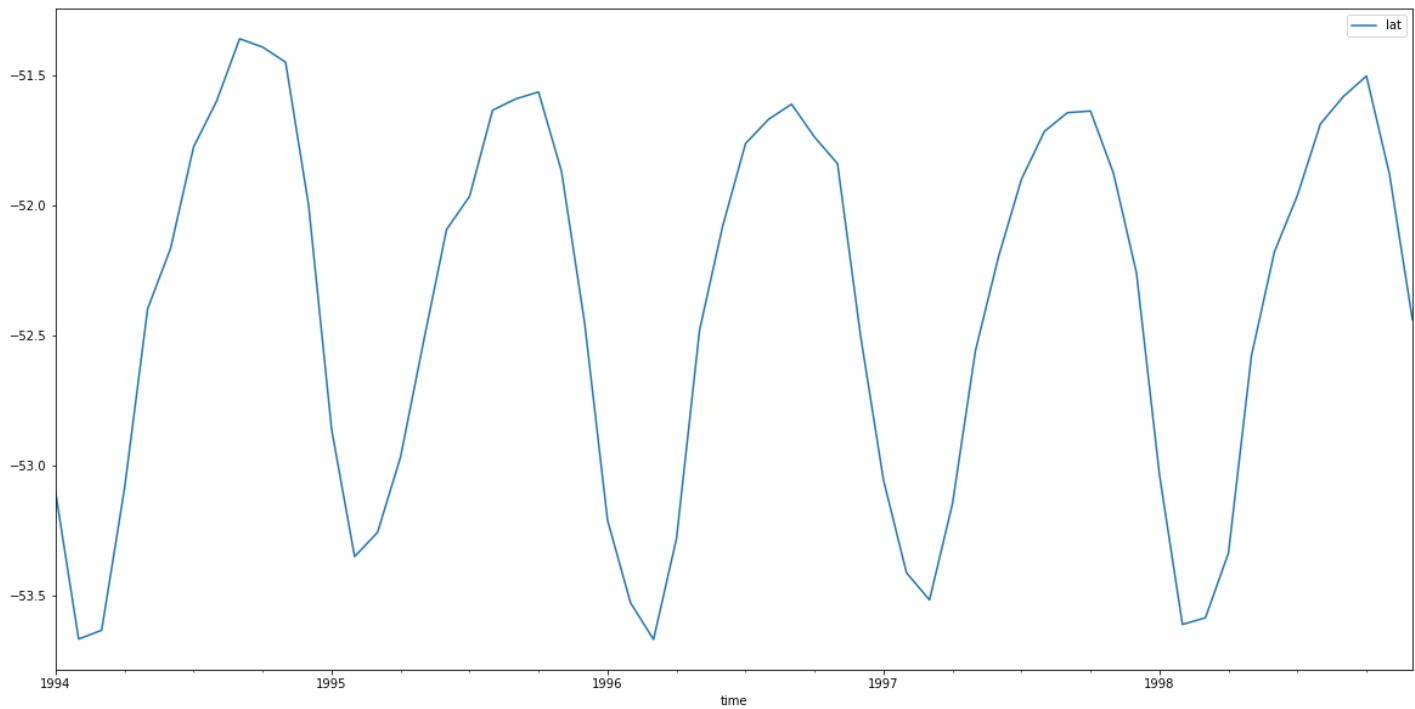


1997 Dip investigation

In [32]:

```
timeRange = np.arange(np.datetime64("1994-01", 'M'), np.datetime64("1999-01", 'M'), np.timedelta64(1, 'M'))
breakDown = monthlyLatMMList[2]
timeRangeDF = breakDown[breakDown.index.isin(timeRange)]
timeRangeDF.plot(figsize=(20, 10))
print("Single Lat mean value for 1994 to 1998")
plt.show()
```

Single Lat mean value for 1994 to 1998



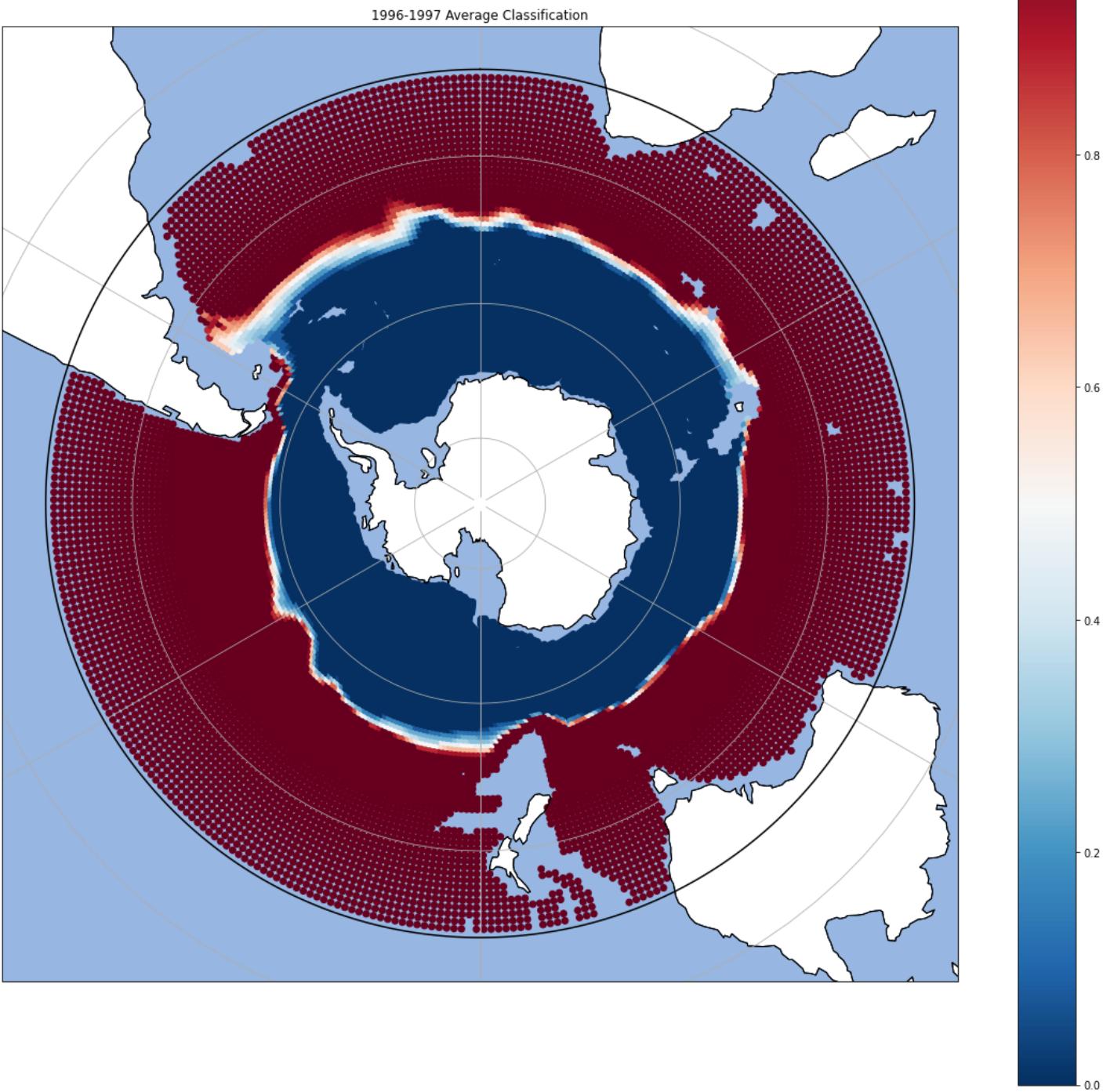
In [33]:

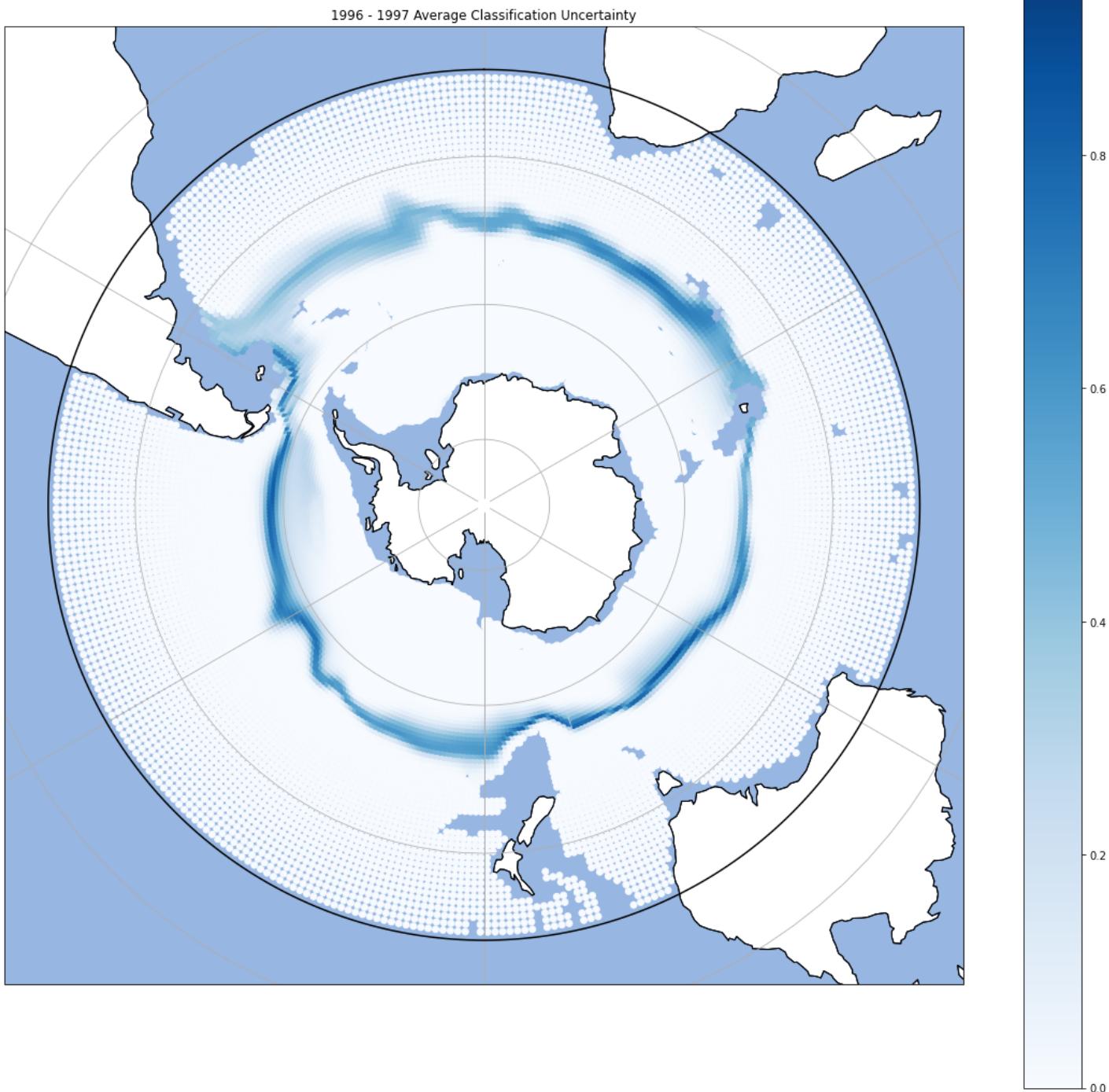
```
timeRangeClose = np.arange(np.datetime64("1996-01", 'M'), np.datetime64("1998-01", 'M'), np.timedelta64(1, 'M'))
timeRangeCloseDF = sampleMetaDFL[sampleMetaDFL["time"].isin(timeRangeClose)]
avgGeoTimeRangeCloseDFG = timeRangeCloseDF.groupby(["lat", "lon"])
avgGeoTimeRangeCloseDF = avgGeoTimeRangeCloseDFG.mean(["lat", "lon"])
avgGeoTimeRangeCloseDF = avgGeoTimeRangeCloseDF.reset_index()
print("Average classification information calculated for 1996 - 1997, stored in avgGeoTimeRangeCloseDF.")
```

Average classification information calculated for 1996 - 1997, stored in avgGeoTimeRangeCloseDF.

In [34]:

```
locationPlotGroupDFLab(avgGeoTimeRangeCloseDF, "1996-1997 Average Classification", (20,20))
locationPlotUncertaintyDF(avgGeoTimeRangeCloseDF, "1996 - 1997 Average Classification Uncertainty")
plt.show()
```





Decadal investigation

In [35]:

```
decades = ["1980", "1990", "2000"]
decade80 = np.arange(np.datetime64("1980", 'M'), np.datetime64("1990", 'M'), np.timedelta64(1, 'M'))
decade90 = np.arange(np.datetime64("1990", 'M'), np.datetime64("2000", 'M'), np.timedelta64(1, 'M'))
decade00 = np.arange(np.datetime64("2000", 'M'), np.datetime64("2010", 'M'), np.timedelta64(1, 'M'))
decadeList = [decade80, decade90, decade00]
print("Decade lists created and stored in decade80, decade90 and decade00.")
```

Decade lists created and stored in decade80, decade90 and decade00.

In [36]:

```
sampleMetaDFDecList = [sampleMetaDFL[sampleMetaDFL["time"].isin(decade80)], sampleMetaDFL[sampleMetaDFL["time"].isin(decade90)], sampleMetaDFL[sampleMetaDFL["time"].isin(decade00)]]
print("Decade split performed and stored in sampleMetaDFDecList.")
```

Decade split performed and stored in sampleMetaDFDecList.

In [37]:

```
decDFIThreshList = []
decDFIThreshLatMList = []
decDFIThreshLatSList = []
threshold = 0.75

for i in range(len(decades)):
    decDF = sampleMetaDFDecList[i]
    decDFIThresh = decDF[decDF["classUncertainty"] > threshold]
    decDFIThreshList.append(decDFIThresh)

    decDFIThreshLonG = decDFIThresh.groupby("lon")
    decDFIThreshLatMList.append(decDFIThreshLonG.mean()[["lat"]])
    decDFIThreshLatSList.append(decDFIThreshLonG.std()[["lat"]])
print("Decadal class uncertainty latitude means calculted and stored in decDFIThreshLatMList")
print("Decadal class uncertainty latitude standard deviations calculted and stored in decDFIThreshLatSList")
```

Decadal class uncertainty latitude means calculted and stored in decDFIThreshLatMList. Standard deviations are stored in decDFIThreshLatSList.

In [38]:

```
decDFIThreshLatMMListDecade = []
decDFIThreshLatMMListYearly = []
for i in range(len(decDFIThreshLatMList)):
    decDFIThreshLatMMListDecade.append(decDFIThreshLatMList[i]["lat"].mean()) #Decadal Mean

    yearlyLatMeans = monthlyLatMMList[2]
    decadeSet = yearlyLatMeans[yearlyLatMeans.index.isin(decadeList[i])]
    decDFIThreshLatMMListYearly.append(decadeSet["lat"].mean()) #Yearly Mean from previous

print("For "+str(decades[i])+" the singular mean latitude for a threshold of "+str(threshold)+" was "+str(decDFIThreshLatMMListYearly[i]))
print("Decade singular mean latitude values calculated.")
```

For 1980 the singular mean latitude for a threshold of 0.75 was -52.45 via decade and -52.53 via yearly.

For 1990 the singular mean latitude for a threshold of 0.75 was -52.33 via decade and -52.39 via yearly.

For 2000 the singular mean latitude for a threshold of 0.75 was -52.3 via decade and -52.31 via yearly.

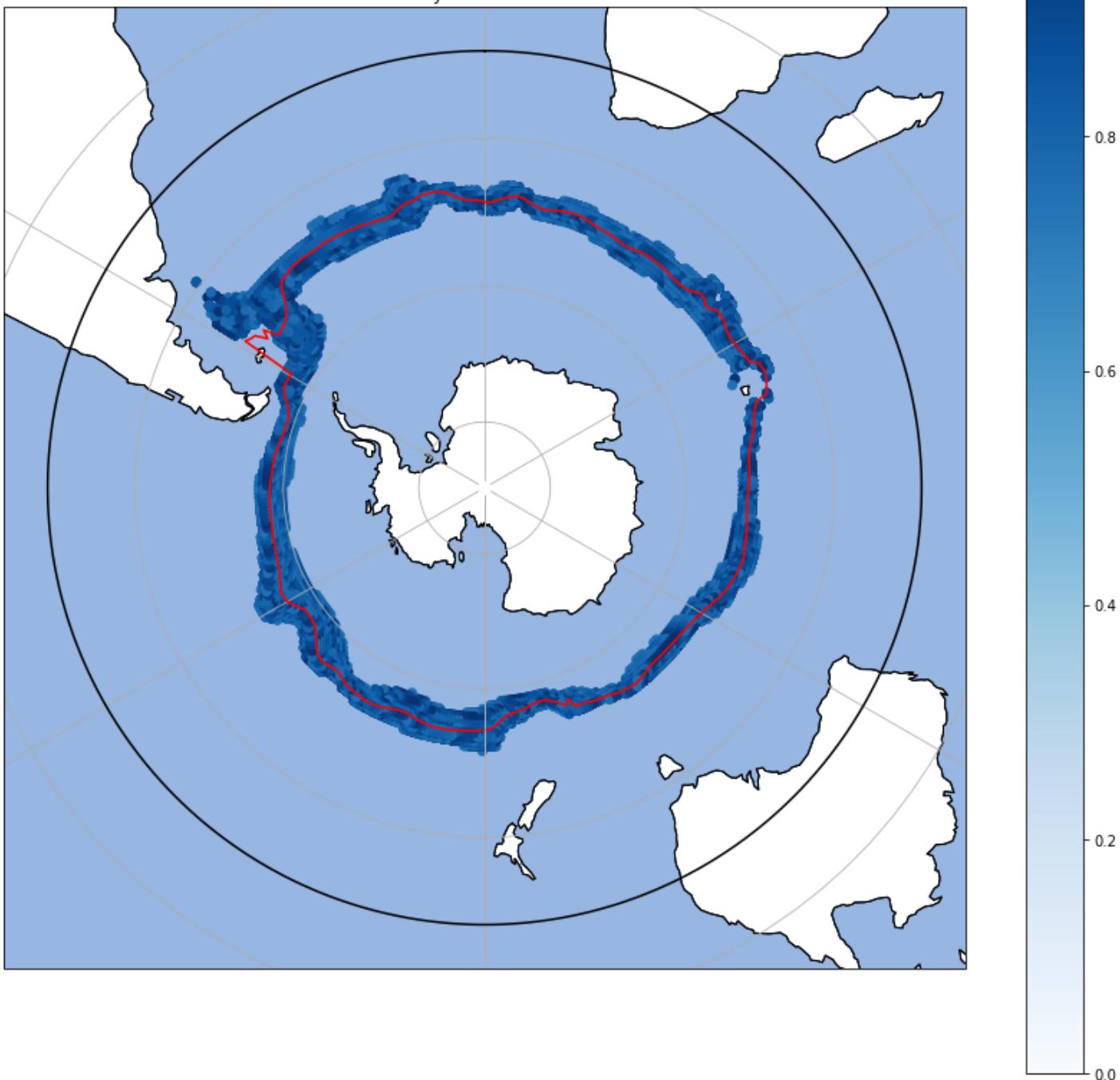
Decade singular mean latitude values calculated.

In [39]:

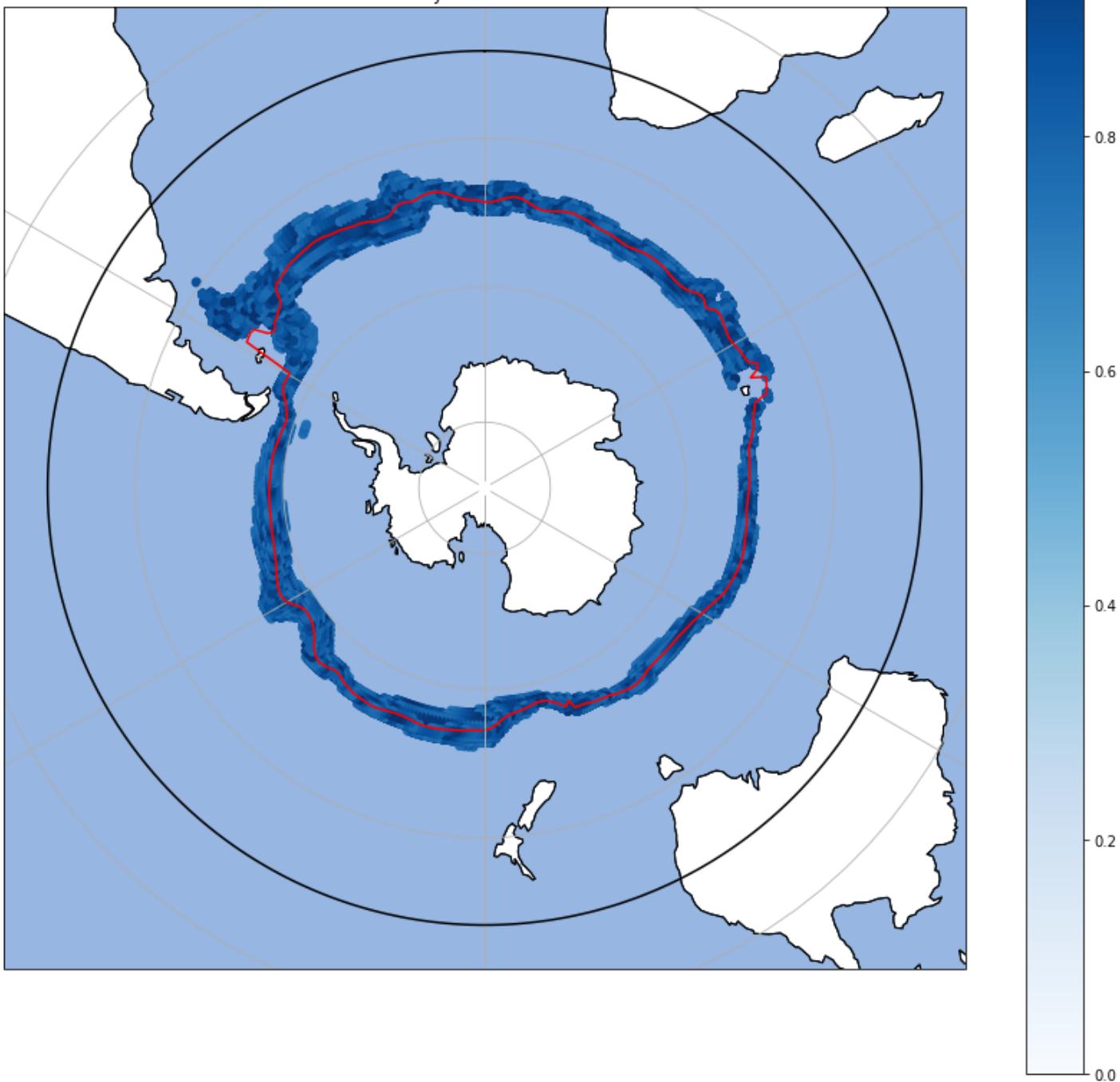
```
for i in range(len(decades)):
    locationPlotUncertaintyDF(decDFIThreshList[i], "Uncertainty above "+str(threshold)+" via decade")
    plt.plot(decDFIThreshLatMList[i].index, decDFIThreshLatMList[i]["lat"], transform=ccrs.PlateCarree())
    plt.show()

locationUncertaintyMean(decDFIThreshLatMList, decDFIThreshLatSList, threshold, (20,20), 4)
```

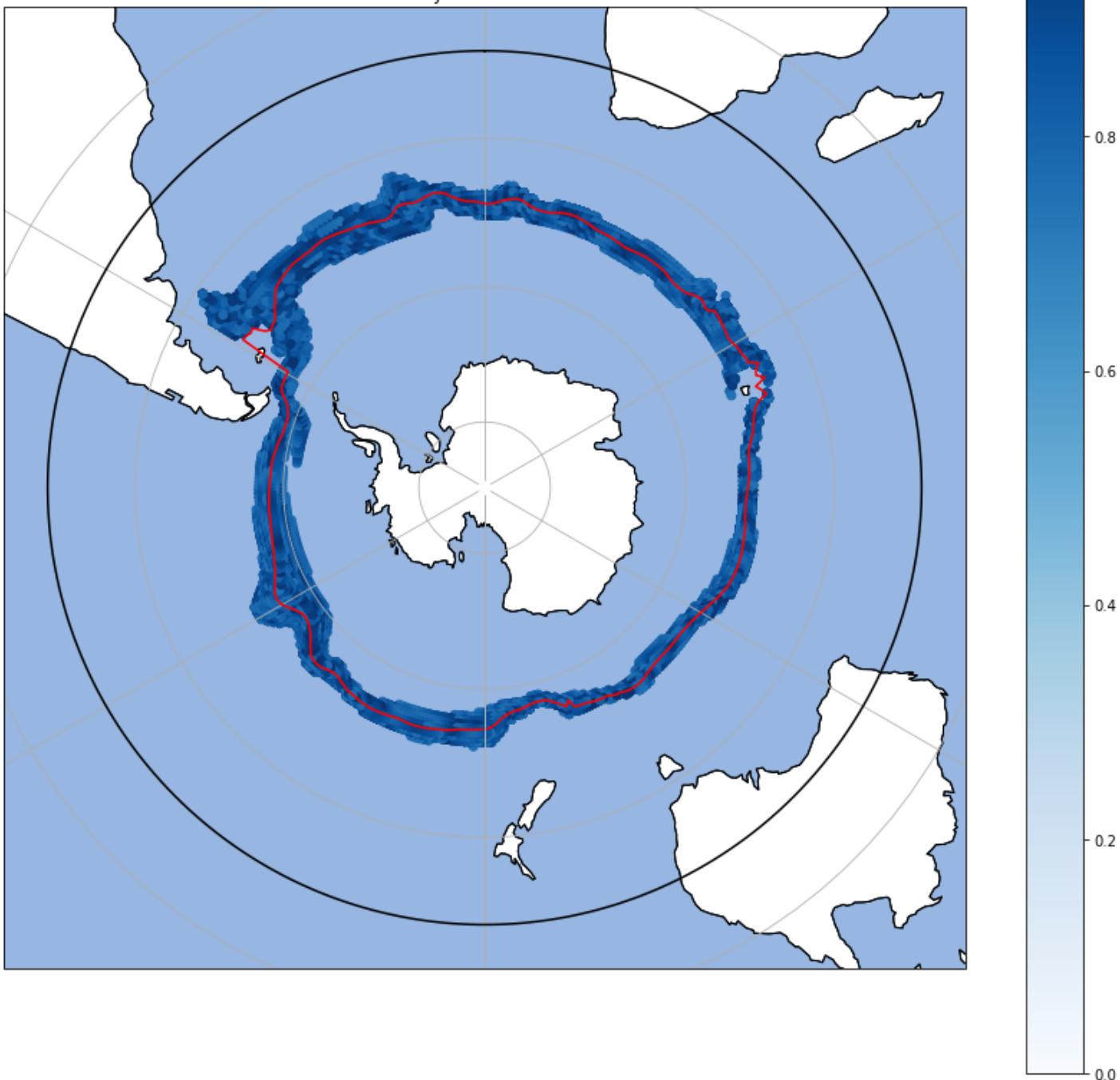
Uncertainty above 0.75 in 1980s

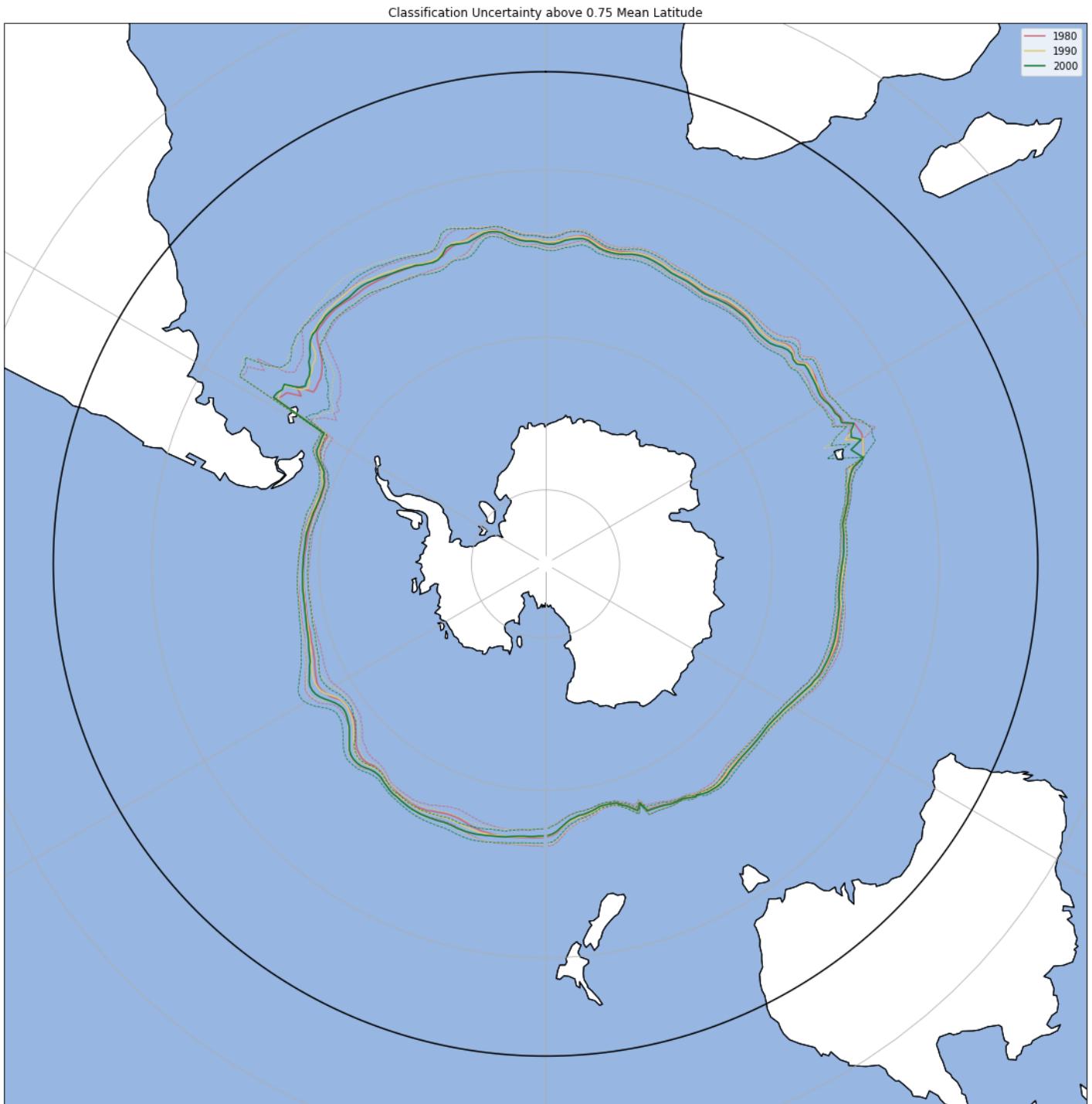


Uncertainty above 0.75 in 1990s



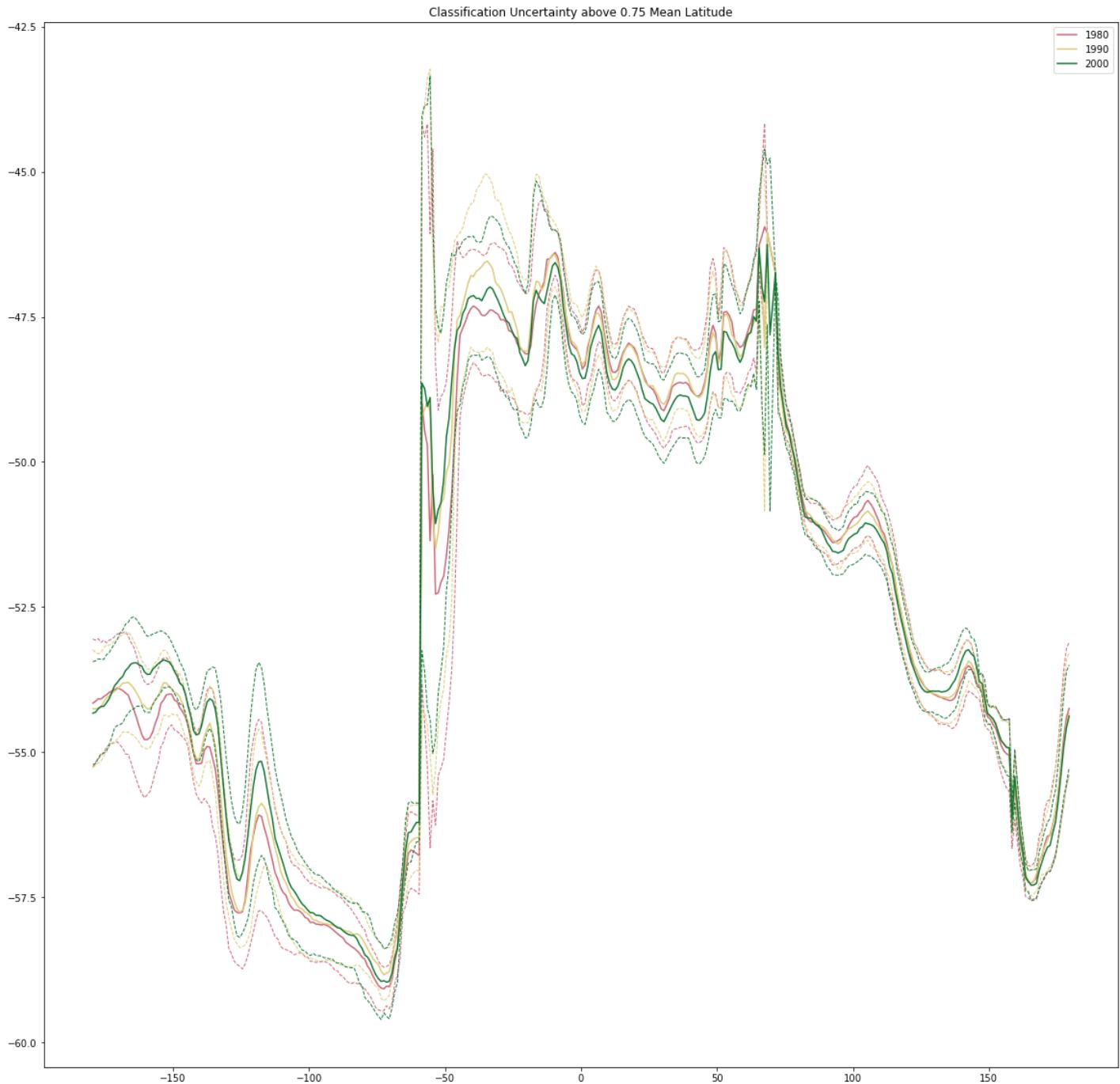
Uncertainty above 0.75 in 2000s





In [40]:

```
locationUncertaintyMeanCart(decDFIThreshLatMList, decDFIThreshLatSList, threshold, (20,20))
plt.show()
```



Exporting Monthly Latitude Means

In [41]:

```
exportName = modelName + "_LatMeans"
np.save(exportName, monthlyLatMMList)
print("Monthly Latitude means (monthlyLatMMList) exported to "+exportName+".")
```

Monthly Latitude means (monthlyLatMMList) exported to GMM_UK_2Class_R2_LatMeans.

In [42]:

```
importName = exportName+".npy"
monthlyLatMMListReload = np.load(importName)
print("Monthly Latitude means reloaded into monthlyLatMMListReload from "+importName+".")
```

Monthly Latitude means reloaded into monthlyLatMMListReload from GMM_UK_2Class_R2_LatMeans.npy.

In [43]:

```
print("Reloaded monthly latitude means.")
monthlyLatMMList
```

```
Reloaded monthly latitude means.
```

```
Out[43]: [lat
          time
          1980-01-01 -53.496106
          1980-02-01 -53.969822
          1980-03-01 -53.881028
          1980-04-01 -53.454682
          1980-05-01 -52.949074
          ...
          ...
          2009-08-01 -52.119879
          2009-09-01 -52.051606
          2009-10-01 -52.044739
          2009-11-01 -52.202208
          2009-12-01 -52.779794

          [360 rows x 1 columns],
          lat
          time
          1980-01-01 -53.295487
          1980-02-01 -53.848568
          1980-03-01 -53.741754
          1980-04-01 -53.288814
          1980-05-01 -52.848369
          ...
          ...
          2009-08-01 -51.944610
          2009-09-01 -51.822196
          2009-10-01 -51.785042
          2009-11-01 -52.014766
          2009-12-01 -52.568497

          [360 rows x 1 columns],
          lat
          time
          1980-01-01 -53.345058
          1980-02-01 -54.055194
          1980-03-01 -53.913183
          1980-04-01 -53.283171
          1980-05-01 -52.851608
          ...
          ...
          2009-08-01 -51.837059
          2009-09-01 -51.691782
          2009-10-01 -51.632764
          2009-11-01 -51.950423
          2009-12-01 -52.410042

          [360 rows x 1 columns],
          lat
          time
          1980-01-01 -53.436115
          1980-02-01 -54.497759
          1980-03-01 -54.237743
          1980-04-01 -53.584481
          1980-05-01 -53.109243
          ...
          ...
          2009-08-01 -52.134495
          2009-09-01 -51.203775
          2009-10-01 -50.813929
          2009-11-01 -51.861067
          2009-12-01 -52.356784

          [360 rows x 1 columns]]
```

End of Notebook