Journal of Geophysical Research:Solid Earth

Supporting Information for

A magnetic data correction workflow for sparse, four dimensional data

Alan R.A. Aitken[1], Lara Nigro Ramos[1], Jason L. Roberts[2], Jamin S. Greenbaum[3], Lenneke M. Jong[2], Duncan A. Young[3], Donald D. Blankenship[3]

1 – The School of Earth Sciences, The University of Western Australia, Crawley, Western Australia 6009, Australia
2 – The Australian Antarctic Division, Kingston, Tasmania 7050, Asutralia
3 – The University of Texas Institute for Geophysics, J.J. Pickle Research Campus, Austin, Texas, USA

Corresponding author: Alan Aitken (alan.aitken@uwa.edu.au)

Contents of this file

Introduction

The supporting information here presents two original algorithms used in the data processing: Text S1 presents the algorithm for multiple base station correction, simplified from the original python script and following python 3 syntax and conventions, however it is not a fully functional script. Text S2 presents the algorithm for median-based line levelling, similarly simplified from the original python script. Figures S1 to S17 show the step-by step changes in the data as a consequence of the corrections and adjustments applied. Finally we show in figure S18 the difference with the ADMAP-2 compilation

Text S1 - Algorithm for multiple base station correction.

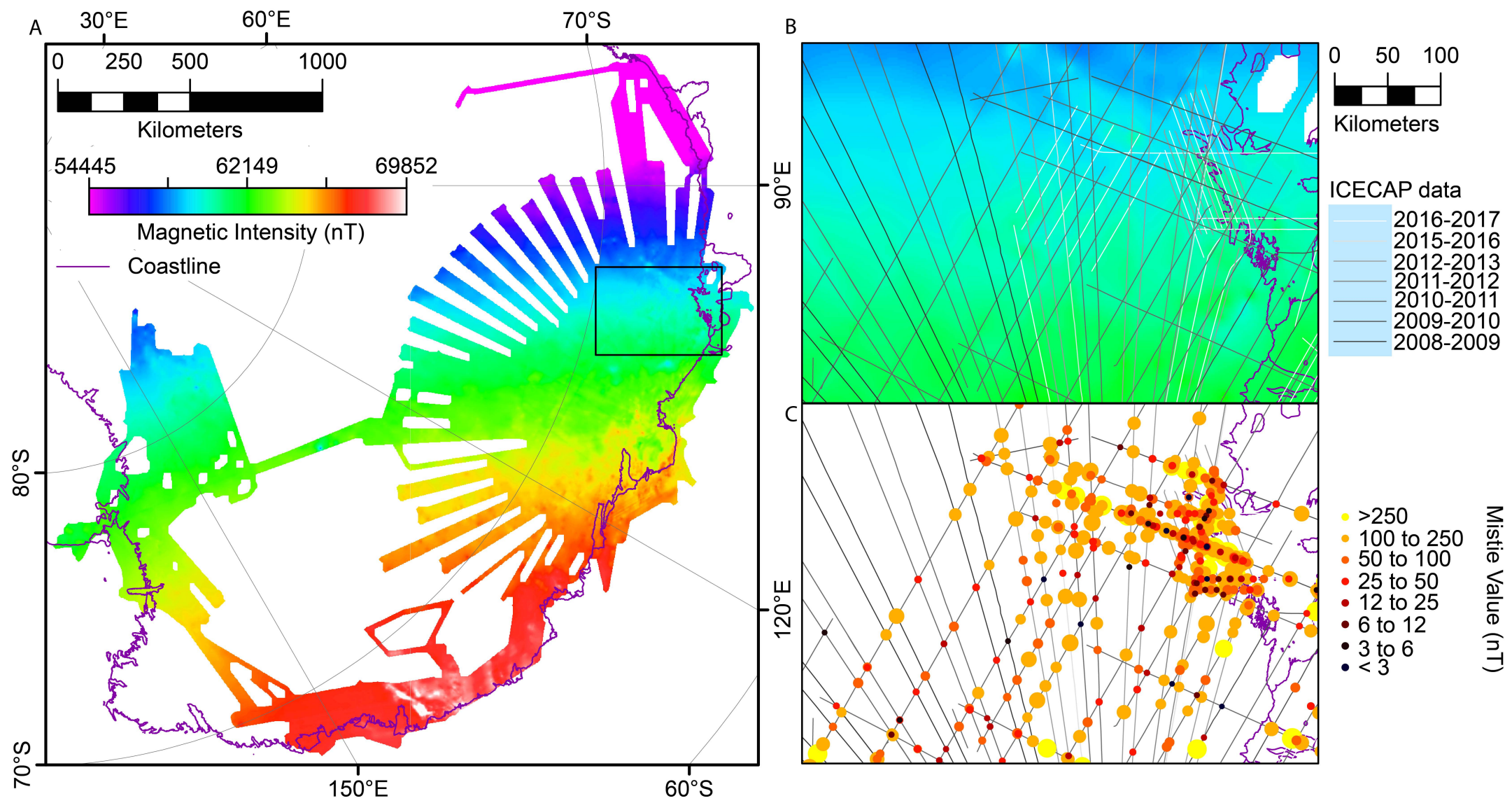
```
#Data
Data = [[FID,X,Y,Z,time,F,inclination]*ndata] # list of data locations, time, and field value
BaseValues = [BS1,BS2,...,BSn] # list of base station field values at corresponding times

#Base Station Site Information
BaseID= [BS1,BS2,...,BSn] #base station IDs
BaseX = [BS1,BS2,...,BSn] #base station x coordinates
BaseY = [BS1,BS2,...,BSn] #base station y coordinates
BaseZ = [BS1,BS2,...,BSn] #base station z elevations
BaseInc = [BS1,BS2,...,BSn] #base station magnetic inclinations
z0 = [BS1,BS2,...,BSn] #source elevation (e.g. bedrock topography)

#Global variables
exponent = n #exponent for inverse distance calculation
maxbase= n #maximum number of bases to be used
maxinc = n #maximum permitted inclination difference
w_h = n #plate half-width

# Define function to compute base weights for given data point
def BaseWeights(X,Y,Z, inclination): #information for specific data point from input file
        #calculate 3D distance and inclination difference to each base
        for i in enumerate BaseID:
                Vector = [X-BaseX[i],Y-BaseY[i],Z-BaseZ[i]]
                BaseDist[i] = sqrt(sum(j**2 for j in Vector)
                IncDiff[i] = abs(BaseInc [i]-inclination)
        #sort distances in proximity order
        BaseDistSorted = sorted(BaseDist)
        #define length scale as a function of the furthest permitted base
        LengthScale = sqrt(BaseDistSorted[maxbase-1]**2)
        #for included bases, define weighting with inverse distance law
        for i in enumerate BaseID:
                if IncDiff[i] < maxinc and BaseDist[i] ≤ LengthScale:
                        BaseWeights[i] = (1-(BaseDist[i])/LengthScale))**exponent
                else:
                        BaseWeights[i] = 0.
        return (BaseWeights)

# Define function to apply vertical damping
def VerticalDamping(Zdata, BaseValues):
        D = []
        #calculate damping for each base
        for i in enumerate BaseID:
                hs = Zdata-z0[i] #station height above source
                bs = Zbase[i] – z0[i] #base height above source
                #trigonometric terms
                A = sin (2* BaseInc[i])
```

```
            B = cos(BaseInc[i])**2- sin(BaseInc[i])**2
            #inverse distance squared from plate-edge to height at central location
            irs = 1/(hs**2+ w_h**2)
            irb = 1/(hb**2+ w_h**2) #ditto for base height
            # damping/amplification adapted from Telford et al., 1990, eq. 3.59b
            D[i] = irs*(A-w_h*B)-irs*(A+w_h*B))/(irb*(A-w_h*B)-irb*(A+w_h*B))
            #apply damping/amplification to base value
            BaseValuesDamped[i] = BaseValues[i]*D[i]
        return (BaseValuesDamped)

# Define function to apply correction for each base
def BaseCorr(BaseValues, BaseWeights):
        #calculate overall correction using values and weights
        Correction = 0.
        for i in enumerate BaseID:
            Correction += BaseWeights[i]*BaseVals[i]
        #calculate overall leverage using values and weights
        Leverage = 0.
        for i in enumerate BaseID:
            Leverage += abs(BaseWeights[i]*(BaseVals[i]-Correction))
        return (Correction, Leverage)

# Apply to Data
for i in enumerate Data:
        #compute weights
        Weights[i] = BaseWeights(X[i],Y[i],Z[i],inclination[i])
        #damp weights to station height
        BaseDataDamped[i] = VerticalDamping(Z[i],BaseValues[i])
        #derive correction
        Correction[i] = BaseCorr(BaseDataDamped[i], Weights[i])
        #apply correction
        CorrectedTMI[i] = TMI[i]-Correction[i]
```

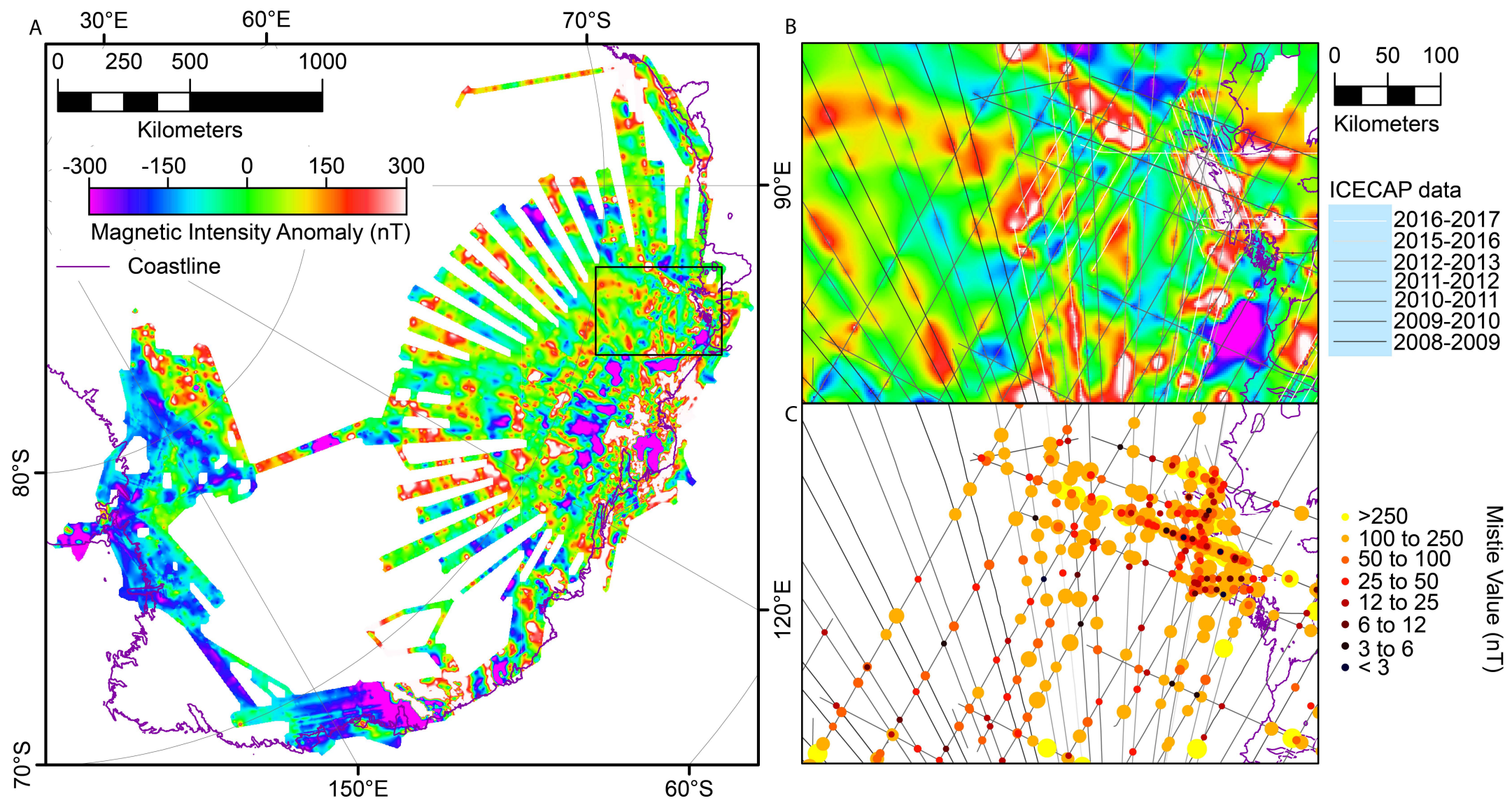
Text S2 - Algorithm for median-based line levelling.

```
#Data
LineList = [L1,L2,...,Ln] #list of lineIDs
TIES = [[[LineID,TieID,LineTMI,TieTMI,CrossDiff]*nties[LineID]]*nlines] #list of lists with intersection data

#Global Variables
DesMaxMedian = n #desired maximum median cross-tie error
MaxCycles = n #maximum number of permitted cycles

#Define function to get median cross-tie error for each line
def GetMedian(ties):
        for (i,d) in enumerate (ties):
            CrossDiffs.append(ties[i][-1]) #get the CrossDiff value for each tie
        CrossDiffs.sort() #sort these in numerical order
        r = len(CrossDiffs)//2 #find the midpoint
        if len(CrossDiffs)%2:
            median = Tievals[r] #central value if odd
        else:
            median = (CrossDiffs [r-1]+ CrossDiffs[r])/2 #average of two if even
        return median

#Define function to apply correction
def DoCorrections(ties,median):
        for i in enumerate ties:
            ties[i][2] += median #correct LineTMI by adding median
            ties[i][4] -= median #correct residual CrossDiff by subtracting median
        return (ties)

#Define function to find co-intersections on other lines
def GetCrosses(ties,median):
        crosses = []
        for i in enumerate ties:
            crosses.append(ties[i][0],ties[i][1],median]) #identify line pairs
        return crosses

#Define function to propagate correction to affected intersections on other lines
def Propagate (ties,lineID,median):
            for i in enumerate ties:
                if ties[i][1] == lineID:
                    ties[i][-2] +=median #correct TieTMI by adding median
                    ties[i][-1] +=median #adjust CrossDiff byadding median
            return ties

#Apply algorithm to data
#Get median CrossDiff values for each line in TIES
MEDS =[]
for (i,d) in enumerate (TIES): #loop across the lines
```

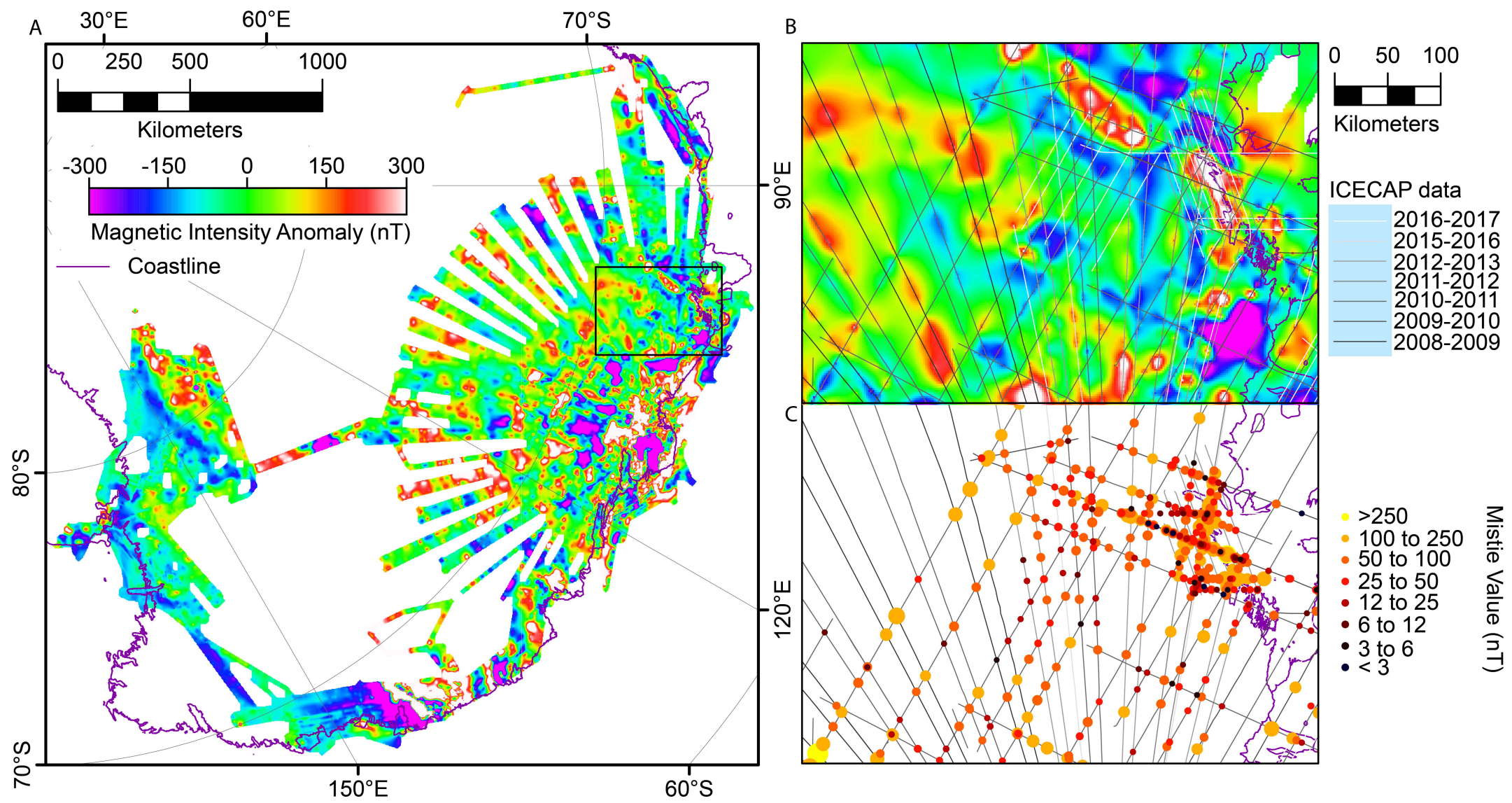
```
        M = GetMedian(TIES[i])
        MEDS.append([i,M])
#sort list of lines by absolute value of median, highest first
LineOrder = sorted(MEDS,key=lambda x:abs(x[1]), reverse = True)
#extract the highest absolute value
MaxMedian = LineOrder[0][1]
#set up outer loop to iterate through all lines over several cycles
Cycle = 0.
while abs(MaxMedian) > DesMM and Cycle < MaxCycles:
        #Set up inner loop to do corrections line by line, worst to best
        for i in enumerate LineOrder:
                lineID = LineOrder[i][0] #get LineID
                median = MEDS[lineID][1] #get associated signed median value
                TIES[lineID] = DoCorrections(TIES[LineID],medval) #apply correction to line
                MEDS[lineID][1] = GetMedian(TIES[LineID]) #recalculate median and update
        list.
                Crosses = GetCrosses (TIES[LineID],median) #identify co-intersections
                for i in enumerate Crosses:
                        lineID = Crosses[i][0]
                        median = Crosses[i][2]
                TIES[tieID] = Propagate(TIES[tieID],lineID,median) #propagate changes to co-
        intersections
                MEDS[tieID][1] = GetMedian(TIES[TieID]) #update medians in unsorted list
                #re-sort medians by absolute value for next cycle
                LineOrder = sorted(MEDS,key=lambda x:abs(x[1]), reverse = True)
                MaxMedian = LineOrder[0][1]
                Cycle += 1
```



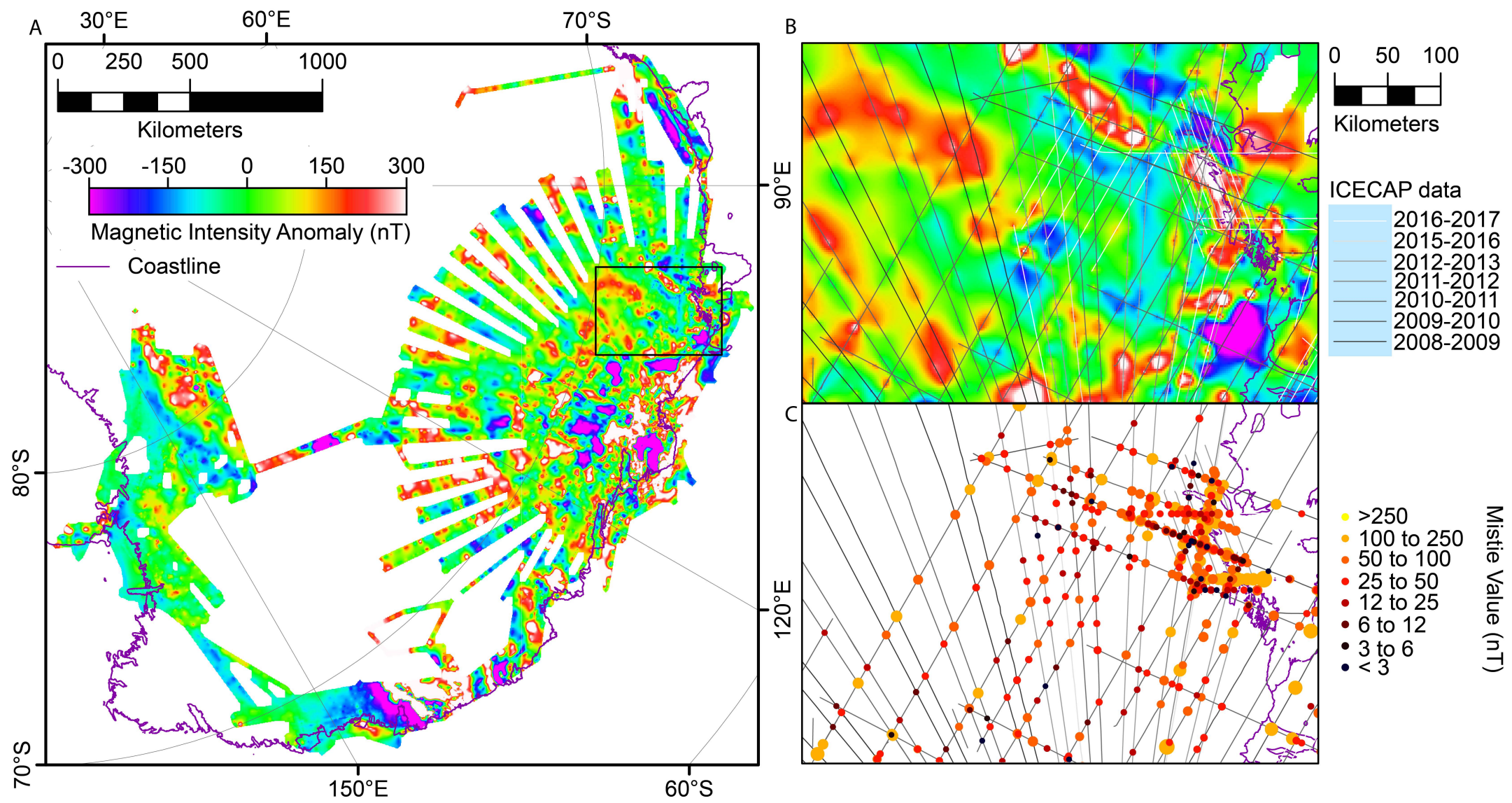
Supplementary Figure 1: Raw Total Magnetic Intensity data
A - over entire dataset, B - in selected region (same colour scale as A), C- residual intersection misties in selected region



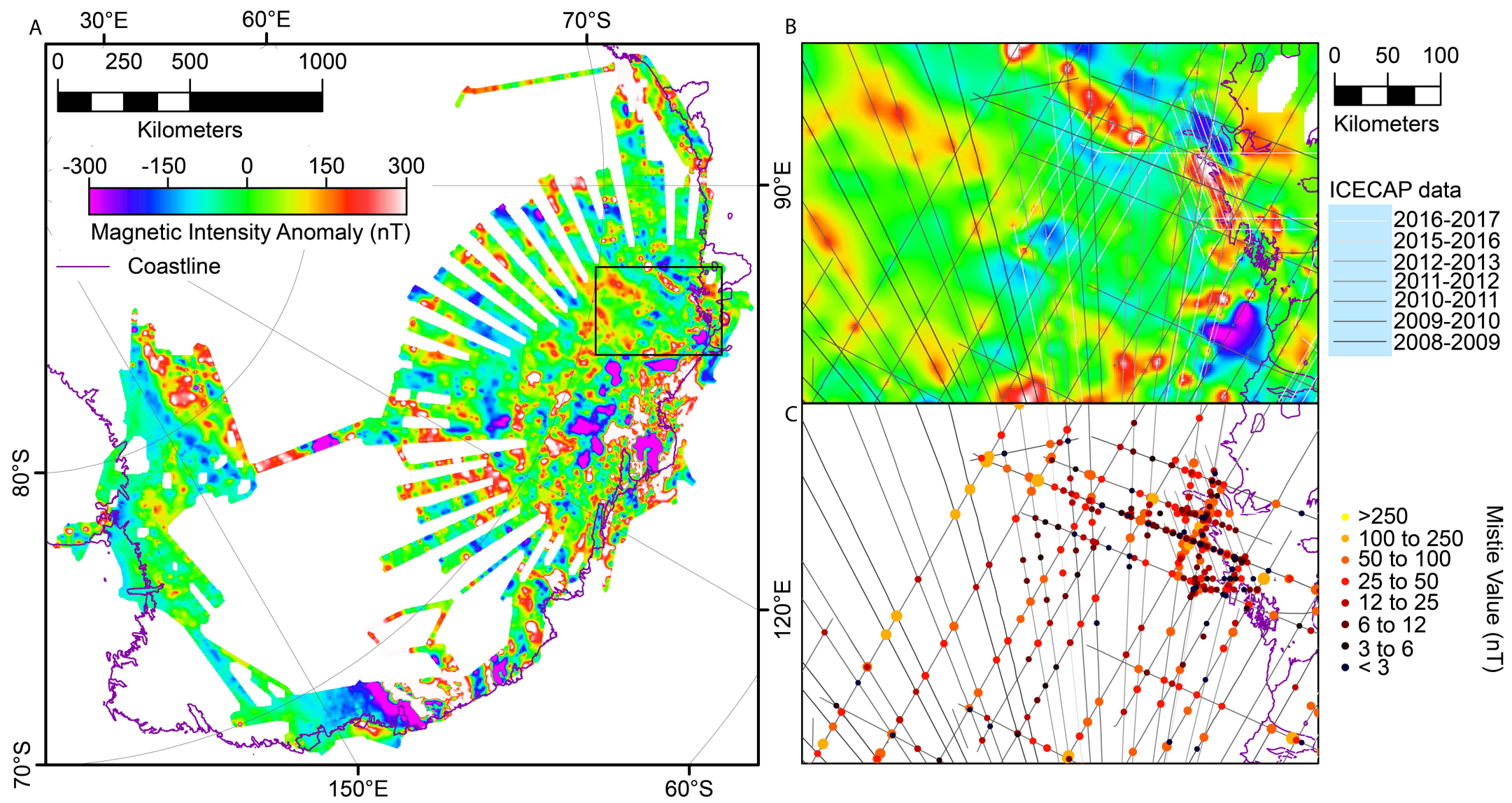
Supplementary Figure 2: IGRF-corrected (t_0) magnetic intensity data
A - over entire dataset, B - in selected region (same colour scale as A), C- residual intersection misties in selected region



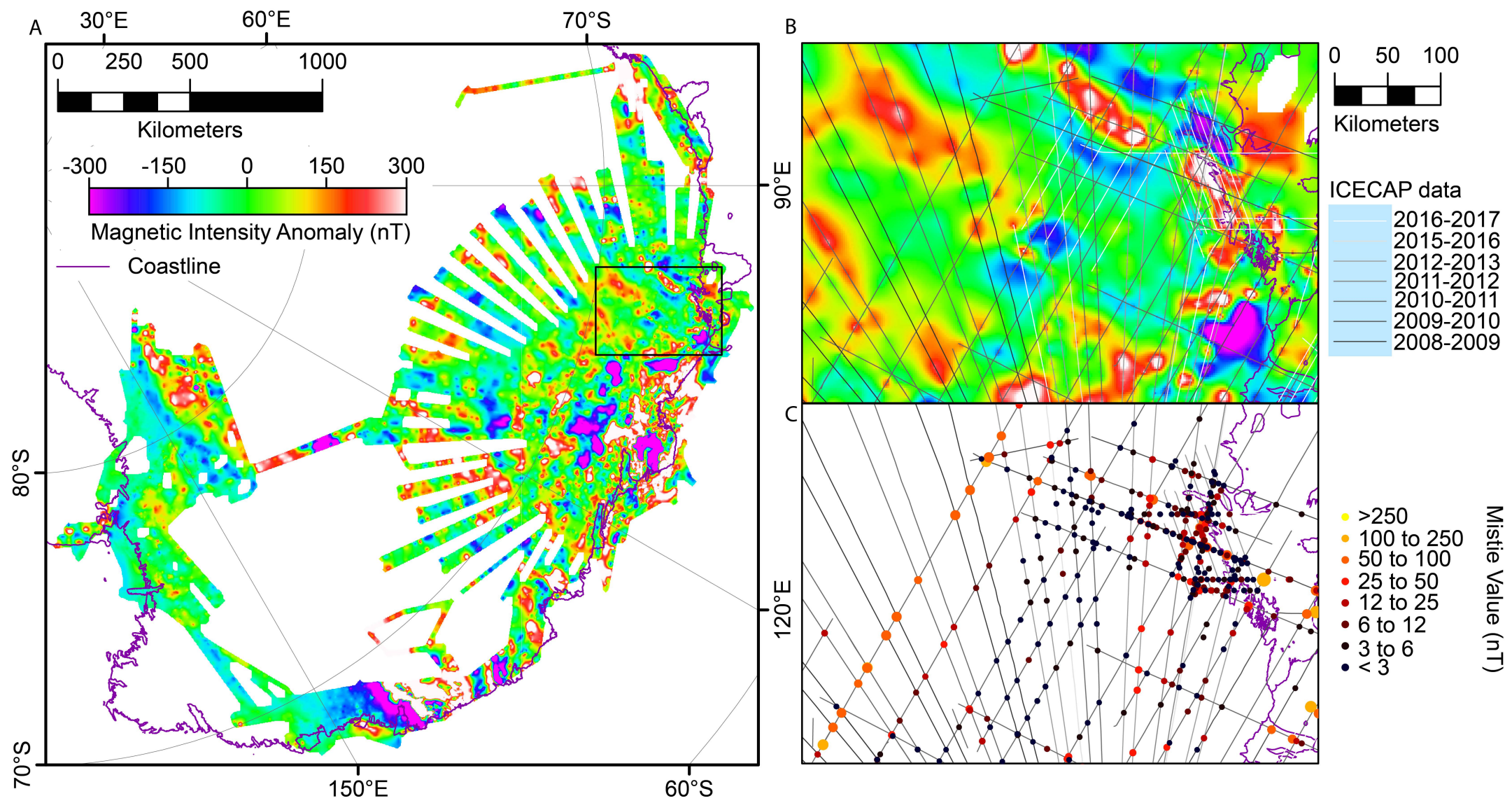
Supplementary Figure 3: IGRF and POMME corrected (t_t) magnetic intensity data
A - over entire dataset, B - in selected region (same colour scale as A), C- residual intersection misties in selected region



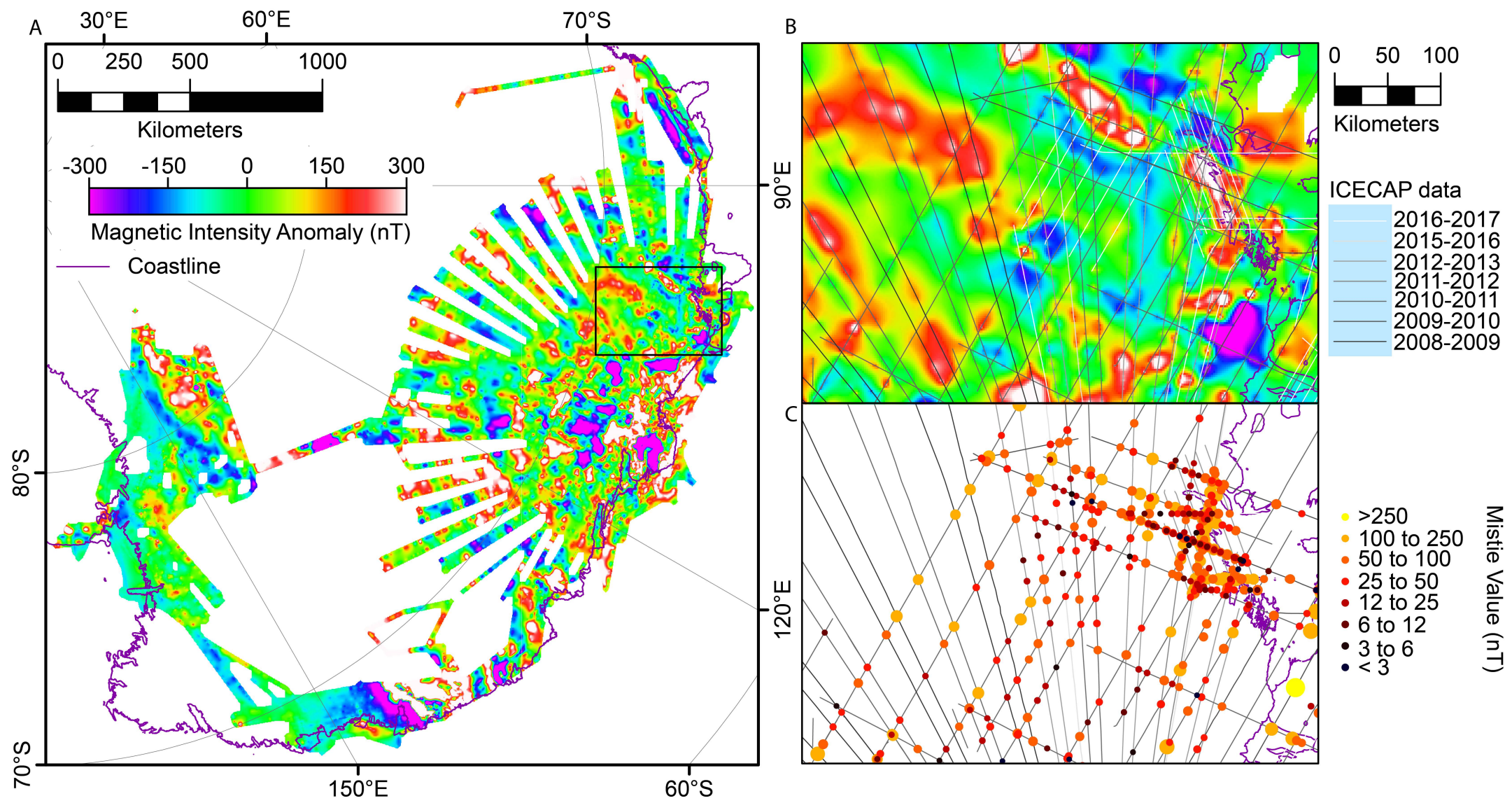
Supplementary Figure 4: IGRF, POMME and Base-station corrected magnetic intensity data
A - over entire dataset, B - in selected region (same colour scale as A), C- residual intersection misties in selected region



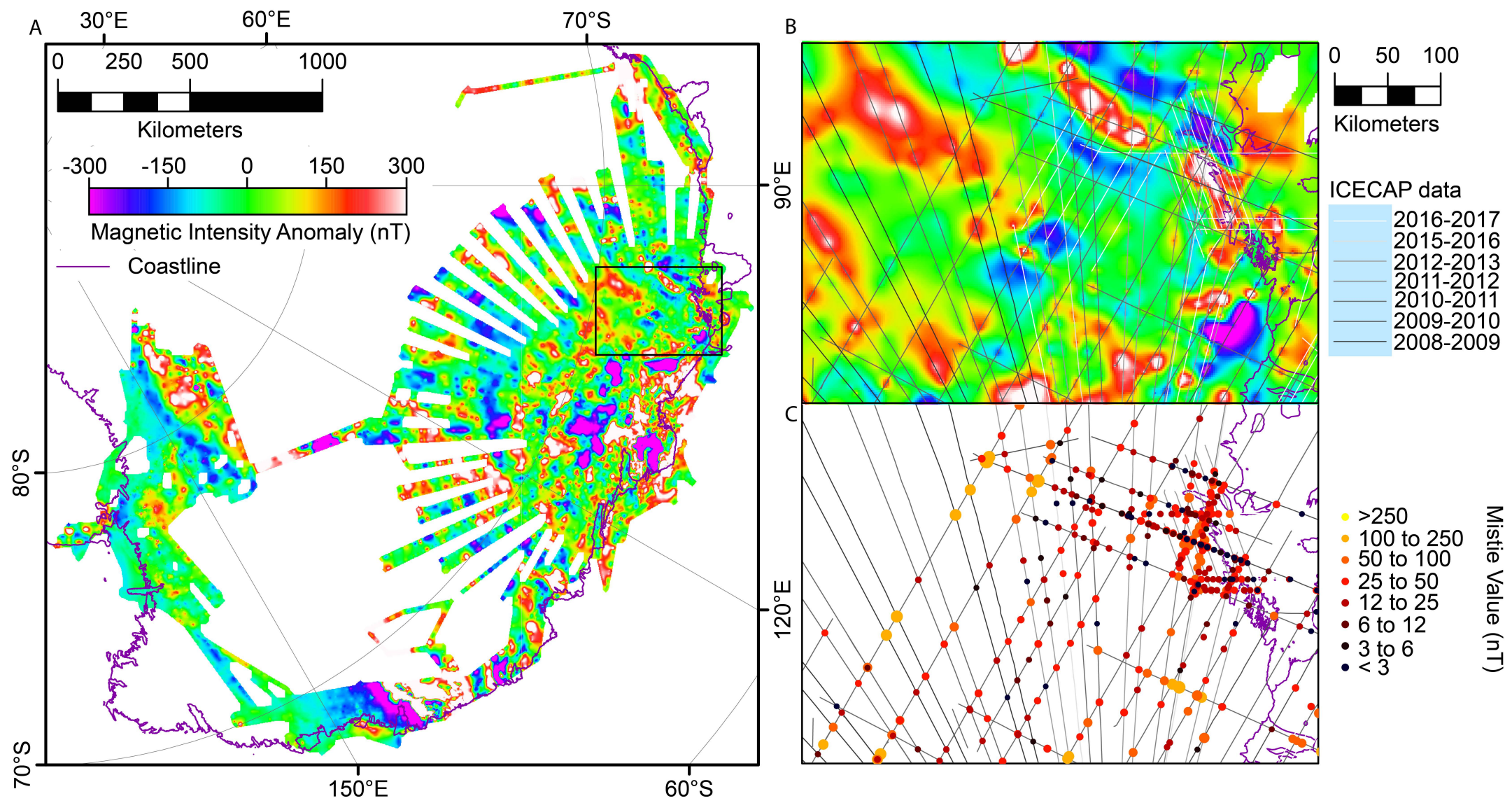
Supplementary Figure 5: IGRF, POMME and Base-station corrected magnetic intensity data, median levelled (DC-shift only)
A - over entire dataset, B - in selected region (same colour scale as A), C- residual intersection misties in selected region



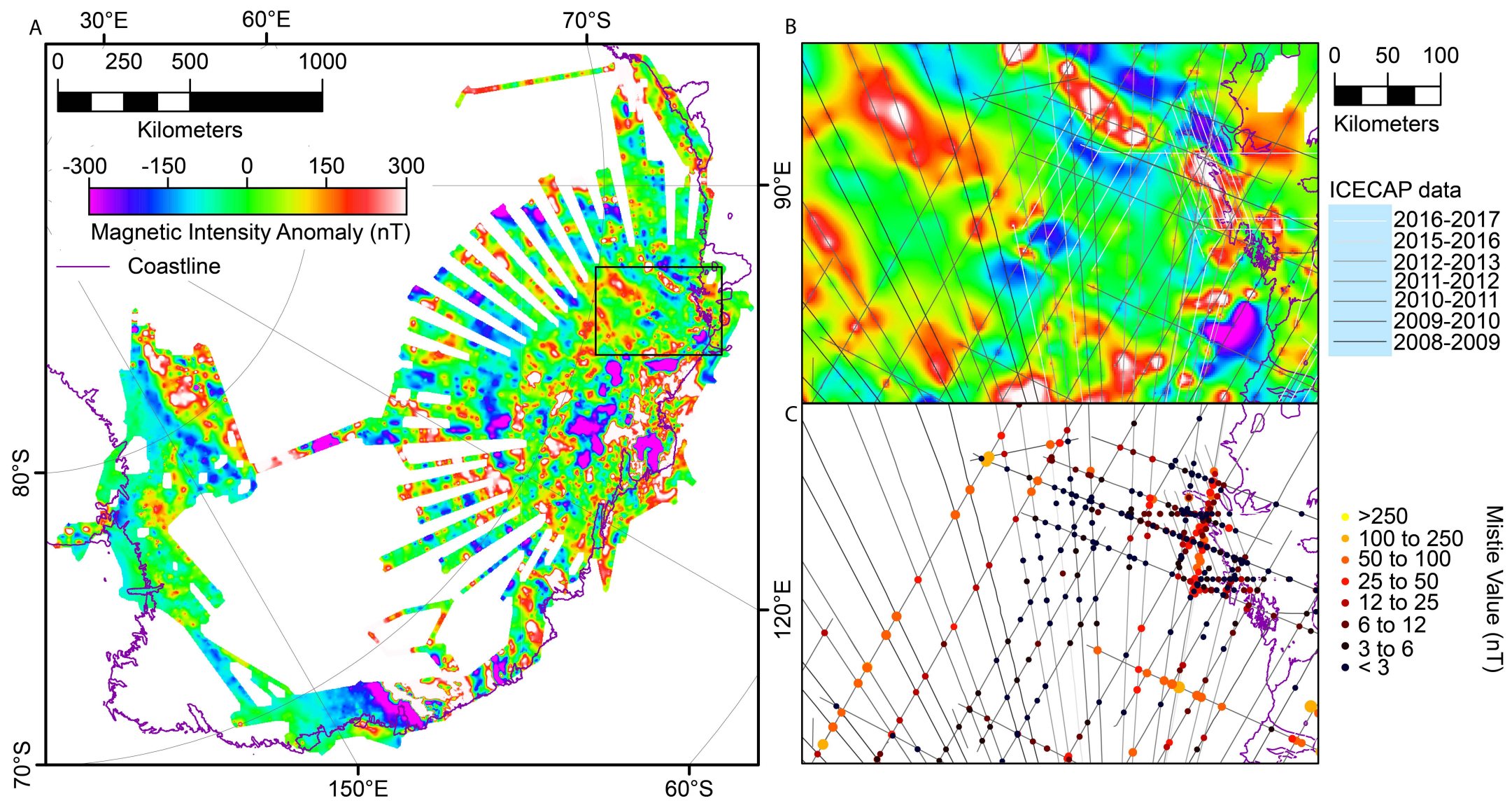
Supplementary Figure 6: IGRF, POMME and Base-station corrected magnetic intensity data, median and spline levelled
A - over entire dataset, B - in selected region (same colour scale as A), C- residual intersection misties in selected region



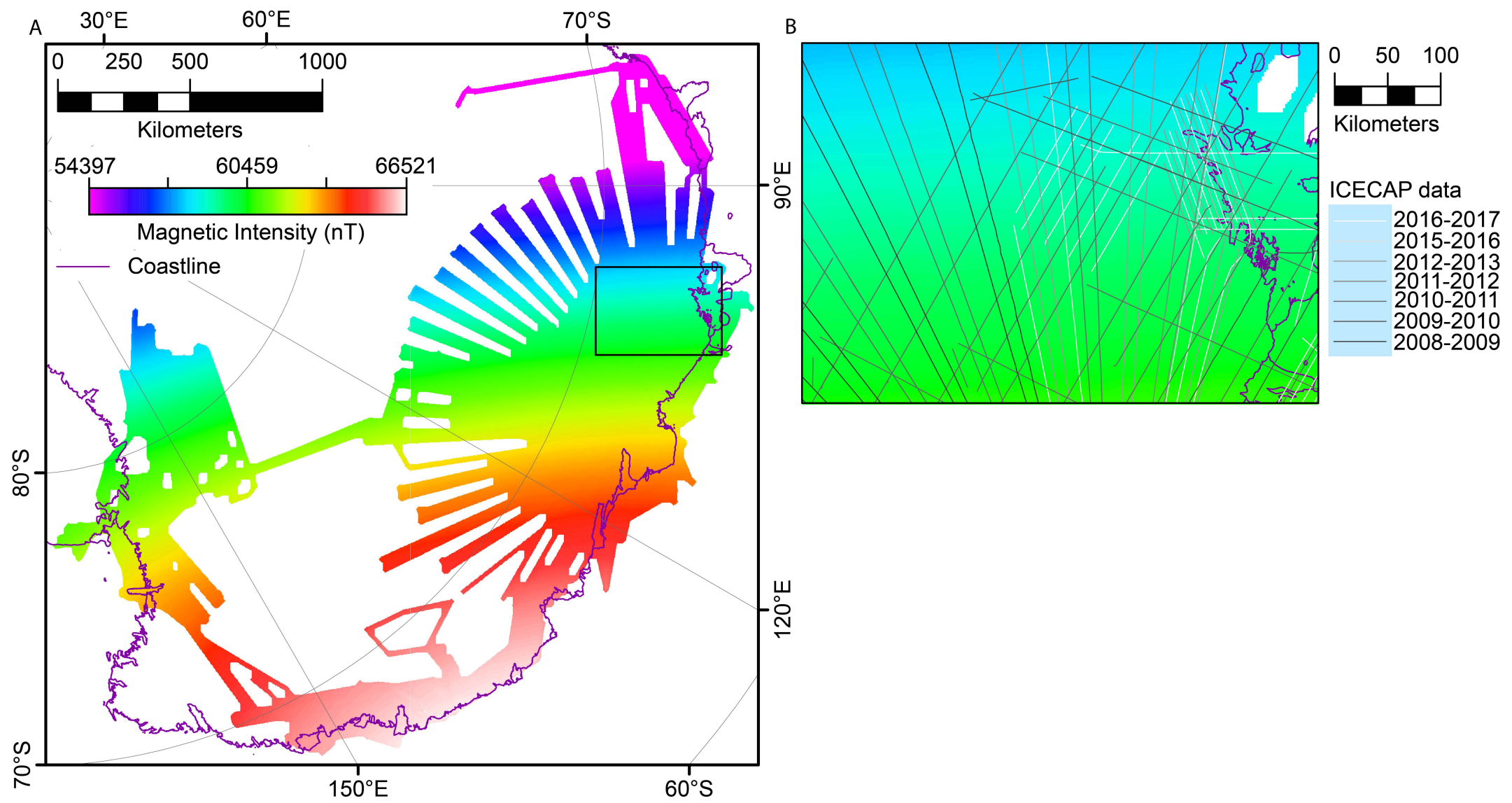
Supplementary Figure 7: IGRF, POMME and Base-station corrected magnetic intensity data, elevation adjusted to 2000m above WGS84 ellipsoid
A - over entire dataset, B - in selected region (same colour scale as A), C- residual intersection misties in selected region



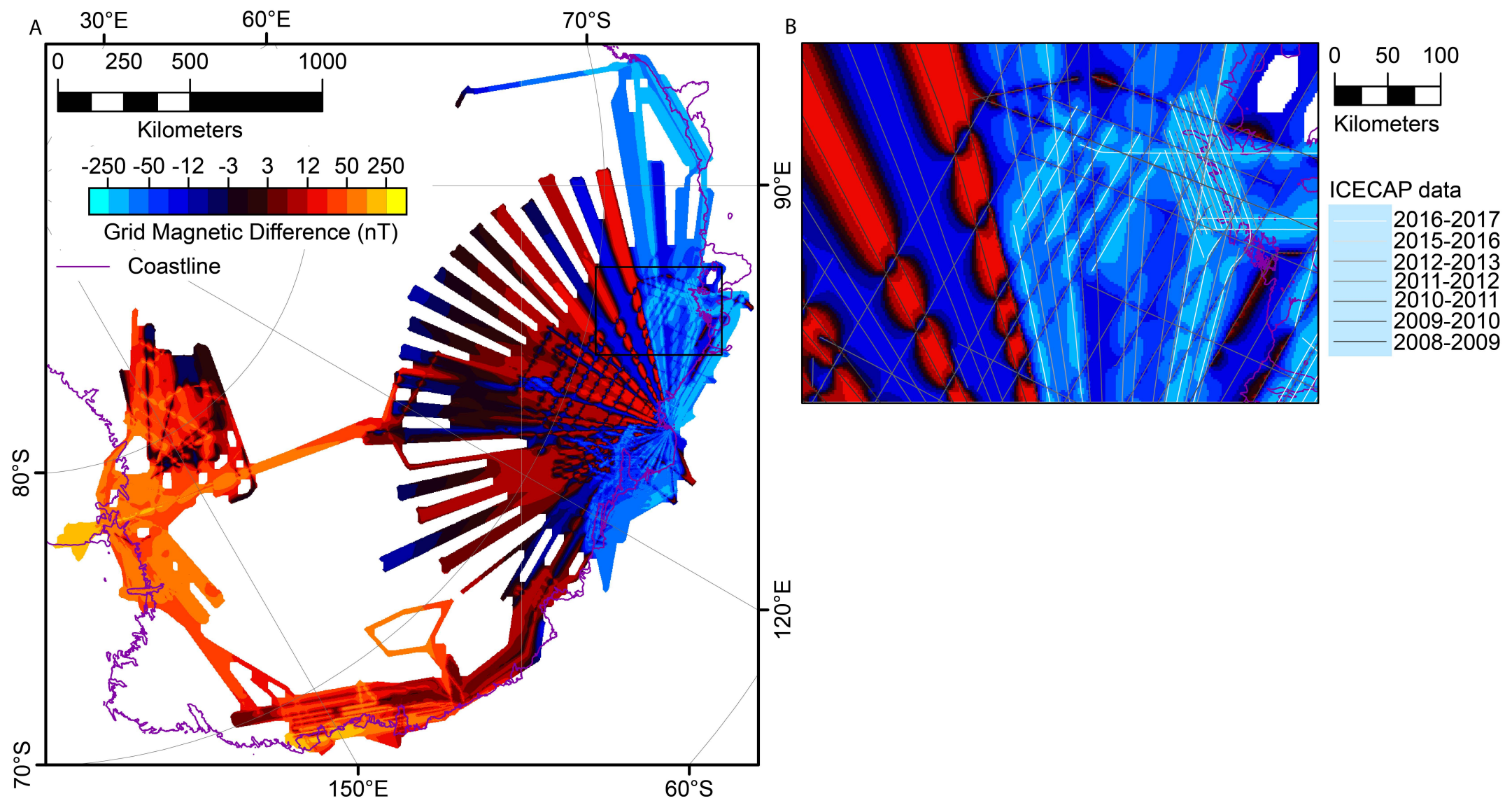
Supplementary Figure 8: IGRF, POMME and Base-station corrected magnetic intensity data, elevation adjusted to 2000m, median levelled (DC-shift only)
A - over entire dataset, B - in selected region (same colour scale as A), C- residual intersection misties in selected region



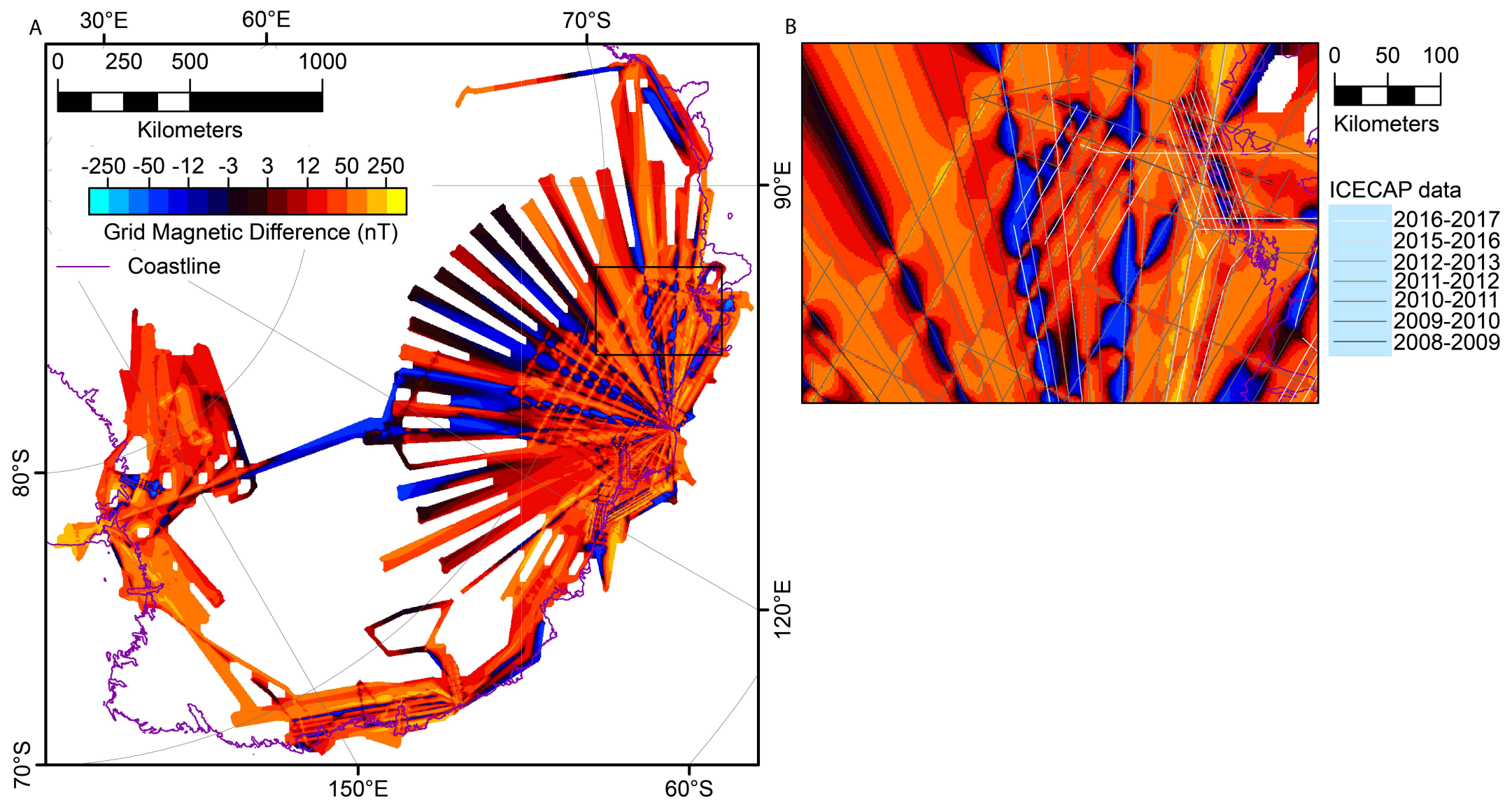
Supplementary Figure 9: IGRF, POMME and Base-station corrected magnetic intensity data, elevation adjusted to 2000m, median and spline levelled
A - over entire dataset, B - in selected region (same colour scale as A), C- residual intersection misties in selected region



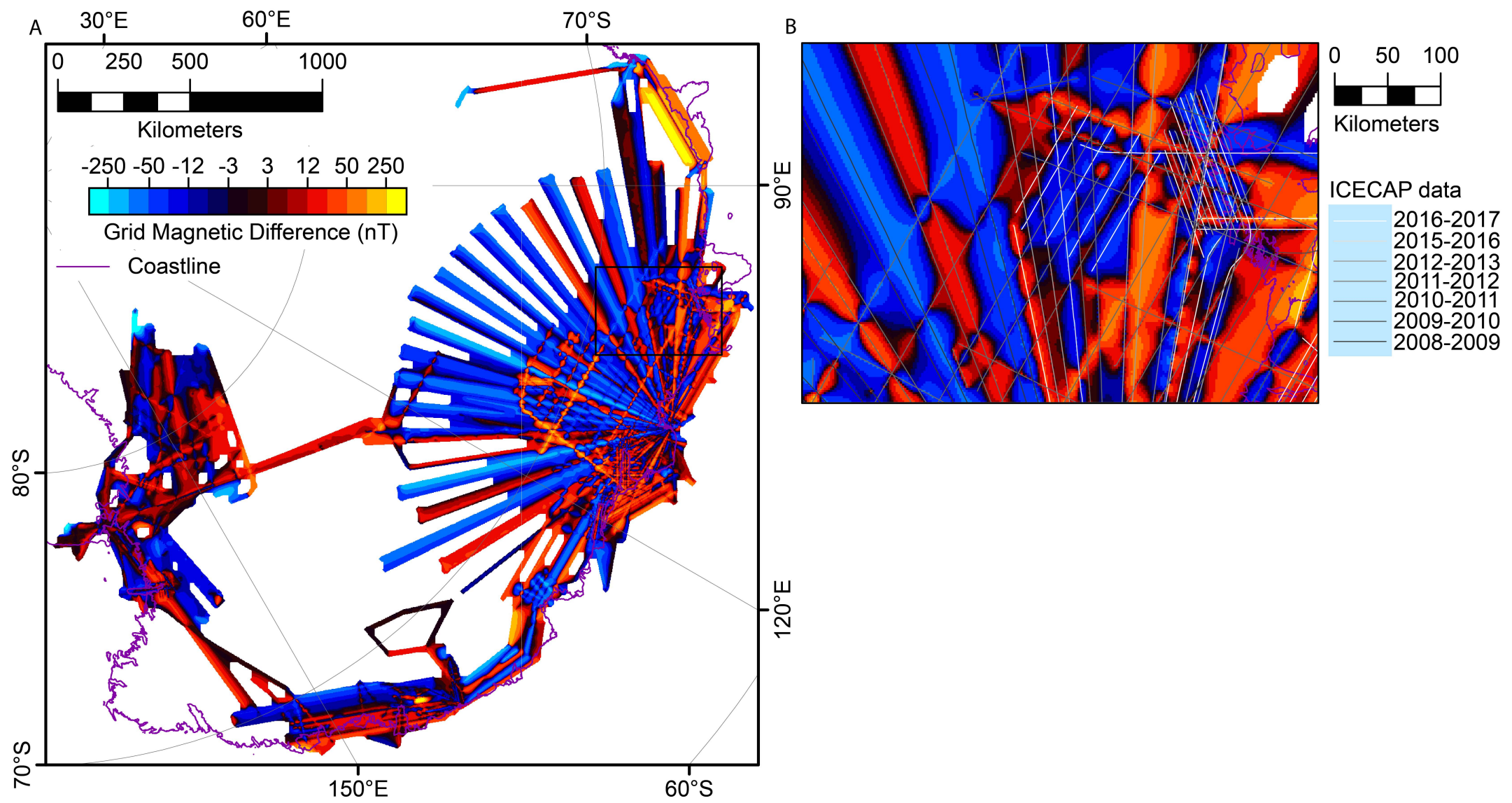
Supplementary Figure 10: Effect of IGRF correction (difference of Supp. Fig. 1 and Supp. Fig. 2)
A - over entire dataset, B - in selected region (same colour scale as A)



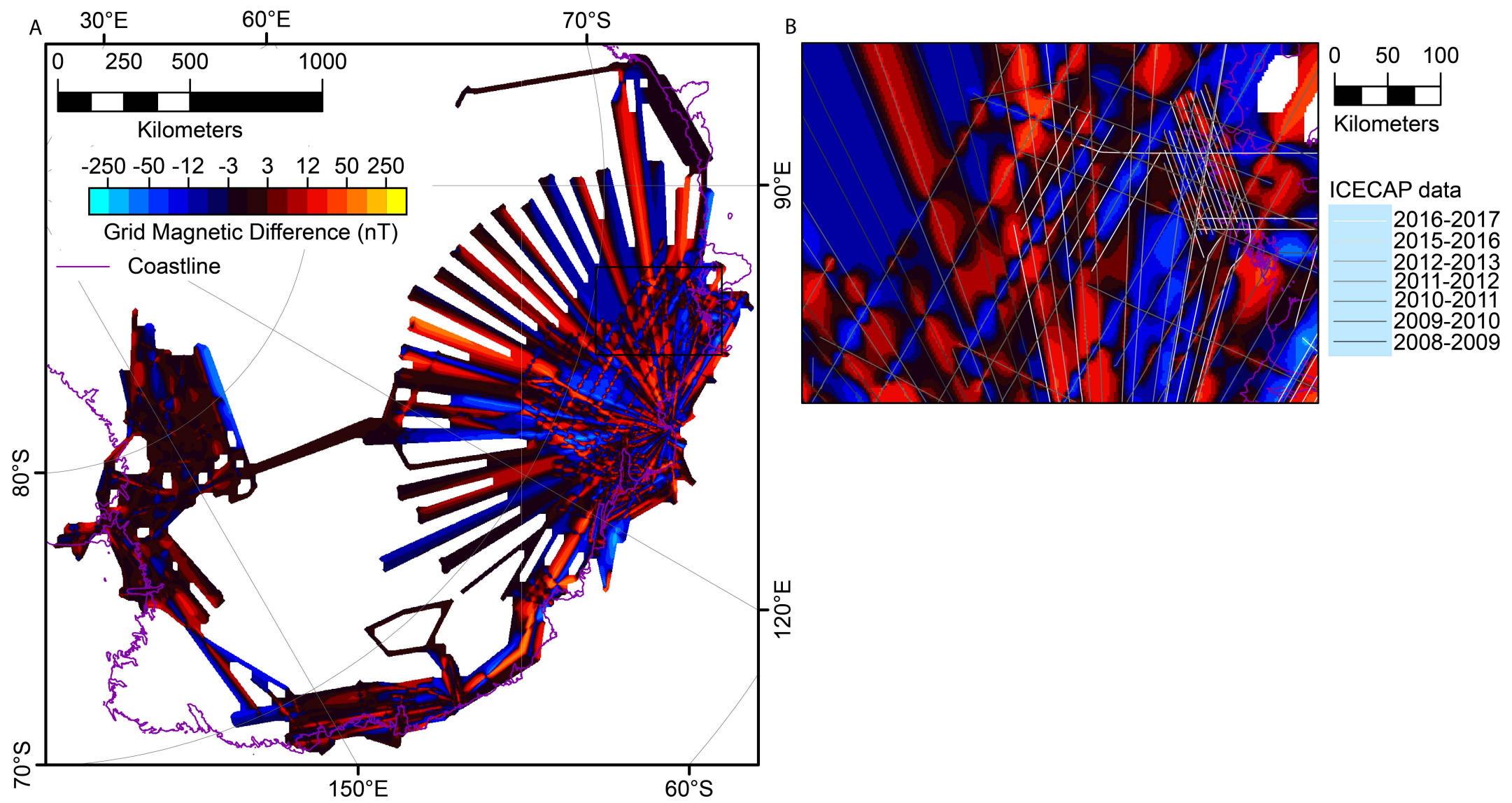
Supplementary Figure 11: Effect of POMME correction (difference of Supp. Fig. 3 and Supp. Fig. 2)
A - over entire dataset, B - in selected region (same colour scale as A)



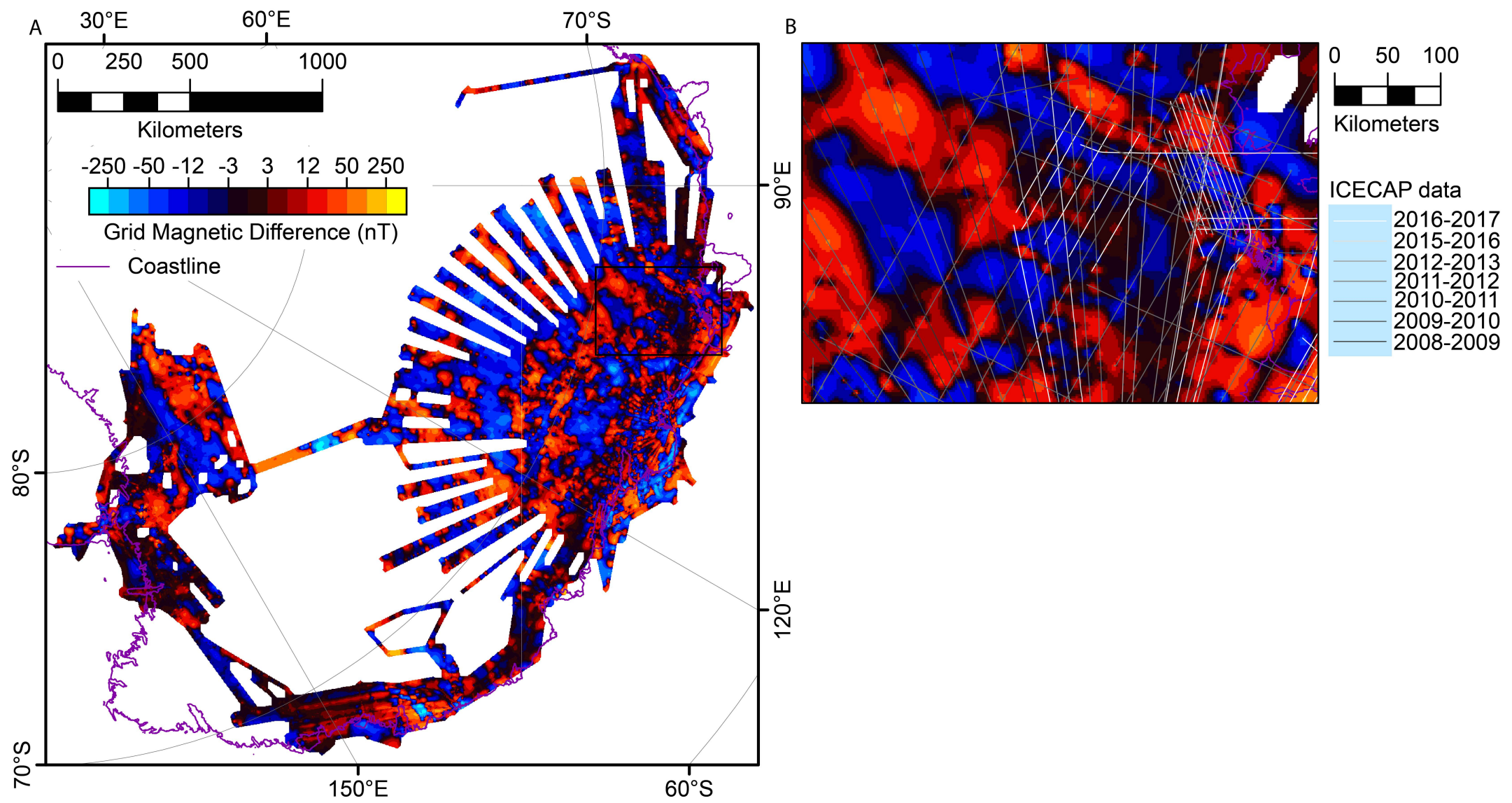
Supplementary Figure 12: Effect of Base Station correction (difference of Supp. Fig. 4 and Supp. Fig. 3)
A - over entire dataset, B - in selected region (same colour scale as A)



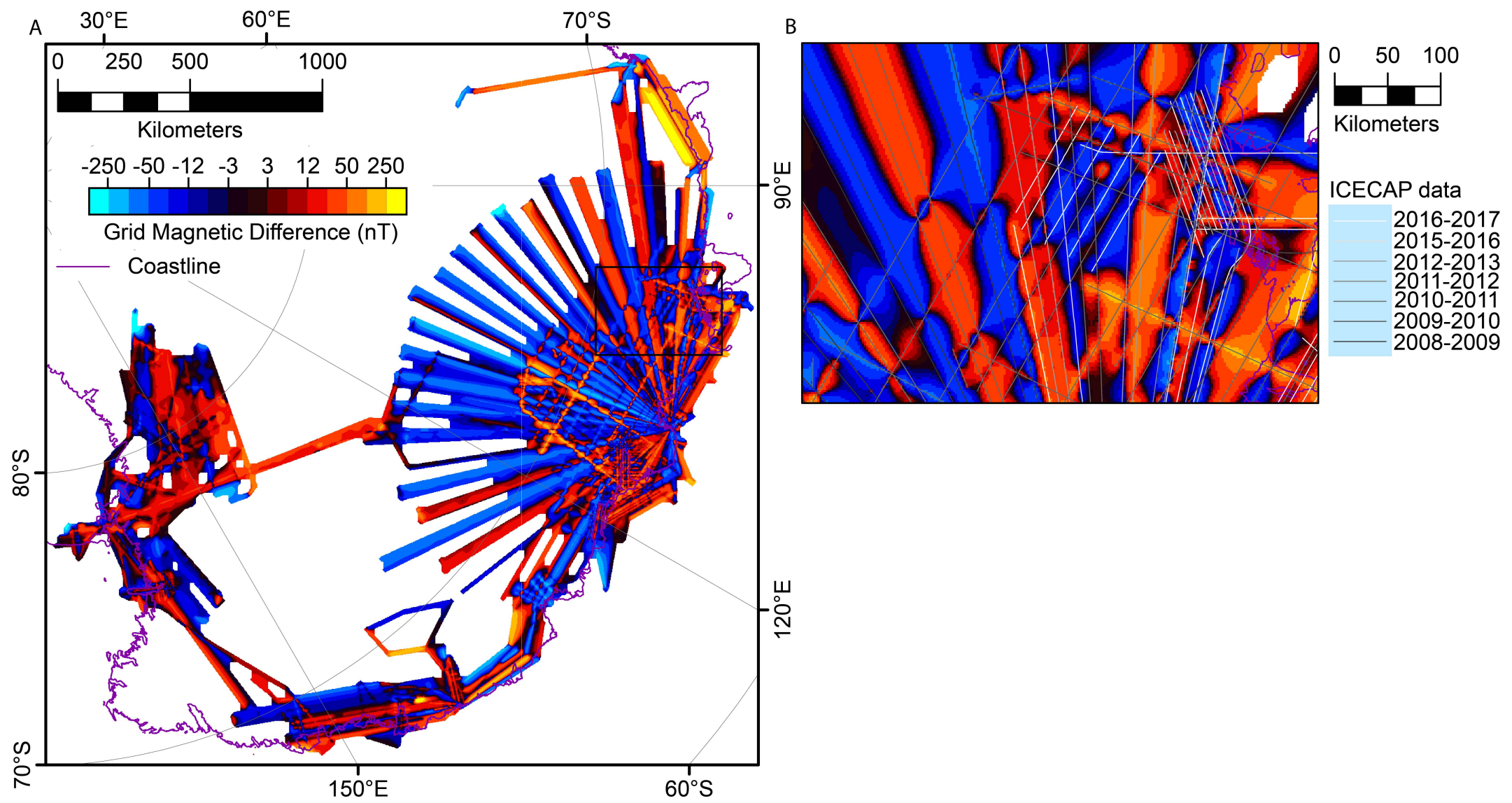
Supplementary Figure 13: Effect of median levelling on corrected TMI (difference of Supp. Fig. 5 and Supp. Fig. 4)
A - over entire dataset, B - in selected region (same colour scale as A)



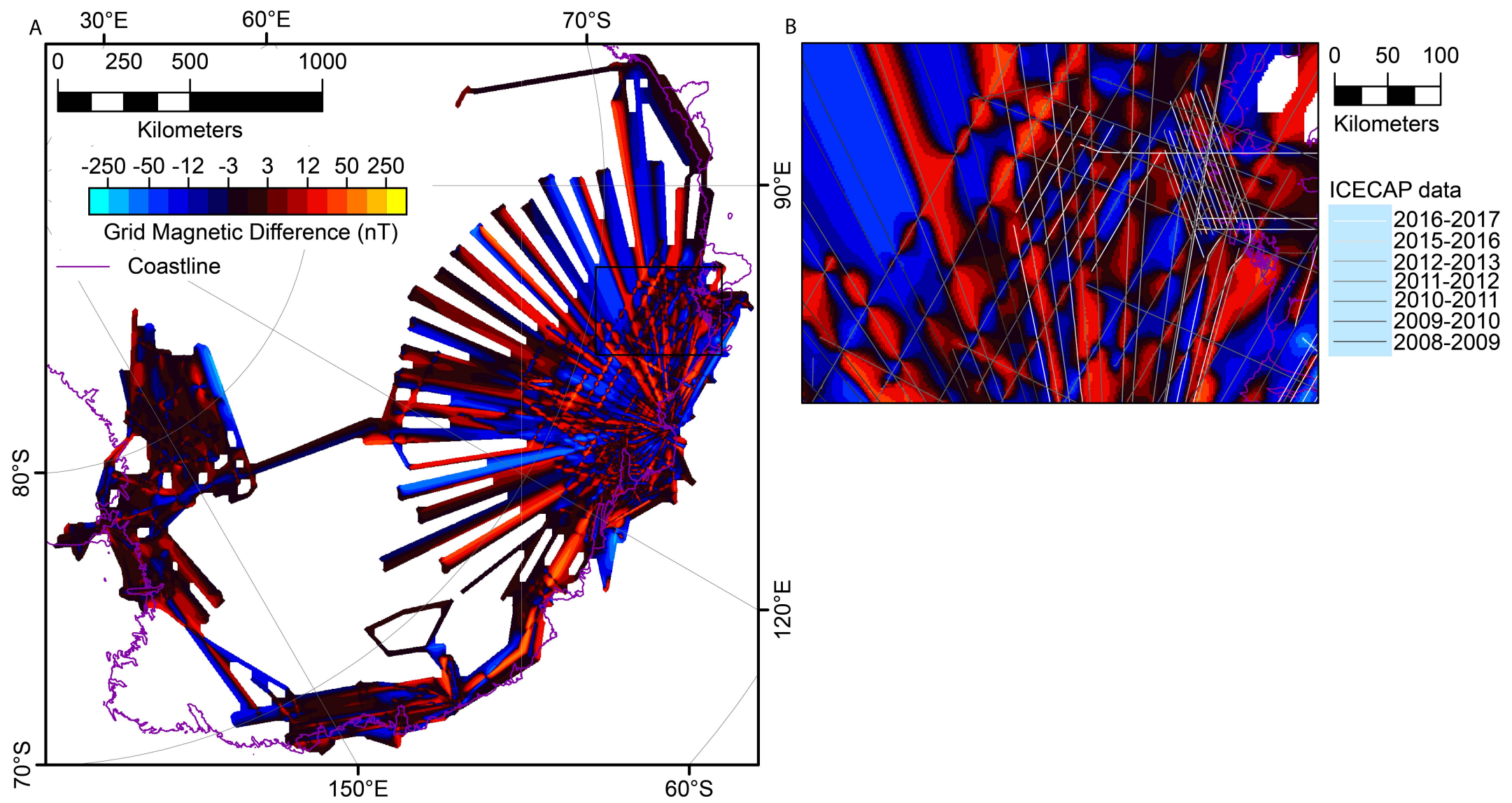
Supplementary Figure 14: Effect of spline levelling on corrected TMI (difference of Supp. Fig. 6 and Supp. Fig. 5)
A - over entire dataset, B - in selected region (same colour scale as A)



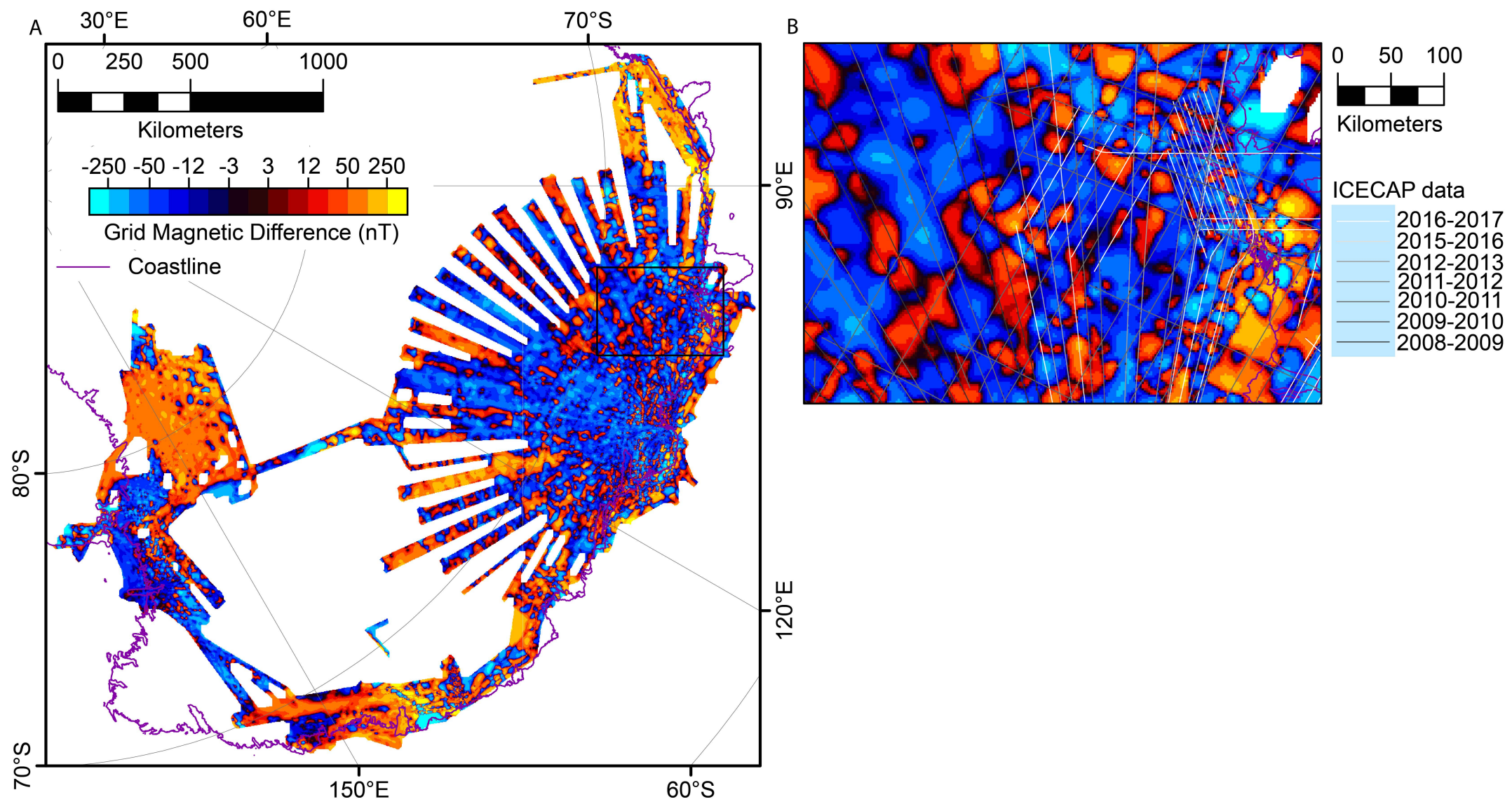
Supplementary Figure 15: Effect of elevation adjustment to 2000m (difference of Supp. Fig. 7 and Supp. Fig. 4)
A - over entire dataset, B - in selected region (same colour scale as A)



Supplementary Figure 16: Effect of median levelling on the elevation adjusted data (difference of Supp. Fig. 8 and Supp. Fig. 7)
A - over entire dataset, B - in selected region (same colour scale as A)



Supplementary Figure 17: Effect of spline levelling on the elevation adjusted data (difference of Supp. Fig. 9 and Supp. Fig. 8)
A - over entire dataset, B - in selected region (same colour scale as A)



Supplementary Figure 18: Difference between corrected TMI and ADMAP-2 grids
A - over entire dataset, B - in selected region (same colour scale as A)