Erklaerung

Moritz Mennenga

10. Juni 2017

analyseMagntic Erklärung

In dieser Vignette soll erklärt werden, welche Schritte bei der Funktion analyseMagnetic durchgeführt werden.

Vorbereiten der Daten

Diese Schritte könnten in R implementiert/durchgeführt werden

Die Geomagnetikdaten wurden aus DLMGPS als .asc Datei exportiert (X Y und nT; inkl. Spurkompensation).

Die folgenden Schritte wurden in SAGA durchgeführt, um die Daten in ein Rasterformat zu überführen. In Qgis ist dies auch möglich, SAGA läuft gerade bei großen Magnetikbilder bzw. großen Datenmengen stabiler und schneller.

- Import/Export -> Tables -> Import Text Table (header=false, separator = tab)
- Shapes -> Points -> Convert Tables to Points (x=F1;y=F2;z=F3)
- Grid -> Spline Interpolation -> Multilevel B-Spline Interpolation (Cell = 0.15)
- Rechtsklick auf erstelltes Raster -> Save As -> *.sgrd
- (Bei UTM) Die .*sgrd mit einem Texteditor öffnen und die führende 32 bei POSITION_XMIN löschen

Nun müssen die Anomalien extrahiert werden, was in QGIS geschehen ist.

- Die erstellte .sdat Datei in QGIS öffnen
- Toolbox -> SAGA Raster Calculator -> a > 2 (es wird ein Raster erstellt, in dem zwischen den Bereichen größer und kleiner 2 getrennt wird - 2nT ist dabei ein Beispielwert)
- Dann Raster -> Conversions -> Vectorize
- Aus dem nun erstellten Vectorlayer können die Bereiche zwischen den Anomalien gelöscht werden
- *Prüfen* Die erste Datenprüfung erfolgt. Der Vectorlayer muss geprüft werden, da es sein kann, dass zwei Anomalien als eine interpretiert wurden. Ggf. Advanced Digitizing Toolbar -> Split Features
- Dann Toolbox -> Polygon centroids

Nun sollten alle Anomalien einen Punkt haben. Diesen Layer als Shapedatei speichern.

Einladen der Daten

Die Daten sind vorbereitet vorhanden. Es handelt sich um den Ausschnitt aus einem Magnetikbild; eine SAGA .sdat Datei, um die Messwerte nutzen zu können. Des weiteren ist eine Shapedatei vorhanden, die für jede Anomalie über 2nT einen Punkt enthält. Die Dateien können mit:

anomalien <- readOGR(“data\example_points.shp”, layer=“example_points”)

bzw:

magnetik <- raster("data\Example_Clip.sdat")

eingeladen werden.

Dazu sind folgende Pakete nötig:

```
library(rgdal)
```

```
## Loading required package: sp
```

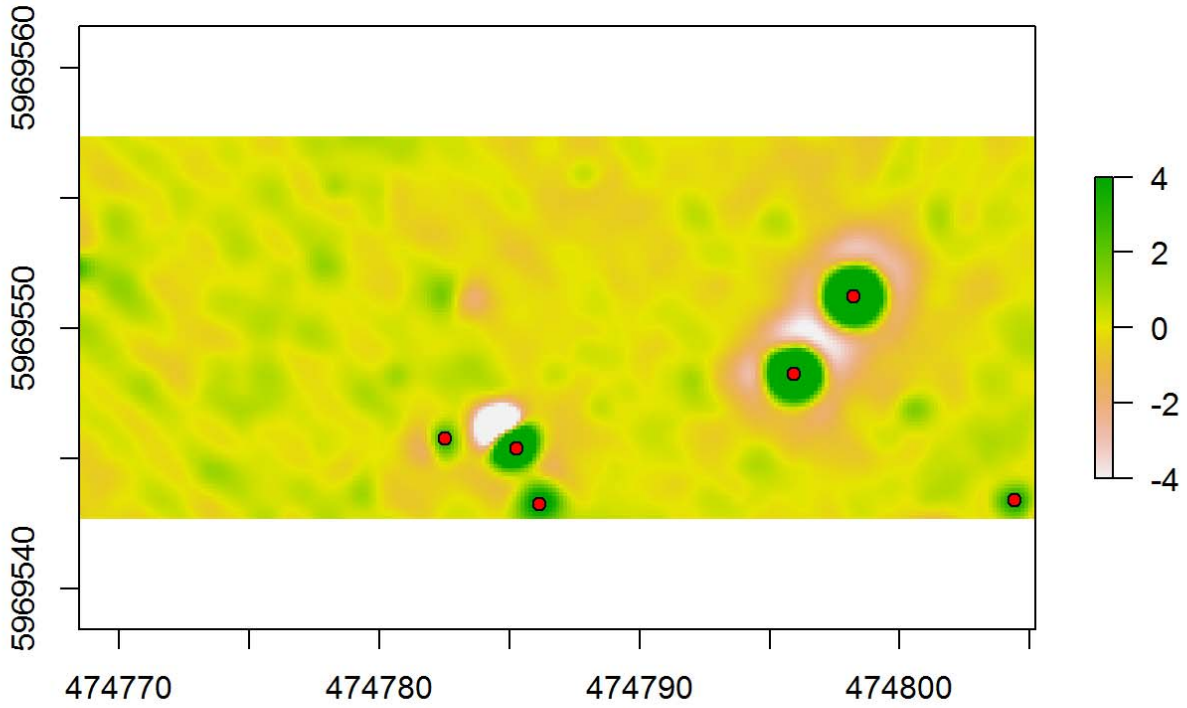
```
## rgdal: version: 1.2-5, (SVN revision 648)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
##  Loaded GDAL runtime: GDAL 2.0.1, released 2015/09/15
##  Path to GDAL shared files: C:/R/R-3.3.2/library/rgdal/gdal
##  Loaded PROJ.4 runtime: Rel. 4.9.2, 08 September 2015, [PJ_VERSION: 492]
##  Path to PROJ.4 shared files: C:/R/R-3.3.2/library/rgdal/proj
##  Linking to sp version: 1.2-4
```

```
library(raster)
#or
devtools::load_all() #< In you case library(magAAR)
```

```
## Loading magAAR
```

Hier das Magnetikbild und die Anomalien:

```
magnetik_disp <- magnetik
magnetik_disp[magnetik_disp > 4] <- 4
magnetik_disp[magnetik_disp < -4] <- -4
plot(magnetik_disp)
points(anomalien, pch = 21, bg = "red", col="black")
```



Markieren der Dipole

Da aktuell alle positiven Anomalien über einem bestimmten Grenzwert mit einem Punkt versehen sind, müssen nun zunächst die Dipole rausgerechnet werden, da es sich bei diesen zum Großteil um modernen Schrott handelt.

Dazu sind einige Parameter nötig, diese wedren im Laufe der Vignette erklärt. Zunächst nur, **get_dipole** = TRUE gibt an, dass Dipole ermittelt werden sollen.

```
get_dipol = TRUE
angle_steps = 10
searchradius = 2.5
dipolfactor = 2
dipol_minima = -8
```

Als nächstes wird der **search_radius** zum Durchmesser, da dieser im Script weiterhin so Verwendung findet. Außerdem heißt der SpatialDataFrame in der Funktion **anomalies_sdf**, in diesem Beispiel soll nur eine Anomalie verwendet werden. Daher sind einige Schleifen usw. nicht und einige Zuweisungen vorhanden, die daher nicht sinnvoll erscheinen. Da die Dipole exportiert werden sollen, wird eine Spalte angehängt, in die das Ergebnis kommt.

```
anomalies_sdf<- anomalien[anomalien@data$id==17, ]

#searcradius was diameter first
searchradius <- searchradius * 2

#empty column for result of dipol test
if(is.null(anomalies_sdf@data$dipol_kB) && get_dipol == TRUE){
 anomalies_sdf@data$di_kB <- 0
}
```

Für jede Anomalie soll eine vordefinierte Menge an Profilen erstellt werden, aus denen die Daten für den Test auf Dipole gewonnen werden.

Dazu muss zunächst ermittelt werden, wie viele Anomalien vorhanden sind und wie die Auflösung des Rasters ist (die Raster haben die gleiche x und y Auflösung)

```
#total amount anomalies
n_anomalies <- length(anomalies_sdf)
#resolution of inputraster = the distance between the points in the profile
magnetic_raster <- magnetik
x_resolution <- xres(magnetic_raster)
print(n_anomalies)
```

```
## [1] 1
```

```
print(x_resolution)
```

```
## [1] 0.15
```

Jetzt wird eine Schleife durchlaufen, die jeden Anomaliepunkt betrachtet. Die Zählvariable ist k. Die Schleife wird im Beispiel weggelassen.

Da die Rechenschritte recht langsam sind, wird ein Status an den Benutzer ausgegeben. Dann werden die Koordinaten der ersten Anomalie in einen dataframe geschrieben, mit dem weitergearbeitet wird Aus der Auflösung und dem **search_radius** lässt sich nun feststellen, wie viele Punkte das Profil haben muss. Damit der mittlere Punkt auch mittig ist, wird bei einer geraden Anzahl an Punkten einer hinzugefügt.

```
  k <- 1
  #Status
#while (k <= n_anomalies){
  if (k %% 10 == 0) {
    print(n_anomalies - k)
  }
  #coordinaten of the first anomaly
  coord <- data.frame(anomalies_sdf@coords[k,1], anomalies_sdf@coords[k,2])
  colnames(coord) <- c("x", "y")
  rownames(coord) <- NULL

  print(coord)
```

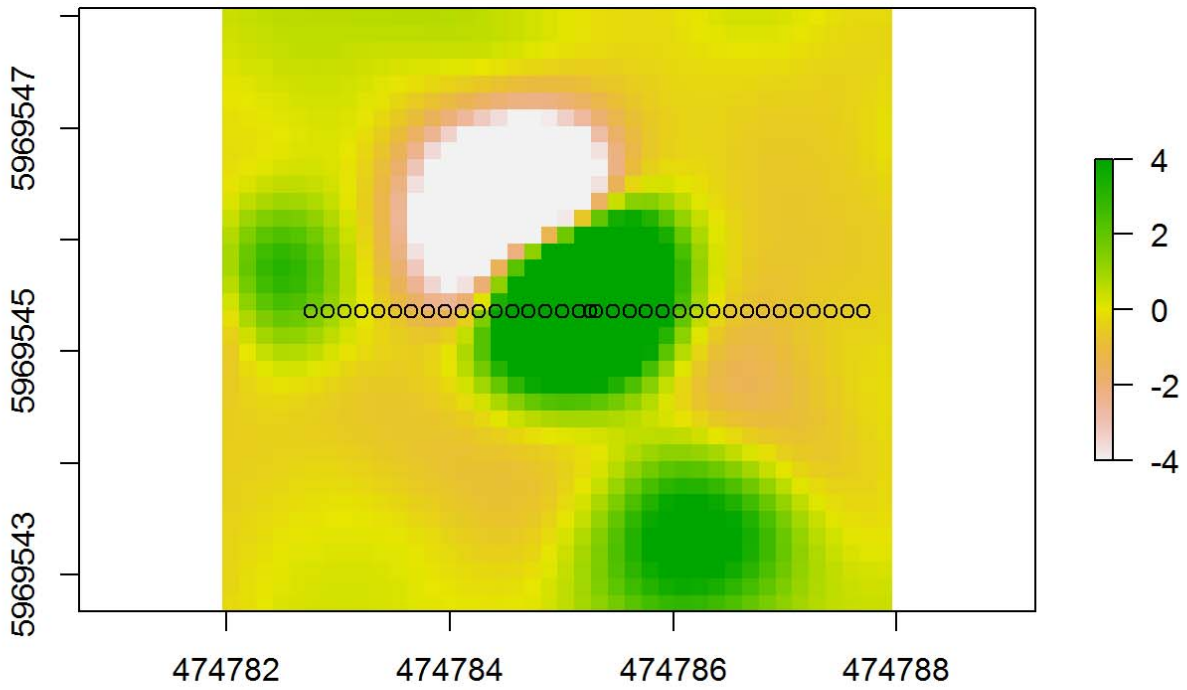
```
##          x       y
## 1 474785.3 5969545
```

```
    #The amount of points that are needed in a profile
    n_newpoints <- round(searchradius / x_resolution)
    #if the value is even, add one. therefore the profile is centric
    if(n_newpoints %% 2 == 0){
      n_newpoints <- n_newpoints + 1
    }

    print (n_newpoints)
```

```
## [1] 33
```

Dann wird der westlichste Punkt des Profiles - das erste Profil liegt immer Ost-West - ermittelt und von diesem immer ein Punkt (max. _newpoints) hinzugefügt, mit dem Abstand der Rasterauflösung.

```
    #For each point a coordinate should be added
    #Starting at the most left point of the profile
    startx <- coord$x - (searchradius / 2)
    #Generating the points
    i <- 2
    increase <- 0
    while (i <= n_newpoints + 2){
      coord[i,] <- c(startx + increase, coord$y[1])
      increase <- increase + x_resolution
      i <- i + 1
    }
    plot(magnetik_disp, xlim=c(474782,474788), ylim=c(5969542,5969548))
    points(coord$x,coord$y)
```

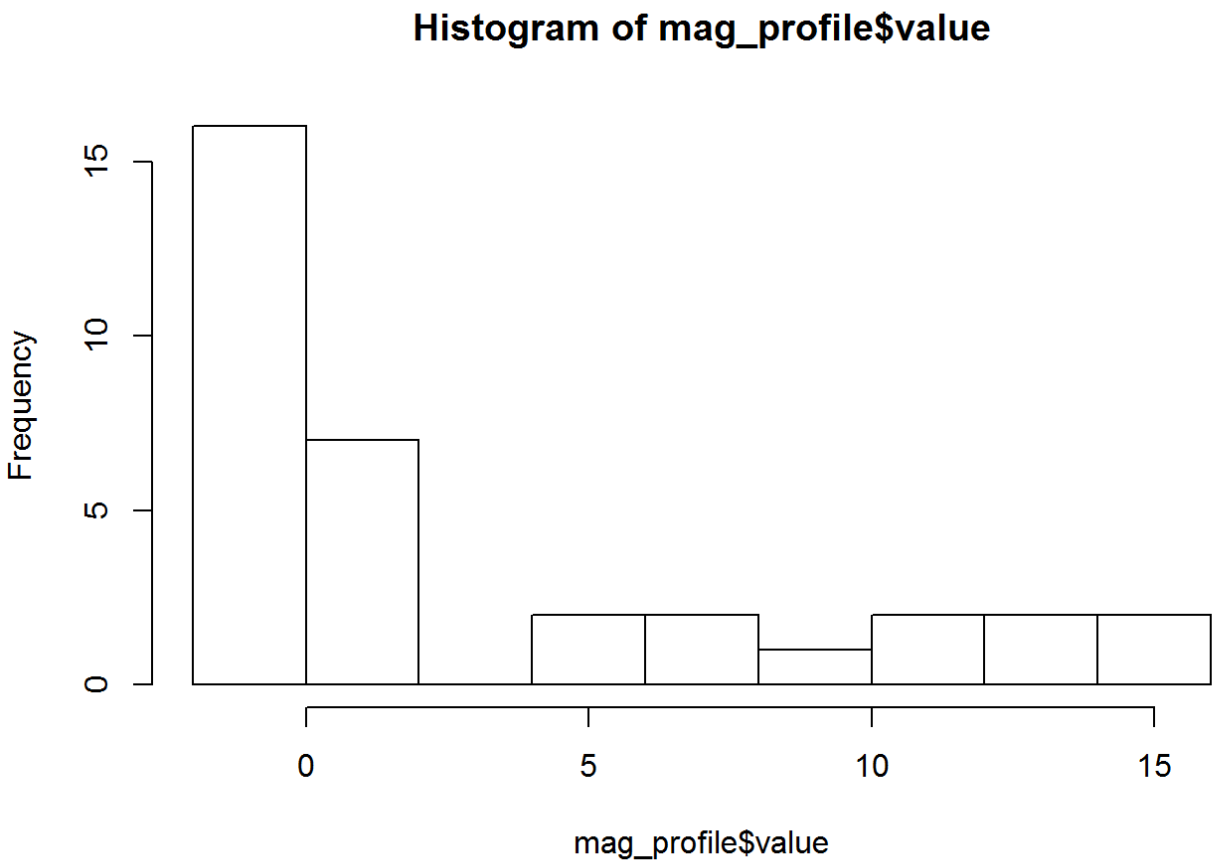


Nun werden die Werte ausgelesen und verglichen. Da ein Dipol nicht immer komplett von einem negativen Bereich umgeben ist, sondern dieser auch nur in eine Richtung liegen kann. Muss dieser Vergleich in mehreren Schritten erfolgen. Dazu werden die Werte aus dem eben angelegten Profil ausgelesen und das Profil dann in **angle_steps** Grad gedreht und dann wieder ausgelesen. Erfüllt eines der so ermittelten Profile die Bedingungen eines Dipols, wird die Anomalie markiert.

Für das erste Profil:

Zunächst werden also die Magnetikwerte an jedem Punkt des Profils ausgelesen. Danach wird der erste Eintrag des Profiles gelöscht, so ist das Profil gleichmäßig und die Wertereihenfolge korrekt.

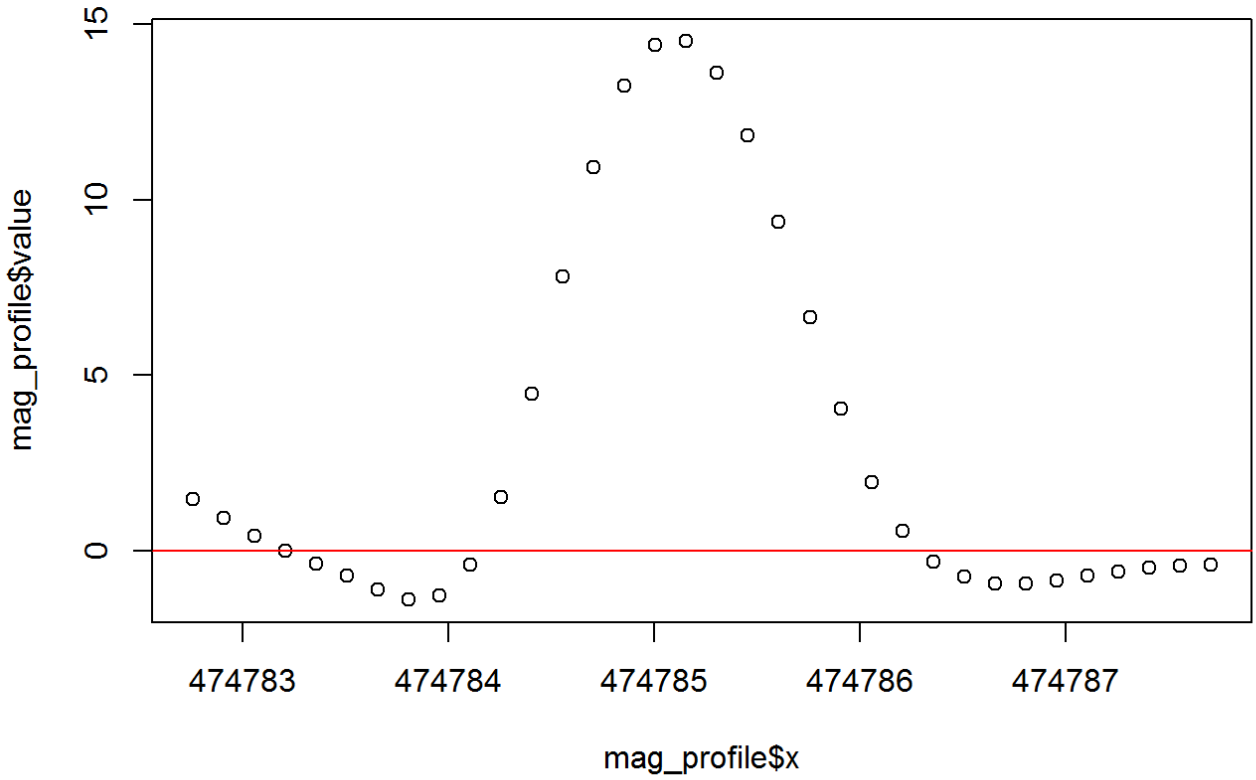
```
      coord_trans <- coord # Schritt wird später erklärt
      #Getting the value of the magnetic raster at everypoint of the profile
      mag_profile <- data.frame(coordinates(coord_trans), extract(magnetic_raster,
coord_trans))
      names(mag_profile) <- c("x", "y", "value")
      #First value can be deleted
      mag_profile <- mag_profile[-1, ]

      hist(mag_profile$value)
```



Nun zu den zwei wichtigsten Schritten bei der Erkennung eines Dipoles.

Die Amplitude einer Anomalie mit potenziell historischem anthropogenen Ursprung ist meist einer Normalverteilung sehr ähnlich, aber auch ein Dipol kann so aussehen, wenn das Profil nicht auf dem positiven und negativen Bereich liegt:

```
plot(mag_profile$x,mag_profile$value)
abline (h=0,col="red")
```



Nach Vergleichen mehrere Datensätze lässt sich ein Dipol natürlich an Hand des Kurvenverlaufes definieren, aber auch anhand des maximalen und minimalen Wertes. Dazu ist der **dipol_factor** nötig. Denn der Dipol ist definiert durch |niedrigster nT-Wert * **dipole_factor**| > maximaler nT-Wert. In diesem Fall also kein Dipol

eine andere Definition durch den Kurvenverlauf o.ä. wäre sicher auch sinnvoll

```
      #A dipol is defined by |minimal value * dipolfactor| > maximal value
      #na.omit, they can be generated be boundary effects
      if (get_dipol == TRUE && abs(min(na.omit(mag_profile$value))) * dipolfactor
> max(na.omit(mag_profile$value)) &&
          min(na.omit(mag_profile$value)) < 0)
      {
        #writing the result in spatial data frame
        anomalies_sdf@data$di_kB[k] <- 1
      }

print(anomalies_sdf@data)
```

```
##   id di_kB
## 4 17     0
```

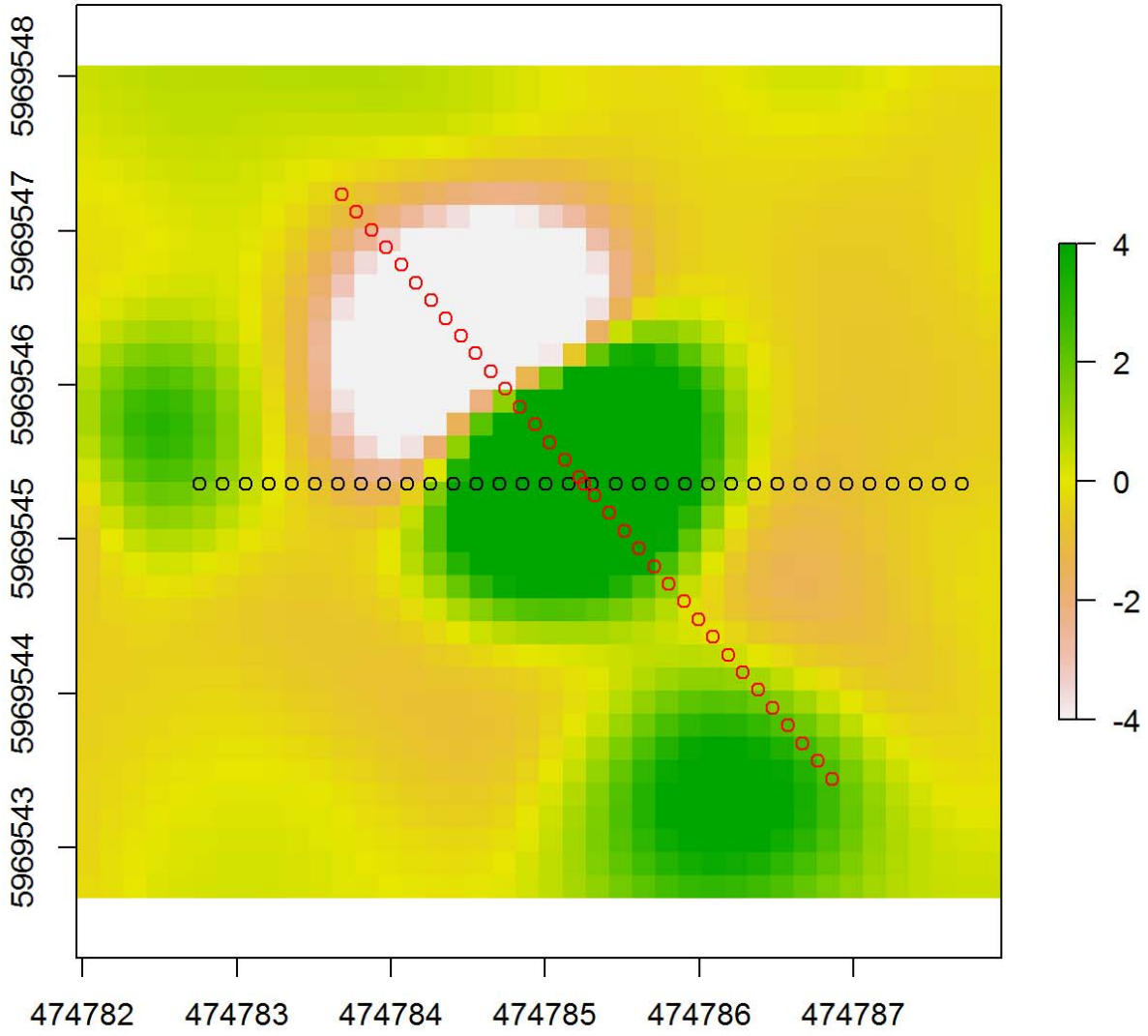
Der zweite Weg einer Definiton ist dadurch begründet, dass in einigen Fällen (dann wenn die minimalen und maximalen nT-Werte extrem waren) dieser Weg nicht funktionert hat. Daher ist festgelegt, dass Anomalien mit Werten unter einem bestimmten nT-Wert ebenfalls markiert werden.Der Grenzwert dafür ist **dipol_minima**.

```
      #The dipol can also be defined as a minimum < dipol_minima.
      if (get_dipol == TRUE && min(na.omit(mag_profile$value)) < dipol_minima){.
        #writing the result in spatial data frame
        anomalies_sdf@data$di_kB[k] <- 2
      }
print(anomalies_sdf@data)
```

```
##   id di_kB
## 4 17     0
```

Nun wird das Profil rotiert und für jeden **angle_step** dieses Vorgehen wiederholt. Degree ist dabei die Zählvariable, die sich bei jedem Durchlauf um **angle_step** erhöht.

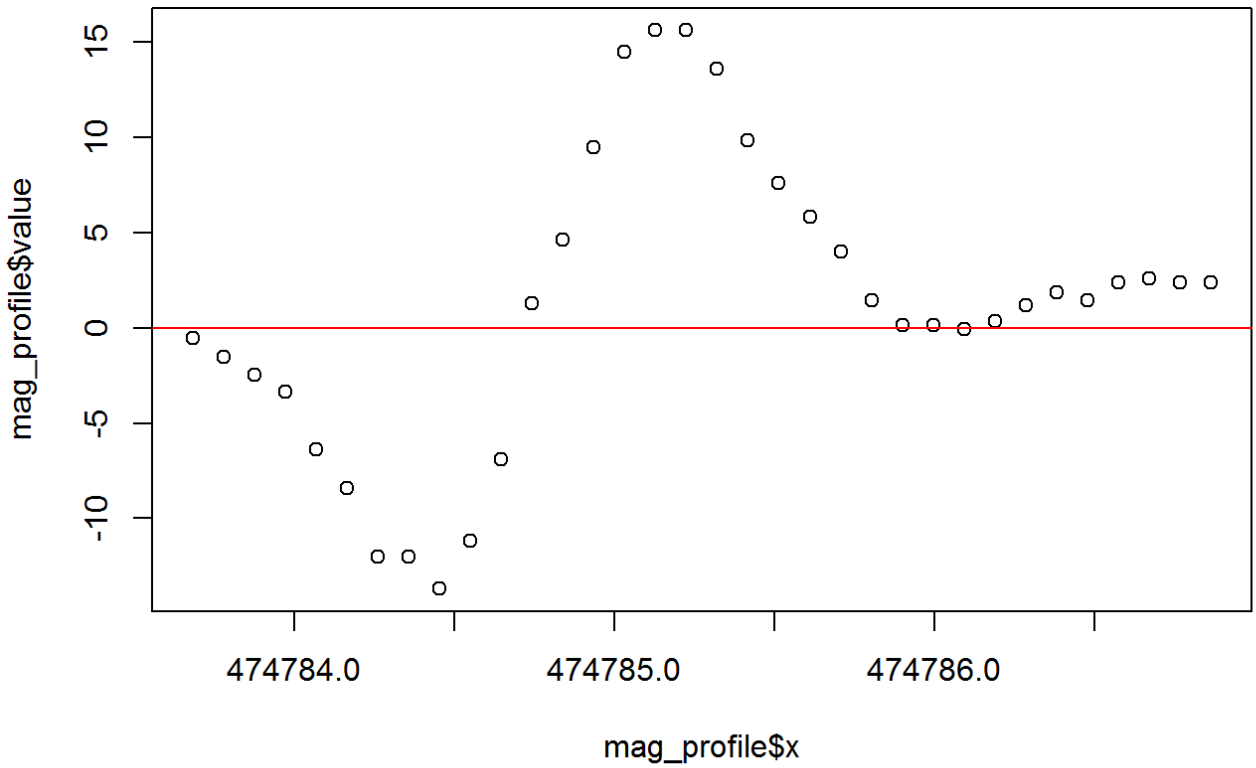
```
        #Starting with the degree 0
        #dregee <- 0
        #For every angle between 0 and 180
        #while (dregee < 180)
        #{
        #if it is the first run, no recalculating of the profile is needed
        #  if (dregee == 0){
        #    #coord_trans is needed for the calculations
        #    coord_trans <- coord

        #  } else {
        degree <- 130 #für das Beispiel, ansonsten Erhöhung fro Durchlauf
        #If the degree is > 0, the coodinates have to be rotated
        #Center is the middle point
        #This is the first value of the data frame
        #For the calculating the new coordinates use translation and rotation
        x_mittel <- coord$x[1]
        y_mittel <- coord$y[1]
        coord_trans <- coord
        for (z in 1:nrow(coord)){
          coord_trans[z,] <- c(
            x_mittel + (coord$x[z] - x_mittel) * cos(degree / 180 * pi) - sin(degr
ee / 180 * pi) * (coord$y[z] - y_mittel),
            y_mittel + (coord$x[z] - x_mittel) * sin(degree / 180 * pi) + (coord$y
[z] - y_mittel) * cos(degree / 180 * pi))
          #http://www.matheboard.de/archive/460078/thread.html
        }
        plot(magnetik_disp, xlim=c(474782,474788), ylim=c(5969542,5969548))
        points(coord$x,coord$y)
        points(coord_trans$x,coord_trans$y, col="red")
```



Nachdem das Profil gedreht wurde, muss der Test auf Dipol wiederholt werden.

Durch das gedrehte Profil wird nun der Kurvenverlauf eines typischen Dipols sichtbar. So wird dieser auch als solcher erkannt, da der Betrag des Minimalwertes x **dipolfactor** (in diesem Fall = 2) größer ist als der Maximalwert. Die Klassifizierung erfolgt in der Spalte *di_kB* (0=kein Dipol; 1=Dipol; 2=Minimum < Grenzwert).

```
      #Getting the value of the magnetic raster at everypoint of the profile
      mag_profile <- data.frame(coordinates(coord_trans), extract(magnetic_raster,
coord_trans))
      names(mag_profile) <- c("x", "y", "value")
      #First value can be deleted
      mag_profile <- mag_profile[-1, ]

      plot(mag_profile$x,mag_profile$value)
      abline (h=0,col="red")
```




```
            #A dipol is defined by |minimal value * dipolfactor| > maximal value
      #na.omit, they can be generated be boundary effects
      if (get_dipol == TRUE && abs(min(na.omit(mag_profile$value))) * dipolfactor
> max(na.omit(mag_profile$value)) &&
          min(na.omit(mag_profile$value)) < 0)
      {
        #writing the result in spatial data frame
        anomalies_sdf@data$di_kB[k] <- 1
      }

      print(anomalies_sdf@data)
```

```
##   id di_kB
## 4 17     1
```

Wird der Wert **dipole_minima** > dem Minimalwert gewählt, so wird die Anomalie ebenfalls klassifiziert

```
      #The dipol can also be defined as a minimum < dipol_minima
      if (get_dipol == TRUE && min(na.omit(mag_profile$value)) < dipol_minima){
        #writing the result in spatial data frame
        anomalies_sdf@data$di_kB[k] <- 2
      }
print(anomalies_sdf@data)
```

```
##   id di_kB
## 4 17     2
```

Nun bietet es sich an die Datei zu exportieren und im GIS zu prüfen, ob alle richtig klassifiziert worden sind.

writeOGR(obj=anomalien, dsn="export", layer="anomalien", driver="ESRI Shapefile")

Ermitteln der Breite und Höhe der Amplitude

Um eine Gruppierung der Anomalien vorzunehmen, werden Werte benötigt, die diese beschreiben. Grundsätzlich sind sich die Kurven in der Form sehr ähnlich und unterscheiden sich vor allem durch ihre Höhe (max. nT) und die Breite. Daher werden diese Werte ermittelt. Um die Breiten und Höhen miteinander vergleichen zu können ist ein Wert nötig (**cut_value**) der als Grundlage für die Ermittlung der Werte genommen wird. In diesem Beispiel 5nT, d.h., dass auf der Höhe von 5 nT die Werte ermitteln werden.

Zunächst definieren wir die Parameter, die nötig sind. Hier wird nur die **method** "avg" vorgestellt. Dabei wird die Breite jedes Profils um eine Anomalie berechnet und der Durchschnitt aller genommen. Bei "median" wird der median verwendet.

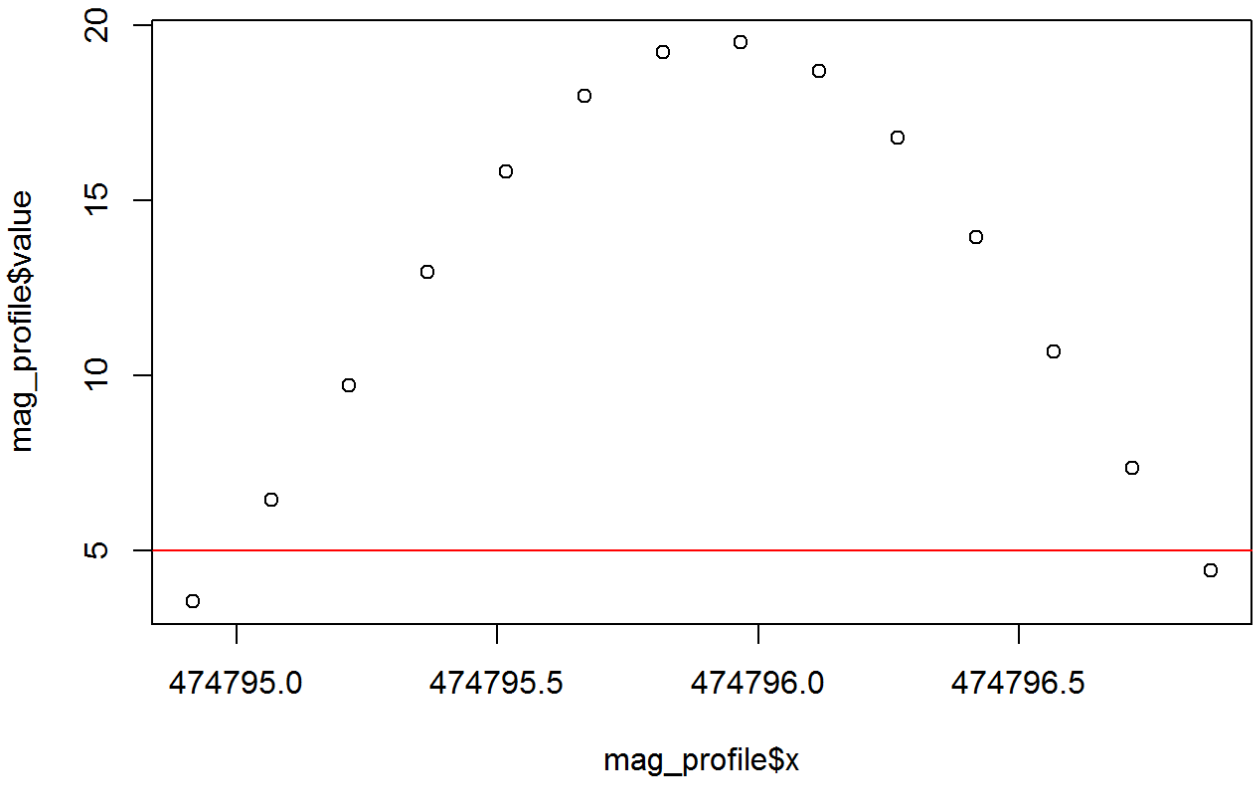
```
anomalies_sdf<- anomalien[anomalien@data$id==13, ]
magnetic_raster <- magnetik
searchradius=2
angle_steps=10
get_profile_values=TRUE
method="avg"
cut_value=5
```

Koordinaten müssen bestimmt werden und die Magnetikwerte ausgelesen

```
        k <- 1

    #while (k <= n_anomalies){
    if (k %% 10 == 0) {
      print(n_anomalies - k)
    }
    #coordinaten of the first anomaly
    coord <- data.frame(anomalies_sdf@coords[k,1], anomalies_sdf@coords[k,2])
    colnames(coord) <- c("x", "y")
    rownames(coord) <- NULL

    print(coord)
```

```
##          x       y
## 1 474795.9 5969548
```

```
    #The amount of points that are needed in a profile
    n_newpoints <- round(searchradius / x_resolution)
    #if the value is even, add one. therefore the profile is centric
    if(n_newpoints %% 2 == 0){
      n_newpoints <- n_newpoints + 1
    }

    #For each point a coordinate should be added
    #Starting at the most left point of the profile
    startx <- coord$x - (searchradius / 2)
    #Generating the points
    i <- 2
    increase <- 0
    while (i <= n_newpoints + 2){
      coord[i,] <- c(startx + increase, coord$y[1])
      increase <- increase + x_resolution
      i <- i + 1
    }
    coord_trans <- coord
    #Getting the value of the magnetic raster at everypoint of the profile
    mag_profile <- data.frame(coordinates(coord_trans), extract(magnetic_raster, c
oord_trans))
    names(mag_profile) <- c("x", "y", "value")
    #First value can be deleted
    mag_profile <- mag_profile[-1, ]
      plot(mag_profile$x,mag_profile$value)
      abline (h=cut_value,col="red")
```



Die Daten dürfen keine NA Werte haben. Dies kann passieren, wenn eine Anomalie am Rand des Bildes liegt.

```
        mag_profile <- na.omit(mag_profile)
        #Getting the highest value (mor eor less the middle of the profile) and th
e starting point
        x_row <- which(mag_profile$value == max(na.omit(mag_profile$value)))[1]
        #result buffer
        distance_right <- 0
        distance_left <- 0
        #Going step by step from the middle point to the right until the nT value
is lower the cutting value
        w <- x_row
        #Stopp if value has reached
        stopp <- FALSE
```

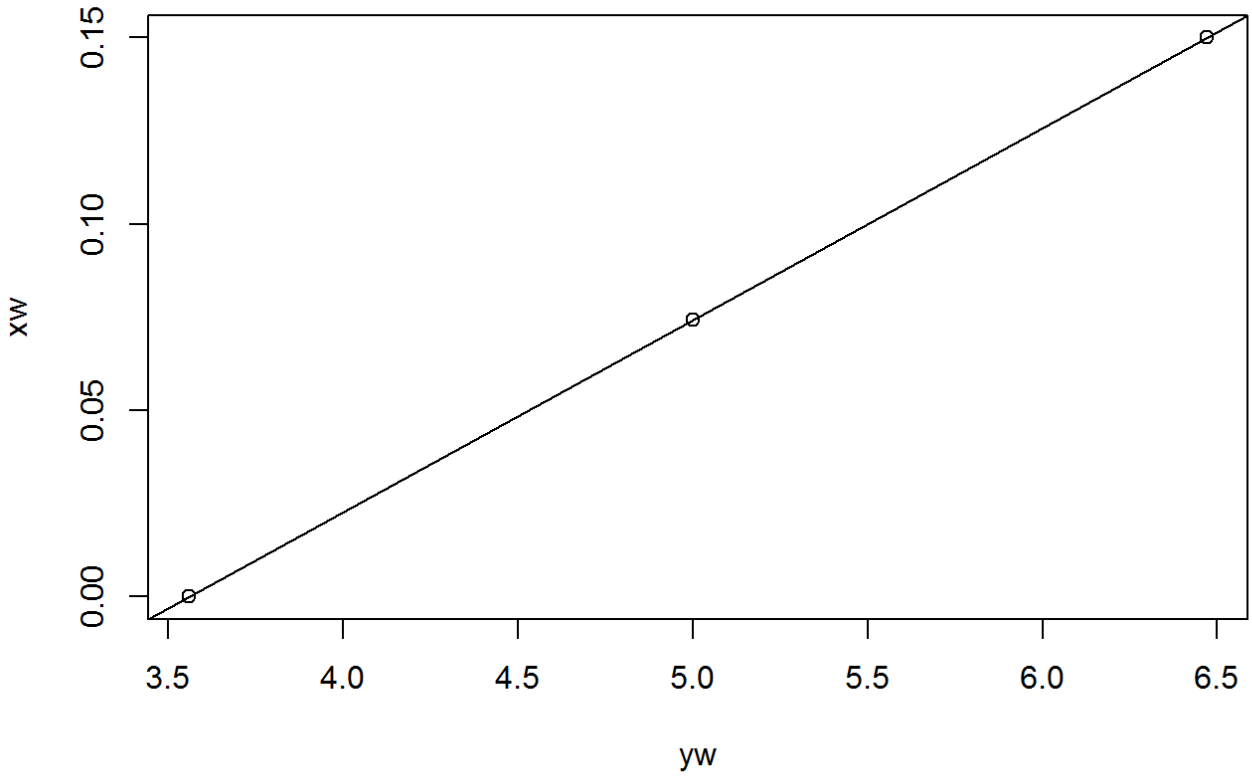
Nun wird die Breite ermittelt, dazu wird der mittlere Punkt des Profils gewählt und von dort das Profil nach rechts und links durchlaufen. Ist der Wert eines Punktes niedriger als **cut_value**, so stoppt der Durchlauf und von diesem Punkt wird weitergerechnet. Um den Abstand zwischen der Kurve zu ermitteln werden folgende Schritte durchgeführt:

- Der Abstand zwischen den beiden Punkten wird ermittelt
- Dann wird ein dataframe erstellt, der besteht aus dem Punkt A(0,nT-Wert des Punktes oberhalb des cut_values) und Punkt B(Distanz zwischen den Punkten ,nT-Wert des Punktes unterhalb des cut_values)
- Diese Werte werden genommen um ein Modell mit einer linearen Regression zu rechnen (*an dieser Stelle könnte durch Modelle, die die ganze Kurve beschreiben mögl. beschleunigt und die Genauigkeit erhöht werden*).
- Nun wird der Abstand zwischen den Punkten vorhergesagt, der am cut_value liegt.
- Die Distanz berechnet sich dann je für links und rechts des Scheitelpunktes anhand des vorhergesagten Abstandes und der Koordinaten
- Der Abstand unter der Kurve ist dann der Abstand der Werte

```
        while(w <= nrow(mag_profile) && stopp == FALSE){
          #HIER PRÜFEN OB DER ERSTE WERT UNTER DEM CUT VALUE LIEGT, WENN JA DANN D
ISTANCE = 0
          #If the cutting value has reached, claculate the nT value at the cutting
value
          if(mag_profile$value[w] < cut_value){
            #getting the distance between the highest point and the value before t
he cutting value had reached
            x1 <- mag_profile$x[w-1]
            y1 <- mag_profile$y[w-1]
            x2 <- mag_profile$x[w]
            y2 <- mag_profile$y[w]
            #getting the slope
            m <- (y2-y1) / (x2-x1)
            #calculating the distance between the two points
            distance <- sqrt(  ((x2 - x1) ^2) + ((y2 - y1) ^2))
            #generate dataframe with values
            #xvalue is the distance between the two points
            xw <- c(0,distance)
            #yvalue is the nT magnetic value at this points
            yw <- c(mag_profile$value[w-1], mag_profile$value[w])
            #to get the magnetic value at the cutting point, a linear regression i
s used
            fm <- lm(xw ~ yw)
            #predict the xvalue(distance) for the yvalue(cutting value)
            a <- predict(fm, data.frame(yw = c(cut_value)), se.fit = TRUE)$fit
            #The distance from the middle point to the cutting value is the distan
ce to the point before
            #reaching the cutting value and the predicted distance
            distance_right <- (w - x_row - 1) * distance + a
            stopp <- TRUE
          }
          w <- w + 1
        }

        #Same will be done for the left site of the graph
        w <- x_row
        stopp <- FALSE

        while(w > 0 && stopp == FALSE){
          if(mag_profile$value[w] < cut_value){
            #HIER PRÜFEN OB DER ERSTE WERT UNTER DEM CUT VALUE LIEGT, WENN JA DANN
DISTANCE = 0
            print(w)
            x1 <- mag_profile$x[w+1]
            y1 <- mag_profile$y[w+1]
            x2 <- mag_profile$x[w]
            y2 <- mag_profile$y[w]
            m <- (y2 - y1) / (x2 - x1)
            stopp <- TRUE
             distance <- sqrt(((x2 - x1) ^2) + ((y2 - y1) ^2))
             xw <- c(0, distance)
             yw <- c(mag_profile$value[w], mag_profile$value[w+1])
             fm <- lm(xw ~ yw)
```

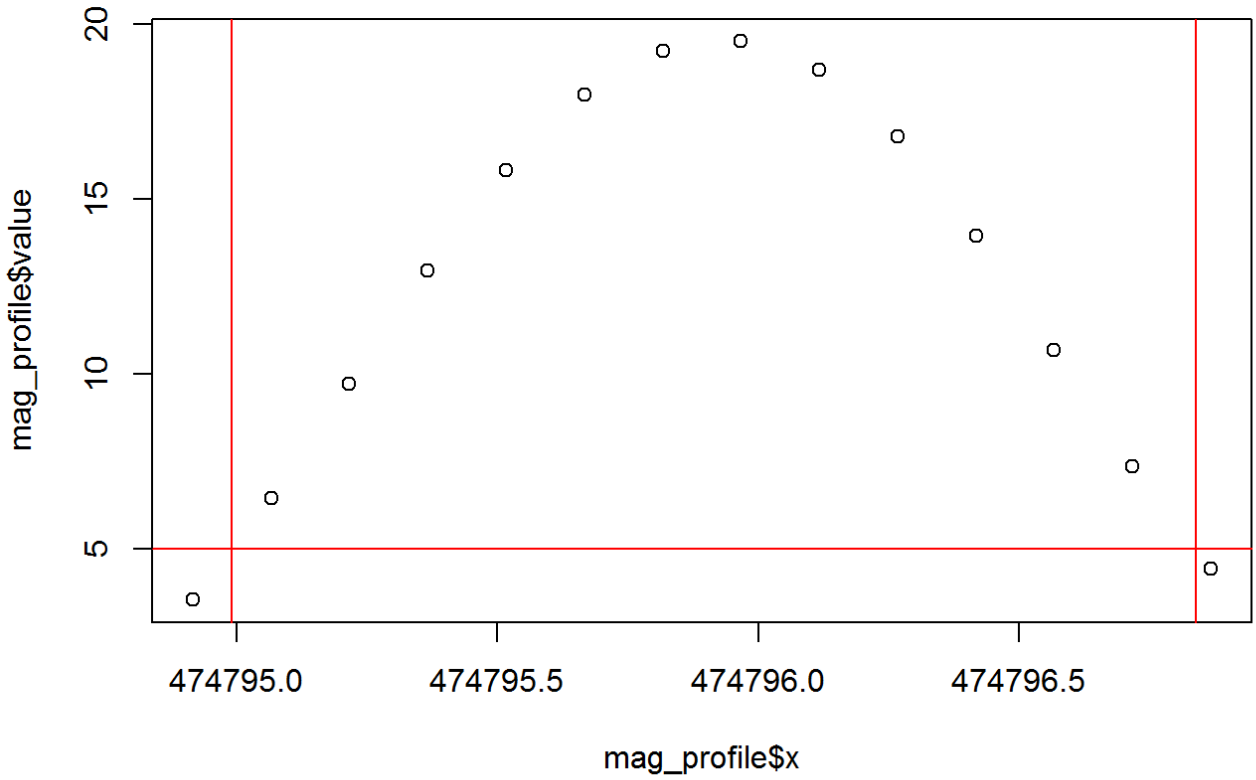
```
            a <- predict(fm, data.frame(yw = c(cut_value)), se.fit = TRUE)$fit
            plot(yw,xw)
            abline(fm)
            points(cut_value,a)
            distance_left <- (x_row - w) * distance - a

          }
          w <- w - 1
        }
```

```
## [1] 1
```



Hier der Scheitelpunkt - bzw. + der Abstände (rot)

```
        plot(mag_profile$x,mag_profile$value)
        abline (v=mag_profile$x[8]-distance_left,col="red")
        abline (v=mag_profile$x[8]+distance_right,col="red")
        abline (h=5, col="red")
```



Jetzt muss noch die Höhe ausgelesen werden, dabei sollte es sich um den höchsten Wert des Profils handeln *zu ergänzen: möglicherweise testen, ob höchster Wert in der Mitte, da sonst Nachbaranomalie der Grund sein könnte*

```
        anomalies_sdf@data$an_breite[k] <- distance_left + distance_right

          #The heigth (magnetic value) is alle the time the highest value

          anomalies_sdf@data$an_hoehe[k] <- max(na.omit(mag_profile$value))-cut_v
alue
```

Das Ergebnis ist nun ein SpatialDataFrame mit den Breiten und Höhen der Aplitude

```
 anomalies_sdf@data
```

```
##   id an_breite an_hoehe
## 2 13  1.847664 14.50933
```

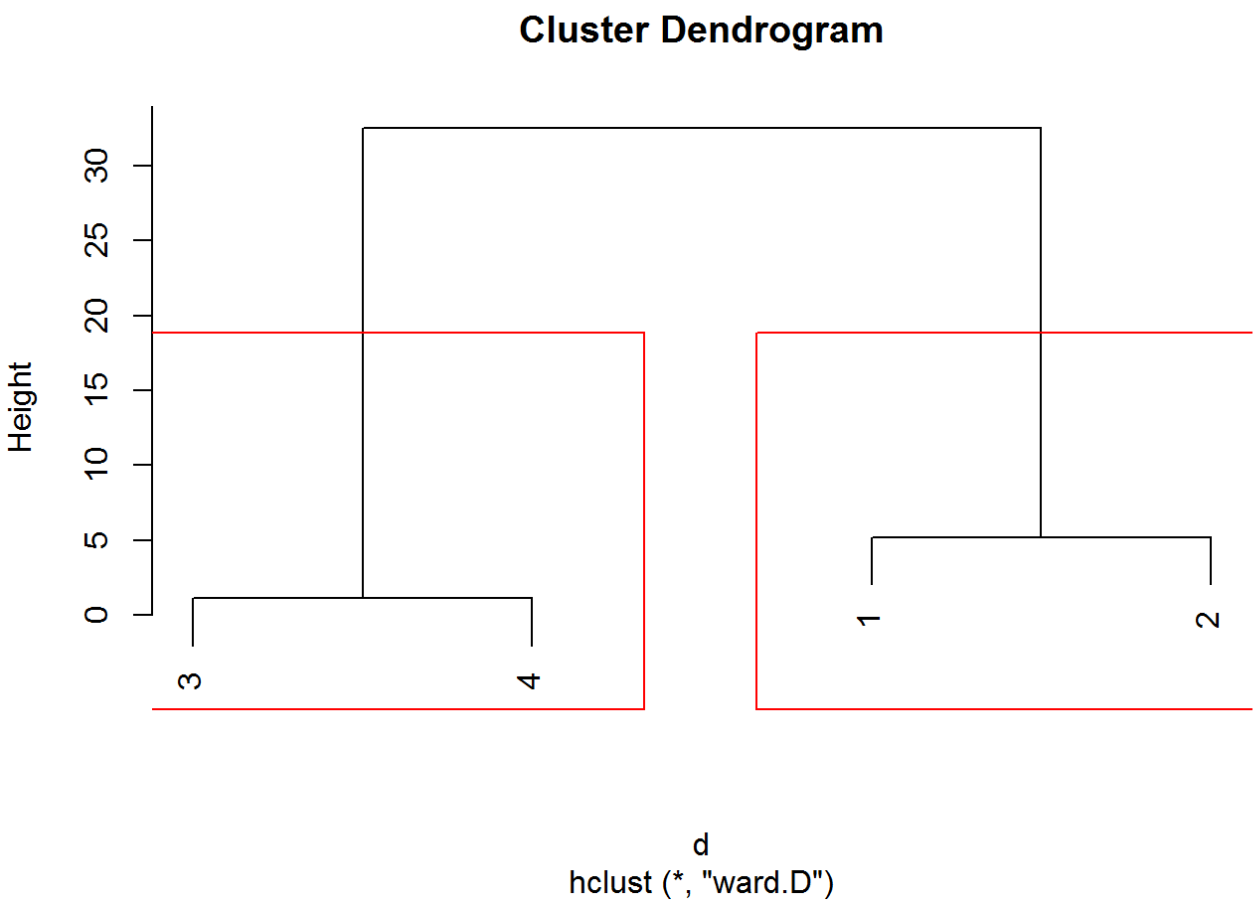
##Nachbereitung Beispiel

Im weiteren Auswerten kann nun mit diesen Daten gearbeitet werden. z.B. mit Hilfe einer Clusteranalyse gruppen gebildet werden. Auch wenn die Daten dafür nicht optimal sind, da zu wenig Anomalien, hier ein Beispiel, ohne weitere Kommentare

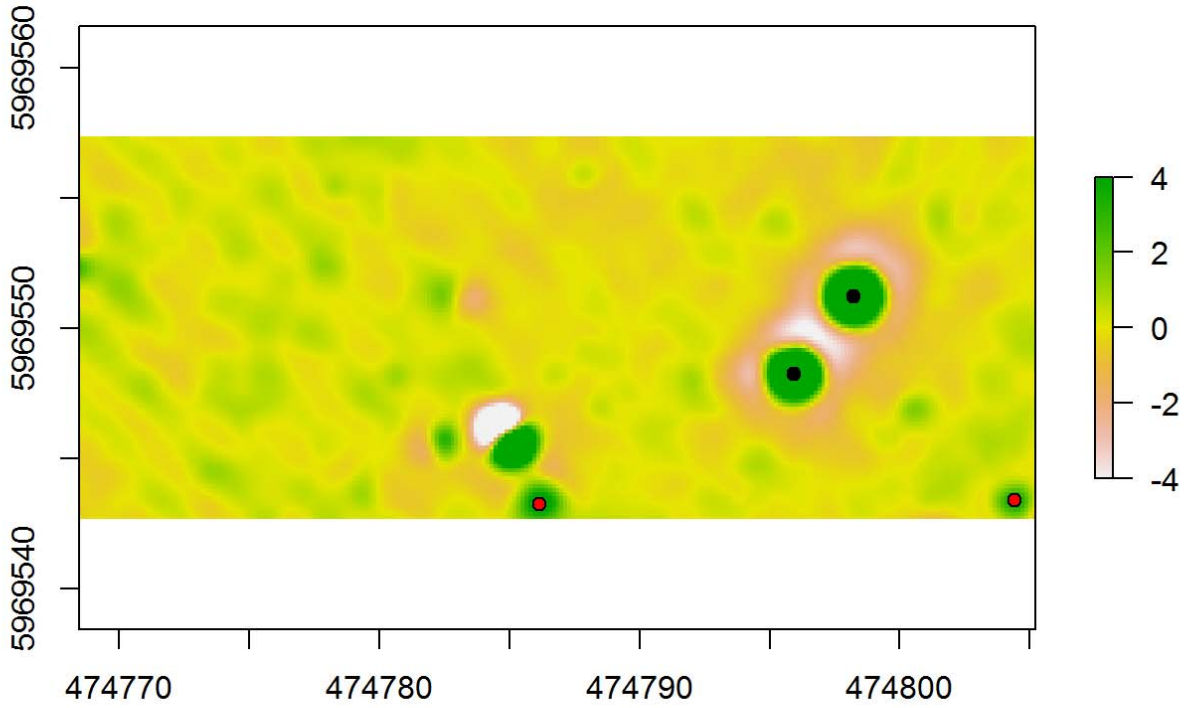
```
anomalie_dp <- analyseMagnetic(anomalies_sdf = anomalien, magnetic_raster = magnet
ik, searchradius = 2.5, angle_steps=10,get_profile_values=FALSE,get_dipol = TRUE,d
ipolfactor = 2,dipol_minima = -8)

 anomalie_db <- anomalie_dp[anomalie_dp@data$di_kB == 0,]
 anomalie_data <- analyseMagnetic(anomalies_sdf = anomalie_db, magnetic_raster = m
agnetik, searchradius = 2.5, cut_value=2,method="avg", angle_steps=10,get_profile_
values=TRUE,get_dipol = FALSE)
 d <- dist(data.frame(anomalie_data@data$an_breite,anomalie_data@data$an_hoehe), m
ethod = "euclidean")
 fit <- hclust(d, method="ward")
```

```
## The "ward" method has been renamed to "ward.D"; note new "ward.D2"
```

```
 plot(fit) # display dendogram
 rect.hclust(fit, k=2, border="red")
```




```
 groups <- cutree(fit, k=2) # cut tree into 5 clusters
 anomalie_data@data$groups <- groups

 plot(magnetik_disp)
 points(anomalie_data[anomalie_data@data$groups==1,], pch = 21, bg = "black", col=
"black")
 points(anomalie_data[anomalie_data@data$groups==2,], pch = 21, bg = "red", col="b
lack")
```


5969560
5969550
5969540
474770
474780
474790
474800
4
2
0
-2
-4