Supplementary Information: A calculator for peatland volume and carbon stock to support area planners and decision makers (Kyrkjeeide et al.)

Anders L. Kolstad, Marte Fandrem

2023-02-13

Contents

Chapter 1

Analyses of peatland carbon stocks

Here I document the data and analyses behind the manuscript *A calculator for peatland volume and carbon stock to support area planners and decision makers.* (Fandrem et al. in prep). The documentation exists both as a dynamic web page, and as a static pdf that came with the original publication.

This web page or pdf was build 2023-02-13 14:52:59

The analyses here is done by Kolstad based on inital work of Fandrem.

Figure S1.1: The analyses described in this book makes it possible, with minimal effort, to estimate the carbon content of peatlands such as this one. Photo: Anders L. Kolstad.

Chapter 2

Test data

2.1 Peat depth and peatland delineation

To create the methodology for mapping peatland depth profiles, and estimating peat volume, carbon stocks and sensitivity to sampling effort, I will use two contrasting test sites: *Tydal* and *Geilo.* Later, in chapter 7 I will validate the generality of this method on an additional four sites. For more details on the justification and the methods used, please the the manuscript.

Libraries used:

```
library(tmap)
library(sf)
library(readr)
library(tmaptools)
library(basemaps)
library(ggplot2)
library(ggpubr)
library(gstat)
library(matrixStats)
library(ggtext)
library(tidyverse)
library(osmplotr)
```

Import shape files with peatland delineations and fix the CRS.

```
SHP_tydal <- sf::read_sf("Data/Tydal/stasjon_Setermyra.shp")
SHP_geilo <- sf::read_sf("Data/Geilo/geilo-dybdef.shp")

# st_crs(SHP_tydal) st_crs(SHP_geilo)# NA I found
# CRS through trial and error. It is UTM 32
st_crs(SHP_geilo) <- 25832

# Transform to UTM33N
SHP_geilo <- st_transform(SHP_geilo, 25833)
```

Import peat depth measurements for the two sites.

```
depths_tydal <- readr::read_csv("Data/Tydal/Torvdybder_Tydal_stasjon.csv")
depths_geilo <- read.csv("Data/Geilo/torvdybder.csv",
    sep = ";")
```

```
# convert data.frames to simple features.
depths_tydal <- sf::st_as_sf(x = depths_tydal, coords = c("coords.x1",
    "coords.x2"), crs = "+init=epsg:25833")
depths_geilo <- st_as_sf(x = depths_geilo, coords = c("x",
    "y"), crs = "+init=epsg:25832")

# Tranbsform to UTM33, same as the shape file
depths_geilo <- st_transform(depths_geilo, 25833)
```

```
# Confirm overlap. Using base plotting to avoid
# auto-transformation
plot(SHP_geilo$geometry, col = "red")
plot(depths_geilo$geometry, pch = 16, col = "grey",
    cex = 0.1, add = T)
```

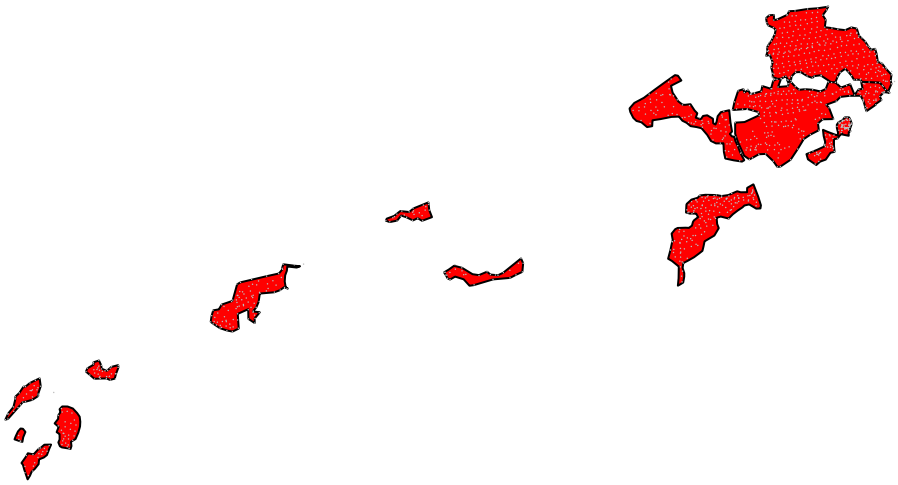


Figure S2.1: Confirming overlap between shape file and depth measurements

That looks fine.

Download basemaps for some context.

```
# osmplotr want bboxes in latlong
SHP_geilo_ll <- sf::st_transform(SHP_geilo, 4326)
SHP_tydal_ll <- sf::st_transform(SHP_tydal, 4326)

bb_Geilo <- sf::st_bbox(SHP_geilo_ll)
```

```
bb_Tydal <- sf::st_bbox(SHP_tydal_ll)

# GEILO
base_geilo_hw <- osmplotr::extract_osm_objects(bbox = bb_Geilo,
    key = c("highway"), sf = T)
base_geilo_building <- osmplotr::extract_osm_objects(bbox = bb_Geilo,
    key = c("building"), sf = T)
base_geilo_ww <- osmplotr::extract_osm_objects(bbox = bb_Geilo,
    key = c("waterway"), sf = T, return_type = "line")

# TYDAL
base_tydal_hw <- osmplotr::extract_osm_objects(bbox = bb_Tydal,
    key = "highway", sf = T)
base_tydal_building <- osmplotr::extract_osm_objects(bbox = bb_Tydal,
    key = c("building"), sf = T)
base_tydal_ww <- osmplotr::extract_osm_objects(bbox = bb_Tydal,
    key = c("waterway"), sf = T, return_type = "line")
```

Plot static map

```
# GEILO
static_geilo <- tm_shape(SHP_geilo_ll) + tm_polygons(col = "green") +
    tm_shape(depths_geilo) + tm_dots() + tm_shape(base_geilo_hw) +
    tm_lines() + tm_shape(base_geilo_ww) + tm_lines(col = "blue",
    size = 2) + tm_shape(base_geilo_building) + tm_polygons(col = "black",
    alpha = 0.3) + tm_compass() + tm_scale_bar() +
    tm_layout(title = "Geilo")


# TYDAL
static_tydal <- tm_shape(SHP_tydal_ll) + tm_polygons(col = "green") +
    tm_shape(depths_tydal) + tm_dots(size = "Dybde",
    col = "Dybde", palette = "-viridis") + tm_shape(base_tydal_ww) +
    tm_lines(col = "blue", size = 2) + tm_compass() +
    tm_scale_bar() + tm_layout(title = "Tydal", legend.show = F,
    inner.margins = c(0.1, 0.02, 0.1, 0.02))

tmap_arrange(static_geilo, static_tydal)
```

These to test cases are very different. Geilo is a set of several unique mire polygons. Usually, in development projects, one would estimate the peat volume and C stock for each of these separately, but we will try now to see if it can be done in one operation. Tydal is a more typical example of a clear peatland delineation. Both cases have dense peat depth measurements taken.

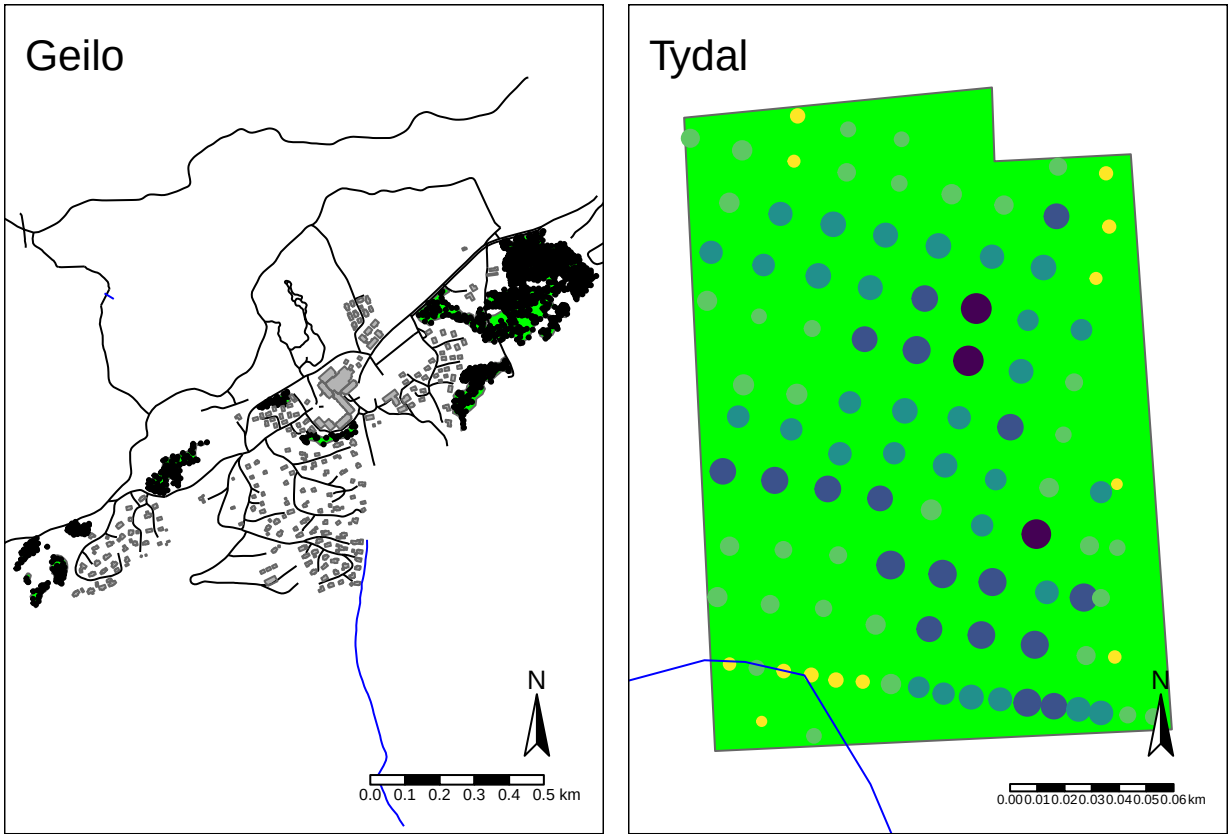


Figure S2.2: Map of Geilo and Tydal test sites. The peatland is deliminated as a green polygon(s). Dots are depth measurements and for Tydal the colour and size of the dots reflects the measured peat depths in meters.

Chapter 3

Find best model

We want to interpolate values from the depth measurement on to a raster grid. From this we can estimate the total volume of the peatland, and also visualize the depth profiles.

The main task in this chapter is the determination of optimal power for IDW by cross validation

3.1 Raster grid

Create raster grid based on the extent of the peatland

```
# Create empty raster grid (stars object) based
# on the extent of the peatland shape files.
# Resolution 1 = 1 m (becasue crs is UTM)
grid_Tydal_stars <- starsExtra::make_grid(SHP_tydal,
    1)
grid_Geilo_stars <- starsExtra::make_grid(SHP_geilo,
    1)
```

Visualise the grid

```
# To plot the grid we can convert the stars
# objects to sf. That way we can plot as polygons
# and visualise the individual cells.  This takes
# too long to perform for Geilo, but we can do it
# for Tydal:
grid_Tydal_sf <- st_as_sf(grid_Tydal_stars)

tm_shape(grid_Tydal_sf) + tm_polygons() + tm_shape(SHP_tydal) +
    tm_polygons(col = "yellow", alpha = 0.5)
```

Crop raster grid using the peatland delineation

```
grid_Tydal_stars_crop <- sf::st_crop(grid_Tydal_stars,
    SHP_tydal)
grid_Geilo_stars_crop <- sf::st_crop(grid_Geilo_stars,
    SHP_geilo)
```

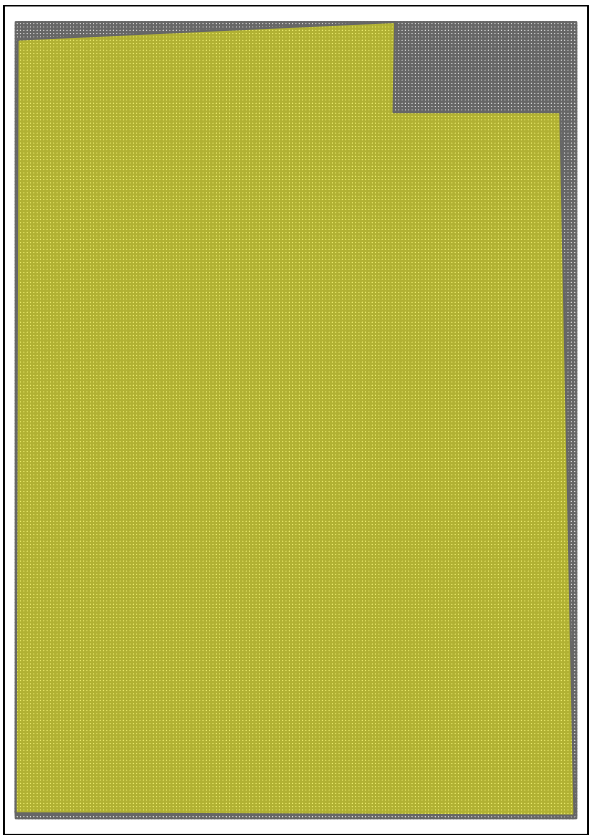


Figure S3.1: A 1x1 m raster grid in grey, overlayed with the peatland delineation in yellow.

```
tmap_arrange(tm_shape(grid_Geilo_stars_crop) + tm_raster(palette = "blue") +
    tm_layout(title = "Geilo", legend.show = F), tm_shape(grid_Tydal_stars_crop) +
    tm_raster(palette = "red") + tm_layout(title = "Tydal",
    legend.show = F))
```

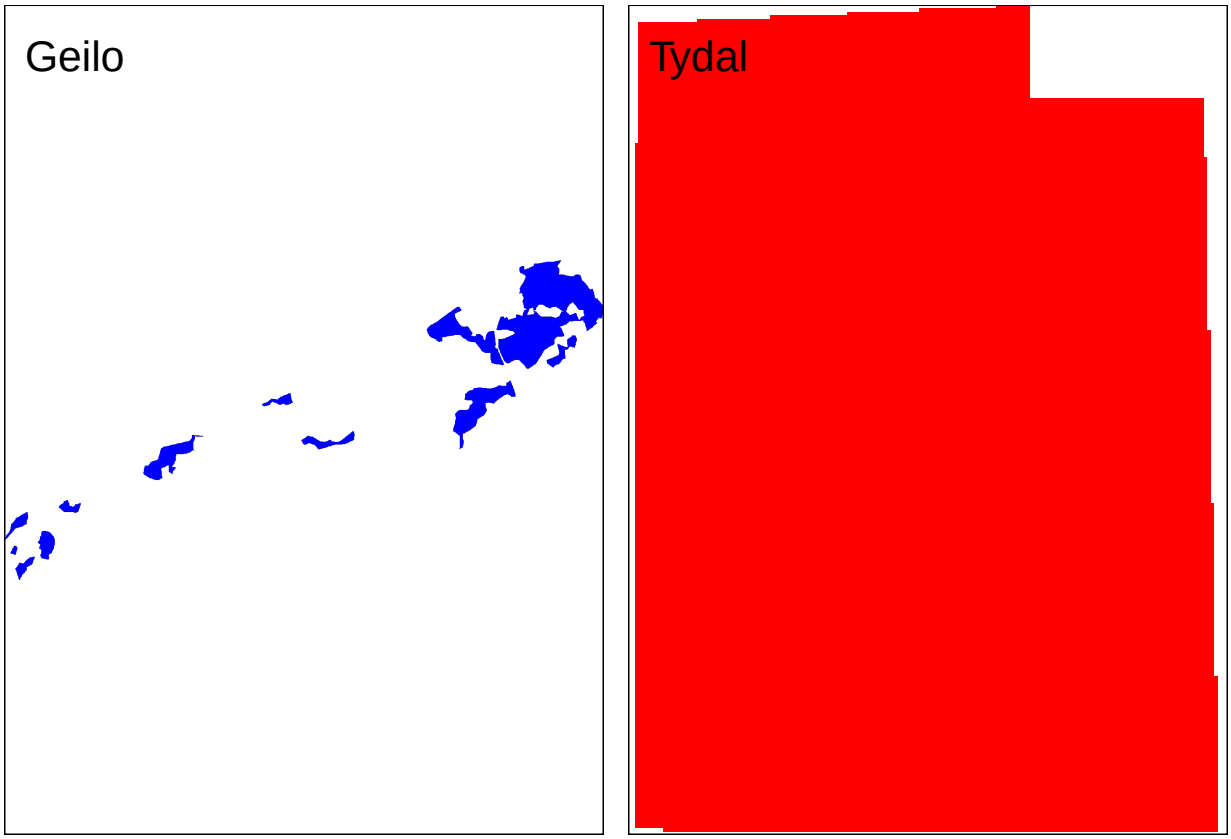


Figure S3.2: Preview of the cropped raster grids.

3.2 Inverse distance weighting

We chose to use Inverse distance weighting (IDW) for the interpolation instead of for example kriging. This is because we do not want to include a separate step with for example calculating a variogram, and the functions for automatically fitting a variogram seems unstable.

The IDW requires setting the number of neighboring points to consider. We could set this to consider all points in the dataset (the default), but this increases calculation times and it is also not sensible in general. I will start by considering the 9 nearest points, as this would mean all the 8 closest points in a systematic grid around a point is included.

```
nmax <- 9
```

IDW also has a power parameter, and this is where we can tune the results by down-weighting data points that are further away. This parameter needs to be optimized for each model. We will fit a model with power varying from 0 to 10 and explore the model fit as a result of the chosen power.

First, lets explore the idw function and what the parameters mean. Tydal is 4.4 meters at it's deepest point. Depending on the idw setting, the prediction map will be further of closer to this number. For example, is we only

consider the closest neighbor, we are essentially performing a nearest neighbor interpolation (Fig. 3.3; left pane). The choice of power then has no effect, and the deepest point in the prediction map equals the deepest point in the measured data.

If we keep the power at zero (ignore distance weighting) and turn up nmax to 9, each raster cell is predicted to be the mean of the nine closest points (Fig. 3.3; middle pane). The overall predicted depth decreases.

If we then also include distance weighting, e.g. by setting power =2, we get smoothing between the points, and a more local prediction than when we had no power (i.e. the deepest point is again close to its true value) (Fig. 3.3; right pane).

```
temp <- gstat::idw(formula = Dybde ~ 1, locations = depths_tydal,
    newdata = grid_Tydal_stars_crop, idp = 0, nmax = 1)
#> [inverse distance weighted interpolation]

temp2 <- gstat::idw(formula = Dybde ~ 1, locations = depths_tydal,
    newdata = grid_Tydal_stars_crop, idp = 0, nmax = 9)
#> [inverse distance weighted interpolation]

temp3 <- gstat::idw(formula = Dybde ~ 1, locations = depths_tydal,
    newdata = grid_Tydal_stars_crop, idp = 2, nmax = 9)
#> [inverse distance weighted interpolation]


temp4 <- gstat::idw(formula = Dybde ~ 1, locations = depths_tydal,
    newdata = grid_Tydal_stars_crop, idp = 2, nmax = 20)
#> [inverse distance weighted interpolation]

temp4.1 <- gstat::idw(formula = Dybde ~ 1, locations = depths_tydal,
    newdata = grid_Tydal_stars_crop, idp = 2)
#> [inverse distance weighted interpolation]

temp5 <- gstat::idw(formula = Dybde ~ 1, locations = depths_tydal,
    newdata = grid_Tydal_stars_crop, idp = 100, nmax = 9)
#> [inverse distance weighted interpolation]

gg_temp <- tm_shape(temp) + tm_raster(col = "var1.pred",
    title = "", style = "fixed", breaks = seq(0, 5,
        1)) + tm_shape(depths_tydal) + tm_symbols(shape = 4,
    col = "black") + tm_layout(legend.outside = F,
    title = paste0("Power = 0\nNmax = 1"))

gg_temp2 <- tm_shape(temp2) + tm_raster(col = "var1.pred",
    title = "", style = "fixed", breaks = seq(0, 5,
        1)) + tm_shape(depths_tydal) + tm_symbols(shape = 4,
    col = "black") + tm_layout(legend.outside = F,
    title = paste0("Power = 0\nNmax = 9"))

gg_temp3 <- tm_shape(temp3) + tm_raster(col = "var1.pred",
    title = "", style = "fixed", breaks = seq(0, 5,
        1)) + tm_shape(depths_tydal) + tm_symbols(shape = 4,
    col = "black") + tm_layout(legend.outside = F,
    title = paste0("Power = 2\nNmax = 9"))

gg_temp4 <- tm_shape(temp4) + tm_raster(col = "var1.pred",
    title = "", style = "fixed", breaks = seq(0, 5,
        1)) + tm_shape(depths_tydal) + tm_symbols(shape = 4,
    col = "black") + tm_layout(legend.outside = F,
```

```
    title = paste0("Power = 2\nNmax = 20"))

gg_temp4.1 <- tm_shape(temp4.1) + tm_raster(col = "var1.pred",
    title = "", style = "fixed", breaks = seq(0, 5,
        1)) + tm_shape(depths_tydal) + tm_symbols(shape = 4,
    col = "black") + tm_layout(legend.outside = F,
    title = paste0("Power = 2\nNmax = Inf"))

gg_temp5 <- tm_shape(temp5) + tm_raster(col = "var1.pred",
    title = "", style = "fixed", breaks = seq(0, 5,
        1)) + tm_shape(depths_tydal) + tm_symbols(shape = 4,
    col = "black") + tm_layout(legend.outside = F,
    title = paste0("Power = 100\nNmax = 9"))
```

```
tmap_arrange(gg_temp, gg_temp2, gg_temp3, ncol = 3)
```

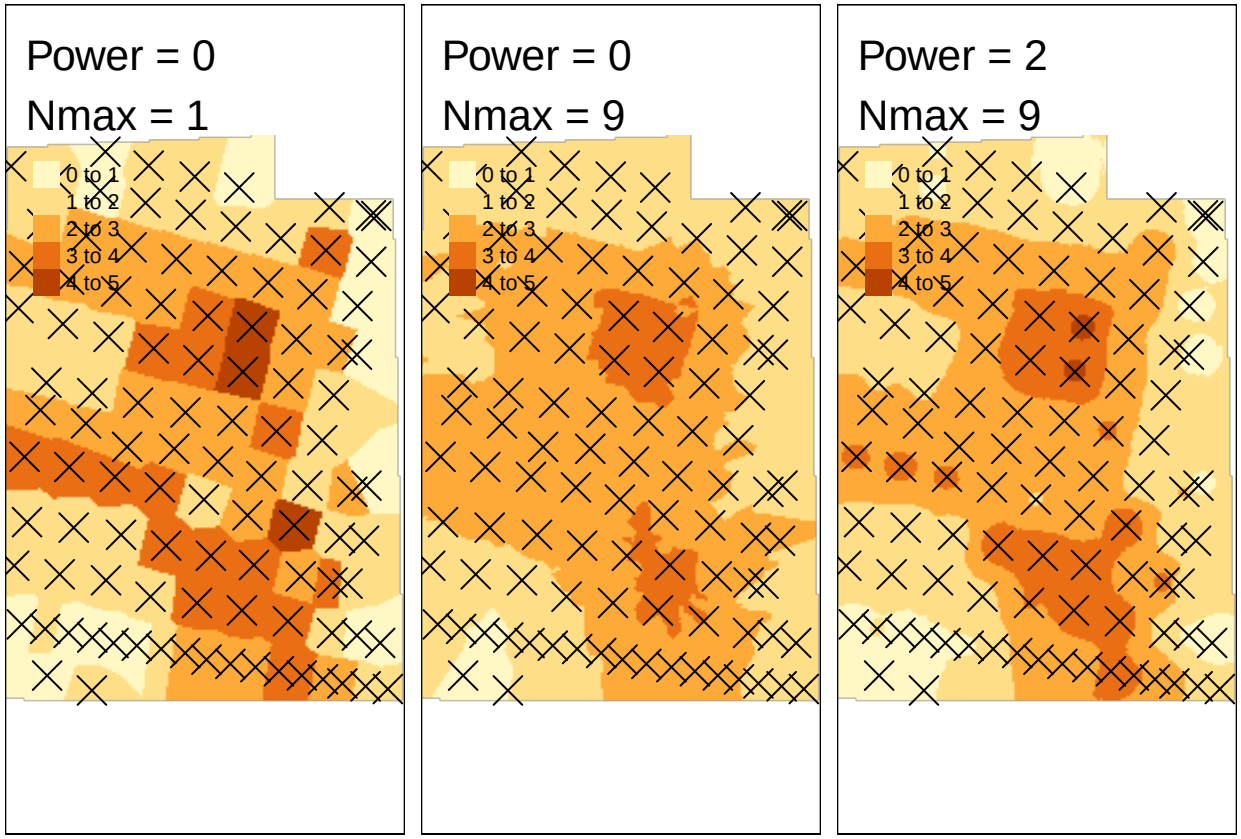


Figure S3.3: IDW with different setting.

```
tmap_arrange(gg_temp4, gg_temp4.1, gg_temp5, ncol = 3)
```

If we increase nmax to 20, and retain the distance weighting, we get a slightly flatter (loss local) estimation than with nmax = 9 (Fig. 3.4; left pane).

If we ramp up the nmax to consider all the points in the dataset, this has little additional effect as long as we have a positive power parameter (Fig. 3.4; middle pane). For bigger data sets however, the computation time might become an issue.

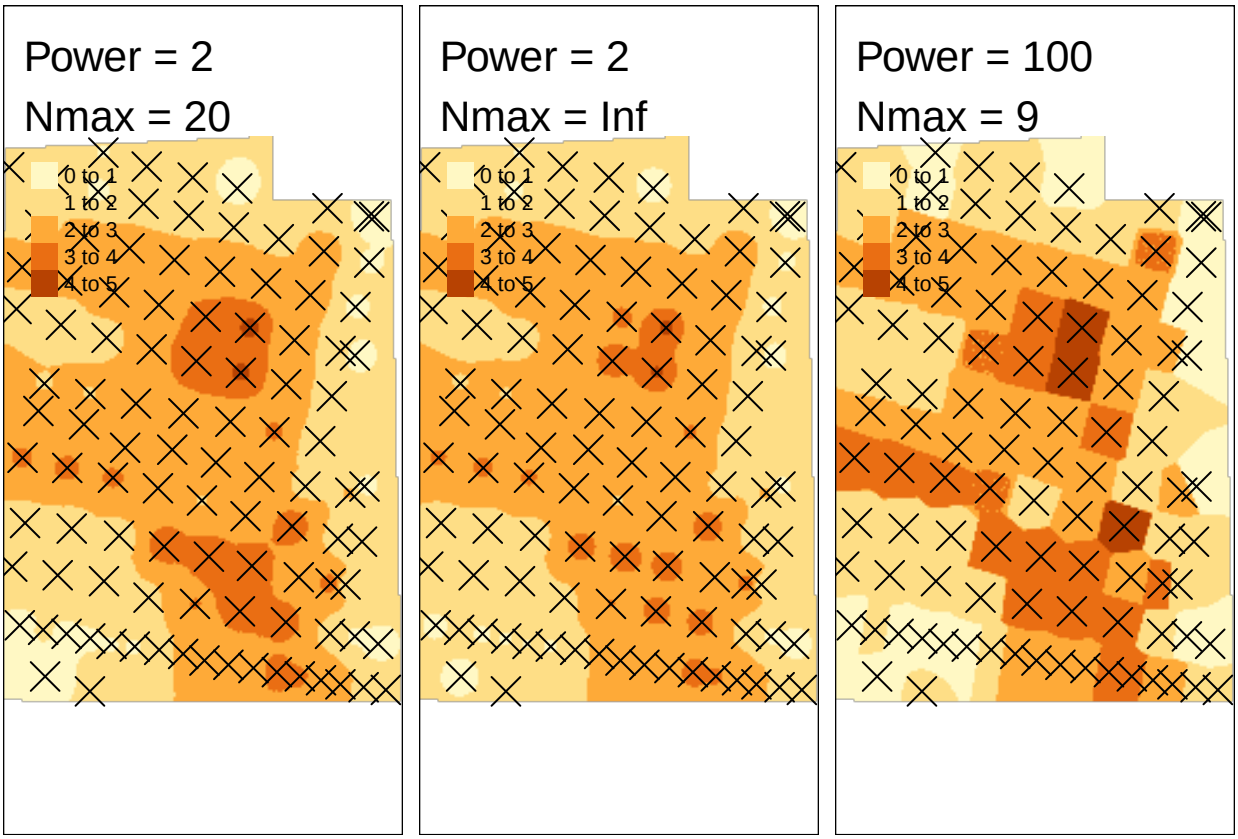


Figure S3.4: IDW with different setting.

If we keep nmax at 9, and ramp up the power, we approach what we had at the beginning, that is a nearest neighbor interpolation (Fig. 3.4; right pane).

Of all the examples above, power = 2 and nmax = 9 is probably the best. However, I don't like these little dots on the map. They are artefacts of the model, and do not exist for real. Therefore I think I would prefer a model with a slightly more local focus. We can increase the power a bit to achieve this.

```
temp <- gstat::idw(formula = Dybde ~ 1, locations = depths_tydal,
    newdata = grid_Tydal_stars_crop, idp = 4, nmax = 9)
#> [inverse distance weighted interpolation]

tm_shape(temp) + tm_raster(col = "var1.pred", title = "",
    style = "fixed", breaks = seq(0, 5, 1)) + tm_shape(depths_tydal) +
    tm_symbols(shape = 4, col = "black") + tm_layout(legend.outside = T,
    title = paste0("Power = 4\nNmax = 9"))
```

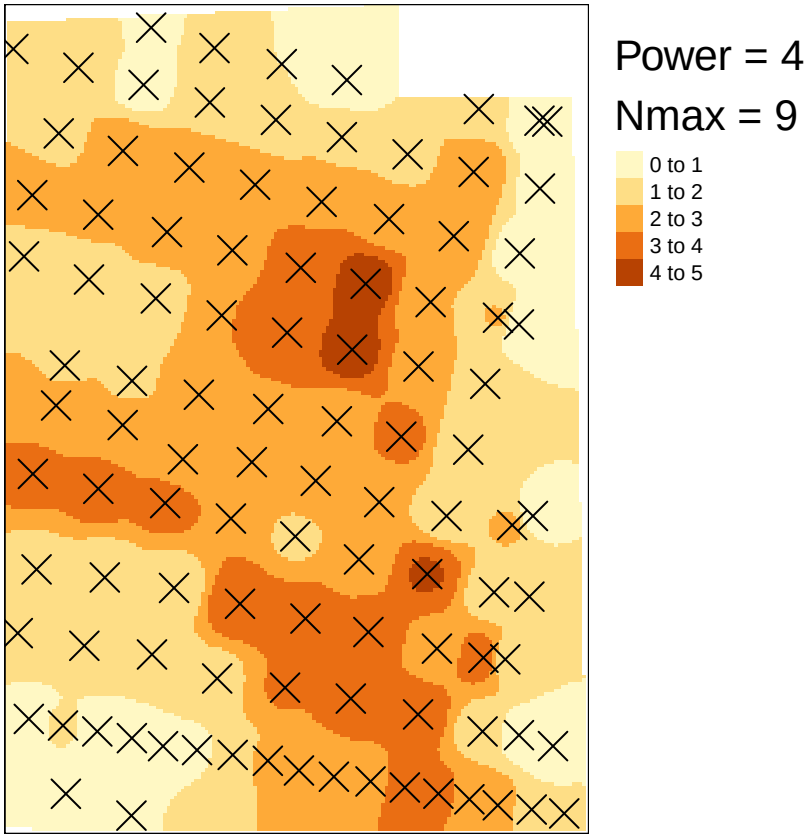


Figure S3.5: IDW with relatively local focus.

After a bit of trial and error, I noticed that by having nmax as low as 9, the optimum power frequently became <2 (which results in model artefacts and biased predictions as you can read about below, esp Fig. 3.9). I therefore increase the nmax.

```
temp <- gstat::idw(formula = Dybde ~ 1, locations = depths_tydal,
    newdata = grid_Tydal_stars_crop, idp = 5, nmax = 20)
#> [inverse distance weighted interpolation]

tm_shape(temp) + tm_raster(col = "var1.pred", title = "",
    style = "fixed", breaks = seq(0, 5, 1)) + tm_shape(depths_tydal) +
```

```
    tm_symbols(shape = 4, col = "black") + tm_layout(legend.outside = T,
    title = paste0("Power = 4\nNmax = 20"))
```

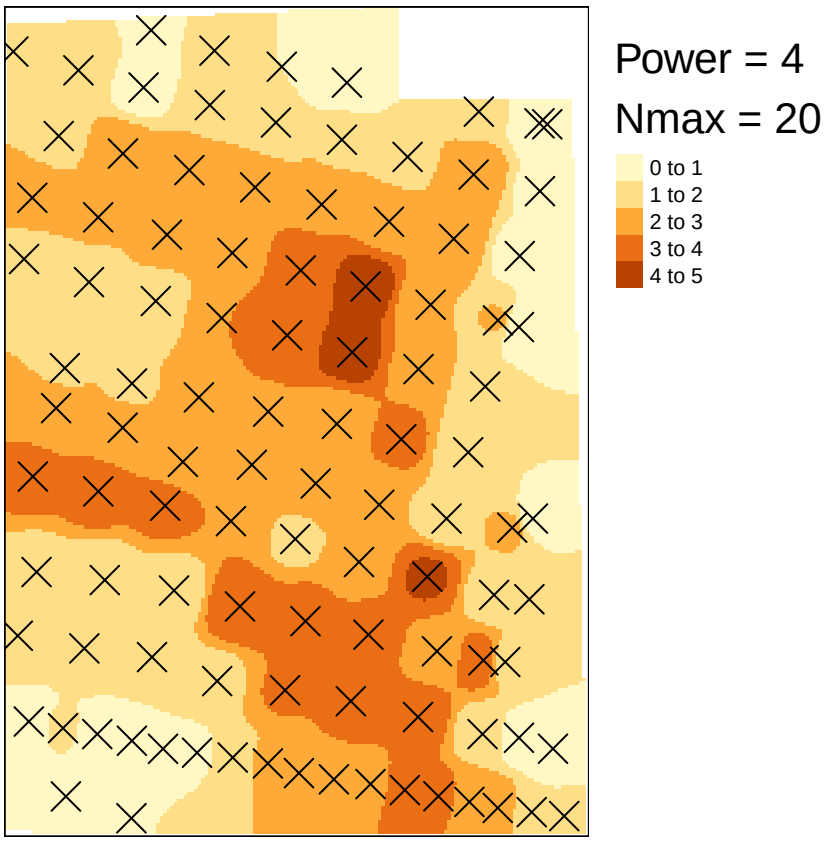


Figure S3.6: IDW with slightly higher nmax and higher power.

```
nmax <- 20
```

This tuning of the power parameter is where we will focus now. Here I perform IDW for power between 0 and 10 to compare the results.

```
myPowers <- c(0:10)
for (i in myPowers) {
    temp <- gstat::idw(formula = Dybde ~ 1, locations = depths_tydal,
        newdata = grid_Tydal_stars_crop, idp = i, nmax = nmax)
    assign(paste0("IDW_tydal_", i), temp)
}

# this one takes a couple of minutes to run
for (i in myPowers) {
    temp <- gstat::idw(formula = Dybde ~ 1, locations = depths_geilo,
        newdata = grid_Geilo_stars_crop, idp = i)
    assign(paste0("IDW_geilo_", i), temp)
}
```

What do these models look like.

```
for (i in myPowers) {

    temp <- get(paste0("IDW_tydal_", i))

    temp <- tm_shape(temp) + tm_raster(col = "var1.pred",
        palette = "-viridis", title = "Interpolated peat\ndepth (m)") +
        tm_layout(legend.outside = T, title = paste0("Power = ",
            i)) + tm_shape(depths_tydal) + tm_symbols(col = "black",
        shape = 4, size = 0.5)
    assign(paste0("tmap_IDW_tydal_power_", i), temp)
}



tmap_arrange(tmap_IDW_tydal_power_0, tmap_IDW_tydal_power_1,
    tmap_IDW_tydal_power_2, tmap_IDW_tydal_power_4,
    tmap_IDW_tydal_power_6, tmap_IDW_tydal_power_8,
    ncol = 3)
```

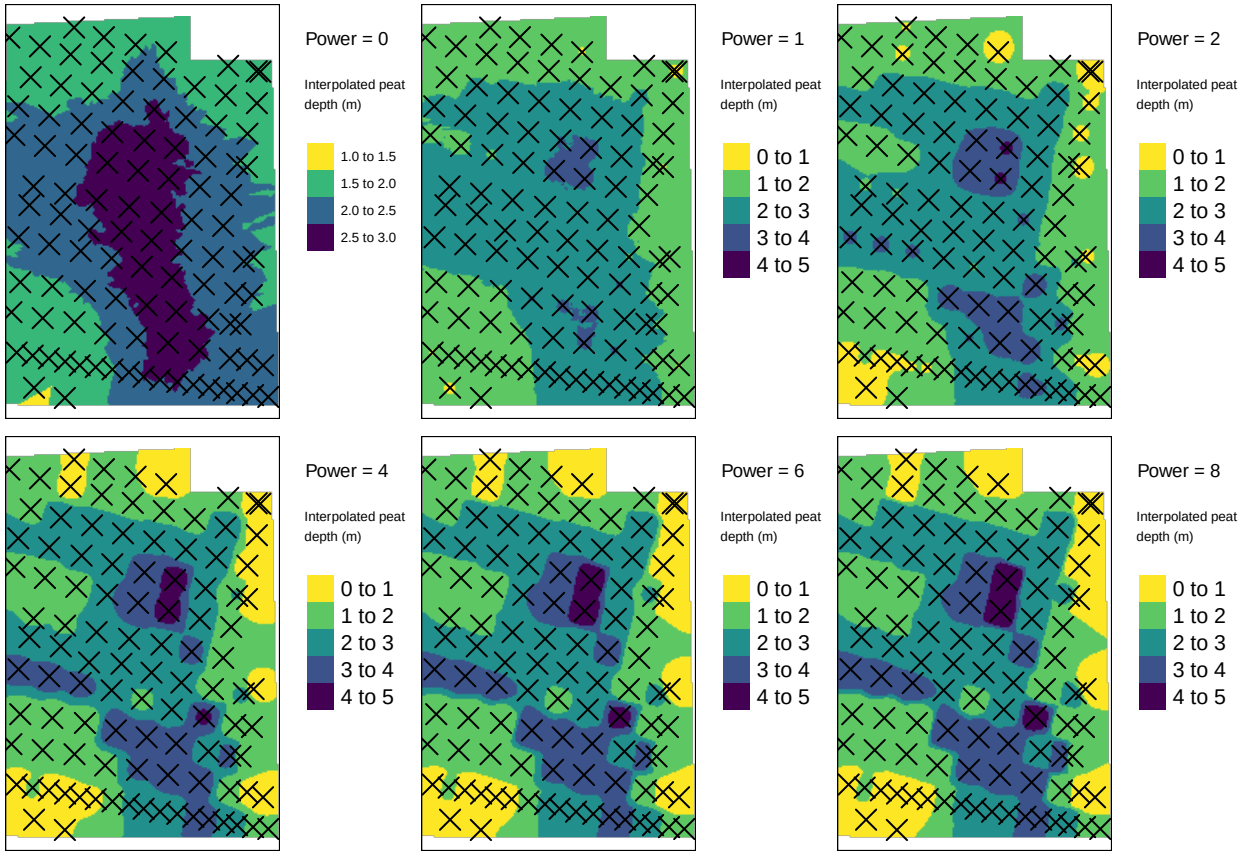


Figure S3.7: First view of how IDW models look like depending on the power parameter. Test site: Tydal.

When plotting Geilo I cannot include the crosses for the depth measurements because there are just too many and the clutter up the image.

```
for (i in myPowers) {

    temp <- get(paste0("IDW_geilo_", i))
```

```
    temp <- tm_shape(temp) + tm_raster(col = "var1.pred",
        palette = "-viridis", title = "Interpolated peat\ndepth (m)") +
        tm_layout(legend.outside = T, title = paste0("Power = ",
            i))
    assign(paste0("tmap_IDW_geilo_power_", i), temp)
}



tmap_arrange(tmap_IDW_geilo_power_1, tmap_IDW_geilo_power_2,
    tmap_IDW_geilo_power_4, ncol = 1)
```

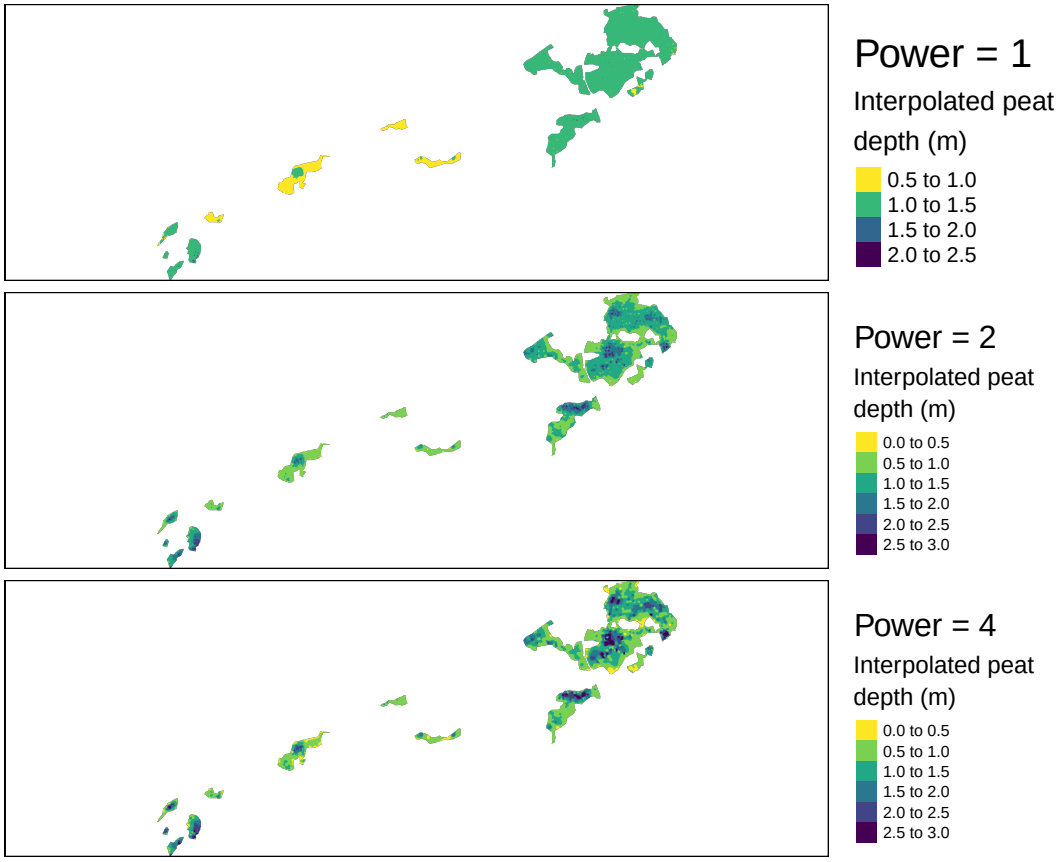


Figure S3.8: First view of how IDW models look like depending on the power parameter. Test site: Geilo.

A low power creates a more flat map where all the depth measurements count equally to any predicted value. With power set to 0 we just predict the average of all the measured peat depths. For both cases, increasing the power beyond 4 has little visual effect.

3.3 Original residuals

Let's check the fit with a plot of the residuals.

```
# Prepare data frame with the measured data
resids_tydal <- data.frame(ID = 1:nrow(depths_tydal))
resids_geilo <- data.frame(ID = 1:nrow(depths_geilo))
```

```
resids_tydal$measured <- depths_tydal$Dybde
resids_geilo$measured <- depths_geilo$Dybde

# Get the predicted values
for (i in myPowers) {
    temp <- get(paste0("IDW_tydal_", i))
    fit <- stars::st_extract(temp, depths_tydal)
    resids_tydal[, i + 3] <- resids_tydal$measured -
        fit$var1.pred
}

for (i in myPowers) {
    temp <- get(paste0("IDW_geilo_", i))
    fit <- stars::st_extract(temp, depths_geilo)
    resids_geilo[, i + 3] <- resids_geilo$measured -
        fit$var1.pred
}

# Lots of NA's produced for depth measurements
# taken outside the peatland polygons
# tmap_mode('plot') tm_shape(IDW_geilo_10)+
# tm_raster(col='var1.pred')+
# tm_shape(fit[is.na(fit$var1.pred),])+
# tm_symbols(size = .1, col='yellow')

resids_geilo <- resids_geilo[!is.na(resids_geilo$V3),
    ]


resids_tydal <- tidyr::pivot_longer(resids_tydal, !ID &
    !measured, names_to = "power", values_to = "residual")
resids_geilo <- tidyr::pivot_longer(resids_geilo, !ID &
    !measured, names_to = "power", values_to = "residual")


# Remove V from column names
resids_tydal$power <- gsub(pattern = "V", replacement = "",
    x = resids_tydal$power)
resids_geilo$power <- gsub(pattern = "V", replacement = "",
    x = resids_geilo$power)

resids_tydal$power <- as.numeric(resids_tydal$power)
resids_geilo$power <- as.numeric(resids_geilo$power)

resids_tydal$power <- resids_tydal$power - 3
resids_geilo$power <- resids_geilo$power - 3


gg_resids_tydal <- ggplot(resids_tydal, aes(x = factor(power),
    y = residual)) + geom_boxplot(fill = "grey", trim = F) +
    theme_bw(base_size = 12) + xlab("Power") + ggtitle("Tydal")
#> Warning in geom_boxplot(fill = "grey", trim = F): Ignoring
#> unknown parameters: `trim`
```

```
gg_resids_geilo <- ggplot(resids_geilo, aes(x = factor(power),
    y = residual)) + geom_boxplot(fill = "grey", trim = F) +
    theme_bw(base_size = 12) + xlab("Power") + ggtitle("Geilo")
#> Warning in geom_boxplot(fill = "grey", trim = F): Ignoring
#> unknown parameters: `trim`
```

```
ggpubr::ggarrange(gg_resids_geilo, gg_resids_tydal)
```

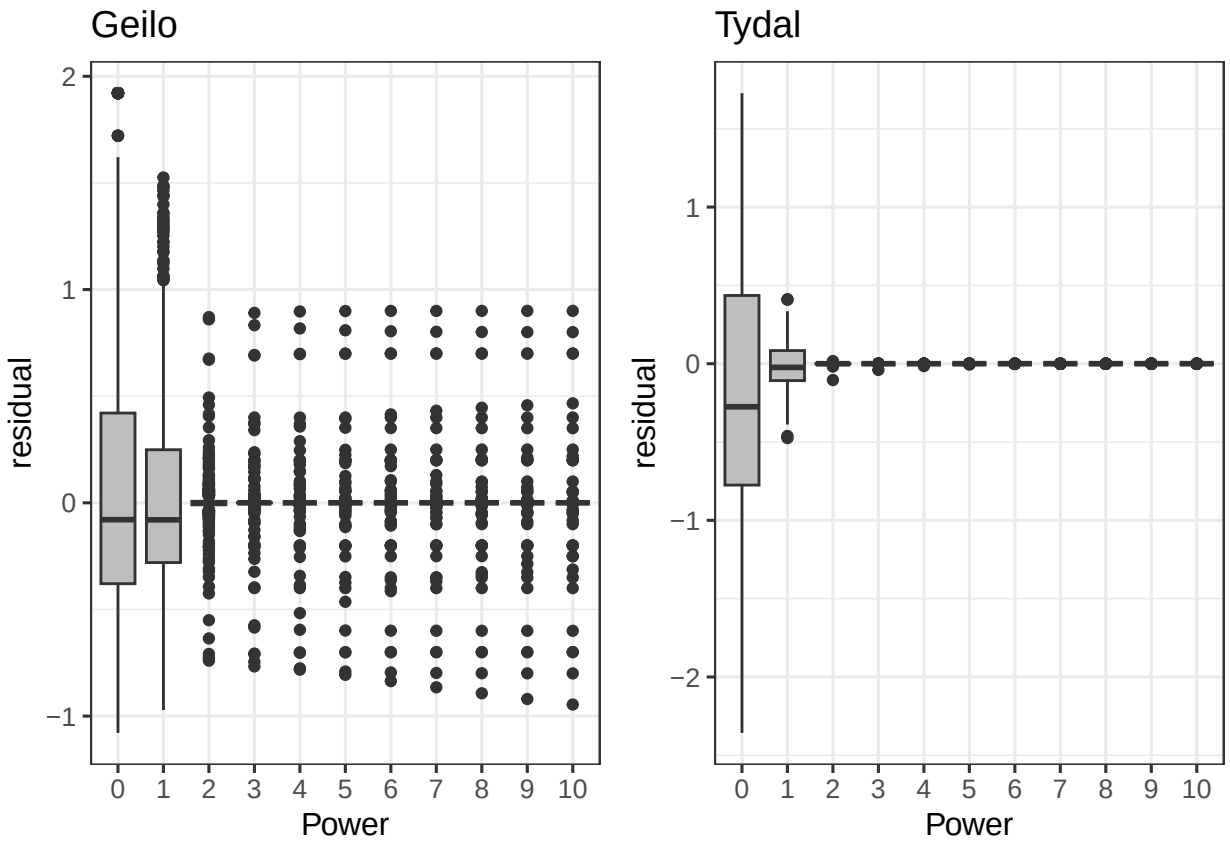


Figure S3.9: Residuals (observed - predicted) stabelize with power greater than 2.

With a power of 2 or greater, the residuals are very stable and unbiased (centered on 0).

3.4 Estimated volume

What about the estimated volume - does that change depending on the power setting? Note that because the data are projected in UTM, the mean depth (in meter) of each 1x1 raster cell equals the volume in m3.

```
volume_tydal <- NULL
for (i in myPowers) {
    temp <- get(paste0("IDW_tydal_", i))
    volume_tydal <- c(volume_tydal, sum(temp$var1.pred,
        na.rm = T))
}
```

```
volume_geilo <- NULL
for (i in myPowers) {
    temp <- get(paste0("IDW_geilo_", i))
    volume_geilo <- c(volume_geilo, sum(temp$var1.pred,
        na.rm = T))
}
```

```
volume_tydal <- data.frame(volume = volume_tydal, power = myPowers)
volume_tydal$relative_volume <- volume_tydal$volume/mean(volume_tydal$volume) *
    100

volume_geilo <- data.frame(volume = volume_geilo, power = myPowers)
volume_geilo$relative_volume <- volume_geilo$volume/mean(volume_geilo$volume) *
    100


gg_predVolume_tydal <- ggplot(volume_tydal, aes(x = factor(power),
    y = relative_volume)) + geom_point(size = 10) +
    xlab("power") + ylab("Peat volume as a percentage of\nmean predicted peat volume") +
    theme_bw(base_size = 12) + ggtitle("Tydal")

gg_predVolume_geilo <- ggplot(volume_geilo, aes(x = factor(power),
    y = relative_volume)) + geom_point(size = 10) +
    xlab("power") + ylab("Peat volume as a percentage of\nmean predicted peat volume") +
    theme_bw(base_size = 12) + ggtitle("Geilo")

ggpubr::ggarrange(gg_predVolume_geilo, gg_predVolume_tydal)
```

The values are stabilizing relatively fast, but a power of less than 3 seem like a bad idea. In general though, the difference between the worst and best models is just 2-4%.

This relationship probably depends a bit on the number of data points and the density of peat depth measurements. So we should not generalise just yet.

3.5 Model fit LOOCV

Now let's do a proper test of model fit, using leave one out cross validation. I will use the mean absolute deviation as my measure of model fit.

```
MAE_tydal <- data.frame(power = myPowers, MAE = as.numeric(NA))
for (i in myPowers) {
    temp <- krige.cv(Dybde ~ 1, depths_tydal, set = list(idp = i),
        nmax = nmax)
    MAE_tydal$MAE[MAE_tydal$power == i] <- mean(abs(temp$residual))
}
#>   |                                                                      |
#>   |                                                                      |
#>   |                                                                      |
#>   |                                                                      |
#>   |                                                                      |
#>   |                                                                      |
#>   |                                                                      |
#>   |                                                                      |
#>   |                                                                      |
```

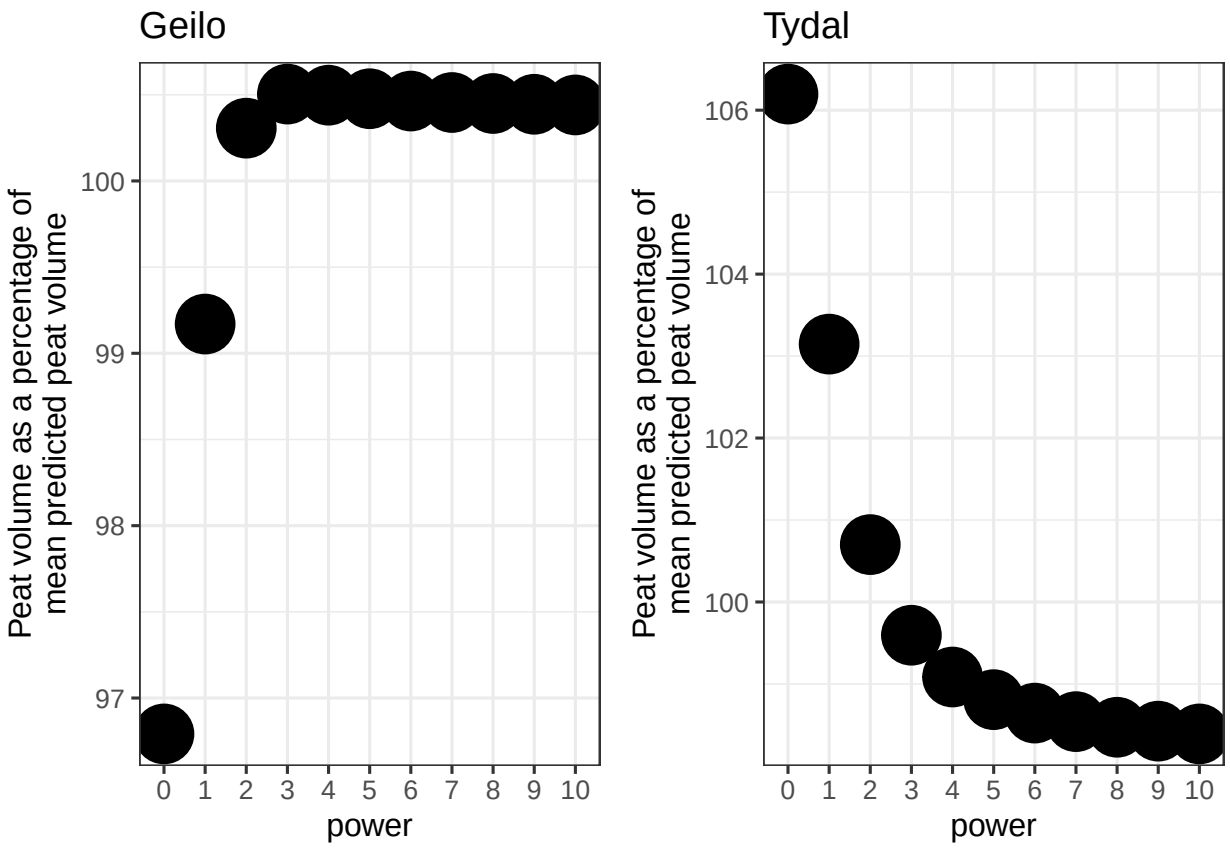


Figure S3.10: Peat volume estimates as a function of the power parameter.

```
#>   |                                                                      |
#>   |                                                                      |

MAE_tydal$best <- ifelse(MAE_tydal$MAE == min(MAE_tydal$MAE),
    "best", "not-best")
```

For Geilo the calculations take longer, so I will not consider as big a range in power.

```
myPowers2 <- seq(0, 5, 1)
MAE_geilo <- data.frame(power = myPowers2, MAE = as.numeric(NA))
for (i in myPowers2) {
    temp <- krige.cv(Dybde ~ 1, depths_geilo, set = list(idp = i),
        nmax = nmax)
    MAE_geilo$MAE[MAE_geilo$power == i] <- mean(abs(temp$residual))
}

MAE_geilo$best <- ifelse(MAE_geilo$MAE == min(MAE_geilo$MAE),
    "best", "not-best")
saveRDS(MAE_geilo, "Data/cache/MAE_geilo_cashe.rds")
```

```
gg_MAE_tydal <- ggplot(MAE_tydal, aes(x = power, y = MAE,
    colour = best, shape = best)) + geom_point(size = 10) +
    theme_bw(base_size = 12) + scale_x_continuous(breaks = myPowers) +
    guides(colour = "none", shape = "none") + scale_color_manual(values = c("darkgreen",
    "grey")) + scale_shape_manual(values = c(18, 19)) +
    ggtitle("Tydal")

gg_MAE_geilo <- ggplot(MAE_geilo, aes(x = power, y = MAE,
    colour = best, shape = best)) + geom_point(size = 10) +
    theme_bw(base_size = 12) + scale_x_continuous(breaks = myPowers2) +
    guides(colour = "none", shape = "none") + scale_color_manual(values = c("darkgreen",
    "grey")) + scale_shape_manual(values = c(18, 19)) +
    ggtitle("Geilo")

ggpubr::ggarrange(gg_MAE_geilo, gg_MAE_tydal)
```

The mean absolute error is not negligible in any of the two cases. The interpretation is, if you leave out one of the peat depth measurement and predict the peat depth in that point (using just the remaining data points) the models are on average 40-60 cm wrong.

The best models have power = 2 for Geilo and 4 for Tydal. These models also look reasonable in terms of predicted peat volume and the spread of the raw residuals (Figs. 3.7 and 3.8). But the map for Geilo is quite *spotted*, and perhaps a higher power would be smart. I will therefore change it to 3. This is justified also because of the minimal impact in MAE and the predicted volume. For Geilo, we probably would get better results if we analysed each polygon separately.

Note: A power less than 2 is not recommended as we have seen above. With nmax set to 9, the optimum power for Geilo was actually 1. I therefore went back up in the script and increased nmax to 20.

3.6 Best models

Based on a criterion of lowest MAE (but above 2) we have identified the best IDW models for the two test sites:

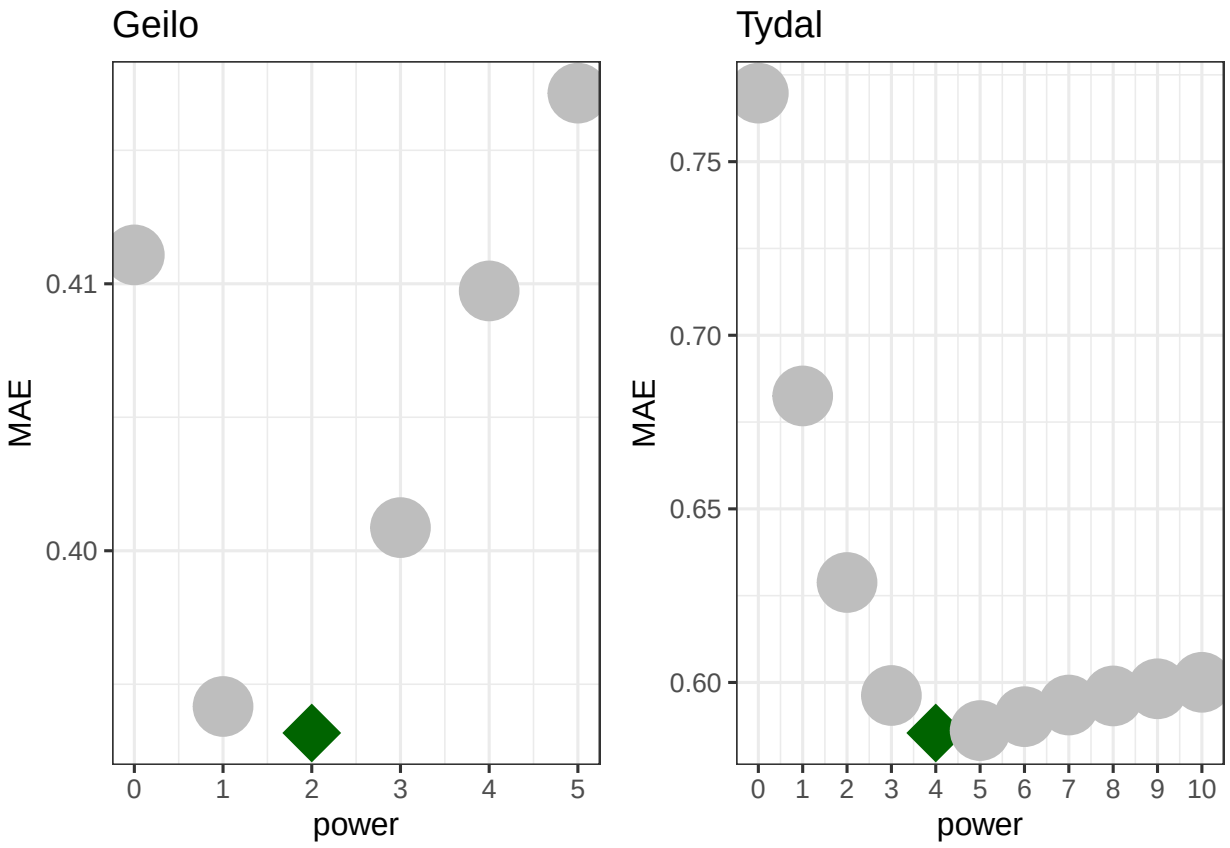


Figure S3.11: Mean absolute error based on leave-one-out cross validation. The optimal power setting is indicated by a green star symbol.

```
tm_tydal <- tm_shape(IDW_tydal_4) + tm_raster(col = "var1.pred",
    palette = "-viridis", title = "Interpolated peat\ndepth (m)") +
    tm_shape(depths_tydal) + tm_symbols(shape = 4,
    col = "black", size = 0.5) + tm_compass(type = "8star",
    position = c("right", "bottom"), size = 2) + tm_scale_bar(position = c("left",
    "bottom"), width = 0.3) + tm_layout(inner.margins = c(0.15,
    0.05, 0, 0.05), legend.outside = T)

# tmap::tmap_save(tm_tydal,
# 'Output/plot_IDW_tydal_bestModel.tiff', width =
# 1600, height = 1600, units = 'px')
tm_tydal
```

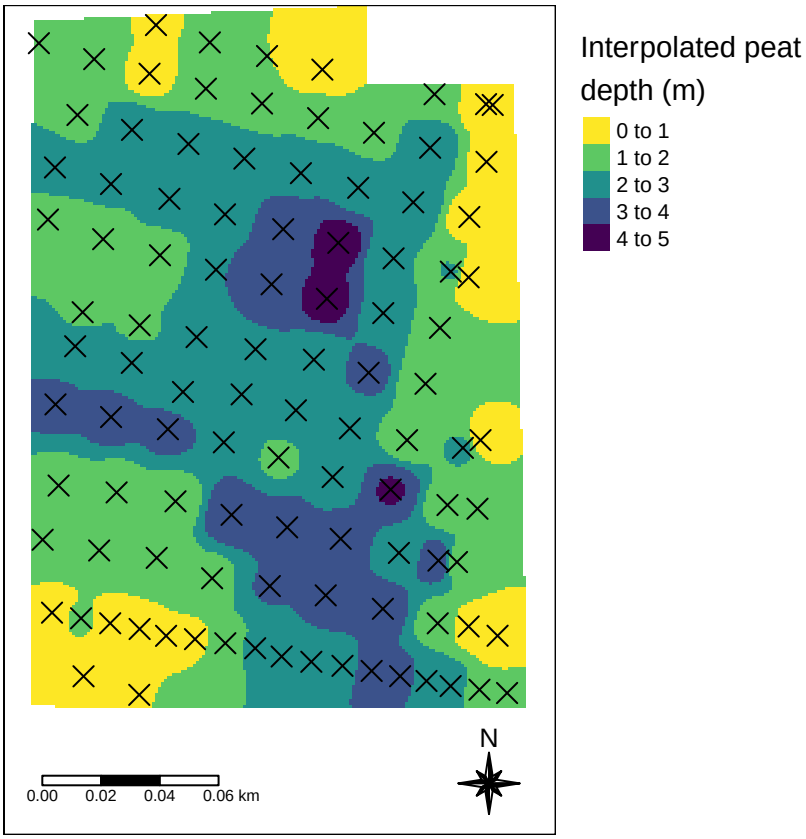


Figure S3.12: Best model for estimating the peat depth profile for the Tydal test site. Dots indicatore peat depth sampling points. MAE = 0.59 meters. Power = 4.

```
IDW_geilo_3 <- gstat::idw(formula = Dybde ~ 1, locations = depths_geilo,
    newdata = grid_Geilo_stars_crop, idp = 2, nmax = nmax)
```

```
tm_geilo <- tm_shape(IDW_geilo_3) + tm_raster(col = "var1.pred",
    palette = "-viridis", title = "Interpolated peat\ndepth (m)") +
    tm_compass(type = "8star", position = c("right",
        "bottom"), size = 2) + tm_scale_bar(position = c("right",
    "bottom"), width = 0.3)

# tmap::tmap_save(tm_geilo,
```

```
# 'Output/plot_IDW_geilo_bestModel.tiff', width =
# 1600, height = 1600, units = 'px')
tm_geilo
#> stars object downsampled to 1369 by 730 cells. See tm_shape manual (argument raster.downsample)
```

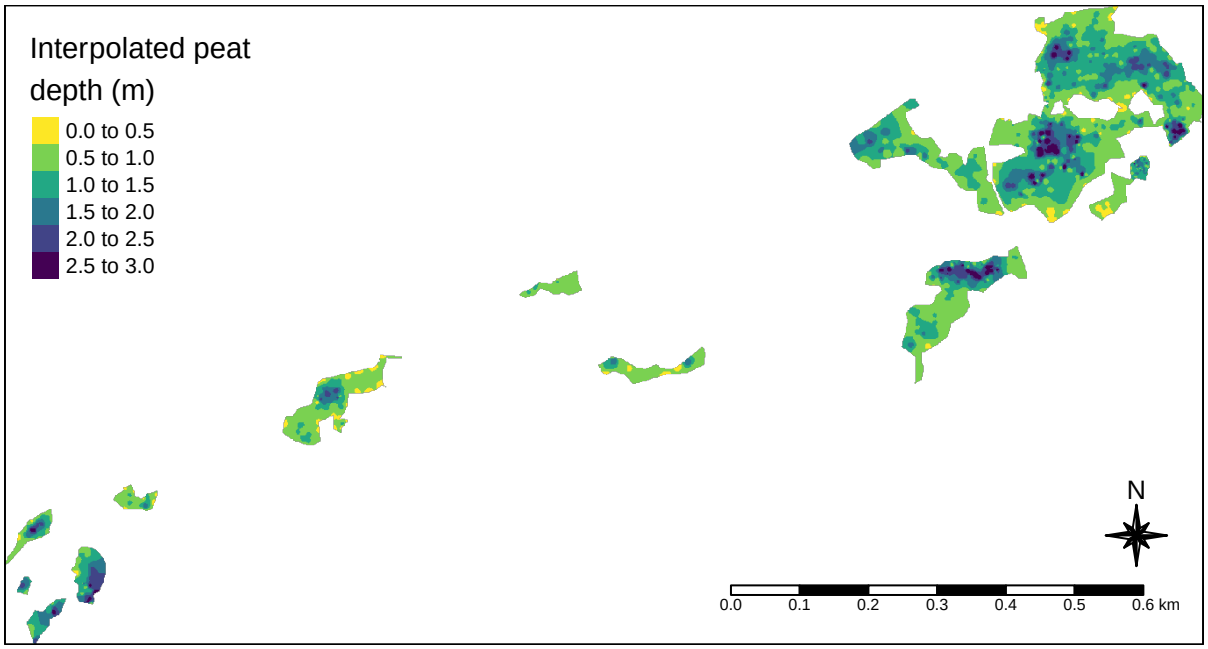


Figure S3.13: Best model for estimating the peat depth profile for the Geilo test site. Dots indicatore peat depth sampling points. MAE = 0.4 meters. Power = 3.

Tydal is 7.7336967×10^4 m^3 and Geilo is 1.0246996×10^5 m^3. We can get the residuals from LOOCV.

```
SE_tydal <- krige.cv(Dybde ~ 1, depths_tydal, set = list(idp = 4),
    nmax = nmax)
#>   |                                                                      |
SE_geilo <- krige.cv(Dybde ~ 1, depths_geilo, set = list(idp = 3),
    nmax = nmax)
#>   |                                                                      |


ggplot(SE_tydal, aes(x = residual)) + geom_histogram()
```

For any point inside the prediction area, we assume an error equal to the mean residual from LOOCV. For Tydal, this is 0.59 meters. There is no clear way how to predict confidence intervals for IDW. The residuals, and the MAE, represents uncertainty for each grid cell value, and not uncertainty in the sum of all these grid cells. Since the mean of the residuals (not the absolute mean, but the normal mean) is quite close to 0, the uncertainty for each cell cancels out.

We can calculate the peat volume based on the best model

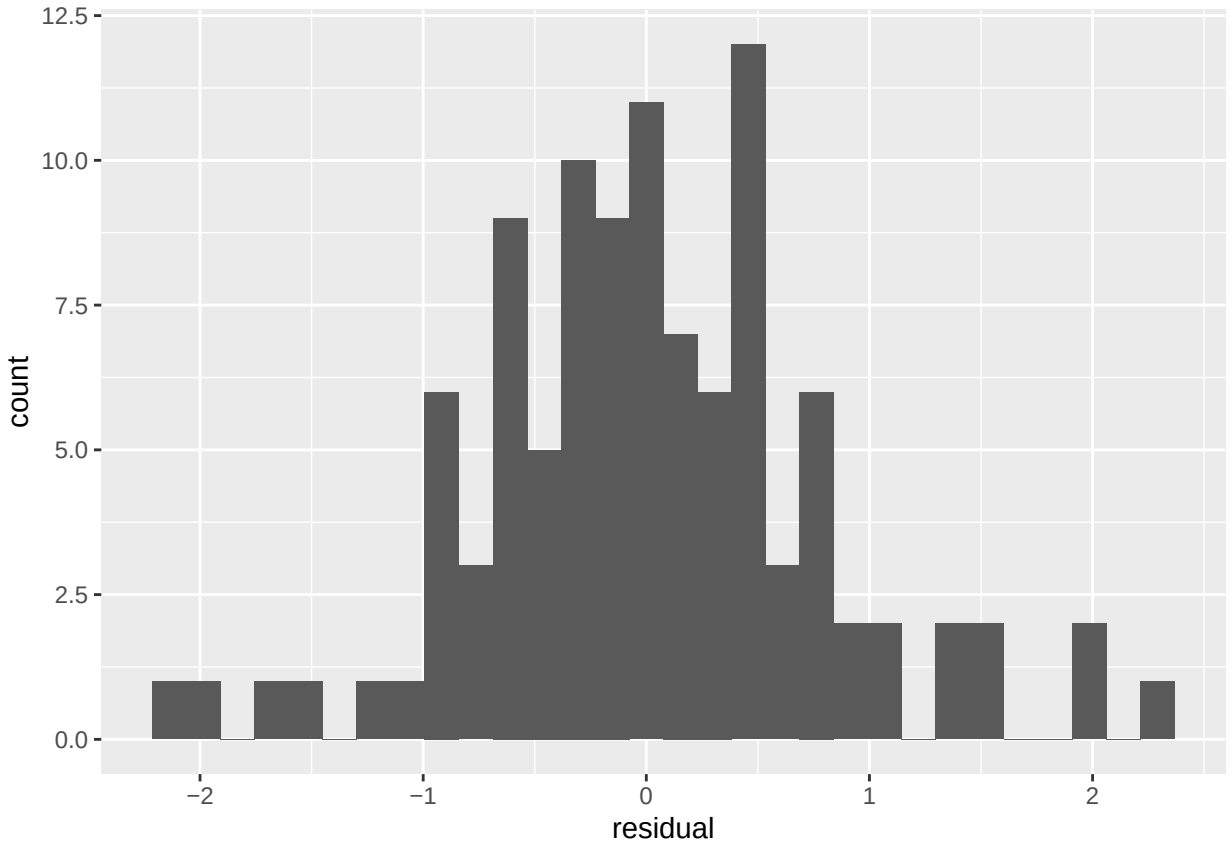


Figure S3.14: Resuduals from LOOCV. Test site: Tydal

```
volume <- data.frame(site = c("Tydal", "Geilo"), volume = c(sum(IDW_tydal_4$var1.pred,
    na.rm = T), sum(IDW_geilo_3$var1.pred, na.rm = T)))
```

```
library(ggtext)  # write superscript as markdown
ggplot(volume, aes(x = site, y = volume)) + geom_bar(stat = "identity",
    colour = "black", fill = "grey") + theme_bw(base_size = 12) +
    theme(axis.title.y = element_markdown()) + ylab("Volume m^(3)")
```

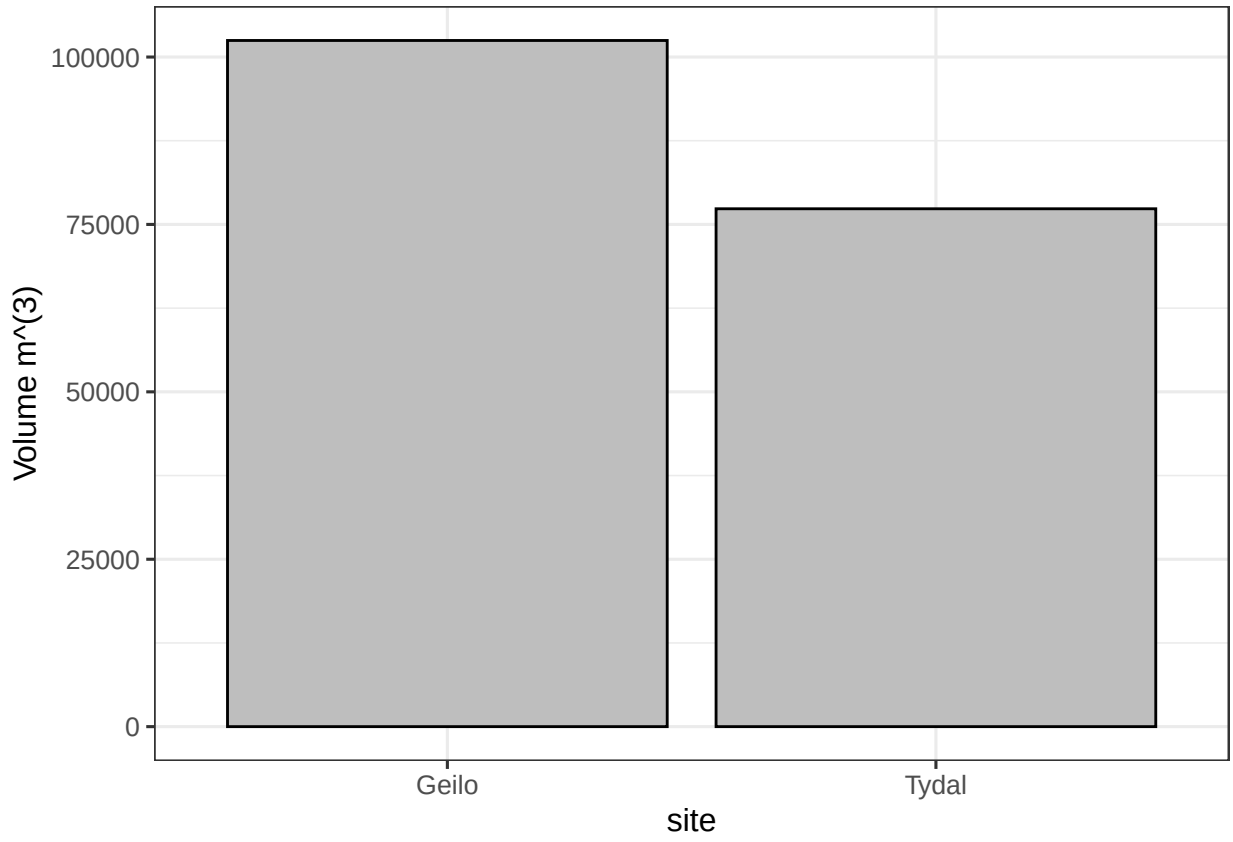


Figure S3.15: Peat volume estimated from IDW.

The rough volume (no iDW) Tydal and Geilo

```
SHP_tydal$myArea <- sf::st_area(SHP_tydal)
mean(depths_tydal$Dybde) * sum(SHP_tydal$myArea)
#> 76994.27 [m^2]
```

```
SHP_geilo$myArea <- sf::st_area(SHP_geilo)
mean(depths_geilo$Dybde) * sum(SHP_geilo$myArea)
#> 98690.12 [m^2]
```

Next I want to try and estimate how many depth measurement I need to get a stable result, both on therms of peat volume and of visual appearance. Then I will also see if the same choice of the power parameter is still the best after removing some data points.

Chapter 4

Optimise sampling intensity

Now I want to resample and reduce the dataset, but try to keep the original sampling design. For the Geilo dataset I want to limit this part of the analyses to only consider one polygon. It is obvious that a peat depth measurement from a neighboring mire cannot replace data collection on the actual mire.

4.1 Subset Geilo data

```
# select a single mire polygon
SHP_geilo_biggest <- SHP_geilo[which.max(SHP_geilo$AREAL),
    ]

# drop the depth measurements from the other
# polygons
depths_geilo_biggest <- sf::st_intersection(depths_geilo,
    SHP_geilo_biggest)

# Crop the raster grid
grid_Geilo_stars_crop_biggest <- sf::st_crop(grid_Geilo_stars,
    SHP_geilo_biggest)
```

```
plot(depths_geilo_biggest$Dybde[order(depths_geilo_biggest$Dybde)],
    ylab = "Peat depth (m)")
```

Looking at the depth measurements in Fig. 4.1, we are reminded that the data contains zeros (measurements taken near the edge) and are truncated above 3 (because that was the length of the peat depth probe).

```
tm_shape(SHP_geilo_biggest) + tm_polygons() + tm_shape(depths_geilo_biggest) +
    tm_symbols(col = "black", shape = 4)
```

4.2 Repeat the best IDW

Here I simply recalculate the best models, as described in the previous chapter

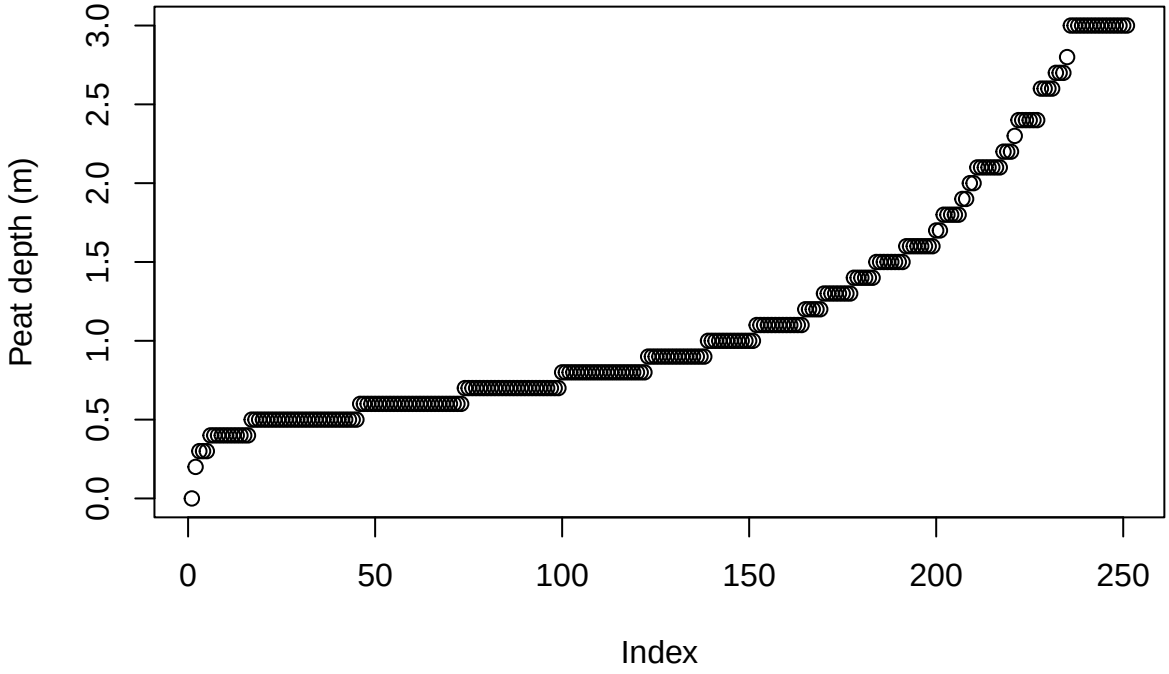


Figure S4.1: Peat depth measurements for the biggest of the Geilo mires.

Figure S4.2: One mire polygon from the Geilo test site used in the assessment of minimum sampling.

```
IDW_tydal_4 <- gstat::idw(formula = Dybde ~ 1, locations = depths_tydal,
    newdata = grid_Tydal_stars_crop, idp = 4, nmax = nmax)
#> [inverse distance weighted interpolation]

IDW_geilo_3 <- gstat::idw(formula = Dybde ~ 1, locations = depths_geilo_biggest,
    newdata = grid_Geilo_stars_crop_biggest, idp = 3,
    nmax = nmax)
#> [inverse distance weighted interpolation]
```

4.3 Reduce N

- by gradually removing the closest point.

First we need to set up some temporary files for the for-loop.

```
distMatrix_tydal <- sf::st_distance(depths_tydal)
distMatrix_tydal <- units::drop_units(distMatrix_tydal)
distMatrix_tydal[distMatrix_tydal == 0] <- NA
distMin_tydal <- matrixStats::rowMins(distMatrix_tydal,
    na.rm = T)

distMatrix_geilo <- sf::st_distance(depths_geilo_biggest)
distMatrix_geilo <- units::drop_units(distMatrix_geilo)
distMatrix_geilo[distMatrix_geilo == 0] <- NA
distMin_geilo <- matrixStats::rowMins(distMatrix_geilo,
    na.rm = T)
```

Create temporary working files

```
distMin_tydal_temp <- distMin_tydal
depths_tydal_temp <- depths_tydal

distMin_geilo_temp <- distMin_geilo
depths_geilo_temp <- depths_geilo_biggest
```

Get the summed volume

```
est_tydal <- sum(IDW_tydal_4$var1.pred, na.rm = T)
est_geilo <- sum(IDW_geilo_3$var1.pred, na.rm = T)
```

Get the median, mean and skewness of the distances to the closest neighbor

```
medianDist_tydal <- median(distMin_tydal)
meanDist_tydal <- mean(distMin_tydal)
skewDist_tydal <- e1071::skewness(distMin_tydal)

medianDist_geilo <- median(distMin_geilo)
meanDist_geilo <- mean(distMin_geilo)
skewDist_geilo <- e1071::skewness(distMin_geilo)
```

Get the MAE

```
MAE_tydal_allData <- krige.cv(Dybde ~ 1, depths_tydal,
    set = list(idp = 4), nmax = nmax)
#>   |                                                                      |
MAE_tydal_allData <- mean(abs(MAE_tydal_allData$residual))

MAE_geilo_allData <- krige.cv(Dybde ~ 1, depths_geilo_biggest,
    set = list(idp = 2), nmax = nmax)
#>   |                                                                      |
MAE_geilo_allData <- mean(abs(MAE_geilo_allData$residual))
```

Put it into a dataframe

```
summaryTable_tydal <- data.frame(medianDist = medianDist_tydal,
    meanDist = meanDist_tydal, skewness = skewDist_tydal,
    n = length(distMin_tydal), estimatedPeatVolume_m3 = est_tydal,
    MAE = MAE_tydal_allData)

summaryTable_geilo <- data.frame(medianDist = medianDist_geilo,
    meanDist = meanDist_geilo, skewness = skewDist_geilo,
    n = length(distMin_geilo), estimatedPeatVolume_m3 = est_geilo,
    MAE = MAE_geilo_allData)
```

Perform a for-loop to gradually remove points based on how close they are to other points. This will hopefully retain most of the systematic design of the points (avoiding clumping of points).

```
# I will not bother going lower than ten data
# points
for (i in 1:c(nrow(depths_tydal) - 10)) {

    print(paste("run number = ", i))

    toRemove <- which(distMin_tydal_temp == min(distMin_tydal_temp))[1]
    print(paste("Removing row number ", toRemove))

    # get some stats distance between neighbours
    medianDist <- median(distMin_tydal_temp)
    meanDist <- mean(distMin_tydal_temp)
    skewDist <- e1071::skewness(distMin_tydal_temp)

    depths_tydal_temp <- depths_tydal_temp[-toRemove,
        ]

    # perform interpolation on tempDF
    int <- gstat::idw(Dybde ~ 1, depths_tydal_temp,
        newdata = grid_Tydal_stars_crop, nmax = nmax,
        idp = 4)

    # Export some predictions for checks
    if (i %in% seq(30, 90, 30)) {
        assign(paste0("IDW_tydal_i", i), int)
    }

    est <- sum(int$var1.pred, na.rm = T)

    # get the MAE as well, even though it takes a
```

```
    # long time to run
    MAE <- krige.cv(Dybde ~ 1, depths_tydal_temp, set = list(idp = 4),
        nmax = nmax)
    MAE <- mean(abs(MAE$residual))

    # paste into the summary table
    summaryTable_tydal[1 + i, "medianDist"] <- medianDist
    summaryTable_tydal[1 + i, "meanDist"] <- meanDist
    summaryTable_tydal[1 + i, "skewness"] <- skewDist
    summaryTable_tydal[1 + i, "n"] <- length(distMin_tydal_temp) -
        1
    summaryTable_tydal[1 + i, "estimatedPeatVolume_m3"] <- est
    summaryTable_tydal[1 + i, "MAE"] <- MAE

    # prepare dataset for next loop
    euclidDist <- sf::st_distance(depths_tydal_temp)
    euclidDist <- units::drop_units(euclidDist)
    euclidDist[euclidDist == 0] <- NA
    distMin_tydal_temp <- rowMins(euclidDist, na.rm = T)

}
saveRDS(summaryTable_tydal, "Data/Tydal/tydal_cvAnalysisData.rds")

saveRDS(IDW_tydal_i30, "Output/Tydal/IDW_tydal_i30.rds")
saveRDS(IDW_tydal_i60, "Output/Tydal/IDW_tydal_i60.rds")
saveRDS(IDW_tydal_i90, "Output/Tydal/IDW_tydal_i90.rds")
```

```
# now doing the same for the Geilo test site
for (i in 1:c(nrow(depths_geilo_biggest) - 10)) {

    print(paste("run number = ", i))

    toRemove <- which(distMin_geilo_temp == min(distMin_geilo_temp))[1]
    print(paste("Removing row number ", toRemove))

    # get some stats distance between neighbours
    medianDist <- median(distMin_geilo_temp)
    meanDist <- mean(distMin_geilo_temp)
    skewDist <- e1071::skewness(distMin_geilo_temp)

    depths_geilo_temp <- depths_geilo_temp[-toRemove,
        ]

    # perform interpolation on tempDF
    int <- gstat::idw(Dybde ~ 1, depths_geilo_temp,
        newdata = grid_Geilo_stars_crop_biggest, nmax = nmax,
        idp = 3)

    # Export some predictions for checks
    if (i %in% seq(40, 240, 50)) {
        assign(paste0("IDW_geilo_i", i), int)
    }

    est <- sum(int$var1.pred, na.rm = T)

    # get the MAE as well, even though it takes a
```

```
    # long time to run
    MAE <- krige.cv(Dybde ~ 1, depths_geilo_temp, set = list(idp = 3),
        nmax = nmax)
    MAE <- mean(abs(MAE$residual))

    # paste into the summary table
    summaryTable_geilo[1 + i, "medianDist"] <- medianDist
    summaryTable_geilo[1 + i, "meanDist"] <- meanDist
    summaryTable_geilo[1 + i, "skewness"] <- skewDist
    summaryTable_geilo[1 + i, "n"] <- length(distMin_geilo_temp) -
        1
    summaryTable_geilo[1 + i, "estimatedPeatVolume_m3"] <- est
    summaryTable_geilo[1 + i, "MAE"] <- MAE

    # prepare dataset for next loop
    euclidDist <- sf::st_distance(depths_geilo_temp)
    euclidDist <- units::drop_units(euclidDist)
    euclidDist[euclidDist == 0] <- NA
    distMin_geilo_temp <- rowMins(euclidDist, na.rm = T)

}
saveRDS(summaryTable_geilo, "Data/Geilo/geilo_cvAnalysisData.rds")

saveRDS(IDW_geilo_i40, "Output/Geilo/IDW_geilo_i40.rds")
saveRDS(IDW_geilo_i90, "Output/Geilo/IDW_geilo_i90.rds")
saveRDS(IDW_geilo_i140, "Output/Geilo/IDW_geilo_i140.rds")
saveRDS(IDW_geilo_i190, "Output/Geilo/IDW_geilo_i190.rds")
saveRDS(IDW_geilo_i240, "Output/Geilo/IDW_geilo_i240.rds")
```

Import the data back in

```
summaryTable_tydal <- readRDS("Data/Tydal/tydal_cvAnalysisData.rds")
summaryTable_geilo <- readRDS("Data/Geilo/geilo_cvAnalysisData.rds")
```

A code block just for getting the reduced datasets

```
distMin_tydal_temp <- distMin_tydal
depths_tydal_temp <- depths_tydal

for (i in 1:c(nrow(depths_tydal) - 10)) {

    # print(paste('run number = ', i))

    toRemove <- which(distMin_tydal_temp == min(distMin_tydal_temp))[1]
    # print(paste('Removing row number ',
    # toRemove))

    depths_tydal_temp <- depths_tydal_temp[-toRemove,
        ]

    # Export some data set for checks
    if (i %in% seq(30, 90, 10)) {
        assign(paste0("depths_tydal_i", i), depths_tydal_temp)
    }
```

```
    # prepare dataset for next loop
    euclidDist <- sf::st_distance(depths_tydal_temp)
    euclidDist <- units::drop_units(euclidDist)
    euclidDist[euclidDist == 0] <- NA
    distMin_tydal_temp <- rowMins(euclidDist, na.rm = T)

}



#### Geilo

distMin_geilo_temp <- distMin_geilo
depths_geilo_temp <- depths_geilo_biggest

for (i in 1:c(nrow(depths_geilo_biggest) - 10)) {

    # print(paste('run number = ', i))

    toRemove <- which(distMin_geilo_temp == min(distMin_geilo_temp))[1]
    # print(paste('Removing row number ',
    # toRemove))

    depths_geilo_temp <- depths_geilo_temp[-toRemove,
        ]

    # Export some data set for checks
    if (i %in% c(1, 50, 100, 200)) {
        assign(paste0("depths_geilo_i", i), depths_geilo_temp)
    }

    # prepare dataset for next loop
    euclidDist <- sf::st_distance(depths_geilo_temp)
    euclidDist <- units::drop_units(euclidDist)
    euclidDist[euclidDist == 0] <- NA
    distMin_geilo_temp <- rowMins(euclidDist, na.rm = T)

}
```

```
tmap_arrange(tm_shape(SHP_tydal) + tm_polygons() +
    tm_shape(depths_tydal) + tm_symbols(col = "black",
    shape = 4, size = 0.5) + tm_layout(title = "All data points",
    inner.margins = c(0.1, 0.02, 0.1, 0.02)), tm_shape(SHP_tydal) +
    tm_polygons() + tm_shape(depths_tydal_i30) + tm_symbols(col = "black",
    shape = 4, size = 0.5) + tm_layout(title = "-30 data points",
    inner.margins = c(0.1, 0.02, 0.1, 0.02)), tm_shape(SHP_tydal) +
    tm_polygons() + tm_shape(depths_tydal_i60) + tm_symbols(col = "black",
    shape = 4, size = 0.5) + tm_layout(title = "-60 data points",
    inner.margins = c(0.1, 0.02, 0.1, 0.02)), tm_shape(SHP_tydal) +
    tm_polygons() + tm_shape(depths_tydal_i80) + tm_symbols(col = "black",
    shape = 4, size = 0.5) + tm_layout(title = "-80 data points",
    inner.margins = c(0.1, 0.02, 0.1, 0.02)))
```

```
tmap_arrange(tm_shape(SHP_geilo_biggest) + tm_polygons() +
    tm_shape(depths_geilo_biggest) + tm_symbols(col = "black",
```

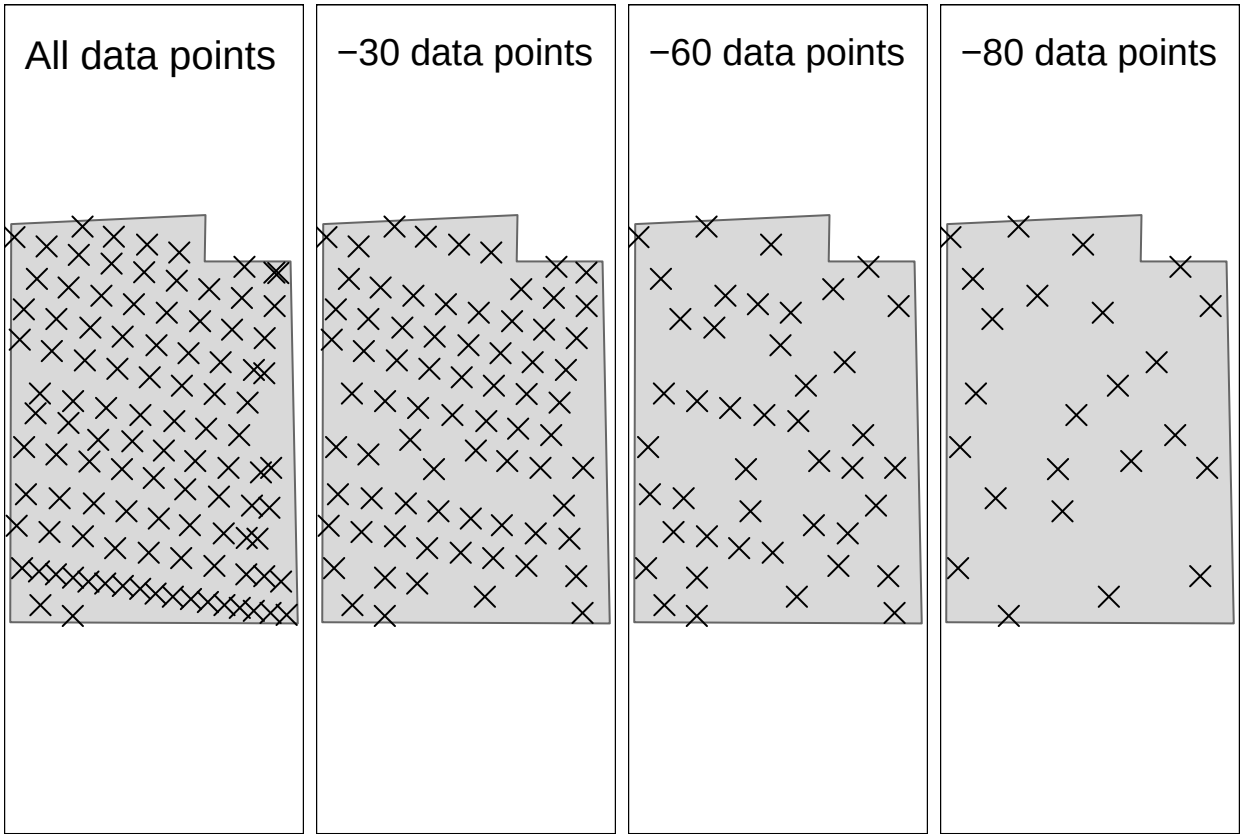


Figure S4.3: Demonstrating the gradual reduction in sampling points. after removing 80 data points, the median distance between the points is 28 meters. Test site: Tydal

```
    shape = 4, size = 0.5) + tm_layout(title = "All data points",
    inner.margins = c(0.1, 0.02, 0.1, 0.02)), tm_shape(SHP_geilo_biggest) +
    tm_polygons() + tm_shape(depths_geilo_i50) + tm_symbols(col = "black",
    shape = 4, size = 0.5) + tm_layout(title = "-50 data points",
    inner.margins = c(0.1, 0.02, 0.1, 0.02)), tm_shape(SHP_geilo_biggest) +
    tm_polygons() + tm_shape(depths_geilo_i100) + tm_symbols(col = "black",
    shape = 4, size = 0.5) + tm_layout(title = "-100 data points",
    inner.margins = c(0.1, 0.02, 0.1, 0.02)), tm_shape(SHP_geilo_biggest) +
    tm_polygons() + tm_shape(depths_geilo_i200) + tm_symbols(col = "black",
    shape = 4, size = 0.5) + tm_layout(title = "-200 data points",
    inner.margins = c(0.1, 0.02, 0.1, 0.02)))
```

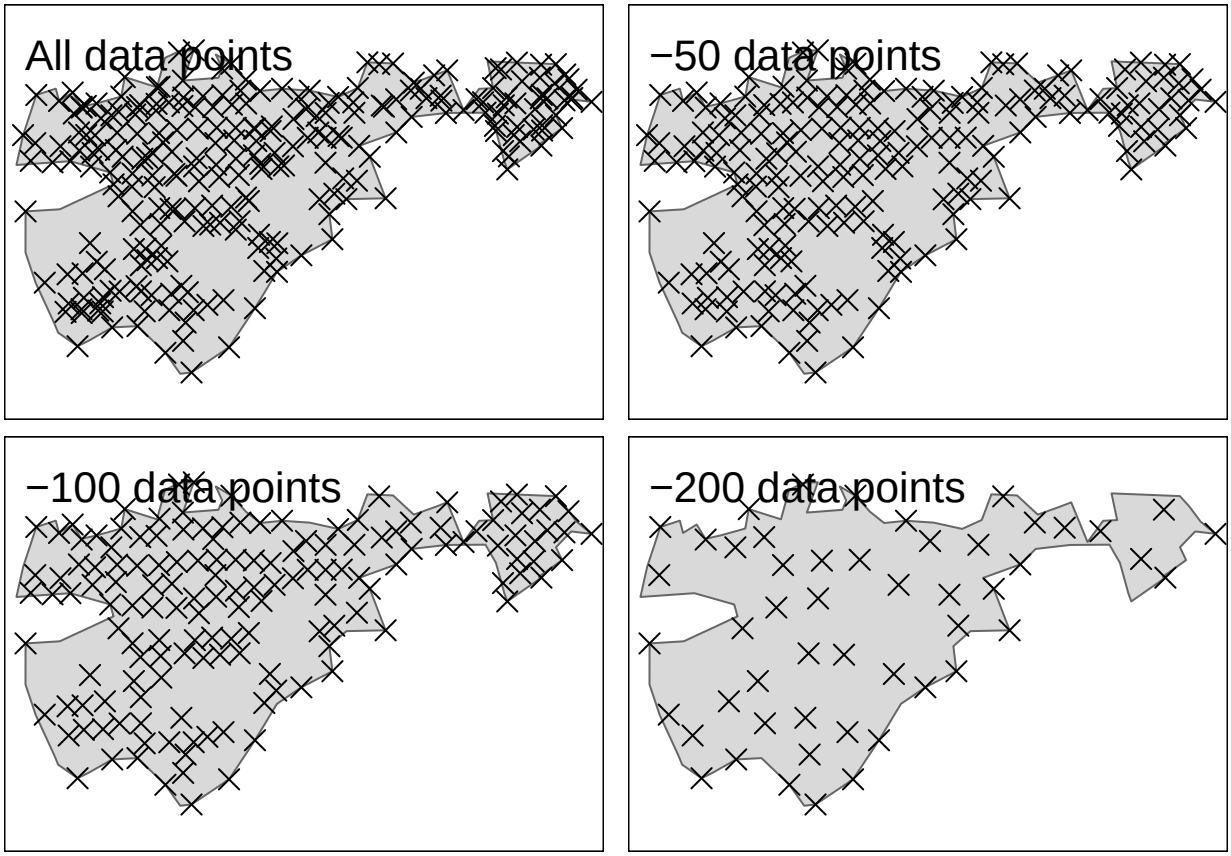


Figure S4.4: Demonstrating the gradual reduction in sampling points. after removinf 200 data poins, the median distance between points is 19 meters. Test site: Geilo

4.4 Explore results

4.4.1 Sampling design

```
IDW_tydal_i30 <- readRDS("Output/Tydal/IDW_tydal_i30.rds")
IDW_tydal_i60 <- readRDS("Output/Tydal/IDW_tydal_i60.rds")
IDW_tydal_i90 <- readRDS("Output/Tydal/IDW_tydal_i90.rds")
```

```
tmap_arrange(tm_shape(IDW_tydal_i30) + tm_raster(col = "var1.pred",
    palette = "-viridis", title = "Interpolated peat\ndepth (m)"),
    tm_shape(IDW_tydal_i60) + tm_raster(col = "var1.pred",
        palette = "-viridis", title = "Interpolated peat\ndepth (m)"),
    tm_shape(IDW_tydal_i90) + tm_raster(col = "var1.pred",
        palette = "-viridis", title = "Interpolated peat\ndepth (m)"))
```

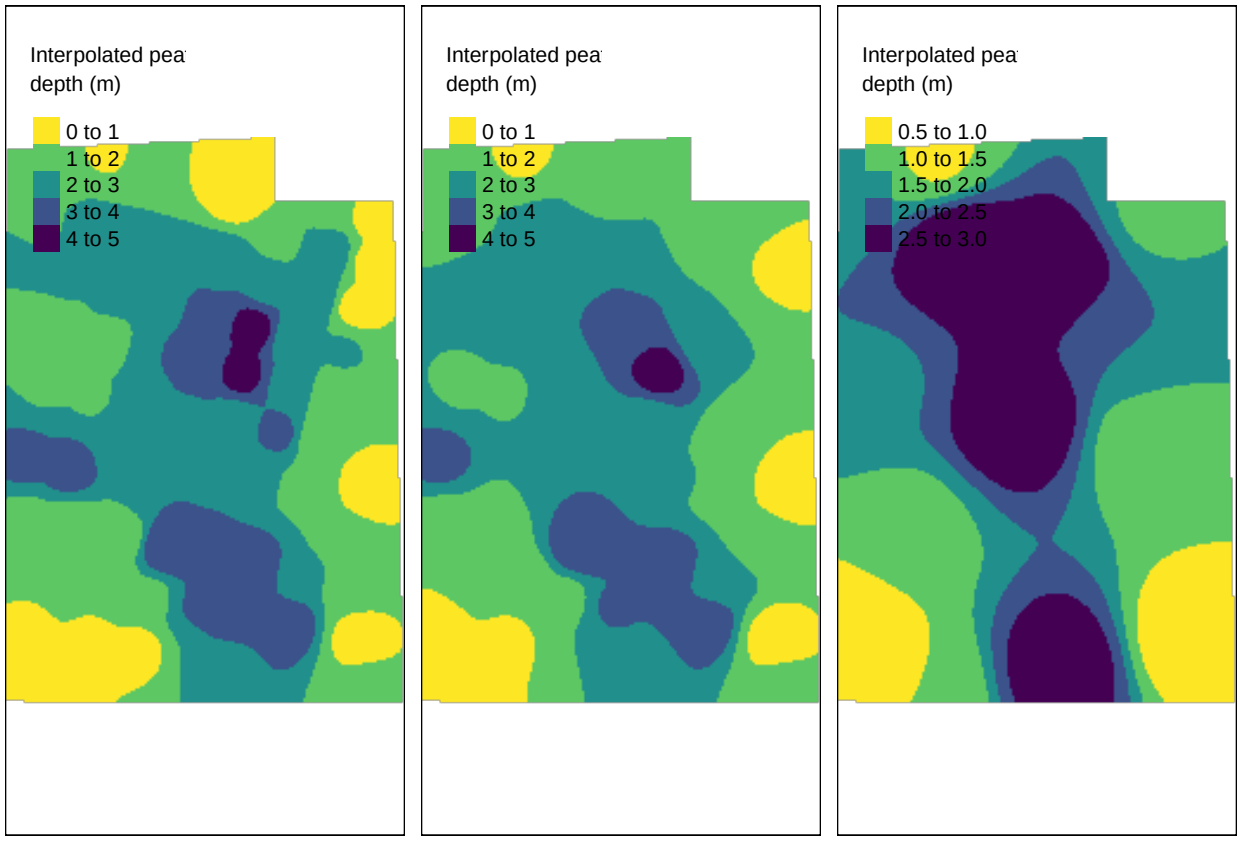


Figure S4.5: IDW with mean distance between data points fromleft to right: 19.4, 20.2 and 40 m. Test site: Tydal

```
IDW_geilo_i90 <- readRDS("Output/Geilo/IDW_geilo_i90.rds")
IDW_geilo_i140 <- readRDS("Output/Geilo/IDW_geilo_i140.rds")
IDW_geilo_i190 <- readRDS("Output/Geilo/IDW_geilo_i190.rds")
IDW_geilo_i240 <- readRDS("Output/Geilo/IDW_geilo_i240.rds")
```

```
tmap_arrange(tm_shape(IDW_geilo_i90) + tm_raster(col = "var1.pred",
    palette = "-viridis", title = "Interpolated peat\ndepth (m)"),
    tm_shape(IDW_geilo_i140) + tm_raster(col = "var1.pred",
        palette = "-viridis", title = "Interpolated peat\ndepth (m)"),
    tm_shape(IDW_geilo_i190) + tm_raster(col = "var1.pred",
        palette = "-viridis", title = "Interpolated peat\ndepth (m)"),
    tm_shape(IDW_geilo_i240) + tm_raster(col = "var1.pred",
        palette = "-viridis", title = "Interpolated peat\ndepth (m)"))
```

4.4.2 Distance to neighbors

Combine datasets

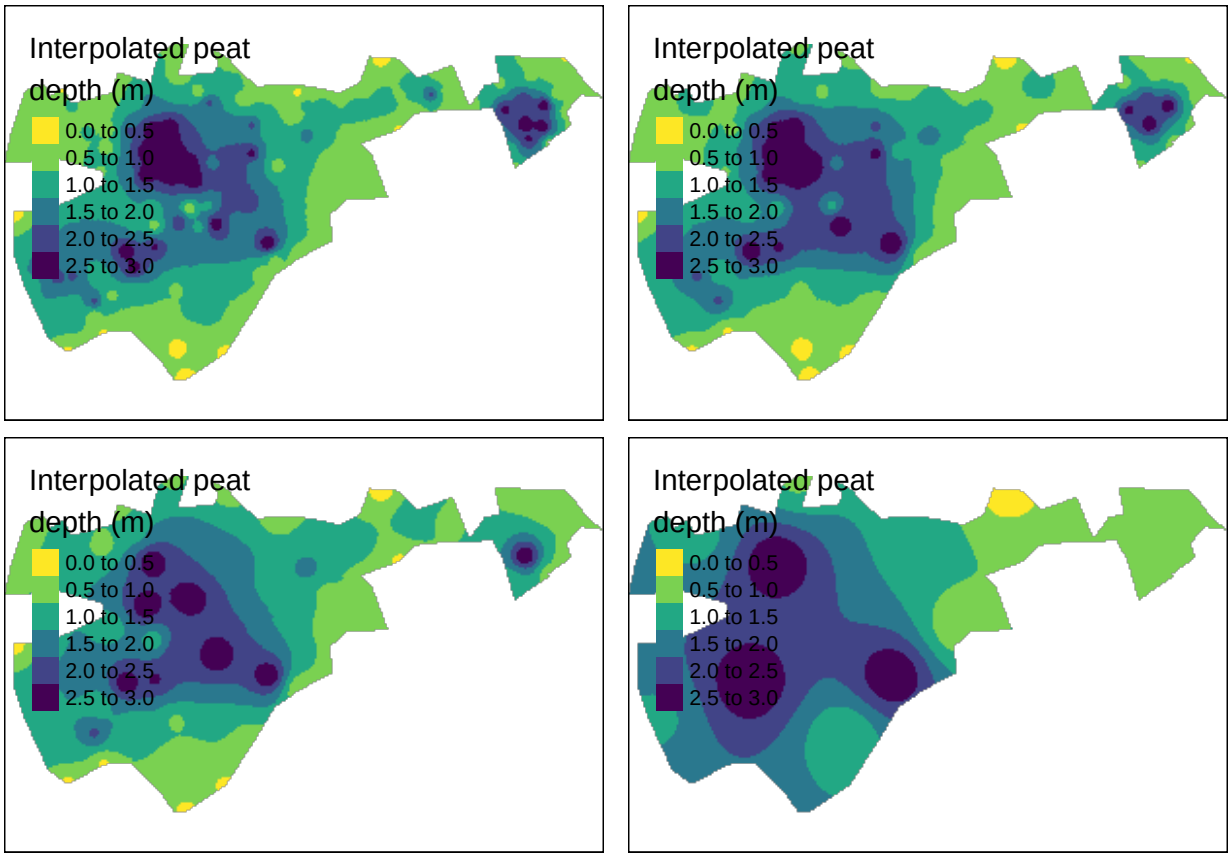


Figure S4.6: IDW with mean distance between data points from left to right, row by row: 9.1, 11.0, 17.3 and 44.5 m. Test site: Geilo

```
summaryTable_tydal$site <- "Tydal"
summaryTable_geilo$site <- "Geilo"
summaryTable_twoSites <- rbind(summaryTable_tydal,
    summaryTable_geilo)
```

```
ggplot(data = summaryTable_twoSites) + geom_point(aes(x = n,
    y = medianDist)) + geom_point(aes(x = n, y = meanDist),
    pch = 21, fill = "grey") + theme_bw(base_size = 16) +
    ylab("Distance to nearest neighbour (m)") + ylim(0,
    60) + facet_wrap(vars(site), scales = "free_x")
```

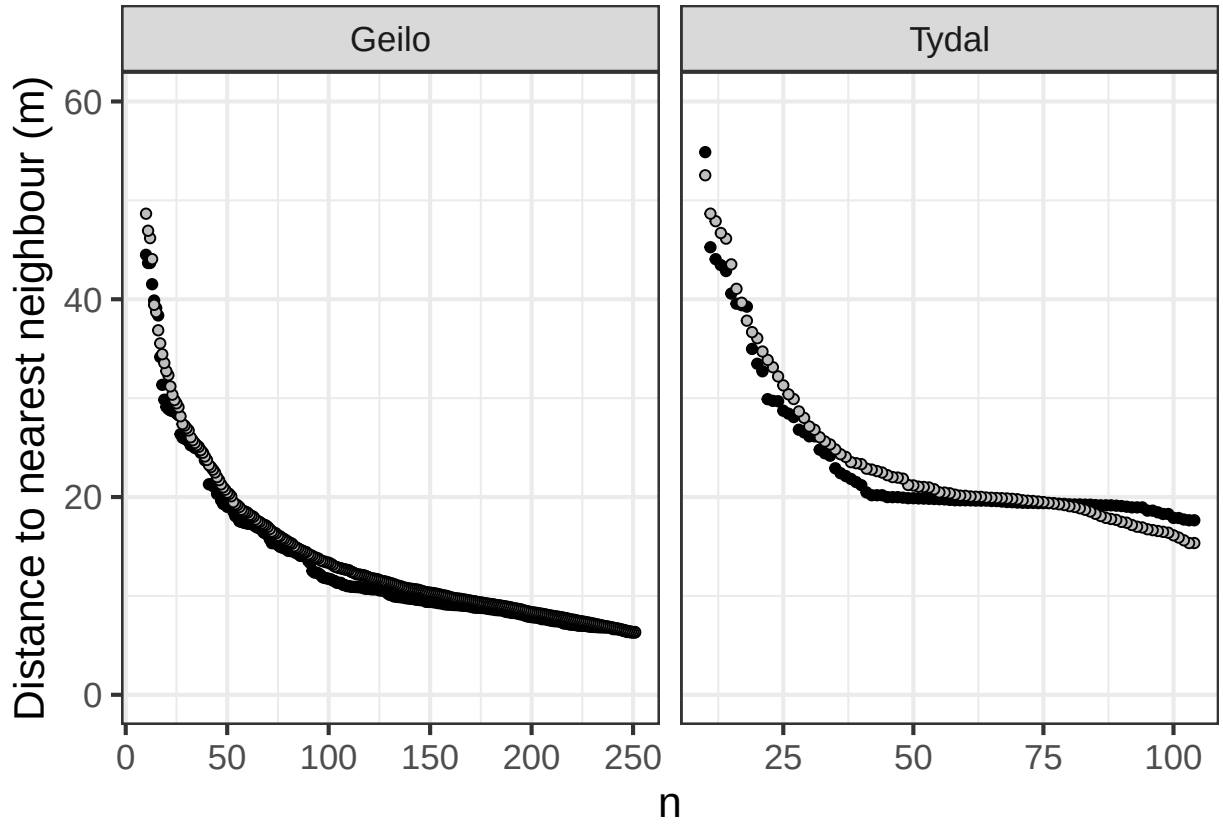


Figure S4.7: Mean (grey) and median (black) distance between peat depth measurements along a gradient of sampling intensity.

This figure tells us that the median (black) and the mean (grey) distance to nearest neighbor are very similar, meaning there is little skew, meaning again that the n reduction process has retained the systematic sampling design.

4.4.3 Skewness

```
ggplot(data = summaryTable_twoSites) + geom_point(aes(x = n,
    y = skewness), pch = 21, fill = "grey") + theme_bw(base_size = 16) +
    ylab("Skewness") + facet_wrap(vars(site), scales = "free_x")
```

Contrary to the above, this figure indicates perhaps that the sampling designs is compromised when I reduce n. It starts out with a negative skew due to some points being very close together. There are then 'weeded out' first, and

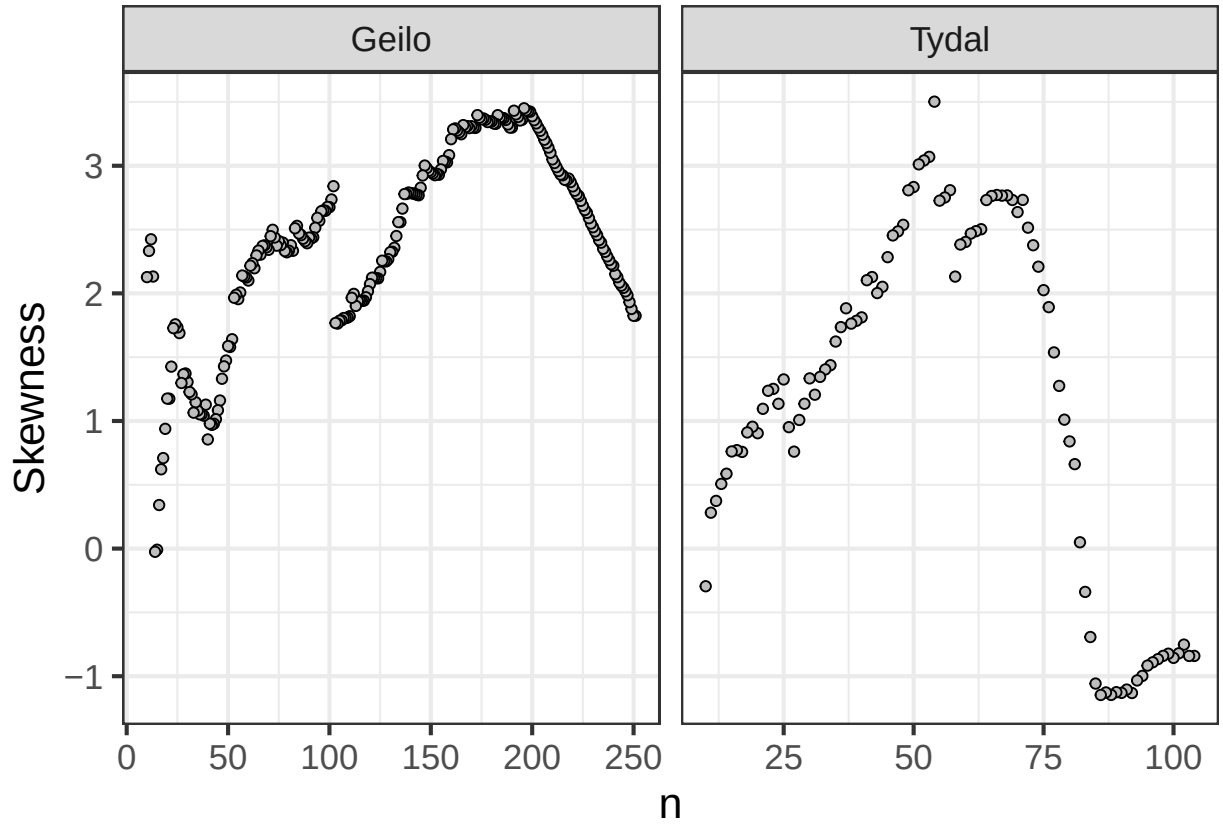


Figure S4.8: Skewness in the distribution of euclidian distances as a response to varying sapling intensity.

the skew is flipped around to become positive. I will mostly rely on figure 4.3 and 4.4 and say that the reduction in sampling intensity retains the systematic design.

4.4.4 Peat volume estimates

```
gg_volume <- ggplot(data = summaryTable_twoSites, aes(x = medianDist,
    y = estimatedPeatVolume_m3)) + geom_point(pch = 21,
    fill = "grey", size = 2) + theme_bw(base_size = 16) +
    ylab(expression("Est. peat volume (m"^3 * ")")) +
    xlab("Median distance to nearest neighbour (m)") +
    coord_trans(x = "log2") + facet_wrap(vars(site),
    scales = "free")


ggsave("Output/plot_distanceAgainstVolume.tiff", gg_volume,
    width = 1600, height = 1200, units = "px")

gg_volume
```

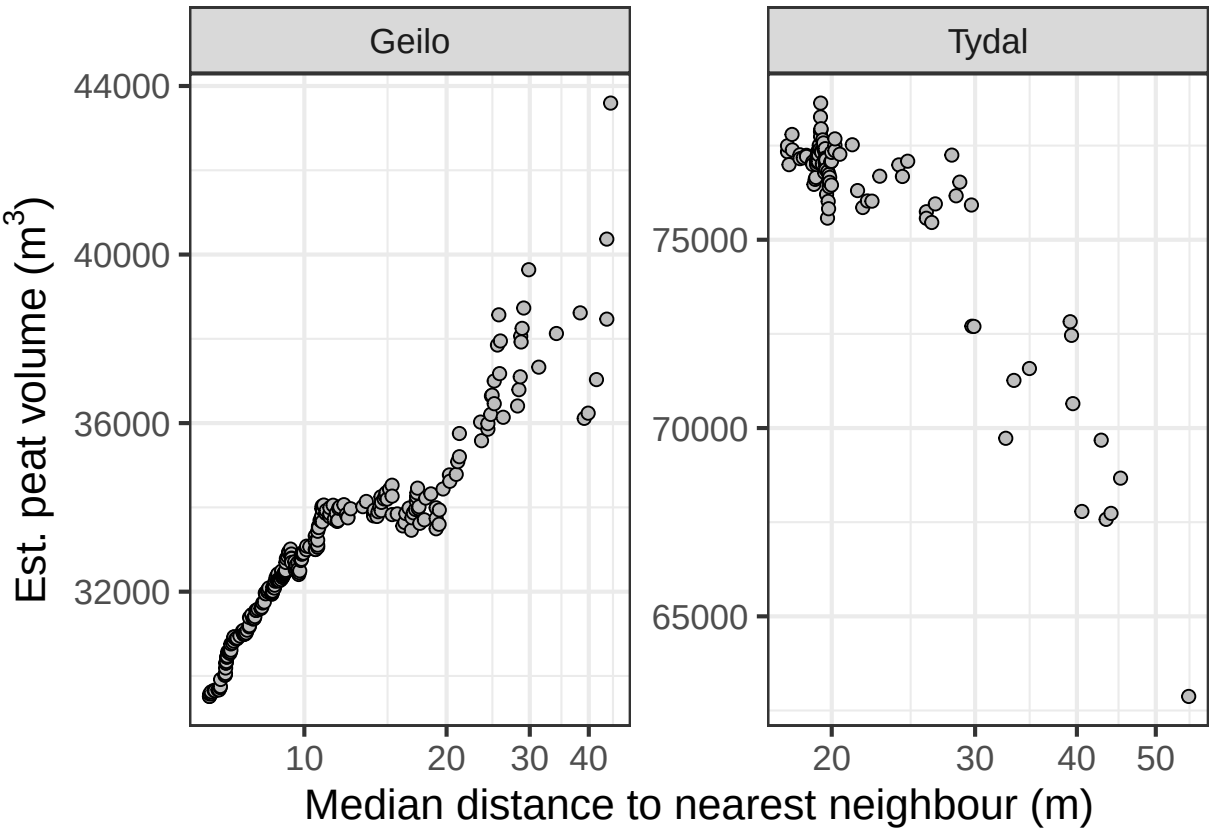


Figure S4.9: Estimated peat volume as a response of the median distance between sampling points.

For Tydal, the estimated peat volume is quite stable up to around 25 meter between sampling points. For Geilo, the estimated volume increases from zero to 10-12 meters, before stabilizing and finally increasing again with distances above 29 meters. The most correct prediction comes with a median distance between 12-20 meters. The original sampling density was biased towards the edges, resulting in a disproportionate large number of samples from shallow

areas (Fig. 4.4). The IDW does not seem to be able to adequately control for this, and care should be taken to ensure a balanced and systematic sampling design.

4.4.5 CV

In addition to seeing the peat volume estimates becoming biased with increasing distance between sampling points, we also we the variation increasing. Now we will calculate this in terms of the coefficient of variation.

```
max(summaryTable_tydal$medianDist)
#> [1] 54.87518
```

```
max(summaryTable_geilo$medianDist)
#> [1] 44.49275
```

```
tydal_cv <- summaryTable_tydal
tydal_cv$group <- ifelse(tydal_cv$medianDist < 10,
    "<10 m", ifelse(tydal_cv$medianDist < 15, "10-15 m",
        ifelse(tydal_cv$medianDist < 20, "15-20 m",
            ifelse(tydal_cv$medianDist < 25, "20-25 m",
                ifelse(tydal_cv$medianDist < 30, "25-30 m",
                  ifelse(tydal_cv$medianDist < 35,
                    "30-35 m", ifelse(tydal_cv$medianDist <
                      40, "35-40 m", "40-55 m")))))))
tydal_cv <- tydal_cv[tydal_cv$medianDist < 100, ]
(tydal_n <- table(tydal_cv$group))
#>
#> 15-20 m 20-25 m 25-30 m 30-35 m 35-40 m 40-55 m
#>      60      13      10       3       3       6
```

There's quite few data points in the higher categories. Perhaps too few.

```
geilo_cv <- summaryTable_geilo
geilo_cv$group <- ifelse(geilo_cv$medianDist < 10,
    "<10 m", ifelse(geilo_cv$medianDist < 15, "10-15 m",
        ifelse(geilo_cv$medianDist < 20, "15-20 m",
            ifelse(geilo_cv$medianDist < 25, "20-25 m",
                ifelse(geilo_cv$medianDist < 30, "25-30 m",
                  ifelse(geilo_cv$medianDist < 35,
                    "30-35 m", ifelse(geilo_cv$medianDist <
                      40, "35-40 m", "40-55 m")))))))
geilo_cv <- geilo_cv[geilo_cv$medianDist < 100, ]
(geilo_n <- table(geilo_cv$group))
#>
#>   <10 m 10-15 m 15-20 m 20-25 m 25-30 m 30-35 m 35-40 m
#>     120      56      29      13      15       2       3
#> 40-55 m
#>       4
```

CV function

```
cv <- function(x) {
    sd(x)/mean(x)
}
```

```
tydal_cv_tbl <- tapply(tydal_cv$estimatedPeatVolume_m3,
    tydal_cv$group, cv)
tydal_cv_tbl <- data.frame(cv = tydal_cv_tbl, label = names(tydal_cv_tbl),
    order = c(3, 4, 5, 6, 7, 8))
tydal_cv_tbl <- tydal_cv_tbl[order(tydal_cv_tbl$order),
    ]
tydal_cv_tbl$n <- tydal_n

geilo_cv_tbl <- tapply(geilo_cv$estimatedPeatVolume_m3,
    geilo_cv$group, cv)
geilo_cv_tbl <- data.frame(cv = geilo_cv_tbl, label = names(geilo_cv_tbl),
    order = c(1, 2, 3, 4, 5, 6, 7, 8))
geilo_cv_tbl <- geilo_cv_tbl[order(geilo_cv_tbl$order),
    ]
geilo_cv_tbl$n <- geilo_n
```

Join tables

```
tydal_cv_tbl$site = "Tydal"
geilo_cv_tbl$site = "Geilo"
cvTab <- rbind(tydal_cv_tbl, geilo_cv_tbl)
```

```
gg_cv <- ggplot(data = cvTab, aes(x = order, y = cv,
    fill = site, shape = site)) + geom_line(lty = 2) +
    scale_shape_manual(values = c(21, 24)) + geom_point(size = 3,
    stroke = 1.5, position = position_dodge(width = 0.2)) +
    theme_bw(base_size = 16) + scale_x_continuous(breaks = cvTab$order,
    labels = cvTab$label) + theme(axis.text.x = element_text(angle = 90)) +
    xlab("")

ggsave("Output/plot_distanceAgainstCV.tiff", gg_cv,
    width = 1600, height = 1200, units = "px")

gg_cv
```

4.5 MAE

```
gg_mae <- ggplot(data = summaryTable_twoSites, aes(x = medianDist,
    y = MAE)) + geom_point(pch = 21, fill = "grey",
    size = 2) + theme_bw(base_size = 16) + ylab("Mean absolute error (m)") +
    xlab("Median distance to nearest neighbour (m)") +
    coord_trans(x = "log2") + facet_wrap(vars(site),
    scales = "free")


ggsave("Output/plot_distanceAgainstMAE.tiff", gg_mae,
    width = 1600, height = 1200, units = "px")

gg_mae
```

This figure I think also support that sampling distances above about 25 m is a bad idea.

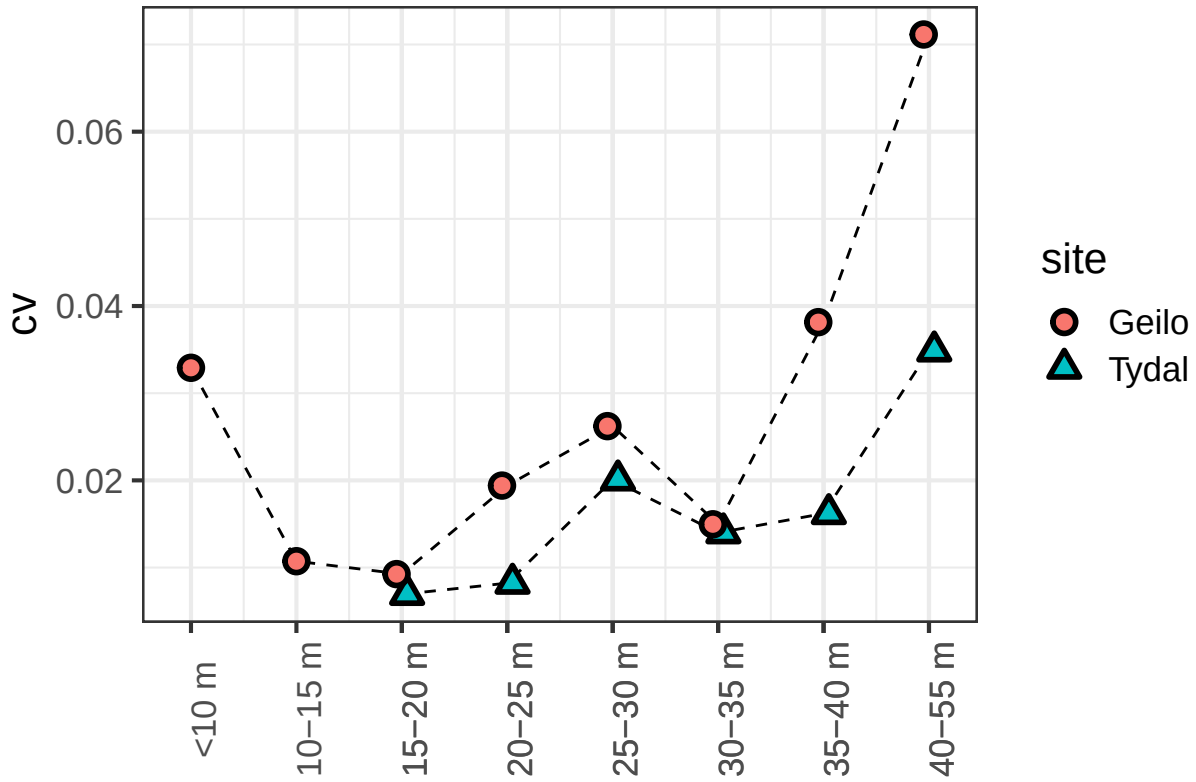


Figure S4.10: Coefficient of variation in peat volume estimates as a response to median distance between sampling points

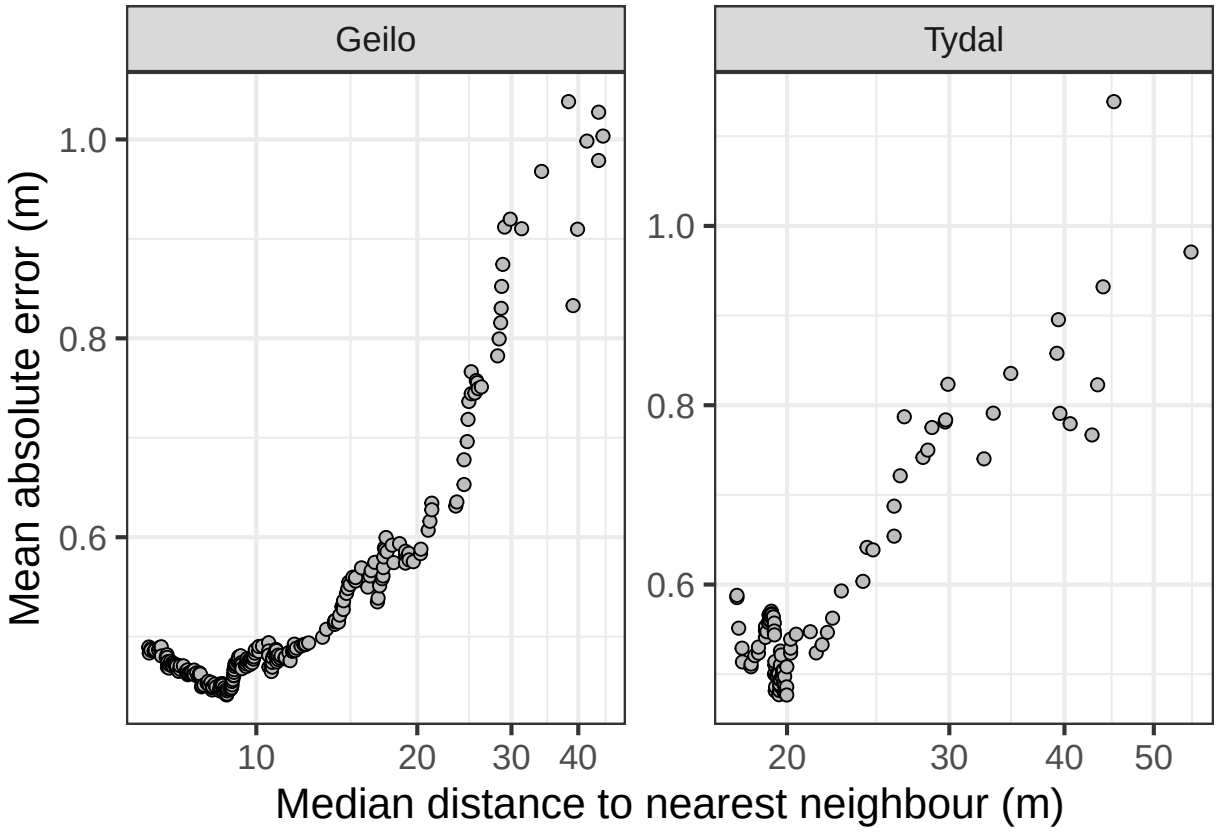


Figure S4.11: MAE in IDW predictions as a response of median distance between sampling points.

Chapter 5

Validate using reduced sampling data

In chapter 2 we found the optimal power settings for the Tydal and Geilo sites. Now I just want to look at how this might have changed when we have removed *superflous* data points as in chapter 3.

Lets say for Tydal that we want to have 20 meters between the sampling points.

```
tm_shape(SHP_tydal) + tm_polygons() + tm_shape(depths_tydal_i60) +
    tm_dots(size = "Dybde", scale = 2, col = "Dybde",
        palette = "-viridis", title = "Measured peat depth (m)") +
    tm_layout(legend.outside = T)
```

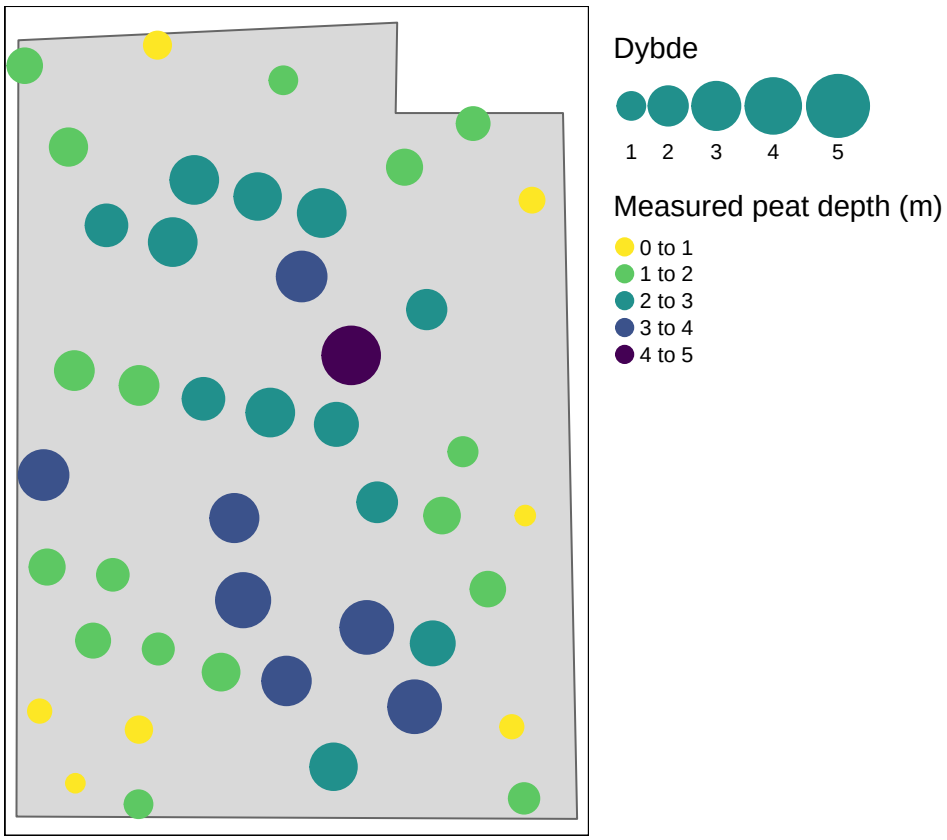


Figure S5.1: Peat depth measurements in the Tydal test site, after trimming the data poins so that the median distance between them is 20 meters.

Confirm the median distance is 20 meters

```
temp <- sf::st_distance(depths_tydal_i60)
temp <- units::drop_units(temp)
temp[temp == 0] <- NA
temp <- rowMins(temp, na.rm = T)
median(temp)
#> [1] 20.18216
```

Then we determine the optimal power setting.

```
ccalc_optimumPower <- function(powerRange = 1:6, nmax = 20,
    peatDepths, peatlandDelimination, title) {

    temp <- data.frame(power = powerRange, MAE = as.numeric(NA))

    vol <- NULL

    myGrid <- starsExtra::make_grid(peatlandDelimination,
        1)
    myGrid <- sf::st_crop(myGrid, peatlandDelimination)

    for (i in powerRange) {

        # Get the MAE
        temp2 <- krige.cv(Dybde ~ 1, peatDepths, set = list(idp = i),
            nmax = nmax)
        temp$MAE[temp$power == i] <- mean(abs(temp2$residual))

        # Get the volume
        vol_temp <- gstat::idw(Dybde ~ 1, peatDepths,
            newdata = myGrid, nmax = nmax, idp = i)

        vol <- c(vol, sum(vol_temp$var1.pred, na.rm = T))
    }

    ifelse(temp$power[which.min(temp$MAE)] < 2, temp$best <- ifelse(temp$power ==
        2, "best", "not-best"), temp$best <- ifelse(temp$MAE ==
        min(temp$MAE), "best", "not-best"))

    # Plot MAE
    gg_out <- ggplot(temp, aes(x = power, y = MAE,
        colour = best, shape = best)) + geom_point(size = 10) +
        theme_bw(base_size = 12) + scale_x_continuous(breaks = powerRange) +
        guides(colour = "none", shape = "none") + scale_color_manual(values = c("darkgreen",
        "grey")) + scale_shape_manual(values = c(18,
        19)) + ggtitle(title)

    # Plot volume
    vol_df <- data.frame(volume = vol, power = powerRange)
    vol_df$relative_volume <- vol_df$volume/mean(vol_df$volume) *
        100

    gg_out_vol <- ggplot(vol_df, aes(x = factor(power),
        y = relative_volume)) + geom_point(size = 8) +
        xlab("power") + ylab("Peat volume as a percentage of\nmean predicted peat volume") +
```

```
        theme_bw(base_size = 12)

    ggpubr::ggarrange(gg_out, gg_out_vol)
}
```

```
ccalc_optimumPower(peatDepths = depths_tydal_i60, title = "Tydal",
    peatlandDelimination = SHP_tydal)
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
```

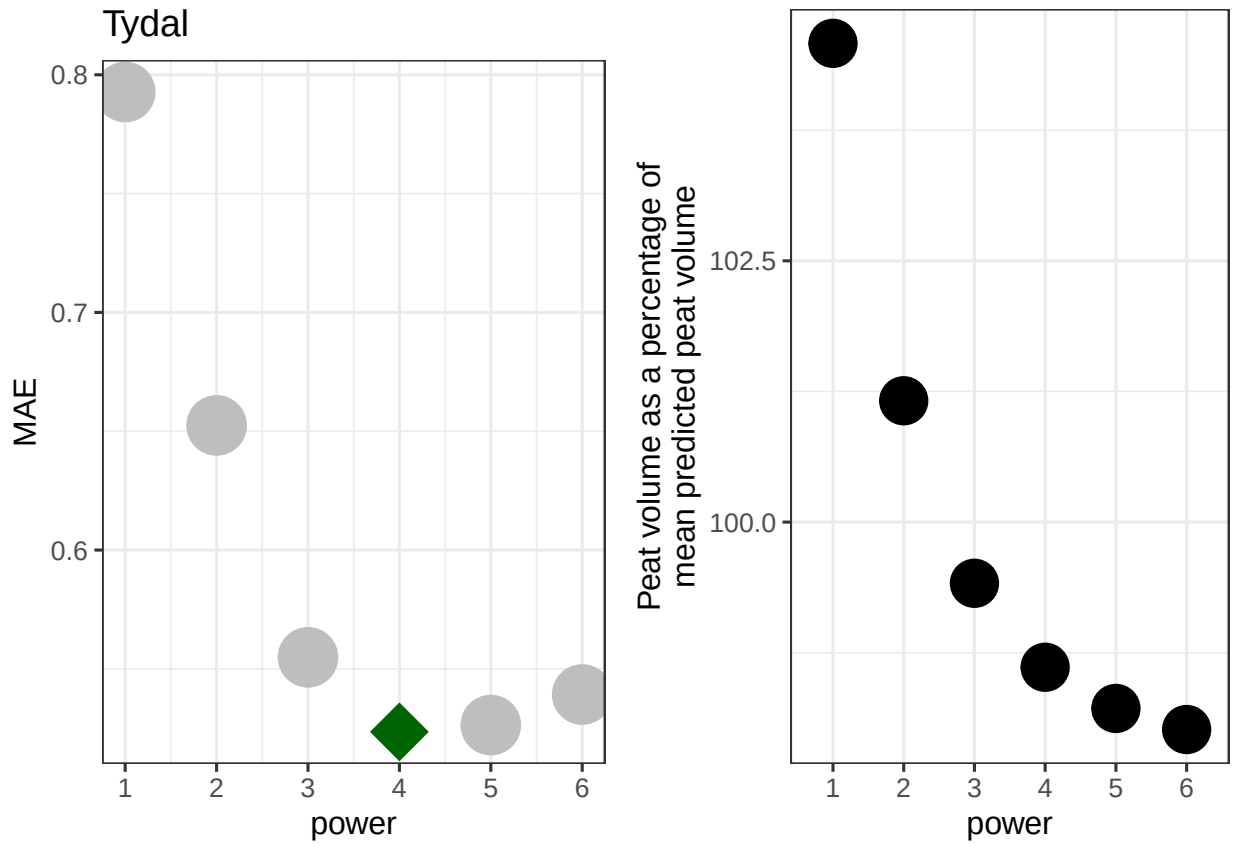


Figure S5.2: Determening the optimal power setting for IDW using a reduced dataset from the Tydal test site.

That's good. The optimum power is the same as when we had all the data points.

Let's plot the IDW predictions for Tydal based on all the data pionts and based on the reduced set of data points.

```
IDW_tydal_4_red <- gstat::idw(formula = Dybde ~ 1,
    locations = depths_tydal_i60, newdata = grid_Tydal_stars_crop,
    idp = 4, nmax = nmax)
#> [inverse distance weighted interpolation]
```

```
tm_tydal_red <- tm_shape(IDW_tydal_4_red) + tm_raster(col = "var1.pred",
    palette = "-viridis", title = "Interpolated peat\ndepth (m)") +
    tm_shape(depths_tydal_i60) + tm_symbols(shape = 4,
    col = "black", size = 0.5) + tm_compass(type = "8star",
    position = c("right", "bottom"), size = 2) + tm_scale_bar(position = c("left",
    "bottom"), width = 0.3) + tm_layout(inner.margins = c(0.15,
    0.05, 0, 0.05), legend.outside = T)

tmap_arrange(tm_tydal, tm_tydal_red)
```

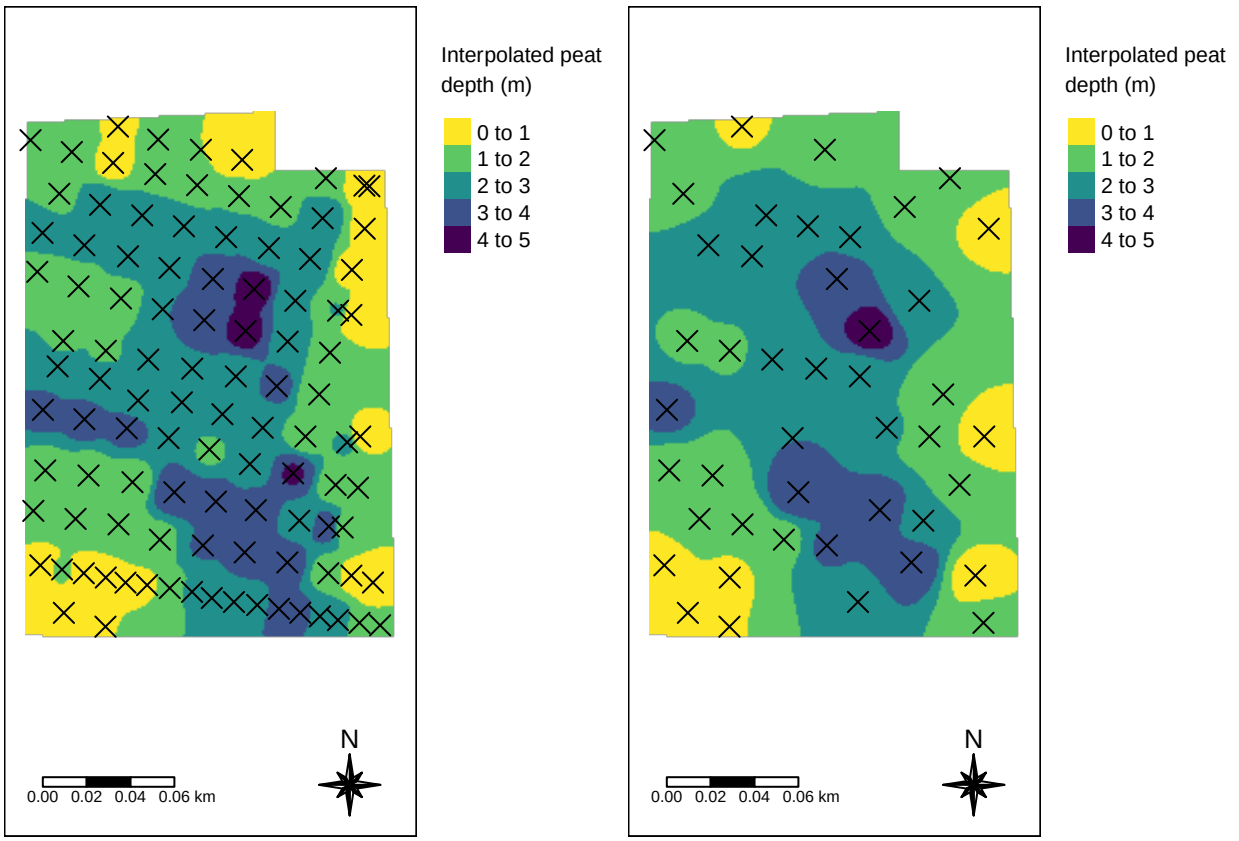


Figure S5.3: Comparing IDW prediction using all data points (left) and a reduced set of data points with median distance between point set to 20 meters.

The two maps in Fig. 5.3 are qualitatively similar.

Chapter 6

Carbon stocks

Now I want to convert the peat volumes into estimates of carbon stock. This conversion depends on the values of some peat characteristics, such as bulk density and C concentrations. We have a data set with these values.

Import dataset with peat characteristics.

```
peatCharacteristics <- read_delim("Data/peatCharacteristics.csv",
    delim = ";", escape_double = FALSE, locale = locale(encoding = "ISO-8859-1"),
    trim_ws = TRUE)
dim(peatCharacteristics)
#> [1] 88 30
```

Each sample consists of multiple sub samples from different depths. Here we will ignore the depth aspect, and simply take te mean from each sample core. This part of the analyses can be improved.

```
peatCharacteristics_summedDepths <- peatCharacteristics %>%
    mutate(perc_SOM = as.numeric(`% SOM`)) %>%
    mutate(BD = as.numeric(`BD (t/m3)`)) %>%
    dplyr::select(`SAMPLE ID2`, perc_SOM, BD) %>%
    group_by(`SAMPLE ID2`) %>%
    summarise(across(.fns = list(mean = ~mean(., na.rm = TRUE))))

# Getting the other variables that I also want to
# keep
df_info <- peatCharacteristics %>%
    dplyr::select(`SAMPLE ID2`, `General Peatland Type`,
        `Specific Peatland Type`) %>%
    group_by(`SAMPLE ID2`) %>%
    unique()
# and join together again (this two-stap
# proceedure could be simplified)
df = full_join(peatCharacteristics_summedDepths, df_info,
    by = "SAMPLE ID2") %>%
    drop_na()
head(df)
#> # A tibble: 6 x 5
#>   `SAMPLE ID2` perc_SOM_mean BD_mean General Peatl~1 Speci~2
#>   <chr>                <dbl>   <dbl> <chr>           <chr>
#> 1 0029                  98.3  0.05   fen             poor f~
#> 2 0030                  99.5  0.102  fen             interm~
#> 3 0031                  98.2  0.0527 bog             bog
```

```
#> 4 0032                  95.9  0.0555 fen             poor f~
#> 5 0033                  96.6  0.067  fen             interm~
#> 6 0034                  99.3  0.066  fen             rich f~
#> # ... with abbreviated variable names
#> #   1: `General Peatland Type`, 2: `Specific Peatland Type`
```

We have 12 samples from bogs, and 14 from fens:

```
table(df$`General Peatland Type`)
#>
#> bog fen
#>  12  14
```

We do not perhaps have enough data points to really compare the different specific peatland types:

```
table(df$`Specific Peatland Type`)
#>
#>              bog intermediate fen      oceanic bog
#>                3                5                2
#>         poor fen       raised bog         rich fen
#>                5                7                4
```

Let's get the total peat volumes for Tydal (and Geilo, but I will not look at Geilo here).

```
volume
#>    site    volume
#> 1 Tydal  77336.97
#> 2 Geilo 102469.96
```

Calculating the c stocks (tons C) for the Tydal site without specifying the mire type

```
c_stock_tydal_uninformed <- volume$volume[volume$site ==
    "Tydal"] * mean(df$perc_SOM_mean/100) * mean(df$BD_mean) *
    0.5  #* 1000 # removing the original conversion from tons to kg
c_stock_tydal_uninformed
#> [1] 3519.665
```

I think we also need to estimate the uncertainties around this number. Note that for the peat volume estimation, we did not calculate an uncertainty. But for the peat characteristics we have an uncertainty in therms of variation within and between samples points. I will calculate the uncertainty between individual peat cores only.

```
c_stock_tydal_uninformed <- NULL
for (i in 1:1000) {
    temp <- volume$volume[volume$site == "Tydal"] *
        mean(sample(df$perc_SOM_mean, replace = T)/100) *
        mean(sample(df$BD_mean, replace = T) * 0.5)

    c_stock_tydal_uninformed <- c(c_stock_tydal_uninformed,
        temp)
}
hist(c_stock_tydal_uninformed, main = "", xlab = "C stock")
```

This distribution is quite wide. Let's summarize the distribution.

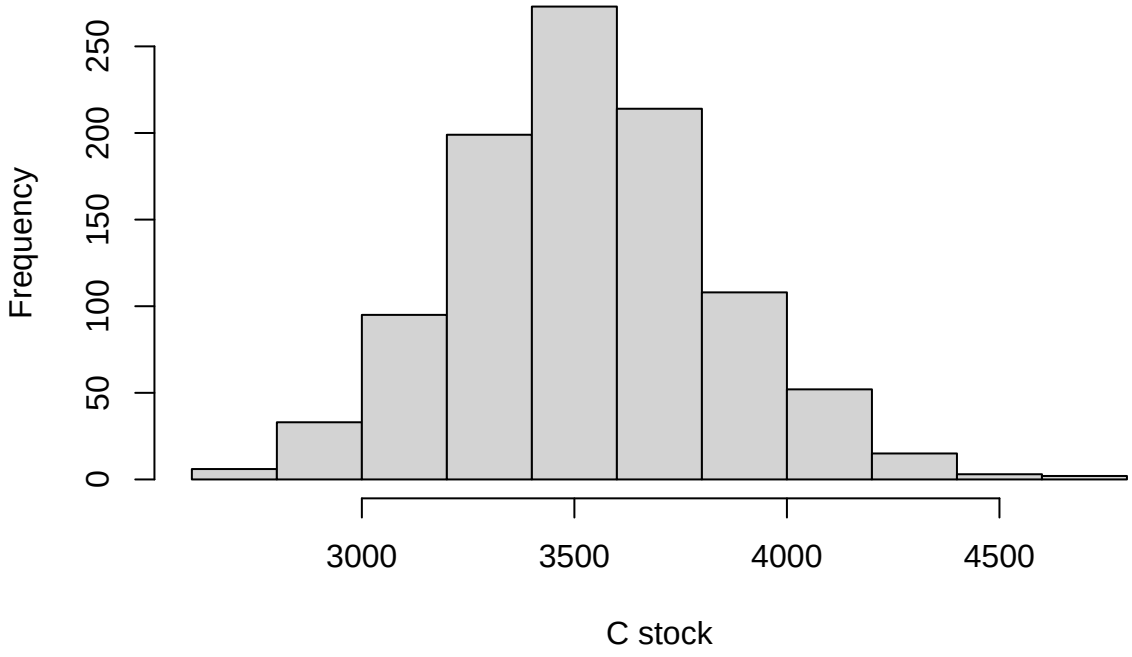


Figure S6.1: Estaimted carbon stock in the Tydal peatland

```
c(quantile(c_stock_tydal_uninformed, c(0.05, 0.5, 0.95)),
    mean = mean(c_stock_tydal_uninformed), sd = sd(c_stock_tydal_uninformed))
#>        5%       50%       95%      mean        sd
#> 3025.9875 3518.3761 4055.6040 3531.1143  307.0405
```

We can put this stuff into a more generic function.

```
ccalc_cStocks <- function(volume, peatlandType = c("fen",
    "bog"), peatData) {
    temp_stocks <- NULL
    temp_peatData <- peatData[peatData$`General Peatland Type` %in%
        peatlandType, ]

    for (i in 1:1000) {
        temp <- volume * mean(sample(peatData$perc_SOM_mean,
            replace = T)/100) * mean(sample(peatData$BD_mean,
            replace = T) * 0.5)

        temp_stocks <- c(temp_stocks, temp)
    }
    return(c(quantile(temp_stocks, c(0.05, 0.5, 0.95)),
        mean = mean(temp_stocks), sd = sd(temp_stocks)))
}
```

Calculating summary statistics for the carbon stocks in Tydal, using both general peatland types.

```
(cstocks_tydal_unspecified <- ccalc_cStocks(volume = volume$volume[volume$site ==
    "Tydal"], peatData = df))
#>        5%       50%       95%      mean        sd
#> 3048.0489 3528.0585 4045.9962 3529.7181  307.5967
```

And using only the *bog* data.

```
(cstocks_tydal_bog <- ccalc_cStocks(volume = volume$volume[volume$site ==
    "Tydal"], peatData = df, peatlandType = "bog"))
#>        5%       50%       95%      mean        sd
#> 2992.8414 3521.1338 4030.4238 3515.2569  311.7378
```

And finally, aslo the fen data.

```
(cstocks_tydal_fen <- ccalc_cStocks(volume = volume$volume[volume$site ==
    "Tydal"], peatData = df, peatlandType = "fen"))
#>        5%       50%       95%      mean        sd
#> 3023.4134 3515.4305 4030.6718 3526.8163  306.9057
```

```
cstocks_tydal_unspecified_df <- as.data.frame(cstocks_tydal_unspecified)
names(cstocks_tydal_unspecified_df) <- "C stocks"
cstocks_tydal_unspecified_df$summary <- row.names(cstocks_tydal_unspecified_df)
cstocks_tydal_unspecified_df$information <- "Unspesified"


cstocks_tydal_bog_df <- as.data.frame(cstocks_tydal_bog)
names(cstocks_tydal_bog_df) <- "C stocks"
cstocks_tydal_bog_df$summary <- row.names(cstocks_tydal_bog_df)
```

```
cstocks_tydal_bog_df$information <- "Bog"

cstocks_tydal_fen_df <- as.data.frame(cstocks_tydal_fen)
names(cstocks_tydal_fen_df) <- "C stocks"
cstocks_tydal_fen_df$summary <- row.names(cstocks_tydal_fen_df)
cstocks_tydal_fen_df$information <- "Fen"

cstocks_tydal_compare <- rbind(cstocks_tydal_fen_df,
    cstocks_tydal_bog_df, cstocks_tydal_unspecified_df)

# Pivot
cstocks_tydal_compare <- pivot_wider(cstocks_tydal_compare,
    names_from = summary, values_from = "C stocks")
```

```
ggplot(cstocks_tydal_compare, aes(x = information)) +
    geom_point(aes(y = mean), shape = 15, size = 10) +
    geom_linerange(aes(ymin = `5%`, ymax = `95%`),
        size = 2) + theme_bw(base_size = 16) + labs(x = "Peat characteristics",
    y = "Carbon stocks (tons)")
```

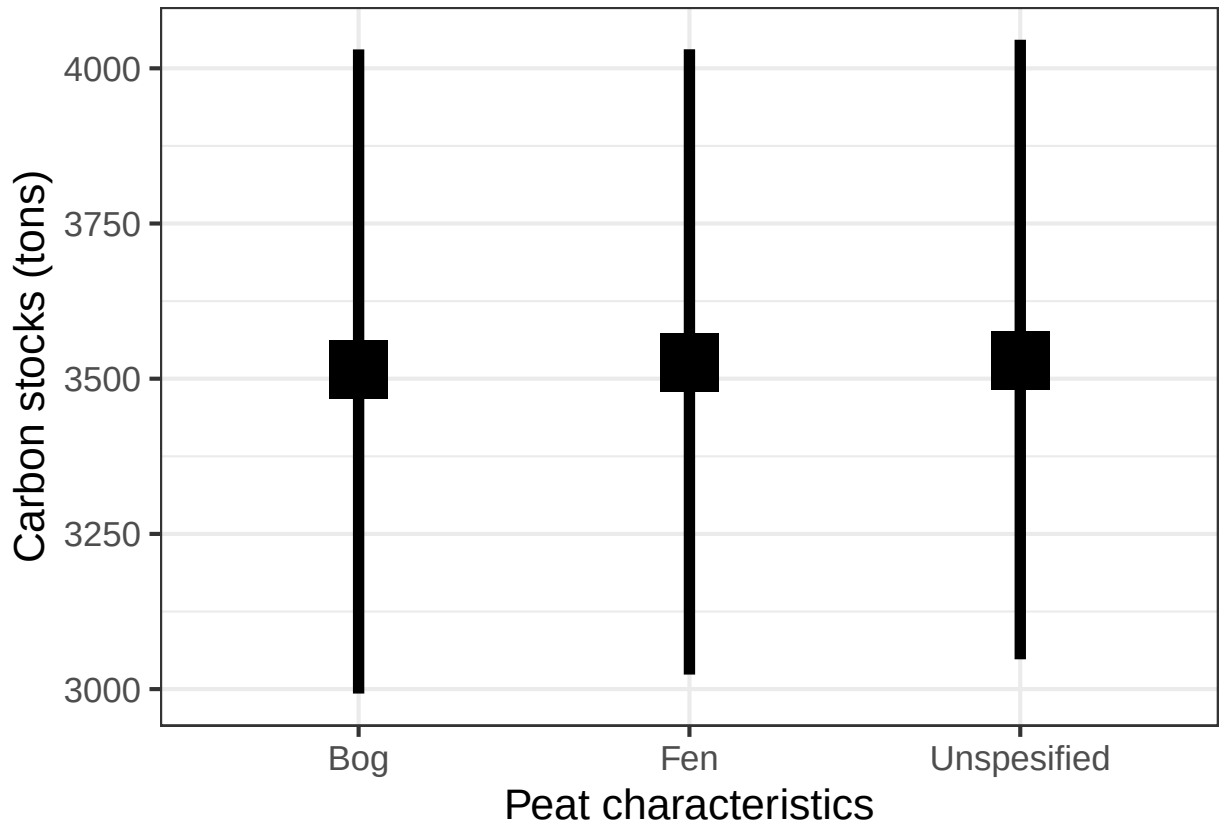


Figure S6.2: Mean (± 95 CI) carbon stock for the Tydal test site.

Specifying the peatland type makes no difference!

Let's just get the (non-informed) C stock estimates for Geilo as well

```
ccalc_cStocks(volume = volume$volume[volume$site ==
    "Geilo"], peatData = df)
#>        5%       50%       95%      mean        sd
#> 4007.5596 4666.1267 5342.8704 4668.1396  408.1592
```

Chapter 7

Additional test sites

We used two contrasting sites, Tydal and Geilo, for the main analyses and testing. Now I will bring in an additional four sites to validate the generality of these finding.

7.1 Modalen

Import shape files with peatland delineations.

```
SHP_modalen <- sf::read_sf("Data/Modalen/Modalen_myrareal.shp")
# st_crs(SHP_modalen) #OK
```

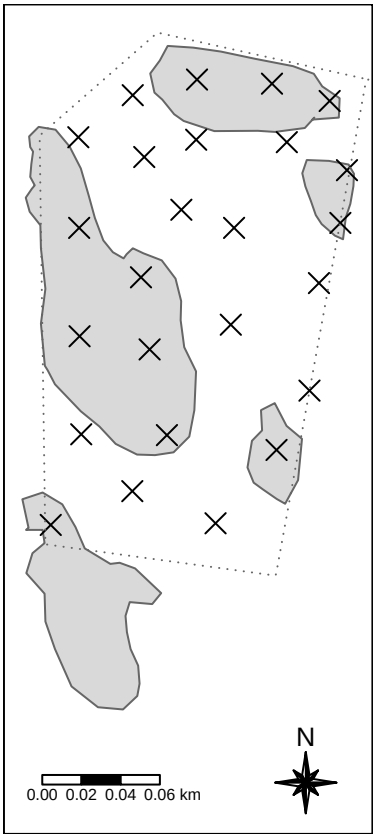
Another shape file with the project area (see below)

```
SHP_modalen_project <- sf::read_sf("Data/Modalen/modalen_shp.shp")
# st_crs(SHP_modalen_project) # OK
```

Import depth samples

```
depths_modalen <- sf::read_sf("Data/Modalen/modalen_punkter.shp")
# st_crs(depths_modalen) #OK
```

```
tm_shape(SHP_modalen) + tm_polygons() + tm_shape(depths_modalen) +
    tm_symbols(col = "black", size = 0.5, shape = 4) +
    tm_shape(SHP_modalen_project) + tm_borders(lty = "dotted") +
    tm_compass(type = "8star", position = c("right",
        "bottom"), size = 2) + tm_scale_bar(position = c("left",
    "bottom"), width = 0.3) + tm_layout(inner.margins = c(0.15,
    0.05, 0.05, 0.05))
```



The peatland delineation (five peatland insde the project area) is mapped using aerial photos and other base maps. There may be peat also between these polygions, however, there si for sure areas between the polygons that are not peat. The depth measurements are performed systematically inside the project area. I don't think we should include points that are taken from smaller isolated mire fragments that are clearly separated from the main peatlands. This example illustrates the importance of appropriate sampling. We are now limited to just a few data points. The smaller mires have only one or two datapoints.

Calculating the combined area of the five mires.

```
SHP_modalen$myArea <- sf::st_area(SHP_modalen)
sum(SHP_modalen$myArea)
#> 16752.75 [m^2]
```

And the area of the project delineation

```
SHP_modalen_project$myArea <- sf::st_area(SHP_modalen_project)
SHP_modalen_project$myArea
#> 36179.43 [m^2]
```

Removing the depth measurement from outside the peatland delinitaion

```
depths_modalen_reduced <- sf::st_intersection(depths_modalen,
    SHP_modalen)
#> Warning: attribute variables are assumed to be spatially
#> constant throughout all geometries
```

```
tm_shape(SHP_modalen) + tm_polygons() + tm_shape(depths_modalen_reduced) +
    tm_symbols(col = "black", size = 0.5, shape = 4) +
```

```
    tm_shape(SHP_modalen_project) + tm_borders(lty = "dotted") +
    tm_compass(type = "8star", position = c("right",
        "bottom"), size = 2) + tm_scale_bar(position = c("left",
    "bottom"), width = 0.3) + tm_layout(inner.margins = c(0.15,
    0.05, 0.05, 0.05))
```

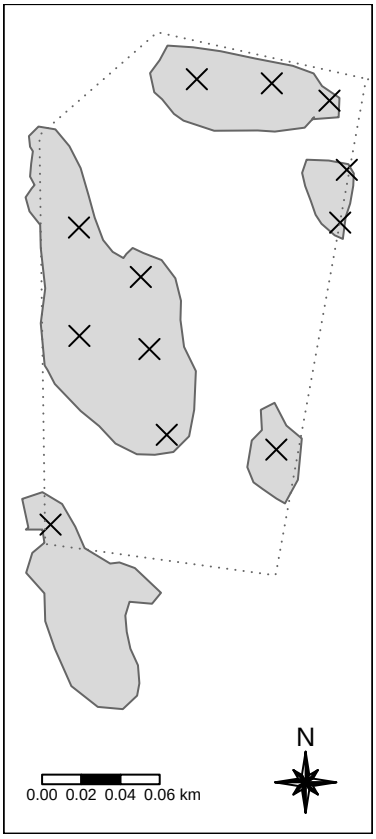


Figure S7.1: The Modalen test site sith peatland delineated as grey polygons. Crosses are peat depth samples, contained with in the project area (dotted line). Only depth measurements that intersects the mire polygons are included

Create a raster grid

```
# First fix polygon closure
SHP_modalen <- sf::st_make_valid(SHP_modalen)


grid_Modalen_stars_crop <- starsExtra::make_grid(SHP_modalen,
    1) %>%
    sf::st_crop(SHP_modalen)
```

```
tm_shape(grid_Modalen_stars_crop) + tm_raster()
```

Predict peat depth

```
ccalc_optimumPower(nmax = nrow(depths_modalen_reduced),
    peatDepths = depths_modalen_reduced, title = "Modalen",
    peatlandDelimination = SHP_modalen)
```

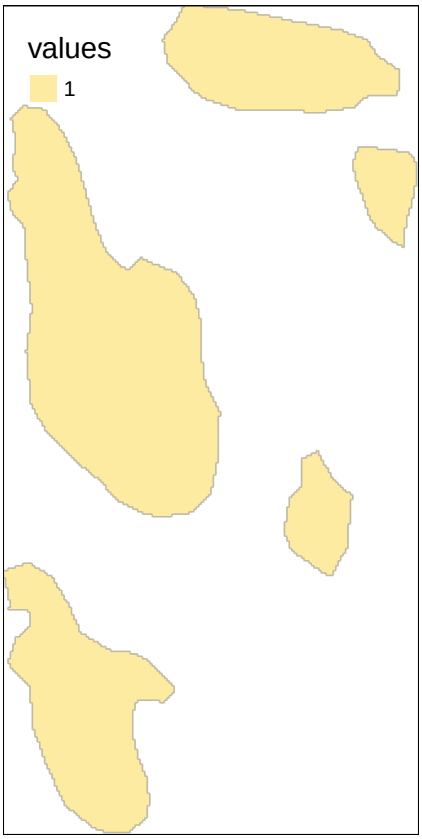


Figure S7.2: Checking that the raster grid for Modalen looks correct.

```
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
```

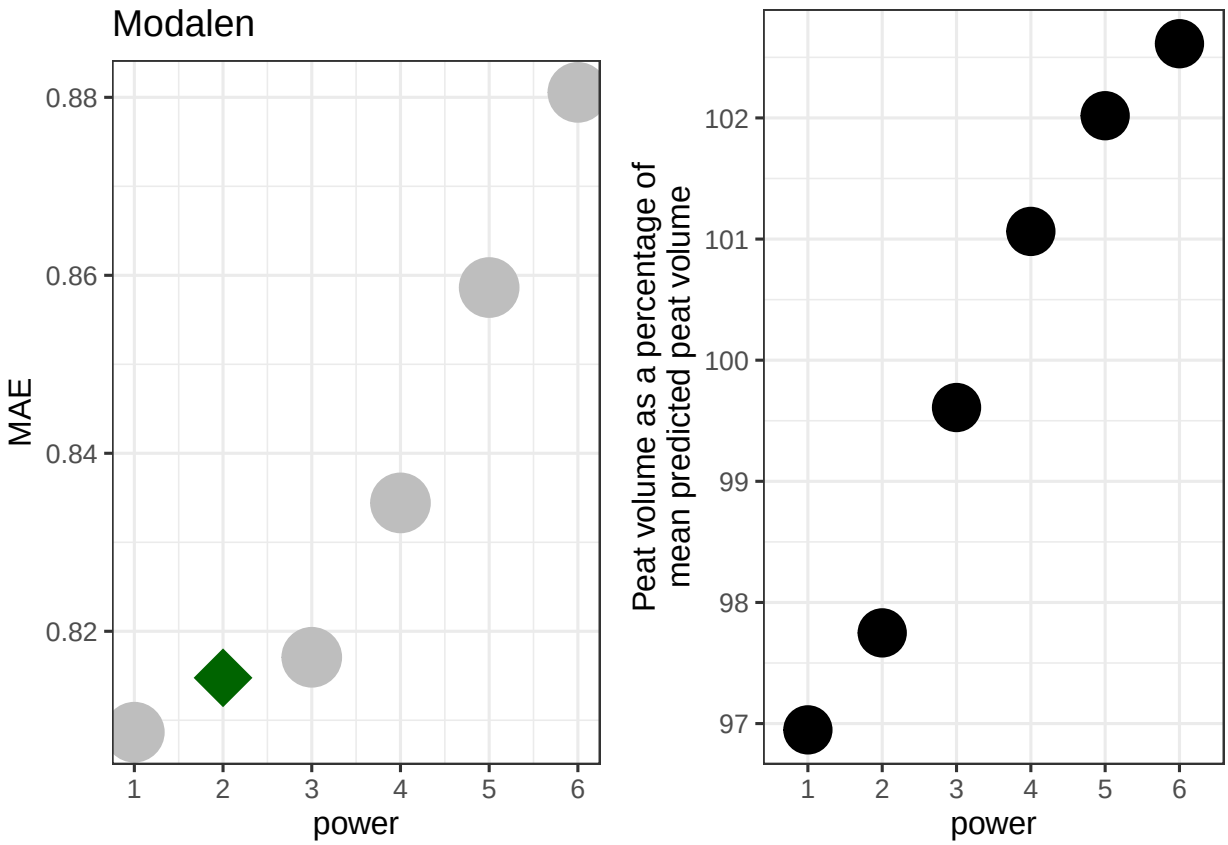


Figure S7.3: Determening the optimal power for the Modelan test site.

Unsurprising, when we have so few data points, the optimum power is very low. I will predict using power 1 through 3 to compare, just because I don't trust power 1 is the best choice.

```
IDW_Modalen_1 <- gstat::idw(formula = Dybde ~ 1, locations = depths_modalen_reduced,
    newdata = grid_Modalen_stars_crop, idp = 1, nmax = 20)  # nmax > than the number of points
#> [inverse distance weighted interpolation]

IDW_Modalen_2 <- gstat::idw(formula = Dybde ~ 1, locations = depths_modalen_reduced,
    newdata = grid_Modalen_stars_crop, idp = 2, nmax = nrow(depths_modalen_reduced))
#> [inverse distance weighted interpolation]

IDW_Modalen_3 <- gstat::idw(formula = Dybde ~ 1, locations = depths_modalen_reduced,
```

```
    newdata = grid_Modalen_stars_crop, idp = 3, nmax = nrow(depths_modalen_reduced))
#> [inverse distance weighted interpolation]
```

```
tmap_arrange(
tm_shape(IDW_Modalen_1)+
  tm_raster(col="var1.pred",
            palette = "-viridis",
            title = "Power = 1\nInterpolated peat\ndepth (m)",
            breaks = seq(0,3,.5))+
  tm_compass(type="8star", position = c("right", "bottom"), size = 2) +
  tm_scale_bar(position = c("right", "bottom"), width = 0.3)+
  tm_shape(depths_modalen_reduced)+
  tm_symbols(shape=4,
             col="black",
             size=.5)+
  tm_layout(legend.outside = T)
,
tm_shape(IDW_Modalen_2)+
  tm_raster(col="var1.pred",
            palette = "-viridis",
            title = "Power = 2\nInterpolated peat\ndepth (m)",
            breaks = seq(0,3,.5))+
  tm_compass(type="8star", position = c("right", "bottom"), size = 2) +
  tm_scale_bar(position = c("right", "bottom"), width = 0.3)+
  tm_shape(depths_modalen_reduced)+
  tm_symbols(shape=4,
             col="black",
             size=.5)+
  tm_layout(legend.outside = T)
,
tm_shape(IDW_Modalen_3)+
  tm_raster(col="var1.pred",
            palette = "-viridis",
            title = "Power = 3\nInterpolated peat\ndepth (m)",
            breaks = seq(0,3,.5))+
  tm_compass(type="8star", position = c("right", "bottom"), size = 2) +
  tm_scale_bar(position = c("right", "bottom"), width = 0.3)+
  tm_shape(depths_modalen_reduced)+
  tm_symbols(shape=4,
             col="black",
             size=.5)+
  tm_layout(legend.outside = T)

)
#> Legend labels were too wide. The labels have been resized to 0.41, 0.41, 0.41, 0.41, 0.41, 0.41. Increa
#> Legend labels were too wide. The labels have been resized to 0.41, 0.41, 0.41, 0.41, 0.41, 0.41. Increa
#> Legend labels were too wide. The labels have been resized to 0.41, 0.41, 0.41, 0.41, 0.41, 0.41. Increa
```

I think power 3 looks more accurate because of the way it gives weight to local points. The recommended practice for this case I think would be to analyse each polyong or mire separately because of how the point from neighboring mires affect the depth predictions.

Summary stats for the peat depth at Modalen.

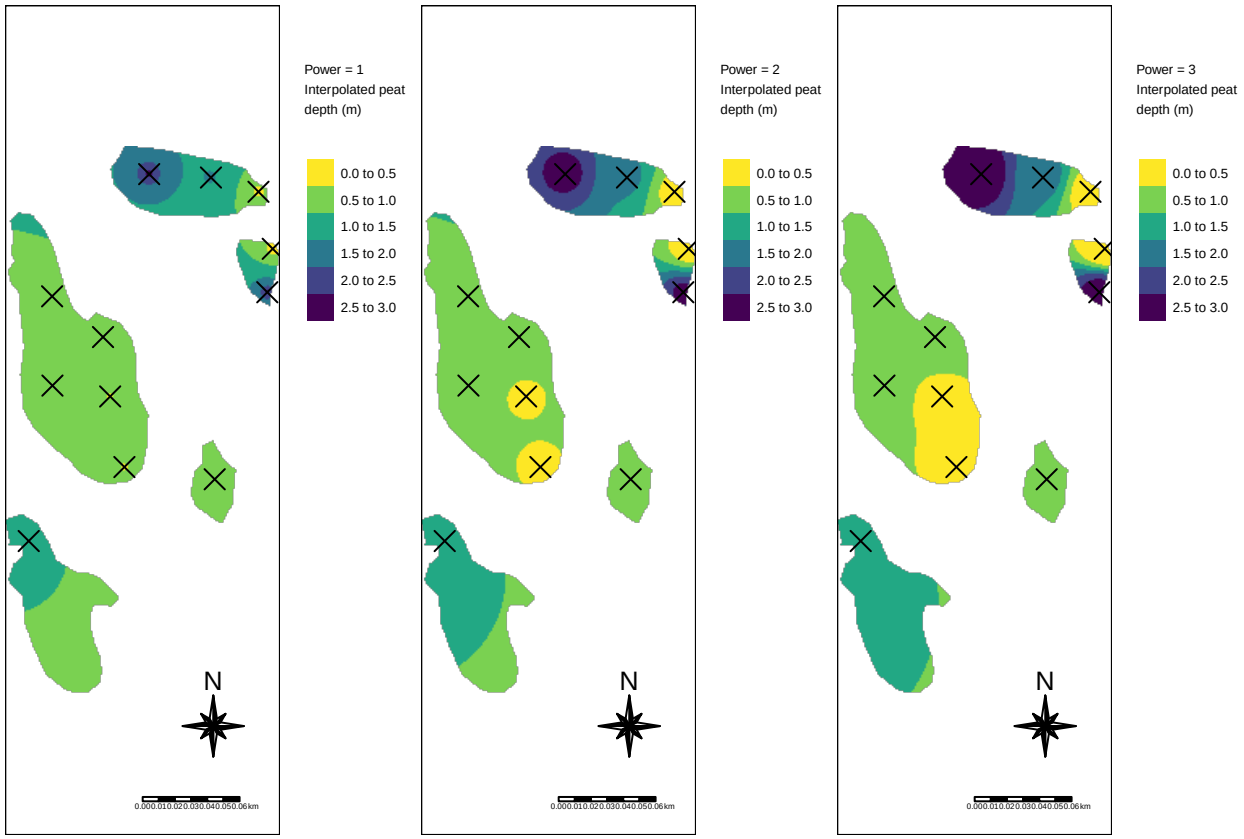


Figure S7.4: IDW for peat depth at Modalen test site.

```
summary(depths_modalen_reduced$Dybde)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   0.000   0.400   0.600   1.017   1.475   2.800
```

The MAE for the power =3 prediction is 0.82 m.

The rough volume (in m3), that is the mean peat depth multiplied by the area, is

```
mean(depths_modalen_reduced$Dybde) * sum(SHP_modalen$myArea)
#> 17031.97 [m^2]
```

Number of depth samples per 100 m2:

```
12/(sum(SHP_modalen$myArea)/100)
#> 0.07163001 [1/m^2]
```

Interpolated volume

```
(volume_modalen <- sum(IDW_Modalen_3$var1.pred, na.rm = T))
#> [1] 16792.6
```

Estimated carbon stocks for the Modalen site:

```
ccalc_cStocks(volume_modalen, peatData = df)
#>        5%       50%       95%      mean        sd
#> 660.91014 762.64323 879.92984 765.22137  67.52858
```

7.2 Opelandsmarka

Import shape file

```
SHP_opelandsmarka <- sf::read_sf("Data/Opelandsmarka/Opelandsmarka Voss.shp")
# st_crs(SHP_opedalsmarka) #32632 UTM 32
```

The depth measurements are in three different files.

```
depths_opelandsmarka1 <- read_delim("Data/Opelandsmarka/Torvdybder_Opelandsmarka_myr1.csv",
    delim = ";", escape_double = FALSE, locale = locale(encoding = "ISO-8859-1"),
    trim_ws = TRUE)
depths_opelandsmarka2 <- read_delim("Data/Opelandsmarka/Torvdybder_Opelandsmarka_myr2.csv",
    delim = ";", escape_double = FALSE, locale = locale(encoding = "ISO-8859-1"),
    trim_ws = TRUE)
depths_opelandsmarka3 <- read_delim("Data/Opelandsmarka/Torvdybder_Opelandsmarka_myr3.csv",
    delim = ";", escape_double = FALSE, locale = locale(encoding = "ISO-8859-1"),
    trim_ws = TRUE)
# The last one is missing a column

depths_opelandsmarka <- bind_rows(depths_opelandsmarka1,
    depths_opelandsmarka2, depths_opelandsmarka3)


depths_opelandsmarka <- sf::st_as_sf(depths_opelandsmarka,
    coords = c("UTM_N_ZONE32N", "UTM_E_ZONE32N"), crs = 32632)
```

```
tm_shape(SHP_opelandsmarka) + tm_polygons() + tm_shape(depths_opelandsmarka) +
    tm_symbols(col = "black", size = 0.5, shape = 4)
```

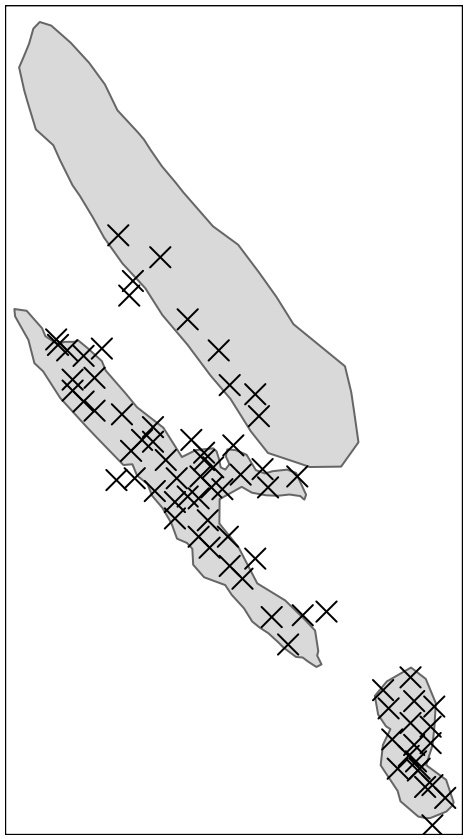


Figure S7.5: Test site: Opelandsmarka. Crosses are peat depth measurements.

I think we can ignore that top mire as it was not sampled nearly as intensive as the other two.

```
SHP_opelandsmarka_twoMires <- SHP_opelandsmarka[-1,
    ]
```

It could be a good idea sometimes to include depth measurements outside the mire polygons (with a depth of 0). Here it is easier for me to just exclude all points outide any polygon.

```
depths_opelandsmarka <- st_intersection(depths_opelandsmarka,
    SHP_opelandsmarka_twoMires)
#> Warning: attribute variables are assumed to be spatially
#> constant throughout all geometries
```

```
tm_shape(SHP_opelandsmarka_twoMires) + tm_polygons() +
    tm_shape(depths_opelandsmarka) + tm_symbols(col = "black",
    size = 0.5, shape = 4)
```

There is one NA in the depth data which we have to remove

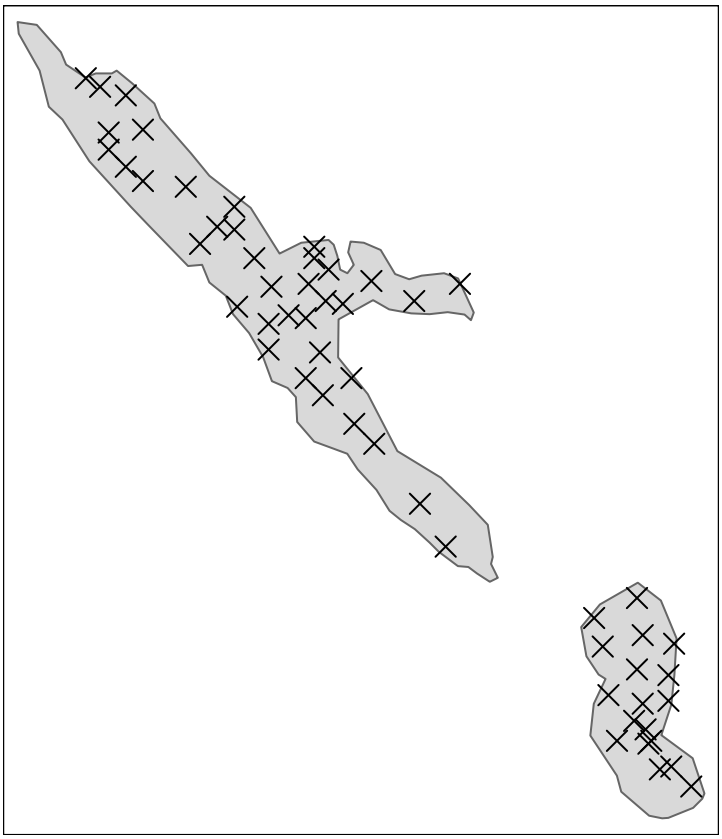


Figure S7.6: Opelandsmarka with two out of three mires. Crosses are peat depth measurements.

```
depths_opelandsmarka <- depths_opelandsmarka[!is.na(depths_opelandsmarka$Dybde),
    ]
```

Try again:

```
ccalc_optimumPower(peatDepths = depths_opelandsmarka,
    title = "Opelandsmarka", peatlandDelimination = SHP_opelandsmarka_twoMires)
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
```

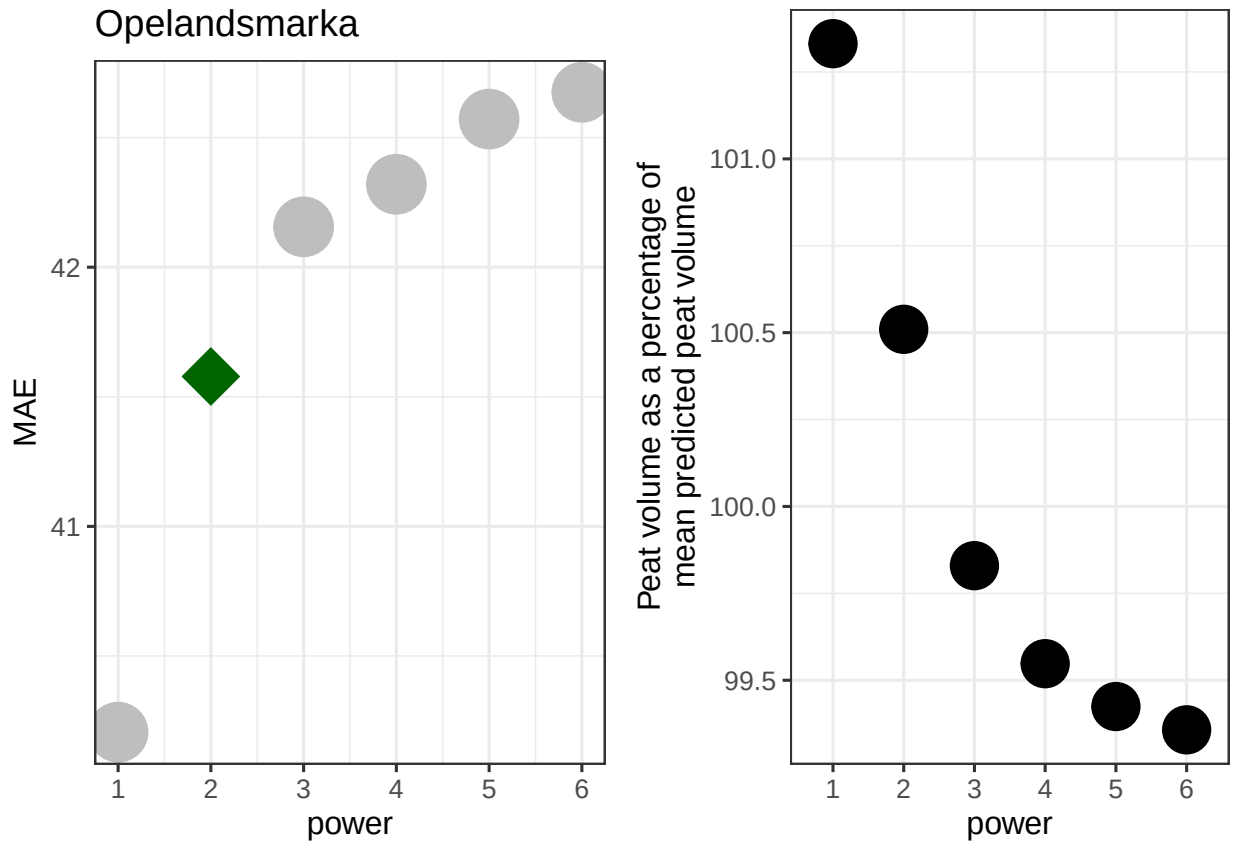


Figure S7.7: Determening the optimal power for the Opelandsmarka test site.

Again the lowest MAE is obtained from a power of 1, but as we have seen, the power should probably be set higher. The predicted peat volume for example is quite unstable at low power settings.

Create a raster grid for the predictions

```
grid_Opelandsmarka_stars_crop <- starsExtra::make_grid(SHP_opelandsmarka_twoMires,
    1) %>%
    sf::st_crop(SHP_opelandsmarka_twoMires)
```

The depths are recorded as cm not meters. Converting to m.

```
depths_opelandsmarka$Dybde <- depths_opelandsmarka$Dybde/100
```

```
IDW_opelandsmarka_1 <- gstat::idw(Dybde ~ 1, depths_opelandsmarka,
    newdata = grid_Opelandsmarka_stars_crop, nmax = nmax,
    idp = 1)
#> [inverse distance weighted interpolation]

IDW_opelandsmarka_2 <- gstat::idw(Dybde ~ 1, depths_opelandsmarka,
    newdata = grid_Opelandsmarka_stars_crop, nmax = nmax,
    idp = 2)
#> [inverse distance weighted interpolation]

IDW_opelandsmarka_3 <- gstat::idw(Dybde ~ 1, depths_opelandsmarka,
    newdata = grid_Opelandsmarka_stars_crop, nmax = nmax,
    idp = 3)
#> [inverse distance weighted interpolation]
```

```
myBreaks <- seq(0,3,.5)
myPos <- c("left", "bottom")

tmap_arrange(
tm_shape(IDW_opelandsmarka_1)+
  tm_raster(col="var1.pred",
            palette = "-viridis",
            title = "Power = 1\nInterpolated peat\ndepth (m)",
            breaks = myBreaks)+
  tm_compass(type="8star", position = myPos, size = 2) +
  tm_scale_bar(position = myPos, width = 0.3)+
  tm_shape(depths_opelandsmarka)+
  tm_symbols(shape=4,
             col="black",
             size=.5)+
  tm_layout(legend.outside = T)
,
tm_shape(IDW_opelandsmarka_2)+
  tm_raster(col="var1.pred",
            palette = "-viridis",
            title = "Power = 2\nInterpolated peat\ndepth (m)",
            breaks = myBreaks)+
  tm_compass(type="8star", position = myPos, size = 2) +
  tm_scale_bar(position = myPos, width = 0.3)+
  tm_shape(depths_opelandsmarka)+
  tm_symbols(shape=4,
             col="black",
             size=.5)+
  tm_layout(legend.outside = T)
,
tm_shape(IDW_opelandsmarka_3)+
  tm_raster(col="var1.pred",
```

```
            palette = "-viridis",
            title = "Power = 3\nInterpolated peat\ndepth (m)",
            breaks = myBreaks)+
  tm_compass(type="8star", position = myPos, size = 2) +
  tm_scale_bar(position = myPos, width = 0.3)+
  tm_shape(depths_opelandsmarka)+
  tm_symbols(shape=4,
             col="black",
             size=.5)+
  tm_layout(legend.outside = T),
ncol = 2

)
```

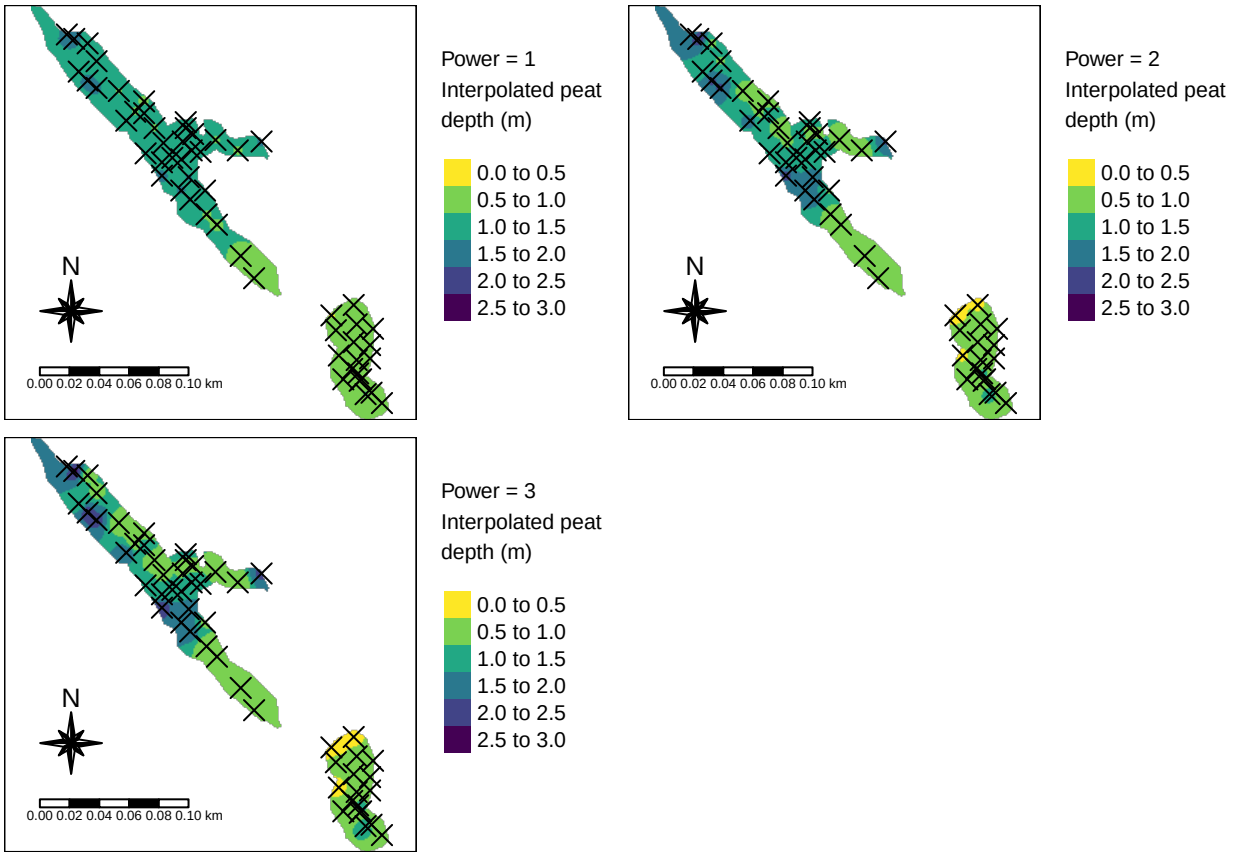


Figure S7.8: Interpolated peat depth at test site Opelandsmarka.

With power set to 1 the model is too flat, and the deep areas are not identified correctly. Power 2 or 3 makes less of a difference. Becaus ethe MAE and volume estimates are so similar, I will chose the modet local model (power = 3)

Getting the total peatland area

```
SHP_opelandsmarka_twoMires$myArea <- sf::st_area(SHP_opelandsmarka_twoMires)
sum(SHP_opelandsmarka_twoMires$myArea)
#> 8700.213 [m^2]
```

Getting the mean peat depth (after excluding points outside the polygons)

```
mean(depths_opelandsmarka$Dybde)
#> [1] 1.058333
```

Rough volume

```
mean(depths_opelandsmarka$Dybde) * sum(SHP_opelandsmarka_twoMires$myArea)
#> 9207.725 [m^2]
```

Number of peat depth measurements

```
nrow(depths_opelandsmarka)
#> [1] 54
```

Depth measurement density (points per 100 m-2)

```
nrow(depths_opelandsmarka)/(sum(SHP_opelandsmarka_twoMires$myArea)/100)
#> 0.6206745 [1/m^2]
```

Interpolated peat volume (best model)

```
(vol_temp <- sum(IDW_opelandsmarka_3$var1.pred, na.rm = T))
#> [1] 9270.232
```

Estimated C stocks

```
ccalc_cStocks(volume = vol_temp, peatData = df)
#>       5%      50%      95%     mean       sd
#> 359.4981 421.1448 484.9453 421.8599  37.5518
```

7.3 Kinn 1

```
# import peatland delineation
SHP_kinn1 <- sf::st_read("Data/Kinn/rikmyr_avgrensing.shp")
#> Reading layer `rikmyr_avgrensing' from data source
#>   `C:\Users\anders.kolstad\Github\carbonCalculator\Data\Kinn\rikmyr_avgrensing.shp'
#>   using driver `ESRI Shapefile'
#> Simple feature collection with 1 feature and 27 fields
#> Geometry type: POLYGON
#> Dimension:     XY
#> Bounding box:  xmin: 5.099473 ymin: 61.48336 xmax: 5.102058 ymax: 61.48463
#> Geodetic CRS:  WGS 84

# import peat deth data
depths_kinn1 <- read_delim("Data/Kinn/data_rikmyr_torvdybder.csv",
    delim = ";", escape_double = FALSE, locale = locale(encoding = "ISO-8859-1"),
    trim_ws = TRUE)
```

The peat depth data doesn't contain coordinated. Importing those now.

```
depths_kinn1_spatial <- sf::st_read("Data/Kinn/rikmyr_punkter.shp")
#> Reading layer `rikmyr_punkter' from data source
#>   `C:\Users\anders.kolstad\Github\carbonCalculator\Data\Kinn\rikmyr_punkter.shp'
#>   using driver `ESRI Shapefile'
#> Simple feature collection with 37 features and 24 fields
#> Geometry type: POINT
#> Dimension:     XYZ
#> Bounding box:  xmin: 5.099712 ymin: 61.48347 xmax: 5.101931 ymax: 61.48458
#> z_range:       zmin: 18.489 zmax: 44.38
#> Geodetic CRS:  WGS 84
```

The column `name` is shared between the two datasets so I can use this to merge them. First I need to remove leading zeros.

```
depths_kinn1_spatial$name <- as.numeric(depths_kinn1_spatial$name)
```

```
depths_kinn1_spatial$Dybde <- depths_kinn1$Dybde[match(depths_kinn1_spatial$name,
    depths_kinn1$name)]
rm(depths_kinn1)
```

There is still one NA to remove

```
depths_kinn1_spatial <- depths_kinn1_spatial[!is.na(depths_kinn1_spatial$Dybde),
    ]
```

Check CRS

```
# st_crs(depths_kinn1_spatial) # lat long
# st_crs(SHP_kinn1) # also lat long

depths_kinn1_spatial <- sf::st_transform(depths_kinn1_spatial,
    25833)
SHP_kinn1 <- sf::st_transform(SHP_kinn1, 25833)
```

```
tm_shape(SHP_kinn1) + tm_polygons() + tm_shape(depths_kinn1_spatial) +
    tm_symbols(col = "black", size = 0.5, shape = 4)
```

This is a nice example data set. I notice however, there are few points near the edges.

Getting the rough volume

```
SHP_kinn1$myArea <- sf::st_area(SHP_kinn1)
SHP_kinn1$myArea
#> 13541.53 [m^2]
```

Mean depth

```
mean(depths_kinn1_spatial$Dybde, na.rm = T)
#> [1] 244.1429
```

The depths are in cm. Converting to m.

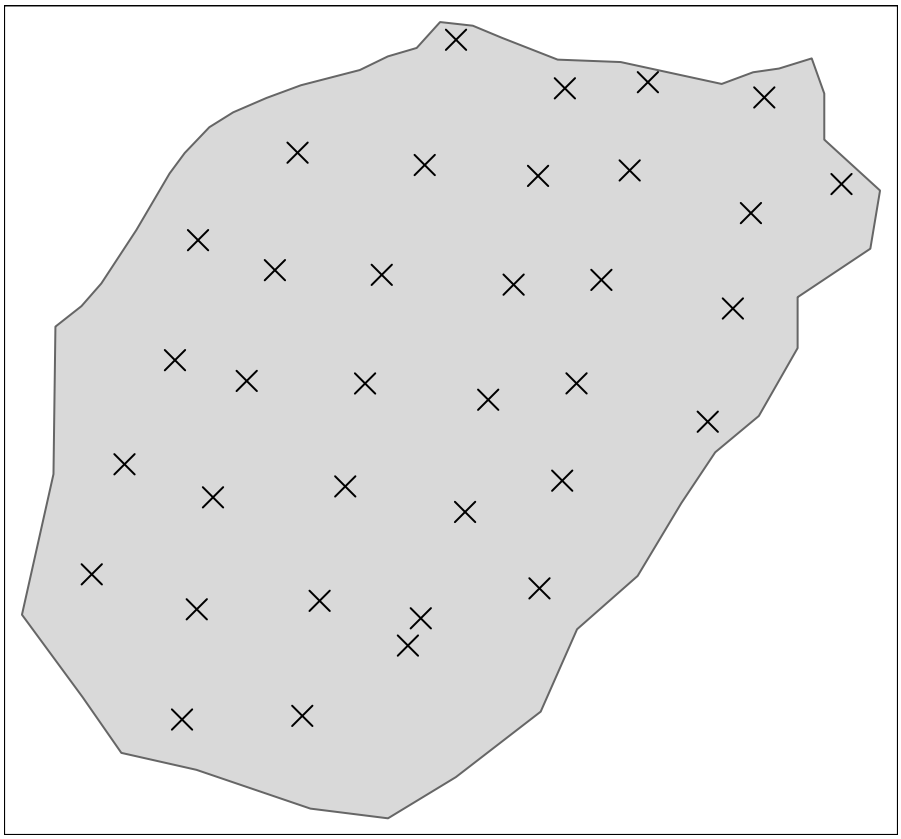


Figure S7.9: Test site Kinn1.

```
depths_kinn1_spatial$Dybde <- depths_kinn1_spatial$Dybde/100
```

Number of depth measurements

```
nrow(depths_kinn1_spatial)
#> [1] 35
```

Point density (100 m-2)

```
nrow(depths_kinn1_spatial)/(SHP_kinn1$myArea) * 100
#> 0.2584641 [1/m^2]
```

Finding optimum power

```
ccalc_optimumPower(peatDepths = depths_kinn1_spatial,
    peatlandDelimination = SHP_kinn1, title = "Kinn1")
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
```

The optimal power is 5.

Create a raster grid for the predictions

```
grid_kinn1_stars_crop <- starsExtra::make_grid(SHP_kinn1,
    1) %>%
    sf::st_crop(SHP_kinn1)
```

```
IDW_kinn1_5 <- gstat::idw(Dybde ~ 1, depths_kinn1_spatial,
    newdata = grid_kinn1_stars_crop, nmax = nmax, idp = 5)
#> [inverse distance weighted interpolation]
```

Max peat depth

```
max(depths_kinn1_spatial$Dybde)
#> [1] 4.7
```

```
myBreaks <- seq(0, 5, 1)
myPos <- c("right", "bottom")

tm_shape(IDW_kinn1_5) + tm_raster(col = "var1.pred",
    palette = "-viridis", title = "Power = 5\nInterpolated peat\ndepth (m)",
    breaks = myBreaks) + tm_compass(type = "8star",
    position = myPos, size = 2) + tm_scale_bar(position = myPos,
    width = 0.3) + tm_shape(depths_kinn1_spatial) +
    tm_symbols(shape = 4, col = "black", size = 0.5) +
    tm_layout(legend.outside = T)
```

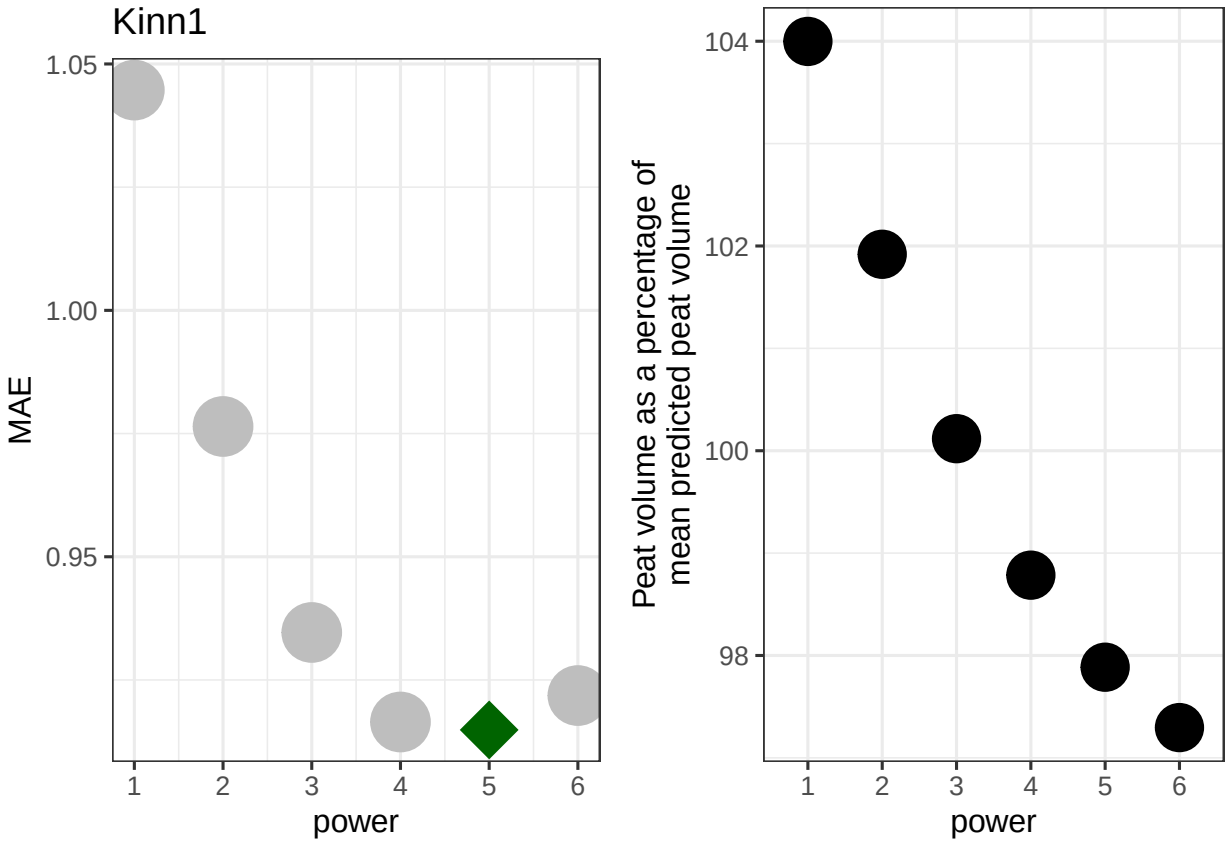


Figure S7.10: Finding the best power setting for the Kinn1 test site.

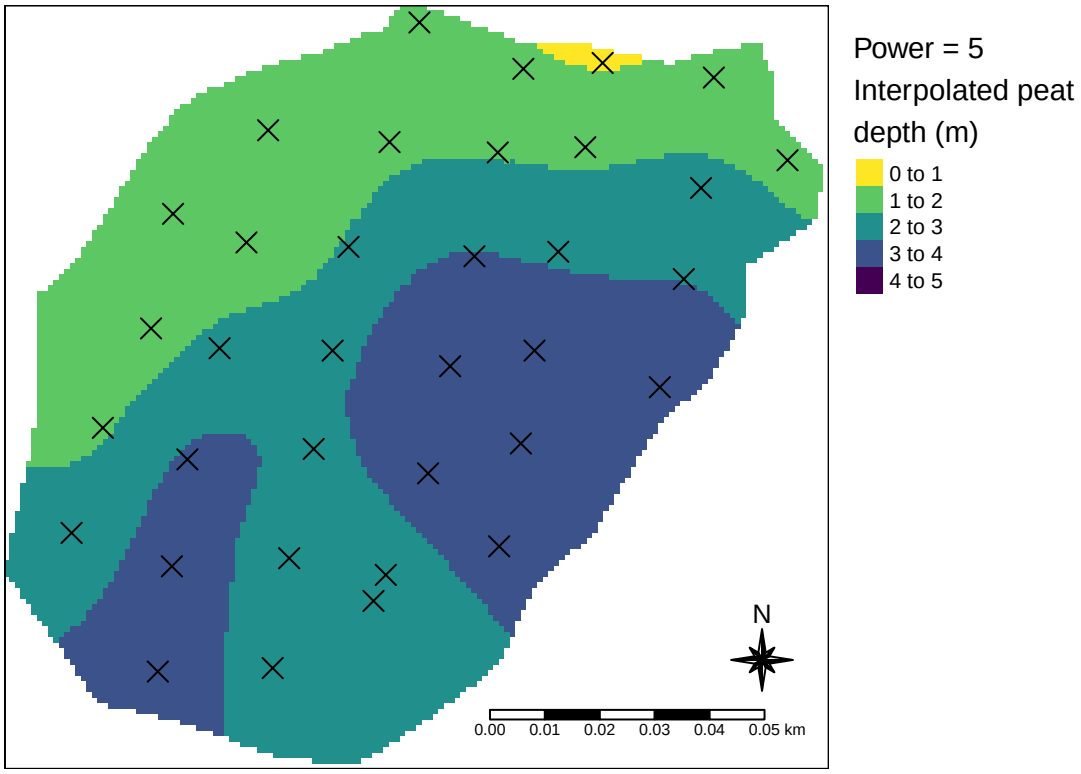


Figure S7.11: Best model for test site Kinn1

Predicted peat volume

```
(vol_temp <- sum(IDW_kinn1_5$var1.pred, na.rm = T))
#> [1] 33530.09
```

Carbon stocks

```
ccalc_cStocks(volume = vol_temp, peatData = df)
#>        5%       50%       95%      mean        sd
#> 1321.9069 1523.9218 1747.6230 1527.1975  130.4528
```

7.4 Kinn2

```
SHP_kinn2 <- sf::st_read("Data/Kinn/maroy_avgrensing.shp")
#> Reading layer `maroy_avgrensing' from data source
#>   `C:\Users\anders.kolstad\Github\carbonCalculator\Data\Kinn\maroy_avgrensing.shp'
#>   using driver `ESRI Shapefile'
#> Simple feature collection with 1 feature and 27 fields
#> Geometry type: POLYGON
#> Dimension:     XY
#> Bounding box:  xmin: 5.102353 ymin: 61.50691 xmax: 5.107625 ymax: 61.50866
#> Geodetic CRS:  WGS 84
```

```
depths_kinn2 <- read_delim("Data/Kinn/data_maroy_torvdybder.csv",
    delim = ";", escape_double = FALSE, locale = locale(encoding = "ISO-8859-1"),
    trim_ws = TRUE)
```

As before, the depths have no coordinates.

```
depths_kinn2_spatial <- sf::st_read("Data/Kinn/maroy_punkter.shp")
#> Reading layer `maroy_punkter' from data source
#>   `C:\Users\anders.kolstad\Github\carbonCalculator\Data\Kinn\maroy_punkter.shp'
#>   using driver `ESRI Shapefile'
#> Simple feature collection with 78 features and 24 fields
#> Geometry type: POINT
#> Dimension:     XYZ
#> Bounding box:  xmin: 5.102523 ymin: 61.50698 xmax: 5.10742 ymax: 61.5086
#> z_range:       zmin: 8.484 zmax: 17.732
#> Geodetic CRS:  WGS 84
```

```
depths_kinn2_spatial$name <- as.numeric(depths_kinn2_spatial$name)
depths_kinn2_spatial$Dybde <- depths_kinn2$Dybde[match(depths_kinn2_spatial$name,
    depths_kinn2$name)]
rm(depths_kinn2)
```

Transform to UTF33

```
depths_kinn2_spatial <- sf::st_transform(depths_kinn2_spatial,
    25833)
SHP_kinn2 <- sf::st_transform(SHP_kinn2, 25833)
```

```
tm_shape(SHP_kinn2) + tm_polygons() + tm_shape(depths_kinn2_spatial) +
    tm_symbols(col = "black", size = 0.5, shape = 4)
```

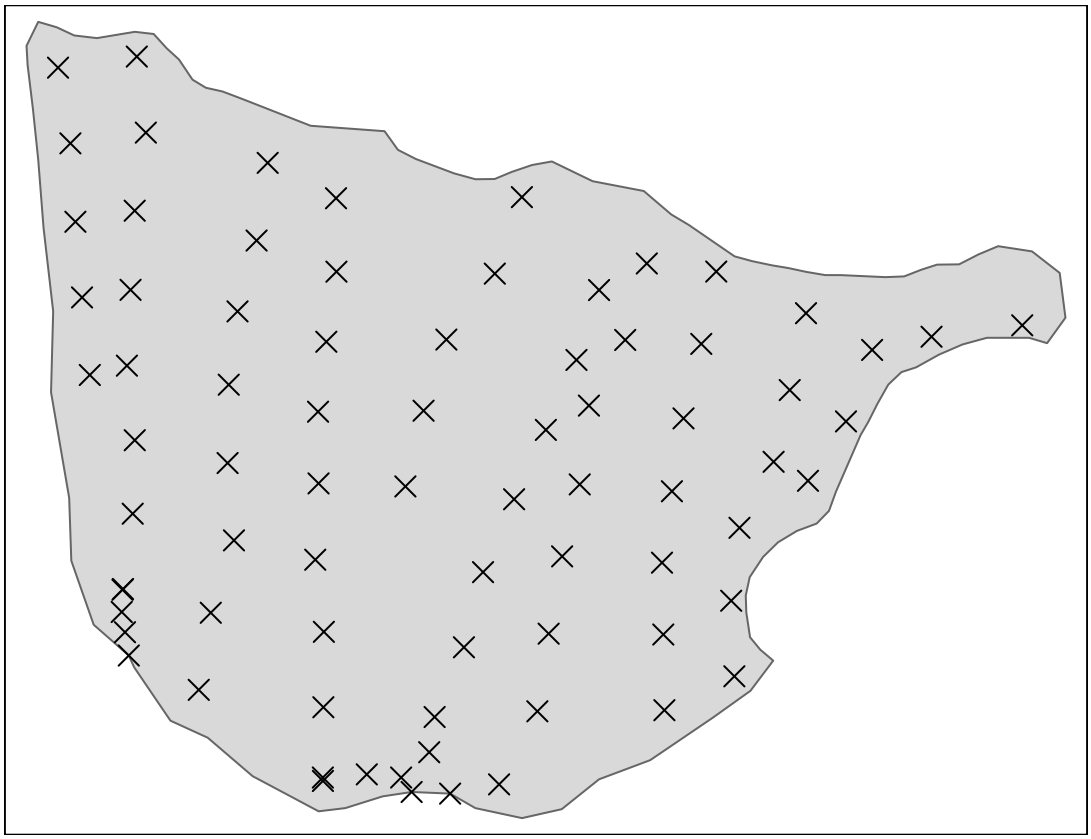


Figure S7.12: Test site Kinn2.

The mean depth

```
summary(depths_kinn2_spatial$Dybde, na.rm = T)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    0.00   41.25   71.00   90.26  130.25  282.00
```

Converting from cm to m

```
depths_kinn2_spatial$Dybde <- depths_kinn2_spatial$Dybde/100
```

Calculate area:

```
SHP_kinn2$myArea <- st_area(SHP_kinn2)
```

Number of peat depth measurements:

```
nrow(depths_kinn2_spatial)
#> [1] 78
```

Density of points (100 m-2)

```
nrow(depths_kinn2_spatial)/SHP_kinn2$myArea * 100
#> 0.2346467 [1/m^2]
```

Find optimal power

```
ccalc_optimumPower(peatDepths = depths_kinn2_spatial,
    peatlandDelimination = SHP_kinn2, title = "Kinn2")
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
#>   |                                                                      |
#> [inverse distance weighted interpolation]
```

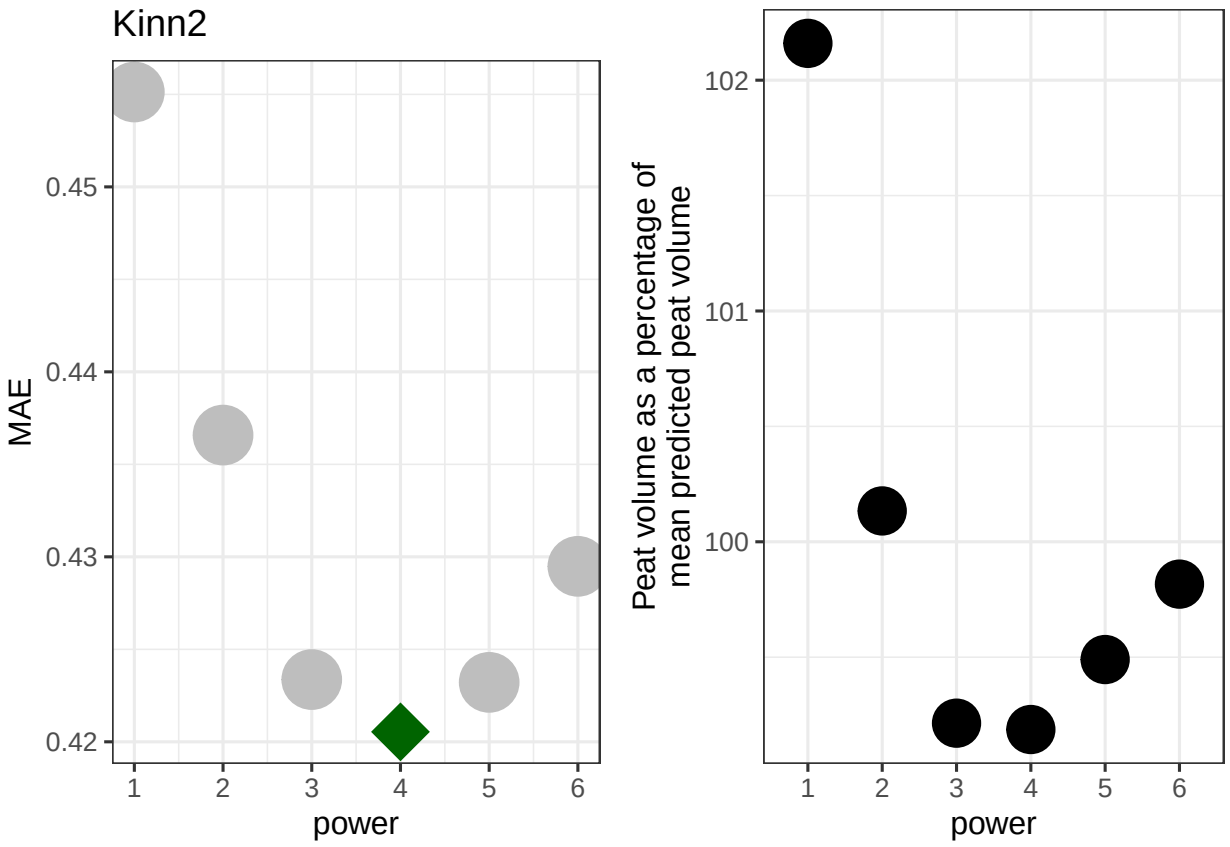


Figure S7.13: Finding the best power setting for the Kinn2 test site.

Power = 4 is best.

Create a raster grid for the predictions

```
grid_kinn2_stars_crop <- starsExtra::make_grid(SHP_kinn2,
    1) %>%
    sf::st_crop(SHP_kinn2)
```

Interpreted peat volume

```
IDW_kinn2_4 <- gstat::idw(Dybde ~ 1, depths_kinn2_spatial,
    newdata = grid_kinn2_stars_crop, nmax = nmax, idp = 4)
#> [inverse distance weighted interpolation]
```

```
myBreaks <- seq(0, 3, 0.5)
myPos <- c("right", "bottom")

tm_shape(IDW_kinn2_4) + tm_raster(col = "var1.pred",
    palette = "-viridis", title = "Power = 4\nInterpolated peat\ndepth (m)",
    breaks = myBreaks) + tm_compass(type = "8star",
    position = myPos, size = 2) + tm_scale_bar(position = myPos,
    width = 0.3) + tm_shape(depths_kinn2_spatial) +
    tm_symbols(shape = 4, col = "black", size = 0.5) +
    tm_layout(legend.outside = T)
```

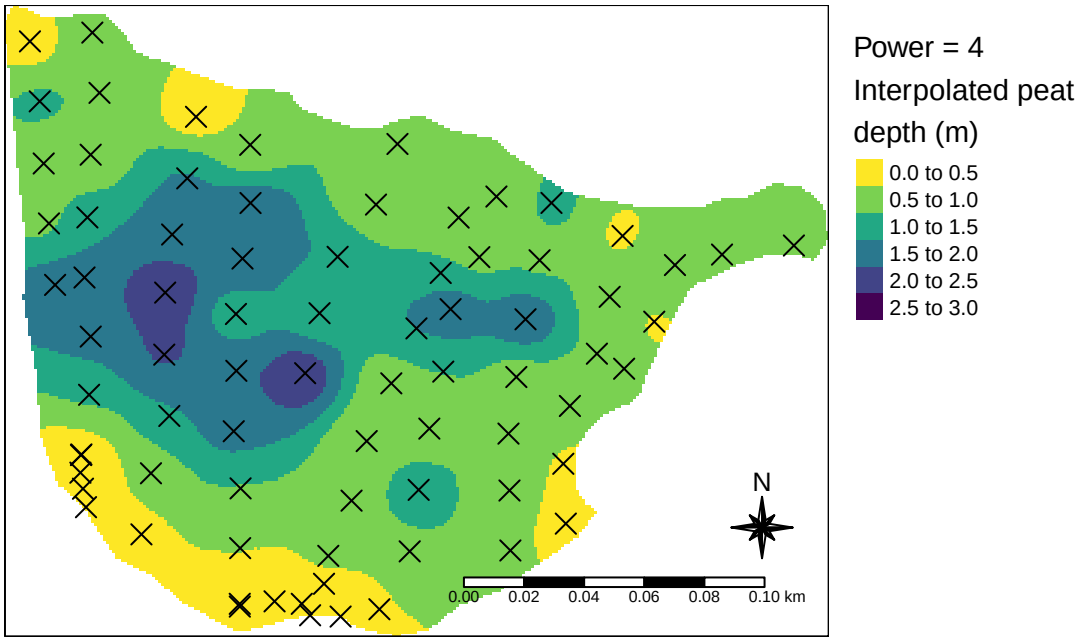


Figure S7.14: Best model for test site Kinn2

And the volume is

```
(vol_temp <- sum(IDW_kinn2_4$var1.pred, na.rm = T))
#> [1] 33037.11
```

And the C stock

```
ccalc_cStocks(volume = vol_temp, peatData = df)
#>        5%       50%       95%      mean        sd
#> 1279.8357 1500.4463 1714.6176 1500.0792  134.4706
```

Chapter 8

Session Info

```
sessionInfo()
#> R version 4.2.2 (2022-10-31 ucrt)
#> Platform: x86_64-w64-mingw32/x64 (64-bit)
#> Running under: Windows 10 x64 (build 19045)
#>
#> Matrix products: default
#>
#> locale:
#> [1] LC_COLLATE=Norwegian_Norway.1252
#> [2] LC_CTYPE=Norwegian_Norway.1252
#> [3] LC_MONETARY=Norwegian_Norway.1252
#> [4] LC_NUMERIC=C
#> [5] LC_TIME=Norwegian_Norway.1252
#> system code page: 65001
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets
#> [6] methods   base
#>
#> other attached packages:
#>  [1] knitr_1.42          osmplotr_0.3.3.022
#>  [3] formatR_1.14        forcats_1.0.0
#>  [5] stringr_1.5.0       dplyr_1.0.10
#>  [7] purrr_1.0.1         tidyr_1.3.0
#>  [9] tibble_3.1.8        tidyverse_1.3.2
#> [11] ggtext_0.1.2        matrixStats_0.63.0
#> [13] ggpubr_0.5.0        ggplot2_3.4.0
#> [15] basemaps_0.0.5      tmaptools_3.1-1
#> [17] readr_2.1.3         sf_1.0-9
#> [19] tmap_3.3-3          bookdown_0.32
#> [21] gstat_2.1-0
#>
#> loaded via a namespace (and not attached):
#>   [1] utf8_1.2.2              spatstat.explore_3.0-6
#>   [3] tidyselect_1.2.0        htmlwidgets_1.6.1
#>   [5] grid_4.2.2              devtools_2.4.5
#>   [7] munsell_0.5.0           ragg_1.2.5
#>   [9] codetools_0.2-18        units_0.8-1
#>  [11] miniUI_0.1.1.1          withr_2.5.0
```

```
#>  [13] spatstat.random_3.1-3  colorspace_2.1-0
#>  [15] highr_0.10             rstudioapi_0.14
#>  [17] wk_0.7.1               ggsignif_0.6.4
#>  [19] tensor_1.5             labeling_0.4.2
#>  [21] httr2_0.2.2            polyclip_1.10-4
#>  [23] farver_2.1.1           bit64_4.0.5
#>  [25] rprojroot_2.0.3        vctrs_0.5.2
#>  [27] generics_0.1.3         xfun_0.36
#>  [29] timechange_0.2.0       markdown_1.4
#>  [31] R6_2.5.1               slippymath_0.3.1
#>  [33] spatstat.utils_3.0-1   cachem_1.0.6
#>  [35] assertthat_0.2.1       promises_1.2.0.1
#>  [37] scales_1.2.1           vroom_1.6.1
#>  [39] googlesheets4_1.0.1    gtable_0.3.1
#>  [41] lwgeom_0.2-11          processx_3.8.0
#>  [43] goftest_1.2-3          rlang_1.0.6
#>  [45] systemfonts_1.0.4      osmdata_0.2.0
#>  [47] splines_4.2.2          rstatix_0.7.1
#>  [49] gargle_1.3.0           dichromat_2.0-0.1
#>  [51] spatstat.geom_3.0-6    broom_1.0.3
#>  [53] s2_1.1.2               yaml_2.3.7
#>  [55] abind_1.4-5            modelr_0.1.10
#>  [57] crosstalk_1.2.0        backports_1.4.1
#>  [59] httpuv_1.6.8           gridtext_0.1.5
#>  [61] tools_4.2.2            usethis_2.1.6
#>  [63] ellipsis_0.3.2         raster_3.6-14
#>  [65] RColorBrewer_1.1-3     proxy_0.4-27
#>  [67] sessioninfo_1.2.2      Rcpp_1.0.10
#>  [69] base64enc_0.1-3        classInt_0.4-8
#>  [71] ps_1.7.2               prettyunits_1.1.1
#>  [73] rpart_4.1.19           deldir_1.0-6
#>  [75] pbapply_1.7-0          cowplot_1.1.1
#>  [77] urlchecker_1.0.1       zoo_1.8-11
#>  [79] haven_2.5.1            fs_1.6.0
#>  [81] leafem_0.2.0           tinytex_0.43
#>  [83] magrittr_2.0.3         magick_2.7.3
#>  [85] spacetime_1.2-8        reprex_2.0.2
#>  [87] googledrive_2.0.0      pkgload_1.3.2
#>  [89] hms_1.1.2              mime_0.12
#>  [91] evaluate_0.20          xtable_1.8-4
#>  [93] XML_3.99-0.13          leaflet_2.1.1
#>  [95] readxl_1.4.1           compiler_4.2.2
#>  [97] maps_3.4.1             KernSmooth_2.23-20
#>  [99] crayon_1.5.2           htmltools_0.5.4
#> [101] mgcv_1.8-41            later_1.3.0
#> [103] tzdb_0.3.0             lubridate_1.9.1
#> [105] DBI_1.1.3              dbplyr_2.3.0
#> [107] rappdirs_0.3.3         Matrix_1.5-3
#> [109] car_3.1-1              cli_3.6.0
#> [111] parallel_4.2.2         pkgconfig_2.0.3
#> [113] sp_1.6-0               terra_1.7-3
#> [115] spatstat.sparse_3.0-0  xml2_1.3.3
#> [117] rvest_1.0.3            callr_3.7.3
#> [119] digest_0.6.31          starsExtra_0.2.7
#> [121] spatstat.data_3.0-0    rmarkdown_2.20
#> [123] cellranger_1.1.0       leafsync_0.1.0
```

```
#> [125] intervals_0.15.2       curl_5.0.0
#> [127] commonmark_1.8.1       shiny_1.7.4
#> [129] lifecycle_1.0.3        nlme_3.1-161
#> [131] jsonlite_1.8.4         carData_3.0-5
#> [133] mapproj_1.2.11         desc_1.4.2
#> [135] viridisLite_0.4.1      fansi_1.0.4
#> [137] pillar_1.8.1           spatstat.linnet_3.0-4
#> [139] lattice_0.20-45        fastmap_1.1.0
#> [141] httr_1.4.4             pkgbuild_1.4.0
#> [143] glue_1.6.2             xts_0.12.2
#> [145] remotes_2.4.2          FNN_1.1.3.1
#> [147] spatstat_3.0-3         png_0.1-8
#> [149] bit_4.0.5              class_7.3-20
#> [151] stringi_1.7.12         profvis_0.3.7
#> [153] textshaping_0.3.6      spatstat.model_3.1-2
#> [155] stars_0.6-0            memoise_2.0.1
#> [157] e1071_1.7-12
```