UPTEC W 17023
Examensarbete 30 hp
Juni 2017

UPPSALA UNIVERSITET

SLU

Modellering av volym samt max- och medeldjup i svenska sjöar

- en statistisk analys med hjälp av geografiska informationssystem

Sara Sandström

REFERAT

Modellering av volym samt max- och medeldjup i svenska sjöar - en statistisk analys med hjälp av geografiska informationssystem

Sara Sandström

Sverige är ett land med ett stort antal insjöar. Av Sveriges ca 100 000 sjöar finns endast uppmätta data på volym, max- och medeldjup för ungefär 8000 sjöar, vilket lämnar ett stort informationsgap. Att provta en så stor mängd sjöar som skulle behövas för att fylla detta gap är väldigt tidskrävande. Det skapar behov av ett alternativt tillvägagångssätt. Tidigare försök att ta fram en modell utifrån kartparametrar har utförts av Sobek m. fl. (2011) vilket resulterade i en modell för volym där medelvolymen hos en grupp av minst 15 sjöar kunde predikteras utifrån kartparametrar med hög säkerhet, medan prediktionerna för volymen i en enskild sjö blev för osäker. Modellen för maxdjup innehöll stora osäkerheter, och även modellen för att prediktera medeldjup blev oanvändbar. Nu finns dock tillgång till nytt kartmaterial med högre upplösning än tidigare. Syftet med examensarbetet har varit att utforska om det är möjligt att uppnå säkrare resultat med det nya kartmaterialet, ta fram vilka kartparametrar som kan förbättra prediktionerna, och om det är möjligt, ta fram säkrare modeller för prediktering av sjövolym, max- och medeldjup.

Variabler till prediktionerna beräknades utifrån analyser med geografiska informationssystem (GIS) samt inhämtades från Svenskt vattenarkiv. Valet av vilka variabler som skulle användas baserades på tidigare studiers resultat samt litteraturstudier.

Utifrån inledande GIS-analyser tillsammans med de multivariata analysmetoderna PCA, PLS-regression och till slut multipel linjär regression kunde en säkrare modell än den som Sobek m. fl. (2011) fann för att prediktera volym i en enskild sjö tas fram (R^2=0.87, p-värde$<$0.00001). Vid tillbakatransformering av den predikterade volymen erhölls en lägre relativ standardavvikelse ($\pm$45 %) i jämförelse med tidigare studie ($\pm$57 %). De variabler som visade bäst korrelation med volymen var sjöarea och medianlutningen i en individuell zon kring varje sjö. En modell för maxdjup erhölls med något högre förklaringsgrad än tidigare (R^2=0.42) där de variabler som förklarade maxdjupet bäst var sjöarea, skillnaden i medelhöjd i närområdet kring sjön samt medianlutningen i en individuell zon kring varje sjö. Den individuella zonen är baserad på varje sjös storlek utifrån det längsta avståndet från mittpunkten i sjön till strandlinjen. Medeldjupet kunde inte predikteras från kartparametrar, men visade starka samband med maxdjupet. Det nya kartmaterialet tillsammans med den individuella zonen för varje sjö bidrog till ett säkrare resultat.

Nyckelord: Maxdjup, sjövolym, medeldjup, PCA, PLS-regression, GIS, multivariat analys, mulitpel linjär regression, SIMCA, Python.

Referens

Sobek, S., Nisell, J. & Fölster J. (2011). Predicting the volume and depths of lakes from map-derived parameters. *Inland Waters*, vol. 1, ss. 177-184.

Institutionen för vatten och miljö, Sveriges lantbruksuniversitet (SLU). Lennart Hjelms väg 9, Box 7050, 750 07 Uppsala. ISSN 1401-5765.

ABSTRACT

Modeling volume, max- and mean-depth in Swedish lakes - a statistical analysis with geographical information systems

Sara Sandström

Lake volume and lake depth are important variables that defines a lake and its ecosystem. Sweden has around 100 000 lakes, but only around 8000 lakes has measured data for volume, max- and mean-depth. To collect data for the rest of the lakes is presently too time consuming and expensive, therefore a predictive method is needed. Previous studies by Sobek et al. (2011) have found a model predicting lake volume from map-derived parameters with high degrees of explanation for mean volume of 15 lakes or more. However, the predictions for one individual lake, as well as max- and mean-depth, were not accurate enough. The purpose with this study was to derive better models based on new map material with higher resolution. Variables used was derived using GIS-based calculations and then analyzed with multivariate statistical analysis with PCA, PLS-regression and multiple linear regression. A model predicting lake volume for one individual lake with better accuracy than previous studies was found. The variables best explaining the variations in lake volume was lake area and the median slope of an individual zone around each lake (R^2=0.87, $p<0.00001$). Also, the model predicting max-depth from lake area, median slope of an individual zone around each lake and height differences in the closest area surrounding each lake, had higher degrees of explanation than in previous studies (R^2=0.42). The mean-depth had no significant correlation with map-derived parameters, but showed strong correlation with max-depth.

Keywords: Lake depth, lake volume, PCA, PLS-regression, GIS, multivariate analysis, multiple linear regression, SIMCA, Python.

Reference

Sobek, S., Nisell, J. & Fölster J. (2011). Predicting the volume and depths of lakes from map-derived parameters. *Inland Waters*, vol. 1, ss. 177-184.

Department of Aquatic Sciences and Assessment, Swedish University of Agricultural Sciences (SLU). Lennart Hjelms väg 9, P.O. Box 7050, SE-75007 Uppsala. ISSN 1401-5765.

FÖRORD

Det här examensarbetet motsvarar 30 hp och är slutprodukten efter fem års studier vid civilingenjörsprogrammet i miljö- och vattenteknik vid Uppsala universitet och Sveriges lantbruksuniversitet (SLU). Arbetet är utfört på institutionen för vatten och miljö, SLU, där Hampus Markensten har varit handledare. Elin Widén-Nilsson vid samma institution har varit ämnesgranskare. Mattias Winterdahl, universitetslektor vid Institutionen för geovetenskaper, Uppsala universitet, har varit examinator.

Detta examensarbete har utan tvekan varit den mest utmanande och lärorika perioden under min studietid i Uppsala och det finns många personer att tacka. Först och främst vill jag rikta ett stort tack till min handledare Hampus Markensten, vars hjälp och stöd under hela tiden har gjort arbetet möjligt, samt för ovärderlig hjälp med Python. Jag vill även tacka Elin Widén-Nilsson som har bidragit med värdefulla kommentarer och hjälp under rapportskrivandets gång. Tack till Lars Sonesten och Jens Fölster vid institutionen för vatten och miljö, SLU, för intressanta diskussioner om, och hjälp med PLS-regression och multipel regression. Till slut vill jag rikta ett tack till min familj och vänner, och där ett speciellt tack till mina vänner på SLU som varit ett stort stöd och bollplank under hela perioden.

Uppsala, juni 2017
Sara Sandström

POPULÄRVETENSKAPLIG SAMMANFATTNING

Sveriges yta består till ca 9 % av sjöarea, vilket motsvarar ungefär 100 000 sjöar. Sjöarna är en viktig del av det svenska landskapet och kulturlivet och gynnar människan genom till exempel naturupplevelser, fiske, bad och dricksvatten. För att sjöarna ska kunna utnyttjas på ett bra och hållbart sätt behöver de övervakas och provtas. Sjövolym, maxdjup och medeldjup är alla tre viktiga egenskaper som påverkar en rad andra faktorer i sjön. Om exempelvis ett föroreningsutsläpp sker i anslutning till en sjö och sjövolymen är känd, kan den användas för att beräkna hur lång tid det tar för föroreningen att transporteras genom sjön. Djupet i sin tur är starkt sammankopplat med skiktningen i sjön, en djupare sjö får tydligare skiktning än en grund sjö. Skiktning innebär att vattenmassorna läggs i skikt, exempelvis efter temperatur eller syrgashalt. Temperaturskiktning är vanligt, där bottenvattnet har en temperatur och ytvattnet en annan.

Då antalet sjöar i Sverige är så pass stort finns inte uppmätta värden på djup och volym för majoriteten. Att göra mätningar i alla de sjöar som saknar värden skulle vara väldigt tidskrävande och kostsamt, vilket gör att behovet av en alternativ metod uppstår. Tidigare försök att ta fram en modell för att förutsäga volym, max- och medeldjup i svenska sjöar har gjorts av Sobek m. fl. (2011). Deras modeller byggde på faktorer framtagna från kartmaterial över Sverige, såsom maxlutningen i närområdet kring sjön. De fann en modell över volym som kunde förutsäga medelvolymen för en grupp med minst 15 sjöar med hög säkerhet, medan om den användes för endast en sjö blev resultatet för osäkert. Modellerna för max- och medeldjup blev för osäkra för att vara användbara. Då det nu finns tillgång till nytt kartmaterial med högre upplösning, var syftet med det här examensarbetet att ta fram nya modeller för att förutsäga volym, maxdjup och medeldjup i svenska sjöar med, om möjligt, ett säkrare resultat.

En modell för att förutsäga sjövolym togs fram utifrån sjöarea och medianlutningen inom en individuell zon kring varje sjö. Modellen som togs fram visade högre säkerhet för förutsägelser för sjövolym i en enskild sjö än tidigare modell. Den individuella zonen togs fram utifrån maxavståndet från mittpunkten i sjön till strandlinjen, och det avståndet användes sedan ut från strandlinjen kring sjön, och bildade på så sätt en zon. I och med att alla sjöar har olika storlek varierar storleken på denna zon för varje sjö. Modellen som togs fram för att förutsäga maxdjup utgick från sjöarea, medianlutningen inom den individuella zonen samt skillnader i medelhöjd i närområdet kring sjön. Modellen fick ett säkrare resultat än tidigare, och sambanden är starka, men den skulle behöva genomarbetas mer för att förutsägelserna ska kunna vara pålitliga för alla mindre sjöar i Sverige. För medeldjup kunde endast en pålitlig modell beroende på maxdjup tas fram, och inga säkra samband med kartparametrar kunde hittas.

Att kunna förutsäga sjövolym i de sjöar där denna information saknas kan vara värdefullt för att exempelvis kunna beräkna uppehållstiden i en sjö, d.v.s. hur lång tid det tar för ett ämne att transporteras genom sjön. Om detta är känt är det lättare att avgöra hur sjön påverkas av inflöde av näringsämnen eller föroreningar. Djupet används också för att dela in sjöar i olika klasser, och på så sätt lättare kunna avgöra deras ekologiska tillstånd.

Arbetet utgick från laserskannat kartmaterial över Sverige med en storlek på 2x2 meter i plan med en avvikelse på mindre än 0.1 meter i höjd samt ett register över de sjöar

som har uppmätt data på sjövolym, maxdjup och medeldjup. Studien baserades också på kartmaterial innehållandes sjöar som polygoner, d.v.s. där varje sjö är representerad och uppritad på en digital karta. Alla sjöar som hade uppmätt information hade dock inte en uppritad polygon, vilket ledde till att endast de sjöarna med polygon användes i studien. För att kunna använda informationen och utföra beräkningar utifrån dessa kartor användes analyser med geografiska informationssystem (GIS), där informationen i kartmaterialet kan tillgodogöras. Utifrån polygonerna och kartan med höjdinformation kunde lutningen i flera bestämda områden kring varje sjö samt vilken höjd över havet sjön är placerad på beräknas. Utifrån detta kunde sedan flertalet statistiska mått tas fram. Detta inkluderade exempelvis max-, medel-, median- och minimumlutning kring sjön. Vilken typ av markanvändning det var i en zon kring varje sjö bestämdes också utifrån kartmaterial. Då alla dessa faktorer hade beräknats, och den önskade statistiken hade erhållits, fanns tillgång till information för totalt 5997 sjöar och 120 olika variabler, där en variabel exempelvis kan vara sjöarea eller maxlutning i en 100 m zon kring varje sjö.

För att kunna analysera en så pass stor mängd data krävs metoder som kan hitta strukturer i datamängden och göra den överskådlig. Till detta ändamål användes multivariata analysmetoder, d.v.s. analysmetoder som tar hänsyn till ett antal olika variabler samtidigt. I denna studie användes tre metoder för att analysera datamängden, i följande ordning: PCA, PLS-regression och till sist multipel linjär regression. PCA användes först för att få en initial överblick över datamängden och för att se om det fanns några variabler som stack ut och visade samband med varandra. Metoden används generellt för att hitta dolda samband mellan olika variabler eller för att hitta mönster mellan exempelvis olika sjöar. Efter detta användes PLS-regression. Denna metod går ut på att hitta vilka variabler som i högst grad påverkar den variabel som ska förutsägas. Med andra ord, vilka övriga variabler påverkar volymen, maxdjupet och medeldjupet? PLS-regressionen åskådliggör inte bara vilka variabler som visar samband med de variabler som önskas förutsägas, utan också vilka som påverkar mest och vilka som inte verkar ha något samband alls. Utifrån det resultat som hittades vid PLS-regressionen kunde de variabler som påverkade volymen, maxdjupet och medeldjupet väljas ut och användas i multipel linjär regression. Med denna metod kunde till slut modeller i form av ekvationer som beskriver alla dessa tre variabler tas fram. Med dessa tre metoder var det därmed möjligt att gå från en datamängd med 120 variabler till tre ekvationer med två variabler för att beskriva sjövolym och tre variabler vardera för att beskriva maxdjup och medeldjup.

Innehåll

1 INLEDNING **1**
- 1.1 SYFTE OCH FRÅGESTÄLLNINGAR . . . 2
- 1.2 TIDIGARE FORSKNING . . . 2
- 1.3 PÅVERKANDE FAKTORER OCH AVGRÄNSNINGAR . . . 3
- 1.4 HYPOTESER . . . 4

2 DATAMATERIAL OCH METODER **5**
- 2.1 DATA . . . 5
- 2.2 GEOGRAFISKA INFORMATIONSSYSTEM . . . 8
- 2.3 PRINCIPALKOMPONENTANALYS . . . 12
- 2.4 PLS-REGRESSION . . . 13
- 2.5 MULTIPEL LINJÄR REGRESSION . . . 15

3 RESULTAT **18**
- 3.1 PCA . . . 18
  - 3.1.1 PCA - alla responsvariabler . . . 19
  - 3.1.2 PCA - volym . . . 20
  - 3.1.3 PCA - maxdjup . . . 22
  - 3.1.4 PCA - medeldjup . . . 24
- 3.2 RESULTAT PLS-REGRESSION . . . 25
  - 3.2.1 PLS - volym . . . 25
  - 3.2.2 PLS - maxdjup . . . 28
  - 3.2.3 PLS - medeldjup . . . 31
- 3.3 MULTIPEL LINJÄR REGRESSION . . . 33
  - 3.3.1 MLR - volym . . . 33
  - 3.3.2 MLR - maxdjup . . . 36
  - 3.3.3 MLR - medeldjup . . . 38

4 DISKUSSION **40**
- 4.1 DISKUSSION - VOLYM . . . 40
- 4.2 DISKUSSION - MAXDJUP . . . 43
- 4.3 DISKUSSION - MEDELDJUP . . . 44
- 4.4 DISKUSSION METOD . . . 44
- 4.5 FÖRSLAG TILL VIDARE FORSKNING . . . 46

5 SLUTSATSER **47**

REFERENSER **48**

A APPENDIX **51**
- A.1 PYTHONKOD . . . 51
  - A.1.1 Skript för bestämda zoner . . . 51
  - A.1.2 Skript för individuella zoner . . . 56
  - A.1.3 Skript för höjdskillnader . . . 73
  - A.1.4 Skript för markanvändning . . . 80
- A.2 SKRIPT FÖR STATISTISKA ANALYSER . . . 86

ORDLISTA

Här nedan följer en ordlista som beskriver begrepp och ord som tas upp i rapporten.

Artificiell marktyp: Mark som är påverkad av människan, som exempelvis jordbruk eller stadsmiljö.

Autoskalning: Ett förbehandlingssteg där data ges variansen 1.

Centrering: Ett förbehandlingssteg som utförs för att ge data medelvärdet 0. För att uppnå detta subtraheras medelvärdet från alla datapunkter, vilket resulterar i att alla datapunkter erhåller medelvärdet 0.

Ekosystemtjänster: De funktioner som ett ekosystem tillhandahåller som gynnar människan.

Heteroskedasticitet: Betyder att residualerna är icke-linjära, och kan därmed uppvisa en konliknande form. Antingen att residualen, d.v.s. variansen, ökar med ökande y-värde, eller att den minskar med ökande y-värde.

Homoskedasticitet: Betyder att residualerna är konstanta över alla observationer, d.v.s. de är randomiserade och uppvisar ingen ökning med ökat y-värde.

Hypolimnion: Beskriver den undre, kallare vattenmassan i en skiktad sjö.

Korsvalidering: Ett sätt att estimera prediktionsfel som går ut på att en del av datasetet används för validering, d.v.s ena delen av datasetet används för prediktion och jämförs sedan med andra delen av datasetet. Detta upprepas för olika delar av datasetet.

Kovarians: Ett statistiskt mått på hur två stokastiska variabler samvarierar.

Morfologi: En sjös morfologi beskriver storleken och formen på sjön.

Objekt: Betyder i denna rapport scores, som används i PCA och PLS-regression för att beskriva raderna i X-matrisen, det vill säga de olika observationerna eller sjöarna.

Perimeter: Motsvarar omkretsen hos en sjö.

Prediktorvariabel: De variabler i en modell som används för att prediktera en annan variabel. I detta fall kan det exempelvis vara sjöarea eller maxlutning i närområdet kring sjön.

Responsvariabel: Den variabel i en modell som ska predikteras, i detta fall volym, maxdjup eller medeldjup.

Strandlinjeutveckling: Beskriver förhållandet mellan längden på strandlinjen och omkretsen på en cirkel med samma area som sjön, med andra ord hur mycket formen på sjön avviker från en cirkelform.

Stratifiering: Synonymt med skiktning. Används för att beskriva en sjö som har sepa-

rerade vattenlager med exempelvis olika temperatur eller syrehalt.

Topografi: Landskapets form.

Vikt: Betyder i denna rapport loading, som används i PCA och PLS-regression för att beskriva de olika variablernas betydelse i analysen.

1 INLEDNING

Sverige är ett land med drygt 100 000 sjöar som tillsammans upptar ca 9 % av hela ytan (SMHI, 2015). Sjöarna verkar både som kulturella och tillförande ekosystemtjänster, genom att bland annat fungera som badplatser och dricksvattenkällor, och är därför viktiga att studera. Ekosystemtjänster är de funktioner som ett ekosystem tillhandahåller som gynnar människan och kan delas in i olika kategorier där tillförande och kulturella är två (Millennium Ecosystem Assessment, 2005). Morfologin, d.v.s. storleken och formen, hos en sjö har stor inverkan på de processer som sker i sjön. Djupet och volymen är två viktiga faktorer som påverkar hur en sjö ser ut och fungerar.

Djupet är starkt kopplat till stratifieringen, skiktningen, i en sjö. Väldigt grunda sjöar har oftast ingen stratifiering alls, medan djupa sjöar kan vara konstant skiktade. Stratifieringen påverkar i sin tur flödet av näringsämnen till och från hypolimnion, det nedre vattenskiktet, samt syretillgången, vilket till exempel avgör typ av bottenfauna (Gorham & Boyce, 1989). Maxdjupet används också för att dela in sjöar i olika limniska typer, där även humushalt, kalkhalt och yta är påverkande parametrar. Indelningen görs för att kunna bedöma sjöar som avviker från referenstillstånd på ett korrekt sätt (Naturvårdsverket, 2007).

Sjödjupet påverkar också vilken volym en sjö har; om medeldjupet är känt kan volymen beräknas med hjälp av sjöarean (Håkanson, 2004). Volymen kan användas för att uppskatta uppehållstid och omsättningstid hos en sjö. Uppehållstiden är den tid det tar för ett ämne att transporteras genom en sjö, och kan beräknas utifrån volym och flöde (Persson m. fl., 2014). Omsättningstiden definieras som tiden det tar för vattnet i en sjö att helt bytas ut (Havs- och vattenmyndigheten, 2013). En känd volym gör det även möjligt att beräkna koncentrationer av eventuella föroreningar eller olika näringsämnen. Detta görs genom massbalansberäkningar baserat på uppmätta koncentrationer vid sjöns in- och utlopp.

I det svenska sjöregistret, som är en del av Svenskt vattenarkiv, SVAR, (SMHI, u.å.), finns tillgängliga data över volym och djup hos ca 8000 sjöar. Detta motsvarar ca 8 % av alla sjöar och innebär att majoriteten av alla sjöar saknar mätningar. Det vanligaste sättet att mäta sjödjup, och bestämma en sjös morfologi, är med ekolod. Denna metod är både tidskrävande och kostsam om den skulle tillämpas på alla sjöar i Sverige, vilket gör att alternativa metoder behövs för att bestämma max- och medeldjup samt volym hos de sjöar där den informationen saknas.

En metod som utgår från kartparametrar är ett bra alternativ för att på ett enkelt sätt prediktera volym och djup i sjöar där det inte finns så mycket tillgänglig djupdata. För att det ska bli möjligt att utföra för en mängd sjöar samtidigt krävs en enkel metod, där detaljkunskap om varje sjö inte är nödvändig. Tidigare försök att ta fram modeller där volym och djup kan beräknas utifrån kartparametrar har gjorts av bland annat Sobek m. fl. (2011). Studien utgick från kartmaterial över Sverige baserat på höjddata med upplösningen 50x50 m, där en modell togs fram som kunde beräkna medelvolymen hos en grupp av sjöar ($n > 15$) med hög säkerhet. För prediktering av volymen i en individuell sjö var säkerheten inte lika hög och en relativ standardavvikelse på $\pm$ 57 % erhölls. Den resulterande modellen för maxdjup innehöll stora osäkerheter och blev därmed inte användbar (Sobek m. fl., 2011).

Nu finns tillgång till ett nytt, bättre kartmaterial med höjddata som är baserat på ett 2x2 m-nät, och utifrån detta förutspås en förbättrad modell för volym och djup i en individuell sjö kunna tas fram.

1.1 SYFTE OCH FRÅGESTÄLLNINGAR

Syftet med examensarbetet var att utifrån det förbättrade kartmaterialet ta fram en säkrare modell för volym, max- och medeldjup.

Målet var att utifrån detta nya kartmaterial, föregående studier samt nya infallsvinklar ta fram en uppdaterad modell för att prediktera volymen, max- och medeldjupet i svenska sjöar. Modellerna tas fram för att komplettera de uppmätta data som finns över djup och volym, och för att på ett snabbt sätt kunna uppskatta max- och medeldjup samt volym för sjöar där data saknas.

Följande frågeställningar ska besvaras:

- Vilka kartparametrar kan förbättra prediktionen av volymen i en sjö jämfört med tidigare modell?
- Vilka kartparametrar kan förbättra prediktionen av max- och medeldjup i en sjö?
- Kan andra faktorer än tidigare konstaterade användas för att prediktera sjödjup och volym, och kan ett säkrare resultat uppnås?

1.2 TIDIGARE FORSKNING

Vilket max- och medeldjup en sjö har beror på många olika faktorer. Morfologin beskriver sjöns form och påverkas av avrinningsområdets topografi. Samma processer som formar topografin i närområdet kring sjön, formar även sjön (Hollister m. fl., 2011). Markanvändningen i närområdet samt i delavrinningsområdet har också en inverkan på sjöns morfologi, där artificiella marktyper kan orsaka bland annat ökad erosion samt högre tillförsel av näringsämnen och död ved. Detta i sin tur kan leda till ökad sedimentation av material till botten och därmed påverka djupet (Naturvårdsverket, 2007). Var sjöar bildas, vilken form samt vilket djup de får beror också på berggrundens brutenhet och jordarten på platsen. Ett mer kuperat landskap ger fler och små sjöar. Många av Sveriges sjöar har bildats i sprickor i berggrunden efter inlandsisen (SMHI, 2008).

Tidigare studier av Håkanson (2004) har med linjär regressionsanalys hittat starka korrelationer mellan maxdjup och medeldjup i sjöar, med bäst korrelation för logaritmerade data. Även relationer mellan medeldjup, siktdjup och fiskproduktion kunde fastställas.

Sobek m. fl. (2011) fann i sin studie att de viktigaste kartparametrarna kopplade till maxdjup var sjöarea, perimeter, strandlinjeutveckling samt maxlutningen i en 50 m-zon från strandlinjen. Studien visade också att vilken geografisk region sjöarna var lokaliserade i samt höjden över havet inte hade någon signifikant betydelse. Ingen signifikant korrelation hittades mellan medeldjup och kartparametrar, dock korrelerade medeldjupet starkt med maxdjupet. Sobek m. fl. (2011) presenterade en modell för prediktion av volymen utifrån kartparametrar där area, perimeter, strandlinjeutveckling samt max- och

minimumlutning i närområdet, var de viktigaste prediktorerna.

Mekanistiska samband mellan sjövolym och sjöarea beroende på jordens Hurst koefficient har även använts för att prediktera global sjövolym och medeldjup med relativt låga osäkerheter (Cael m. fl., 2017). Denna studie utfördes på global skala, där sjöarna delades upp i storleksklasser efter area och medelvolymen och medeldjup för varje klass samt den totala medelvolymen och medeldjupet beräknades.

Ytterligare studier för att ta fram en modell över maxdjup utifrån GIS-baserat material har utförts av Hollister m. fl. (2011) i USA. Liknande som Sobek m. fl. (2011) så antogs lutningen i närområdet kunna approximera lutningen på sjöbottnen. Utifrån detta, samt ett antagande om att djupet i en punkt i sjön är en funktion av avståndet till stranden, togs en ekvation fram. Ekvationen användes för att prediktera djupet på en godtycklig plats i sjön som en funktion av medianlutningen i närområdet samt avståndet till stranden. Här definierades närområdet som det område som omger sjön, som är både inom sjöns avrinningsområde samt inom ett specificerat avstånd från sjön. Det specificerade avståndet är olika för varje sjö och sattes till det längsta avståndet som hittats från stranden till mittpunkten i sjön. Maxdjupet bestämdes sedan till det största värdet som hittades då djupet i varje pixel i sjön beräknades. En korrigering av det predikterade maxdjupet utifrån uppmätta värden utfördes och utifrån detta togs en korrigeringsfaktor fram. Denna användes som ett komplement till modellen för att få bättre säkerhet i resultaten. De fick till slut en modell för maxdjup med ett MSE-värde (ekvation 14) på 5-6 meter.

1.3 PÅVERKANDE FAKTORER OCH AVGRÄNSNINGAR

Då denna studie utförs med svenska sjöar antas samma faktorer ha inverkan på maxdjupet som Sobek m. fl. (2011) fastställde. Det nya kartmaterialet erbjuder dock nya möjligheter. I den tidigare studien användes också maxlutningen i större zoner än 50 och 100 meter under analyserna, men den minsta tillgängliga zonen var den som gav starkast korrelation. Då det nya kartmaterialet kan ta fram zoner 2 meter från strandlinjen bör lutning inom mindre zoner testas för att se om detta kan ge bättre förklaring till djupet och volymen. Lutningen inom en mindre zon nära sjön bör ge en bättre indikation på hur lutningen fortsätter ner i sjön, åtminstone i små sjöar. Med det tidigare kartmaterialet var det inte heller möjligt att testa avstånd mellan 50 och 100 meter från strandlinjen, vilket innebär att även zoner på avstånd från strandlinjen mellan 50 och 100 meter kan korrelera med maxdjupet och volymen.

Delar av metoden som Hollister m. fl. (2011) använde kommer också testas här för att utforska om den är applicerbar på svenska sjöar. I den studien användes avrinningsområdets storlek i beräkningarna, vilket inte är tillgängligt för alla sjöar som används i denna studie. Det nya kartmaterialet har hög noggrannhet och därmed skulle flödesackumulationsberäkningar kunna utföras för att utifrån dessa kunna uppskatta avrinningsområdets storlek. Detta är dock ett stort arbete i sig och ryms inte inom tidsramen för detta projekt, därav görs avgränsningen att ej ta hänsyn till avrinningsområdets storlek. Zoner för lutningsberäkningar kan bestämmas utifrån det specificerade avståndet som är unikt för varje sjö, på liknande sätt som Hollister m. fl. (2011), men utan hänsyn till avrinningsområdets storlek. Zonerna som är av olika storlek för varje sjö hänvisas här till som individuella zonen (avsnitt 2.2).

Då området närmast strandlinjen har stor inverkan på sjöns form, är det intressant att undersöka om formen på området har någon inverkan på djupet eller volymen. Formen representerades dels av lutningen som tidigare nämnt, men även genom skillnader i medelhöjd. För skillnader i medelhöjd valdes det allra närmaste området kring strandlinjen (0–30 meter), se avsnitt 2.2 för närmare beskrivning. Som nämnt tidigare har markanvändning i närområdet inverkan på sjöns morfologi. Därför bör detta tas med i analysen, baserat på vilken typ av markanvändning som finns i närområdet. Närområdet klassificeras i detta fall som 500 meter från strandlinjen för att fånga in ett något större område kring varje sjö.

1.4 HYPOTESER

Utifrån tidigare studier formulerades följande hypoteser:

- Sjövolymen kan modelleras med hög säkerhet utifrån kartparametrarna perimeter, area, strandlinjeutveckling samt max- och minlutning i en zon $<$100 m kring sjön.
- Maxdjupet korrelerar signifikant med sjöarea, maxlutning inom en zon $<$100 m från strandlinjen, sjöperimeter samt strandlinjeutveckling.
- Medeldjupet korrelerar signifikant med maxdjupet, men inte med kartparametrar. Detta baserat på tidigare studiers resultat (Sobek m. fl, 2011).
- Jordbruksmark i närområdet (här ett 500 m brett område kring sjön) har en korrelation med volymen och djupet i sjön.
- Skillnader i medelhöjd i det närmaste området kring sjön har ett samband med djupet i sjön.
- Lutningar inom den individuella zonen kring varje sjö, inspirerad utifrån Hollister m. fl. (2011), visar starkare korrelationer med volymen och djupet än de bestämda zonerna. De bestämda zonerna är skapade utifrån ett bestämt avstånd från strandlinjen kring varje sjö, och är därmed lika för alla sjöar.

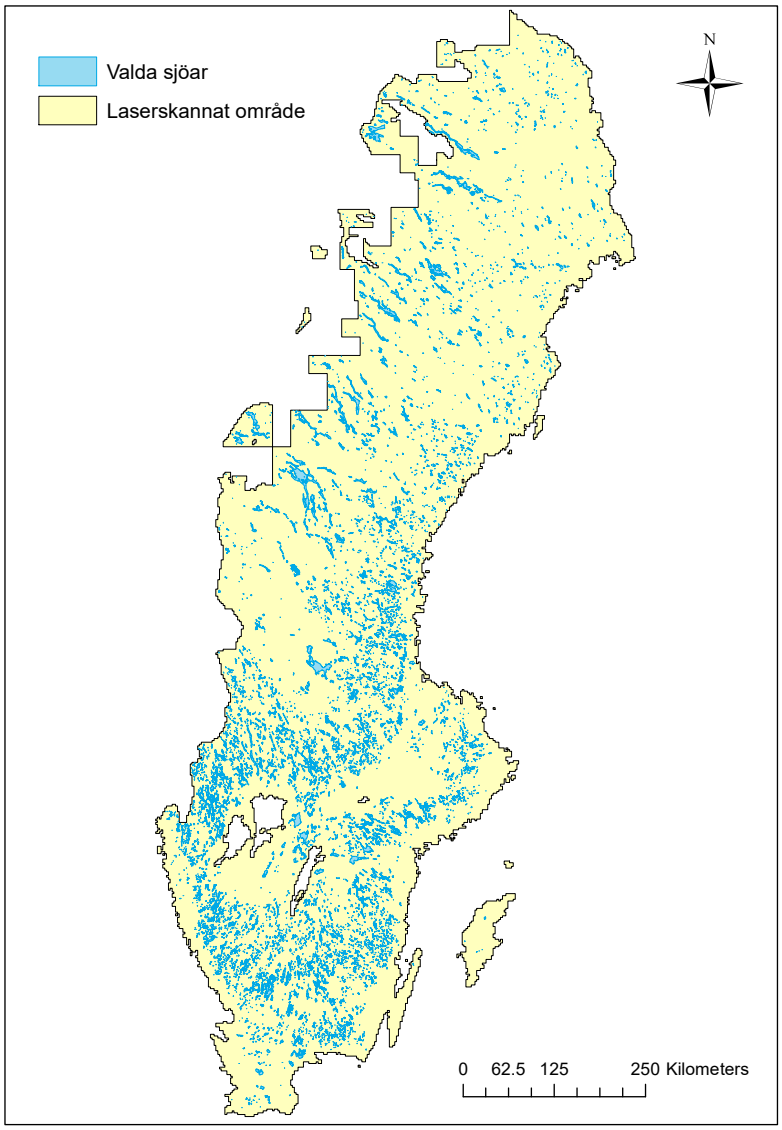
2 DATAMATERIAL OCH METODER

Valet av analysmetoder baserades dels på att tillgängliga data är rumsligt varierande samt att studien bygger på en stor datamängd. Både GIS-baserade metoder och multivariata analysmetoder är väl testade och passar bra för att hantera stora datamängder, vilket gör dem passande för denna studie. Sobek m. fl. (2011) använde även liknande metoder i deras analyser.

2.1 DATA

Det använda kartmaterialet över sjöar kommer från SVAR, där vattenförekomsterna är representerade som polygoner/vektordata i en shapefil med det projicerade koordinatsystemet SWEREF99 TM. Kända sjödjup och volymer hämtas också dessa från SVAR (SMHI, u.å.). De sjöar som valdes för analys har ett känt maxdjup eller medeldjup. Sjöarna som är med i analysen har en motsvarande sjöpolygon i SVAR, vilket resulterade i att vissa sjöar med kända data sorterades bort om de saknade sjöpolygon. Detta val gjordes då skapande av sjöpolygoner för alla sjöar som saknar detta inte ryms inom studiens tidsram. Sobek m. fl. (2011) använde även sjöpolygoner från vägkartan, vilket valdes bort i detta fall på grund av tidsbrist. De valda sjöarna finns utspridda över Sverige och ska därmed ge en representativ modell som kan användas på sjöar i hela Sverige (figur 1).

Det nya, uppdaterade kartmaterialet, GSD-Höjddata (Geografiska Sverigedata), kommer från Lantmäteriet i dataformen ASCII-grid i samma projektion som sjöpolygonerna, SWEREF 99 TM (Lantmäteriet, 2017). Det importerades till ett GIS-program som ett raster, till skillnad från sjöarna som är i vektorformat. GSD-Höjddatasetet är framställt med hjälp av laserskanning från flygplan. Materialet innehåller koordinatsatta höjdpunkter i ett tvåmeters rutnät. Lägesnoggrannheten för terrängmodellen är ca 0.1 m i höjdled och 0.3 m i plan. Noggrannheten är bättre över öppna ytor där avläsningen blir lättare än över ytor med mycket vegetation. Merparten av mätningarna har utförts utanför vegetationssäsongen för att få så bra mätningar som möjligt (Lantmäteriet, 2016). Då det skannade kartmaterialet saknar delar av Sverige, främst fjällområdet som gränsar mot Norge, sorterades sjöar i detta område bort ur analysen. Även de största sjöarna i Sverige uteslöts ur analysen, baserat på att de dels skiljer sig från de mindre sjöarna i form och utseende och dels att de redan är väl undersökta. Modellen är främst tänkt att användas på mindre sjöar där data inte finns tillgängligt i dagsläget, och därmed bör kalibreringsmängden innehålla just detta.



Figur 1: Blåmarkerade områden visar de sjöar som använts i analysen. Det gulmarkerade området visar de delar av Sverige som skannats in och representeras i GSD-höjddata. Vita områden är ej skannade. Datakälla till laserskannat område: Lantmäteriet (2016). Datakälla till sjöpolygoner: SMHI (u.å.).

Data över markanvändning hämtades från PLC6-kartan (*Pollution Load Compilation 6*) framtagen på uppdrag av Havs- och vattenmyndigheten. I PLC6-kartan har olika markanvändningstyper delats in i klasser enligt följande: tätort, skog, hygge, fjäll, vatten, hav, myr, öppen mark och jordbruksmark (Widén-Nilsson m. fl, 2016). Kartan har samma projektion som GSD-Höjddata och sjöpolygonerna samt är baserad på GSD-vägkartan med skala 1:100 000 (Widén-Nilsson m. fl., 2016).

Inför vidare analyser delades all statistik framtagen vid GIS-analyser upp i olika datamängder (tabell 1). Totalt användes data för 5997 unika sjöar och ursprungligen 120 prediktorvariabler. Alla data förbehandlades genom standardisering (centrering och skalning). För att undvika saknade värden gjordes analyser på fyra olika delmängder av hela

datamängden: 1) sjöar som hade data för alla responsvariabler, 2) sjöar med data för volym, 3) sjöar med data för maxdjup och 4) sjöar med data för medeldjup (tabell 1). Under analysens gång gjordes ytterligare en uppdelning av datamängden, där sjöarna delades upp efter sjöarea: sjöar med en area $<$10 km^2 lades i en datamängd och samma uppdelning efter volym, maxdjup och medeldjup som tidigare utfördes också vilket resulterade i data för 4527 unika sjöar (tavell 1). De större sjöarna lades i en separat datamängd som inte analyserades här. Denna indelning gjordes för att göra jämförelse med resultat från Sobek m. fl. (2011) möjlig då de använde samma avgränsning, samt att de erhållna modellerna är tänkta att användas på sjöar av denna storlek och därmed bör kalibreringsdata innehålla liknande sjöar, då större sjöar kan påverka modellernas prediktionskraft negativt. Data för samma sjöar användes även under de övriga modelleringsstegen. Innan analysen rensades data med markanvändningsareor bort, och endast markanvändningsandelar användes (se avsnitt 2.2 för beskrivning av framtagande av dessa). Detta resulterade i att antalet variabler som användes som ingång till de flesta analyserna var 97.

Tabell 1: Datamängder med de olika responsvariablerna använda under statistiska analyser. Kolumnerna i mitten beskriver de datamängder där alla tillgängliga sjöar är representerade, medan de två sista kolumnerna beskriver de datamängder då endast sjöar med en area $<$ 10 km^2 är representerade.

Responsvariabel	**Antal sjöar**	**Sjöarea (km^2)**	**Antal sjöar**	**Sjöarea (km^2)**
Alla	5142	0.008-456	5012	0.008-9.99
Volym	5330	0.008-456	5185	0.008-9.99
Maxdjup	5995	0.008-8309	5794	0.008-9.99
Medeldjup	5148	0.008-8309	5018	0.008-9.99

Inför PLS-regressionen och den multipla regressionen (beskrivna i avsnitt 2.4 och 2.5) delades datamängderna med area $<$ 10 km^2 upp i kalibrerings- och valideringsmängder så att valideringsmängden motsvarade ca 10 $\%$ av ursprungsmängden (tabell 2).

Tabell 2: Uppdelning av kalibrerings- och valideringsmängder inför PLS-regression.

Responsvariabel	**Antal sjöar, kalibrering**	**Antal sjöar, validering**
Volym	4669	516
Maxdjup	5237	557
Medeldjup	4519	500

2.2 GEOGRAFISKA INFORMATIONSSYSTEM

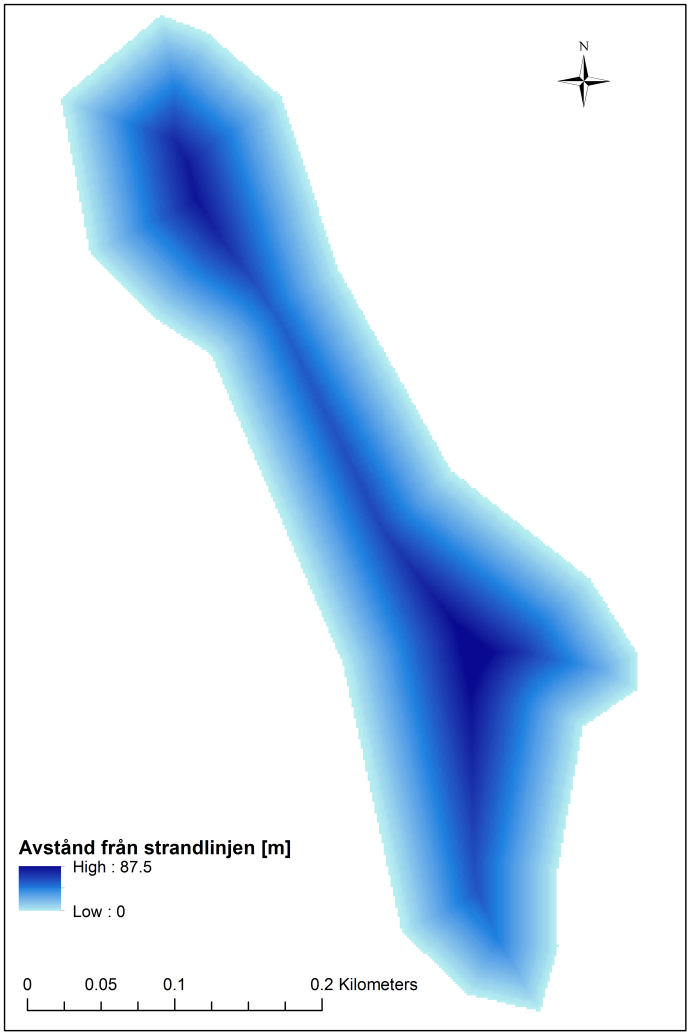
Geografiska informationssystem, vanligen benämnt GIS, är informationssystem som används för att hantera, visualisera och analysera lägesbundna data. Data består av kartor och tabellinformation där informationen i kartorna lagras som koordinater (Pilesjö, u.å.). Det finns ett antal olika program att välja på för att utföra GIS-analyser, och med hjälp av dessa kan olika typer av kartor produceras, och en mängd rumsliga analyser utföras. Med hjälp av olika verktyg kan avstånd mellan två punkter, areor, och mycket annat beräknas.

I denna studie användes programvaran ArcGIS 10.2, och därunder ArcMap 10.2 för analyser där inbyggda verktyg användes för de olika beräkningarna (tabell 3). Öppna GIS-bibliotek för Python användes också i beräkningarna, främst för statistiska beräkningar.

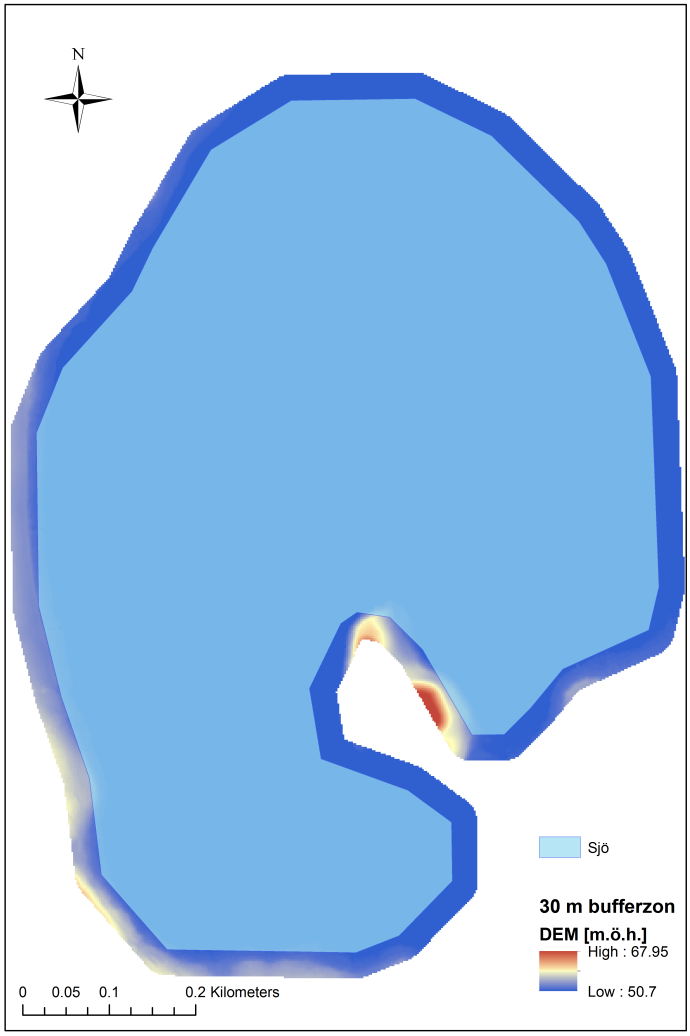
Varje sjö representeras som tidigare nämnts av en polygon. Först beräknades lutningen i grader i varje pixel inom en 1000 m radie kring varje sjöpolygon, detta för att undvika eventuella randeffekter vid lutningsberäkningar i mindre zoner. Då lutningen i alla pixlar beräknats skapades bestämda zoner inom vilka minimum-, max-, medel- och median samt 75-percentilslutning bestämdes. De bestämda zonerna var på följande avstånd från strandlinjen: 10, 20, 30, 40, 50, 60, 70, 80, 90 och 100 meter. För att skapa zoner med individuella avstånd för varje sjö på liknande sätt som Hollister m.fl. (2011) beräknades först avstånd från strandlinjen till olika punkter i sjön, (figur 2), och utifrån detta bestämdes det maximala avståndet (tabell 3). Det maximala avståndet användes sedan som zonavgränsning. För de individuella zonerna beräknades max-, minimum-, median- och medellutning (tabell 4). Alla zoner är hänvisade till som buffertzoner i texten.

Tabell 3: Använda verktyg under beräkningarna i ArcGIS, alla finns i någon av verktygslådorna (toolbox) Spatial analyst eller Analysis toolbox.

Beräkning	**Toolset (del av verktygslåda)**	**Verktyg**
Lutning	Surface toolset	Slope
Zonskapande	Proximity toolset	Buffer analysis
Klippning	Distance toolset/Overlay toolset	Extract by mask/Erase analysis
Avstånd	Distance toolset	Euclidian distance
Sammanläggning	Overlay toolset	Intersect



Figur 2: Exempel på ett raster med uträknade avstånd från strandlinjen. De mörkare blå partierna beskriver avstånd längst från strandlinjen. (Datakälla till sjöpolygon: SMHI (u.å.))



Figur 3: Exempel på buffertzon kring en sjö. Buffertzonen består av ett DEM-raster (Digital Elevation Model), som beskriver höjden över havet i varje pixel. Här är bredden satt till 30 m med start vid strandlinjen. (Datakälla till sjöpolygon: SMHI (u.å.). Datakälla till DEM-raster: Lantmäteriet (2016)).

Arean och perimetern beräknades utifrån sjöpolygonernas storlek (tabell 5). Strandlinjeutvecklingen hos varje sjö, L_d, bestämdes utifrån perimetern, P, samt sjöarean, A_{tot}, enligt ekvation (1) (Håkanson, 2004):

$$L_d = \frac{P}{2\sqrt{\pi A_{tot}}} \tag{1}$$

För att se hur formen på området närmast kring sjön kan påverka djupet och volymen användes höjdskillnader. För att representera närområdet omkring sjön skapades tre zoner, 0-10 m, 10-20 m samt 20-30 m (tabell 3), varefter minimum-, max-, median- och medelhöjden i varje zon beräknades (tabell 4). Skillnaden i medelhöjd mellan 0-10 och 10-20 meters zonerna samt mellan 10-20 och 20-30 m zonerna användes som representation för formen på närområdet och benämns som deltah_{1020} samt deltah_{2030} i rapporten.

Bestämning av hur markanvändningen i närområdet kring sjön ser ut gjordes utifrån kartmaterial beskrivet under avsnitt 2.1. Tre buffertzoner av storlekarna 200 m, 300 m och 500 m klipptes ur markanvändningskartan kring varje sjöpolygon, och buffertzonerna från markanvändningskartan samt sjöpolygonerna lades ihop till en karta (tabell 3 och 4). Varje marktyps areal och andel beräknades. Vid beräkning av andelar uteslöts vatten- och havsarealerna från totalarean, för att endast se hur stor andel de olika marktyperna representerar. Då PLC6-kartan ofta har en högre upplösning än sjöpolygonerna kan en viss förskjutning mellan kartorna ske. PLC6-kartan innehåller även mindre vattendrag och dessa kan då ge ett bidrag till vattenareal inom buffertzonen. Andra sjöar kan också hamna inom detta område och på så sätt bidra. Alla marktyper analyserades, men den marktyp som var av störst intresse här var jordbruksmark. Detta för att se om den särskilde sig och visade mer påverkan än någon annan marktyp. Buffertzonen klassades som jordbrukspåverkad om den innehöll >10 % jordbruksmark. Fler markanvändningstyper och dess andelar i varje buffertzon användes också i analysen för att se om även dessa gav en inverkan på djupet eller volymen.

Tabell 4: Alla beräkningar och behandlingar utförda med GIS-analyser. Subskriptet statistik hänvisar till vilket statistiskt mått som använts, ex-vis min eller max.

Beräkning	**Statistiskt mått**	**Zonstorlek, meter från strandlinje**	**Förkortning**
Lutning	Min, max, medel, median, 75-percentil	10, 20, 30, 40, 50, 60, 70, 80, 90, 100, indv.	$s_{statistikzonstorlek}$
Höjd	Min, max, medel, median	0-10, 10-20, 20-30	$höjd_{statistik10}$ $höjd_{statistik20}$ $höjd_{statistik30}$
Höjdskillnad	Skillnad i medelhöjd mellan zoner	Mellan: 0-10 och 10-20 10-20 och 20-30	$deltah_{1020}$ $deltah_{2030}$
Markanvändning	Andelar, areor, $>10\%$ jordbruksmark	100, 200, 500	andel $marktyp_{zonstorlek}$

Tabell 5: Variabler använda i analysen som ej beräknats utifrån olika buffertzoner.

Variabel	**Källa**	**Förkortning**
Sjöns höjd över havet	SVAR	v.y. höjd
Sjöarea	GIS-beräkning	A_{tot}
Perimeter	GIS-beräkning	P
Strandlinjeutveckling	Beräkning	L_d

Beräkningarna utfördes med hjälp av skriptspråket Python 2.7.3, där verktygen från ArcMap implementerades och automatiserades. Se appendix A.1 för använda skript.

2.3 PRINCIPALKOMPONENTANALYS

Principalkomponentanalys, PCA, är en metod för att analysera data, där metoden bygger på linjär algebra och används inom en mängd olika områden som exempelvis neurovetenskap och datorgrafik (Shlens, 2005). Det är en icke-parametrisk metod som används för att reducera komplexa datamängder till mer hanterbara former och hitta strukturer i dataserierna. Enligt Shlens (2005) utgår PCA från fyra antaganden. Det första är linearitet. Det går även att använda PCA på icke-linjära problem, men vanligast är ett antagande om linearitet. Antagande nummer två är att medelvärde och varians är tillräckliga statistiska mått för att beskriva datamängden. Det tredje antagandet är att stora varianser innehåller viktig dynamik. Det sista antagandet är att principalkomponenterna är ortogonala.

Metoden bygger på att stor varians kan innehålla viktig information. Datamängden representeras i ett vektorrum, en matris X, och reduceringen av komplexitet och dimensioner går i stort sett ut på att ett byte av vektorbas utförs genom att en ny bas bildas som är en linjär kombination av den gamla vektorbasen. En ny vektor etableras i samma riktning som den största variansen i X och bildar därmed den första basvektorn och principalkomponenten, PC1. En till vektor spänns upp, ortogonal mot den första, i den riktning med näst störst varians och bildar därmed den andra basvektorn och principalkomponenten, PC2 (Åmand, 2016). Ju fler dimensioner som finns, desto fler principalkomponenter kan bildas. I teorin kan det bildas en principalkomponent för varje dimension. Dock, då syftet med PCA är att reducera dimensionerna hos en stor datamängd till mer hanterbara nivåer, önskas så få principalkomponenter som möjligt utan förlust av viktig information. Oftast kan de två första principalkomponenterna göra just detta och förklara tillräckligt stor del av variansen i datamängden (Shlens, 2005). Då PCA också kan beskrivas som ett egenvärdesproblem innebär detta att varje principalkomponent, d.v.s. varje dimension, har ett egenvärde. Ju mindre egenvärde principalkomponenten har, desto mindre viktig information innehåller den, d.v.s. mindre varians är förklarad. Detta leder till att egenvärdet utgör ett bra verktyg för att analysera vilka principalkomponenter som behövs samt hur många (Joliffe, 1992).

Enligt Åmand (2016) utförs först en förbehandling av data i form av centrering och autoskalning vid tillämpning av PCA. Detta för att få en representativ bild av data samt för att analysen ska kunna fånga upp viktiga mönster och korrelationer. Detta är speciellt viktigt då datamängden innehåller variabler med olika enheter. Därefter delas datamängden upp i en matris, X, där raderna kallas för objekt, eller *scores*, i det här fallet olika sjöar. Kolumnerna i matrisen representerar olika variabler, som i det här fallet är exempelvis djup eller maxlutning i närområdet kring sjöarna. Då analysen genomförs bildas en ny matris, där varje variabels plats i matrisen får en vikt (*loading*). Detta representerar hur stor inverkan den platsen har på varje principalkomponent. Varje kolumn i denna matris bildar en egenvektor (Åmand, 2016).

Analys av resultatet från en PCA utförs genom studier av grafer av PC1 mot PC2 samt vikter och objekt. Objekt som har liknande variationsmönster kommer att grupperas tillsammans, vilket kan hjälpa till för att se om det finns mönster mellan olika typer av, i detta fall, sjöar. Dessa grafer kan också användas för att upptäcka extremvärden, som placerar sig långt ifrån de övriga sjöarna. Grafer över vikter studeras för att upptäcka korrelation mellan variabler. Enligt Åmand (2016) så tolkas två variablers vikter placerade nära varandra i samma kvadrant som positivt korrelerade med varandra, är de i kvadranten

diagonalt motsatt är de negativt korrelerade med varandra. Är de däremot vinkelräta mot varandra finns ingen märkbar korrelation mellan variablerna. Ju närmare origo variablerna hamnar, ju mindre vikt har de och därmed mindre inverkan på principalkomponenterna.

Här användes PCA för att analysera dataserierna framtagna vid beräkningar och analyser av kartmaterial samt övrig tillgänglig data från SVAR. Detta för att få en översikt över datamängderna och hitta eventuella strukturer inför vidare analyser. Först analyserades datamängderna med alla sjöar. Efter detta analyserades de datamängder innehållandes sjöar med en area < 10 km^2. Endast resultat från analyserna av sjöar med area < 10 km^2 presenteras. De statistiska analyserna utfördes i R version 3.3.1 samt i SIMCA-P 14. Graferna är producerade i SIMCA då de gav ett bättre visuellt resultat. Använda skript för PCA i R finns i appendix A.2.

2.4 PLS-REGRESSION

PLS, Partial Least Squares, eller Projection to Latent Structures (Wold, 1982) är en statistisk metod som används till liknande syften som PCA. Det finns olika PLS-metoder, och alla bygger på antagandet att observerade data genereras av ett system eller en process som drivs av ett litet antal dolda variabler. Datamängderna förbehandlas på samma sätt, och av samma anledning, som vid PCA, genom centrering och autoskalning (Rosipal & Krämer, 2006).

Vid PLS-regression är målet att utifrån ett antal variabler som är lättillgängliga, kunna prediktera andra variabler som inte är lika lättillgängliga. Det kan till exempel vara som i detta fall, att utgående från bland annat sjöarea och lutning i närområdet prediktera sjödjup eller sjövolym. Variablerna som används för att prediktera placeras i X-matrisen, och de variabler som ska predikteras placeras i en Y-matris. Det som skiljer PLS från PCA är att vid PLS är det kovariansen mellan X- och Y-matrisen som maximeras, och inte enbart variansen i X-matrisen som vid PCA. I PLS används en samling variabler, beskrivet i X-matrisen, för att prediktera en annan samling av variabler, Y-matrisen. Detta istället för att beskriva och hitta samband inom en uppsättning variabler, X, som vid PCA. Antalet variabler som ska predikteras kan variera, det kan vara bara en variabel, det kallas då PLS1, två variabler som då blir PLS2 eller fler variabler, PLS-regression (Rosipal & Krämer, 2006). I denna rapport användes PLS1, som är likt multipel linjär regression (avsnitt 2.5), men kan hantera variabler som samvarierar och därmed ta hänsyn till fler variabler samtidigt.

X- och *Y*-matriserna kan beskrivas enligt (2) och (3):

$$X = TP^T + E \tag{2}$$

$$Y = UQ^T + F \tag{3}$$

På samma sätt som i PCA används objekt och vikter. *T* och *U* betecknar objekt för respektive matris och *P* och *Q* betecknar vikter för respektive matris, medan *E* och *F* beskriver residualerna. Upphöjt T (^{T}) står för transponat. Här hittar dock PLS en objektvektor i *X*, som har maximal kovarians med en objektvektor i *Y*. Med andra ord så maximerar PLS kovariansen mellan objektvektorerna i X och Y (Rosipal & Krämer, 2006).

Vid analys av resultat används liknande grafer som vid PCA. För att visualisera hur variablerna i X förhåller sig till Y kan grafen med objekt som visar de första objektvektorerna från X och Y studeras (Sawatsky m. fl., 2015). Grafen med objekt studeras för att hitta grupperingar i X, samt extremvärden. Grafer med vikter kan studeras för att se korrelationerna mellan X- och Y-variablerna. Enligt Eriksson m. fl. (2006) är graferna konstruerade så att de variabler som får höga vikter, positiva eller negativa, och därmed är placerade långt ifrån origo uppvisar stark korrelation med Y-variabeln, där Y-variabeln i detta fall exempelvis är sjövolym. Det innebär att de variabler som placeras kring origo är dåligt korrelerade med Y-variabeln. För att tydliggöra korrelationen mellan prediktorvariablerna och Y-variabeln så kan en tänkt linje dras genom origo och Y-variabeln. Prediktorvariablerna projiceras sedan på denna tänkta linje (Eriksson m. fl., 2006). De variabler som hamnar på samma sida från origo som responsvariabeln har en positiv korrelation med Y-variabeln, medan de som hamnar på den andra sidan har en negativ korrelation (Eriksson m. fl., 2006). Prediktorvariabler som hamnar nära varandra uppvisar samvariation, d.v.s. korrelation med varandra, vilket innebär att de kan även ha liknande påverkan på Y-variabeln (Eriksson m. fl., 2006).

Ytterligare figurer som studeras vid PLS är VIP-grafer, där VIP står för *Variable Importance for the Projection* (Eriksson m. fl., 2006). Denna graf visar vilka variabler som är viktigast för att dels förklara X-matrisen och dels för korrelation med Y. Grafen är ett stapeldiagram med varje variabels VIP-värde. Enligt Eriksson m. f. (2006) innebär ett VIP-värde större än 1 att variabeln är viktig för modellen medan ett VIP-värde mindre än 0.5 tyder på att variabeln är mindre viktig. Variabler som får ett VIP-värde mellan 0.5-1 kan fortfarande ha en betydelse för modellen, beroende på storleken på datamängden (Eriksson m. fl., 2006).

Linearitet mellan respons- och prediktorvariablerna är viktigt för att få en bra modell. Om detta inte uppnås kan en transformering av data vara nödvändig. För att upptäcka icke-linearitet kan en graf av den första objektvektorn för X mot den första objektvektorn för Y studeras (Eriksson m. fl., 2006). Vid linearitet bildas ett ungefärligt 1:1 förhållande. Om detta inte uppnås kan en transformering förbättra modellen. En liten utspridning kring en tänkt 1:1 linje tyder på ett starkt samband mellan Y-variabeln och prediktorvariablerna (Eriksson m. fl., 2006).

Anpassning av en modell med PLS-regression och val av det optimala antalet komponenter följer någon av följande tre metoder enligt Åmand (2016), intern validering, korsvalidering eller testsets-validering. Här används korsvalidering. Som vid all modellering är modellvalidering en central del, att testa modellen på en annan datamängd än den modellen kalibrerats med är viktigt för att se hur bra prediktioner som modellen kan utföra. Vid val av antal komponenter är det viktigt att undvika överanpassning, där mer komponenter än nödvändigt används för att få en bättre anpassning, men som till slut leder till att modellen anpassar sig efter brus istället för efter data (Åmand 2016).

Då modellen är framtagen kan den uttryckas enligt följande regressionssamband (4):

$$Y_{pred.} = b_0 + b'_{PLS}X \tag{4}$$

där $Y_{pred.}$ är det som predikteras, b_0 är skärningen med y-axeln, X är prediktorn och b_{PLS} är modellkoefficienterna. För att avgöra hur bra modellen är, modelleringsgraden

och prediktionsgraden, används måtten R_y^2 och Q_y^2 som beskrivs enligt (Miljöstatistik, u.å.):

$$R_Y^2 = 1 - \frac{\sum F^2}{\sum Y^2} \tag{5}$$

$$Q_Y^2 = 1 - \frac{\sum prediktionsfel_Y^2}{\sum Y^2} \tag{6}$$

F är, enligt (3), residualen för Y-matrisen. Vid beräkning av Q_Y^2 används prediktionsfelet i $Y_{pred.}$ i jämförelse med uppmätt värde på Y. Prediktionsgraden fås då korsvalideringen utförs. R_Y^2 ökar med ökande antal komponenter, vilket gör den något missledande. Men genom att även se på Q_Y^2 som också ökar i början, men sedan minskar om modellen blir överanpassad så kan överanpassning undvikas. En modell med ett Q_Y^2-värde över 0.5 anses generellt som en användbar modell (Eriksson m. fl., 2006). Dock är detta beroende på tillämpningsområde, och även modeller med lägre värden på Q_Y^2 kan vara användbara. En bra modell bör ej ha en större skillnad mellan R_Y^2 och Q_Y^2 än 0.2-0.3 (Eriksson m. fl., 2006). För att avgöra hur väl modellen beskriver variansen i X-matrisen kan också R_X^2-värdet användas, som är motsvarande R_Y^2, fast för prediktorvariablerna.

Tolkning av PLS-regression sker grafiskt. Analyserna utfördes i SIMCA-P 14 samt i programmet R. Skript för PLS-regression i R finns i appendix A.2. Endast resultat producerat i SIMCA-P redovisas då det gav bättre visuellt resultat.

2.5 MULTIPEL LINJÄR REGRESSION

Multipel linjär regression, MLR, har sitt ursprung i vanlig linjär regression, där linjära orsaks-respons samband modelleras. Detta modelleras även med MLR, men till skillnad från enkel linjär regression används flera orsaksvariabler. Det man vill uppnå med MLR är att kunna förklara så mycket som möjligt av variansen i responsen, för att minimera det som endast kan beskrivas som bakgrundsbrus. En MLR-modell beskrivs generellt enligt (7) (Helsel & Hirsch, 2002).

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k + \epsilon \tag{7}$$

där y är responsvariabeln, β_0 är skärningen med y-axeln, β_i, i=1,2..,k, är lutningskoefficienten för de olika orsaksvariablerna x_i och ϵ är den varians som ej går att förklara med modellen, felet. Då linjär regression används görs ett antal antaganden: att variansen hos residualerna är konstanta över alla observationer (homoskedastiska), att de är oberoende av varandra samt normalfördelade. Vid linjär regression antas inte normalfördelning hos respons- eller orsaksvariablerna, endast dess residualer (Helsel & Hirsch, 2002). Detta gör residualanalys till en viktig del i analysprocessen. Om residualerna är heteroskedastiska, d.v.s. uppvisar konliknande form, kan transformering av datamängden användas för att uppnå homoskedasticitet (Helsel & Hirsch, 2002). En vanlig transformering är log-transformering, med exempelvis den naturliga logaritmen eller logaritmen med basen 10. Då detta används för att uppnå en bättre fördelning av residualerna, ska tillbakatransformering ske med försiktighet. För att inte få tillbaka samma skevhet i residualerna vid tillbakatransformering av det slutliga predikterade värdet kan en korrigering utföras (ekvation 8 och 9) (Helsel & Hirsch, 2002). Denna tillbakatransformering gäller vid transformering med den naturliga logaritmen, ln.

$$Y_{korr.} = exp(ln(Y_{pred.})) \times exp(0.5s^2) \tag{8}$$

$$s^2_{korr} = (exp(ln(Y_{pred.})) \times exp(0.5s^2))^2 \times (exp(s^2) - 1) \quad (9)$$

$Y_{korr.}$ är det korrigerade, predikterade värdet på responsvariabeln, $ln(Y_{pred.})$ är det predikterade värdet som erhålls då logaritmerade data används, s^2 är variansen i residualerna hos $ln(Y_{pred.})$ och $s^2_{korr.}$ är variansen i residualerna hos $Y_{korr.}$. För att ge ett mått på spridningen och variansen i $Y_{korr.}$ kan den relativa standardavvikelsen användas, RSD (ekvation 10), där $s^2_{korr.}$ relateras till medelvärdet av $Y_{korr.}$, $\overline{Y}_{korr.}$.

$$RSD = \frac{s^2_{korr}}{|\overline{Y}_{korr.}|} \times 100 \quad (10)$$

Det finns ett antal olika sätt och mått för att avgöra vilken modell som bäst beskriver responsvariabeln, exempelvis mellan modeller med olika prediktorvariabler samt olika antal prediktorvariabler. Ett mått på modellsäkerhet som används här är R^2-värdet som beskriver hur stor del av variansen som förklaras av regressionen och kan beskrivas enligt (11) (Helsel & Hirsch, 2002):

$$R^2 = \frac{SS_y - s^2(n-2)}{SS_y} = 1 - \frac{SSE}{SS_y} \quad (11)$$

där *n* är antalet mätningar eller observationer, s^2 står för standardavvikelsen, SS_y står för summan av kvadrater i y enligt:

$$SS_y = \sum_{i=1}^{n} (y_i - \overline{y})^2 \quad (12)$$

varav $\overline{y}$ är medelvärdet av y_i. Slutligen står SSE för kvadratsumman av alla fel. Ett R^2-värde på 1 innebär att modellen beskriver datamängden perfekt. Det finns dock ett problem med R^2-värdet som mått på modellsäkerhet då multipel regression används. För varje variabel som läggs till ekvationen ökar R^2-värdet, oavsett om den variabeln adderar till förklaringsgraden eller ej. Därför är det justerade R^2-värdet, $R^2_{a,}$ ett bättre mått vid MLR (13). Det justerade värdet tar hänsyn till minstakvadratfelet, MSE (ekvation 14, och ökar därmed när minstakvadratfelet minskar (Helsel & Hirsch, 2002).

$$R^2_a = 1 - \frac{MSE}{(SS_y/(n-1))} \quad (13)$$

$$MSE = \frac{1}{(n-2)} \sum (y_i - y_{pred})^2 \quad (14)$$

R^2_a-värdet beräknas automatiskt vid analyser i programmet R. Det är också viktigt att ha i åtanke att R^2-värdet kan ge en felaktig bild av modellen om residualerna inte uppfyller tidigare nämnda antaganden.

Det mått som används för att bestämma om modellen är signifikant eller ej är p-värdet. Ett vanligt val, som även användes här, är en säkerhet med ett 95 %-konfidensintervall, vilket motsvarar ett α-värde på 0.05. För en signifikant modell ska p-värdet vara strikt mindre än värdet på α, vilket här innebär ett p-värde på <0.05 för en signifikant modell. Ett p-värde kan även tas fram för varje enskild variabel i modellen, detta för att avgöra om variabeln har ett signifikant bidrag till modellen.

Det finns ett antal sätt att beskriva osäkerheter i modellen där minstakvadratfelet är ett. Ett annat sätt är det absoluta medelfelet (ekvation 15), som beskriver medelvärdet av de absoluta prediktionsfelen. Variansen och standardavvikelsen av prediktionsfelen är också vanliga mått på osäkerhet.

$$|\overline{pred.fel}| = \frac{1}{n}\sum_{i=1}^{n}|y_i - y_{pred.i}| \tag{15}$$

Om flera modeller av olika komplexitet erhålls och det ska avgöras om den mer komplexa modellen, med fler antal variabler, tillför högre förklaringsgrad till y än den enklare modellen, kan ett F-test utföras. F-värdet, som här ej beskriver residualen som i PLS, beräknas enligt (16) (Helsel & Hirsch, 2002):

$$F = \frac{(SSE_s - SSE_c)/(df_s - df_c)}{(SSE_c/df_c)} \tag{16}$$

där underskriften s betecknar den enklare modellen och c den mer komplexa modellen. F-värdet som erhålls vid analysen jämförs med ett tabellerat F-värde med (df_s - df_c) och df_c antal frihetsgrader för det valda värdet på α, här 0.05. Om det erhållna värdet överstiger det tabellerade värdet så ska den mer komplexa modellen väljas (Helsel & Hirsch, 2002). F-värdet erhålls automatiskt vid analyser i programmet R och skrivs som $F_{df s, df c}$.

Här utfördes multipel linjär regression i programmet R baserad på de orsaksvariabler som identifierades som viktigast och som inte uppvisade samvariation vid PLS-regression. Detta då MLR ej kan hantera variabler som samvarierar, till skillnad från PLS-regression. I de fall där samvarierande variabler visade starkast påverkan på Y-variabeln vid PLS-regressionen testades alla en i taget och den som bidrog med högst förklaringsgrad till Y-variabeln valdes. Alla identifierade variabler testades enligt en stegvis bakåt process, där variabler tas bort, och förändringen i förklaringsgrad relaterat till antal variabler studeras. Se Helsel & Hirsch (2002) för närmare beskrivning. MLR användes för att ta fram slutgiltiga modellekvationer för varje responsvariabel.

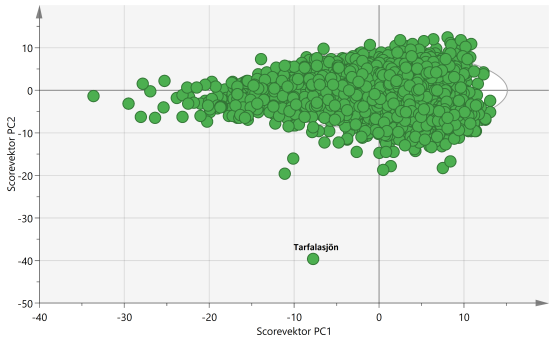
Olika datamängder till kalibrering och validering av modell användes (tabell 2). Valideringsmängderna användes för att testa modellen och avgöra prediktionskraften.

3 RESULTAT

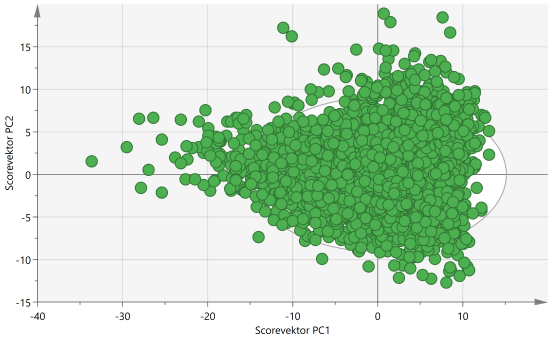
I de följande avsnitten betecknas lutningen med $s_{statistik\,zonstorlek}$ i figurer, skillnaden i medelhöjd med deltah$_{1020}$ samt deltah$_{2030}$, sjöns meter över havet med v.y. höjd, perimeter med P, strandlinjeutveckling med L_d, sjöarea med A_{tot} och de olika markanvändningarnas zonstorlekar nedsänkt inom parantes.

3.1 PCA

Vid alla analyser kontrollerades datamängderna efter extremvärden. En av sjöarna visar tecken på att vara avvikande då dess objekt placerades långt ifrån de övriga (figur 4). Sjön identifierades som Tarfalasjön. Den studerades närmare och sticker ut med ett stort maxdjup och medeldjup. En bättre fördelning utan extremvärden erhölls då sjön togs bort ur analysen (figur 5). Analyserna utfördes inkluderat och exkluderat denna sjö för jämförelse.



Figur 4: Graf över scores från PCA med volym, maxdjup och medeldjup, endast sjöar med area < 10 km^2.

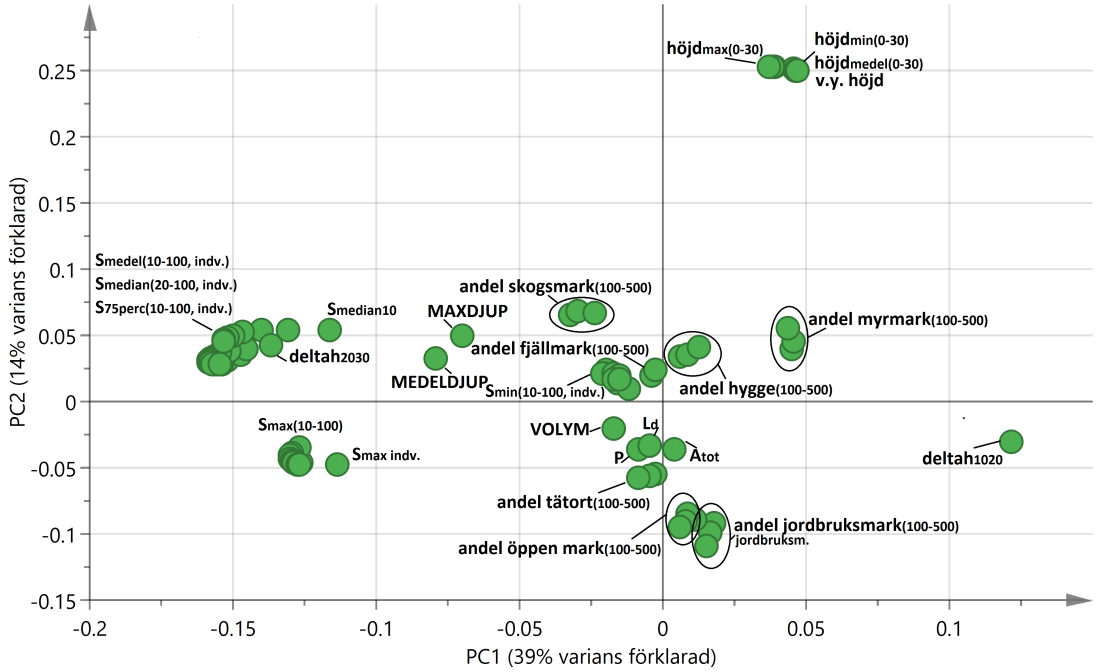


Figur 5: Erhållna objekt från PCA med volym, maxdjup och medeldjup, endast sjöar med area < 10 km^2, här exkluderat extremvärde.

Samma sjö placerades som ett extremvärde vid alla analyser, men då ingen av analyserna i övrigt påverkades nämnvärt om sjön var inkluderad eller inte, så redovisas här endast resultat för de fall då sjön är inkluderad.

3.1.1 PCA - alla responsvariabler

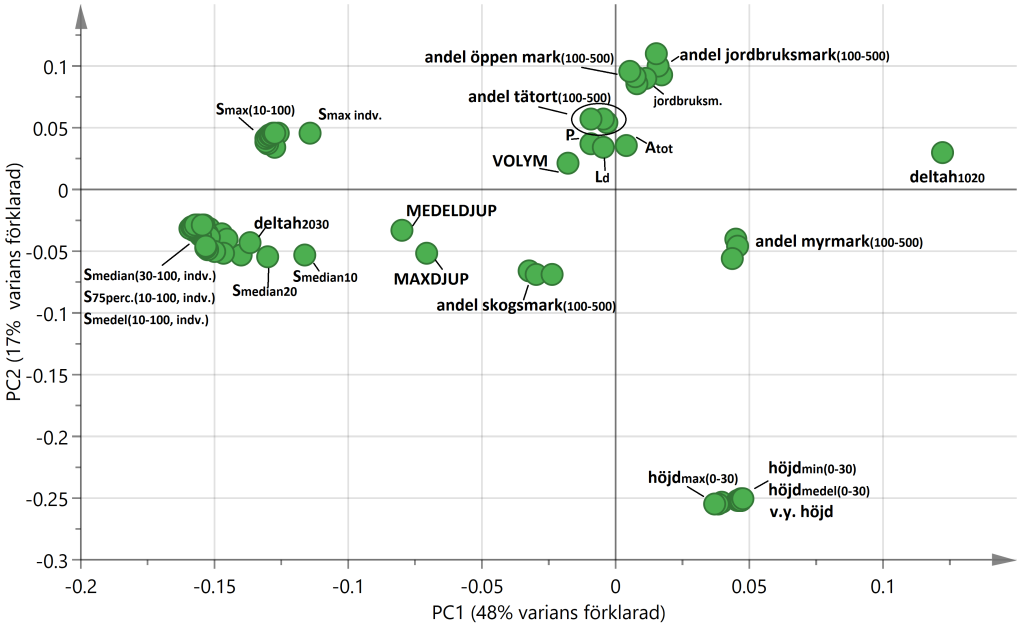
Korrelation mellan volym, strandlinjeutveckling och perimeter blev tydlig då de fick liknande vikter och placerades nära varandra (figur 6). Även maxdjup och medeldjup uppvisade korrelation med varandra.



Figur 6: Vikter från PCA med volym, maxdjup och medeldjup, endast sjöar med area < 10 km^2. Alla variabelförkortningar förklaras i tabell 4 och 5. Antal variabler i analysen var 97.

Den totala variansen som förklaras av de två första principalkomponenterna blev 53 %. Samma resultat erhölls då PCA utfördes på datamängden som också inkluderade sjöar med större area än 10 km^2.

För att se om en större varians kan förklaras av de två första principalkomponenterna rensades de variabler som gav minst påverkan i form av låga vikter bort. Detta gällde framförallt minimumlutningarna i alla buffertzoner samt andel hygge och andel fjällmark (figur 7).

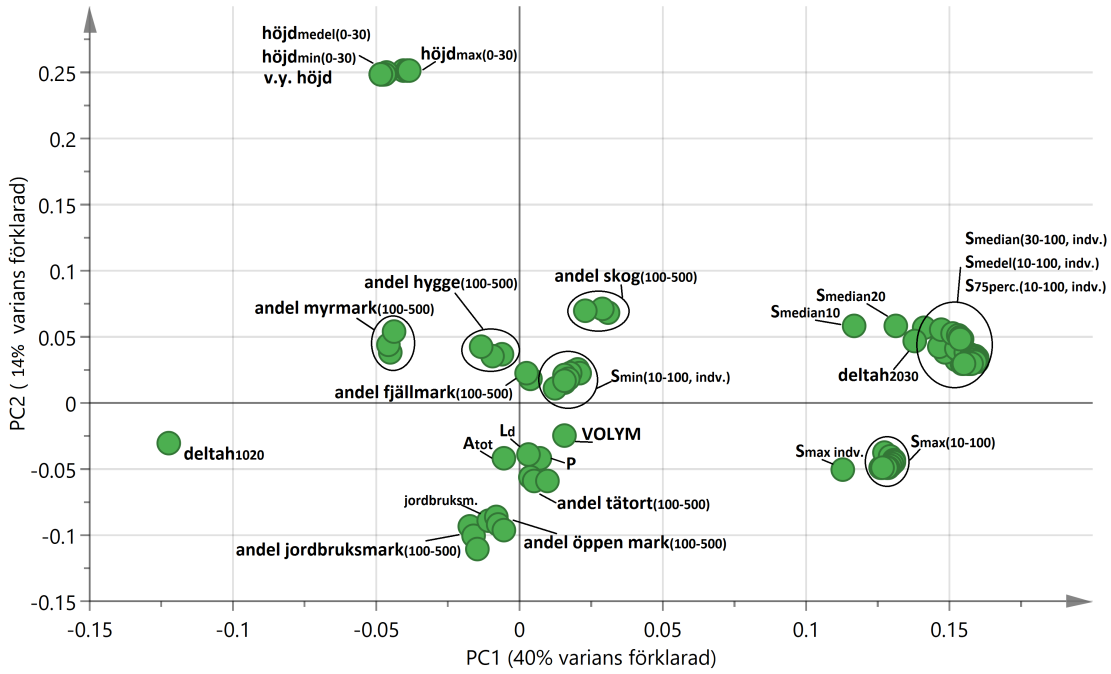


Figur 7: Vikter för alla variabler från PCA med volym, maxdjup och medeldjup, endast sjöar med area < 10 km^2. Alla variabelförkortningar förklaras i tabell 4 och 5. Här har variabler som visat minst påverkan på principalkomponenterna plockats bort, vilket ledde till att det totala antalet variabler var 80.

Efter rensningen blev korrelationerna tydligare, och förklaringsgraden ökade till 65 %. Max-, min- och medelhöjderna inom buffertzonerna 0-10, 10-20 samt 20-30 meter samt sjöns höjd över havet visade stor påverkan på principalkomponenterna (längst ned till höger i figur7). En antydan till negativ korrelation med volymen kan ses då de placeras diagonalt mot varandra, men ingen korrelation mellan höjderna och medel- eller maxdjup hittas. En antydan till korrelation mellan max- och medeldjup och andel jordbruksmark kan ses då de hamnar diagonalt mot varandra.

3.1.2 PCA - volym

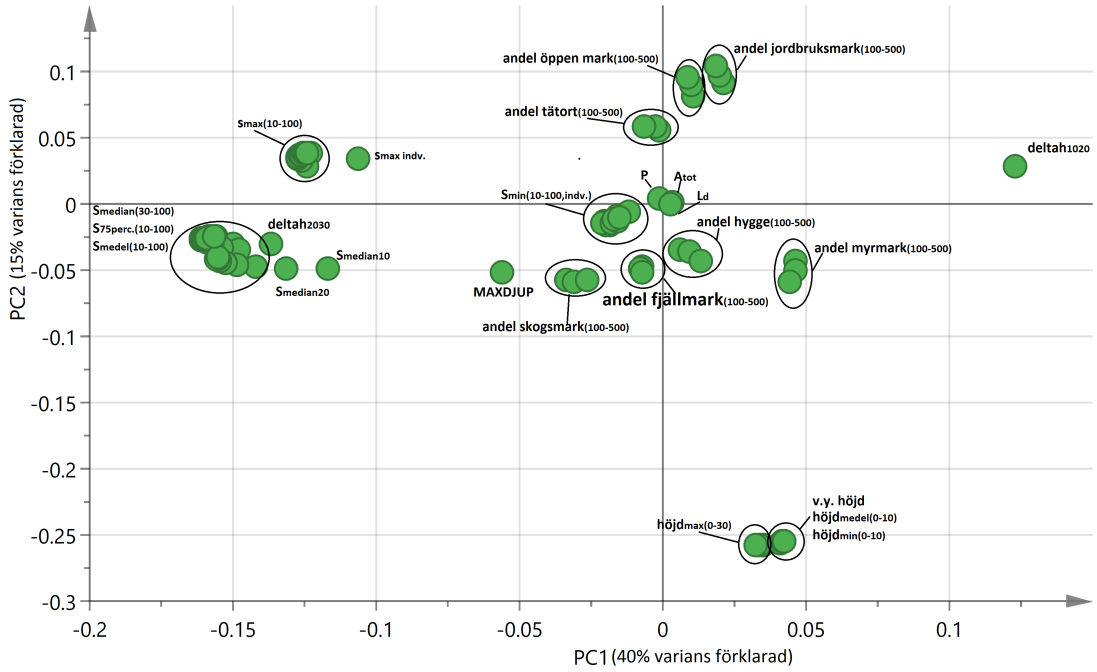
Sjöns höjd över havet samt höjderna inom 0-10, 10-20 och 20-30 m zonerna placerades diagonalt mot volymen och uppvisade därmed tendens till negativ korrelation med volymen (högst uppe i figur 8). Maxlutningarna visade också korrelation med volymen, men här positiv (längst ned till höger). Strandlinjeutvecklingen, perimetern samt sjöarean samvarierar med volymen och placerades därmed nära volymen (figur 8). De två första principalkomponenterna förklarar 54 % av variansen i datamängden.



Figur 8: Vikter för variabler från PCA med volym, endast sjöar med area < 10 km^2. Alla variabelförkortningar förklaras i tabell 4 och 5. Antal variabler var 95.

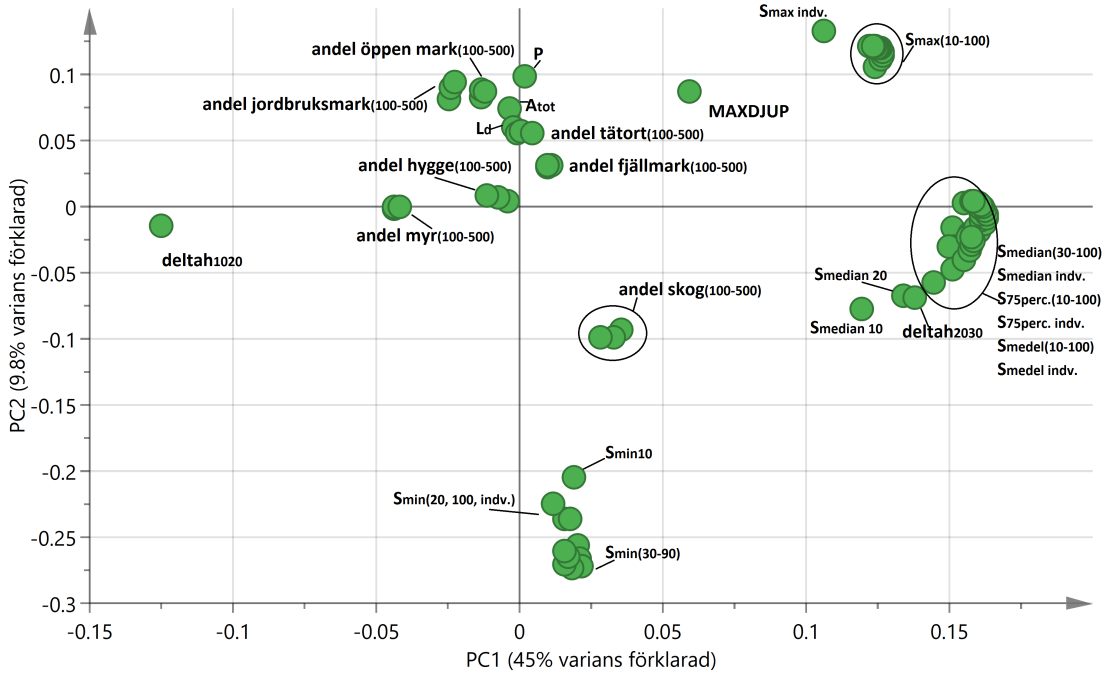
3.1.3 PCA - maxdjup

Max-, min- och medelhöjderna i buffertzonerna närmast strandlinjen, samt sjöns höjd över havet, har stor påverkan på principalkomponenterna (längst ned, figur 9). Dock hittades ingen korrelation med maxdjupet då de är placerade i kvadranten bredvid maxdjupet. Andel skogsmark hamnar ganska nära maxdjupet och kan därmed ha en viss korrelation med maxdjupet. De olika lutningarna placeras för långt ifrån maxdjupet, och bredvid, för att uppvisa tydlig korrelation.



Figur 9: Vikter för variabler från PCA med maxdjup, endast sjöar med area < 10 km^2. Alla variabelförkortningar förklaras i tabell 4 och 5. Totalt antal variabler var 94.

För att se om höjderna hindrar andra variabler från att komma fram i analysen rensades de bort och analysen gjordes om (figur 10).

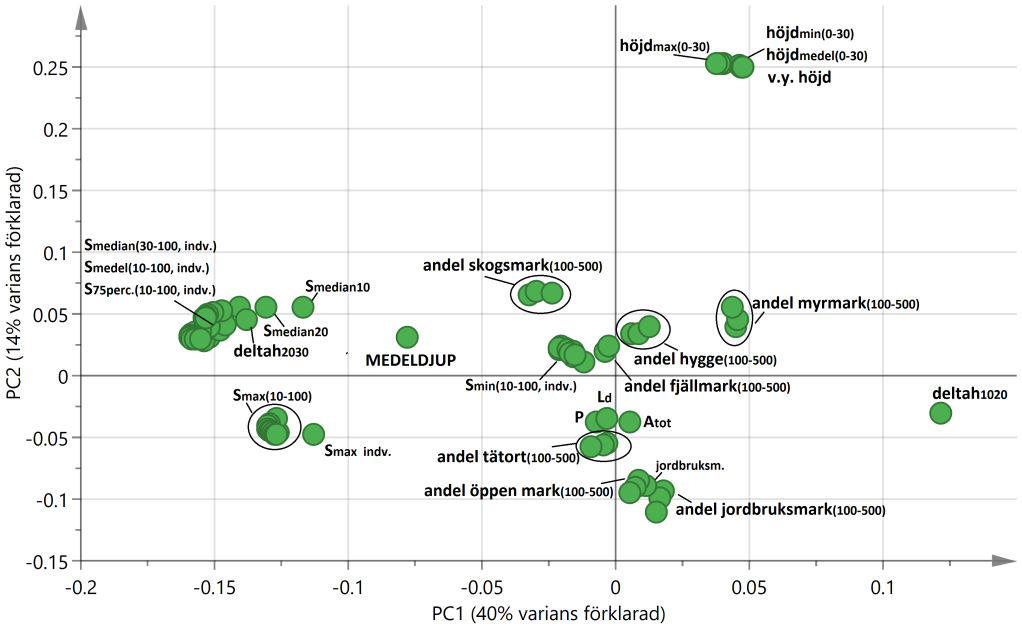


Figur 10: Vikter för variabler från PCA med maxdjup, endast sjöar med area < 10 km^2. Alla variabelförkortningar förklaras i tabell 4 och 5. Här har de variabler som hade minst vikt rensats bort vilket ledde till att antal variabler var 81.

Efter rensningen visar maxdjupet positiv korrelation med maxlutningarna dels inom den individuella zonen och dels de bestämda zonerna. Även andel fjällmark visar viss positiv korrelation med maxdjupet. Skillnaden i medelhöjd mellan 0-10 och 10-20 meters zonerna däremot visade negativ korrelation med maxdjupet.

3.1.4 PCA - medeldjup

Medeldjupet uppvisade positiv korrelation med 75-percentils-, medel och medianlutningarna i de olika bestämda buffertzonerna, samt med skillnaden i medelhöjd mellan 10-20 och 20-30 meters zonerna (figur 11). En viss negativ korrelation hittades också med skillnaden i medelhöjd mellan 0-10 och 10-20 meters zonerna. Även här har min-, max- och medelhöjden i buffertzonerna 0-10, 10-20 och 20-30 meter samt sjöns höjd över havet stor påverkan på principalkomponenterna då de får höga vikter (högst upp i figur 11), men ingen korrelation mellan höjderna och medeldjupet kunde fastställas då de inte placeras nära varandra eller i diagonalt motsatta kvadranter. Median-, 75-percentils och medellutningarna i de olika buffertzonerna visar en eventuell antydan till korrelation med medeldjupet då de placeras relativt nära medeldjupet (figur 11). De två första principalkomponenterna förklarar 54 % av variansen i datamängden.



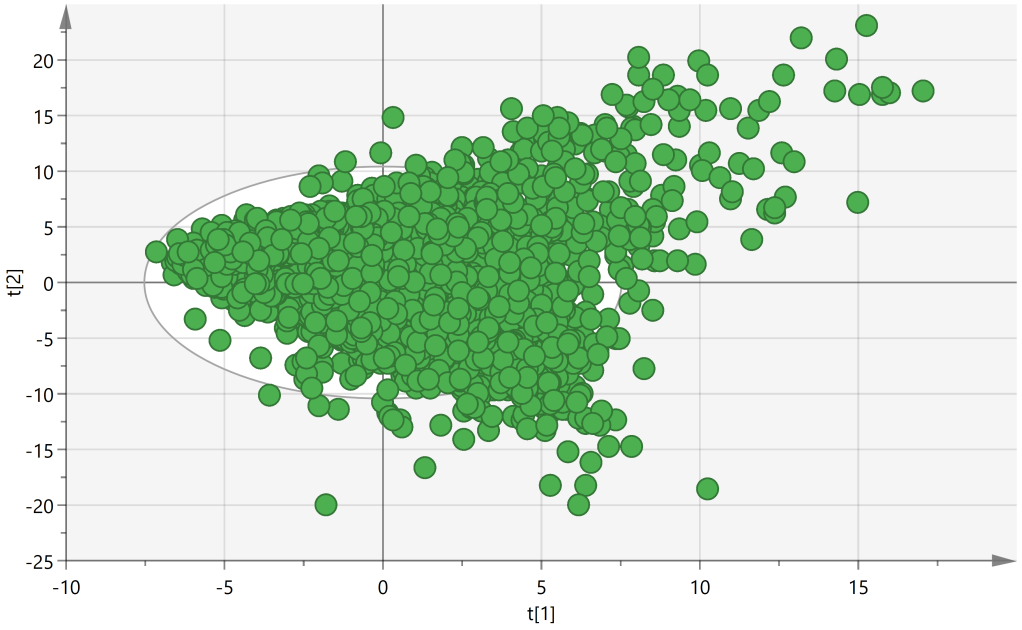
Figur 11: Vikter för de olika variablerna från PCA med medeldjup, endast sjöar med area < 10 km^2. Alla variabelförkortningar förklaras i tabell 4 och 5. Totalt antal variabler var 95.

3.2 RESULTAT PLS-REGRESSION

PLS-regressionen utfördes på datamängderna med volym, maxdjup och medeldjup beskrivna i tabell 1. Utgångspunkten var resultatet från PCA av dessa datamängder. Extremvärdet som identifierades vid PCA (figur 4) behölls även vid PLS-regressionen då den inte stack ut tillräckligt mycket för att det skulle vara motiverat att ta bort den. Främst resultat från PLS-regressionen av datamängderna med sjöar med en area mindre än 10 km^2 redovisas här. PLS-regressionen utförd i R redovisas ej då samma resultat erhölls som i SIMCA. Det som avvek vid PLS-regressionen i R var värden på Q^2 som i R blev någon decimal högre i alla analyser. Det optimala antalet PLS-komponenter tas fram automatiskt av SIMCA vid analysen.

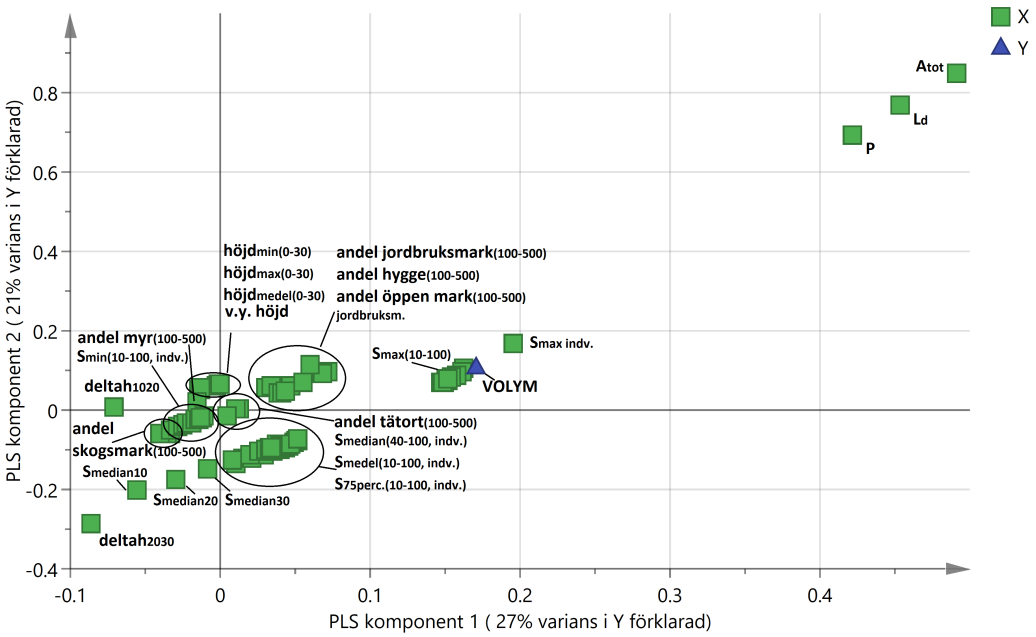
3.2.1 PLS - volym

En del extremvärden, mer eller mindre extrema, hittades vid PLS-regressionen av datamängden med volym (figur 12). Ingen av dessa urskiljer sig dock tillräckligt mycket ur datamängden för att motivera en uteslutning ur analysen. Tarfalasjön skiljde inte ut sig som ett extremvärde, till skillnad mot vid PCA (figur 4).



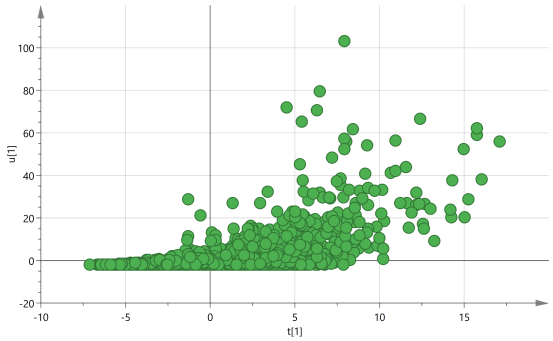
Figur 12: Objekt för de två första PLS-komponenterna för data över volym, här endast med sjöar med area mindre än 10 km^2. t står för objektvektor.

Den tydligaste positiva korrelationen med volymen i både första och andra PLS-komponenten är sjöarea, strandlinjeutveckling och perimeter (figur 13). Även maxlutningen i den individuella zonen, $s_{maxindv.}$, korrelerar positivt med volymen. Skillnaden i medelhöjd mellan 10-20 och 20-30 m zonerna visar en negativ korrelation med volymen både i första och andra komponenten (figur 13). Till skillnad från vid PCA så har de olika höjderna här en liten inverkan på datamängden och placerar sig kring origo i figuren. Median- och medellutningar visar viss påverkan på volymen. En modell med 9 PLS-komponenter fick R^2_X=0.811, R^2_Y=0.677 och Q^2_Y=0.654.

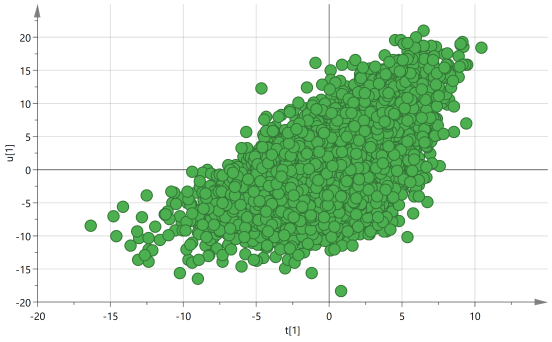


Figur 13: Vikter för de olika variablerna för data över volym, här endast med sjöar med area mindre än 10 km^2. Alla variabelförkortningar förklaras i tabell 4 och 5. Antal prediktorvariabler var 94.

Förhållandet mellan responsvariabeln (volymen) och prediktorvariablerna är inte linjärt (figur 14). Detta innebär att en transformering av datamängden behövs för att uppnå linearitet, här en log10-transformering.



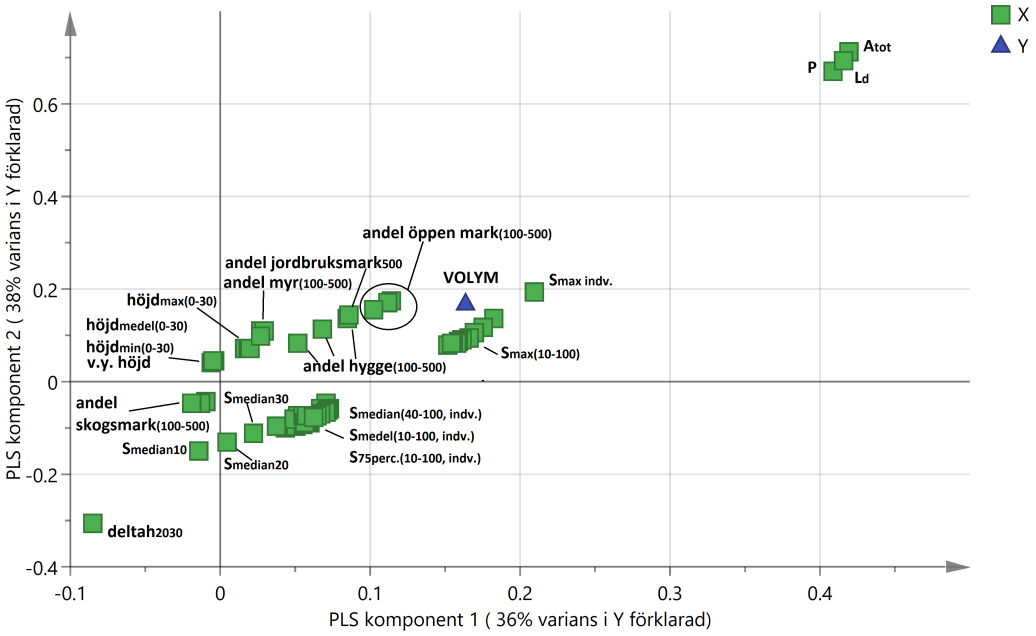
Figur 14: Graf med första objektvektorn i X-matrisen mot första objektvektorn i Y-matrisen för volymen. Studeras för att upptäcka icke-linearitet. Producerad med data över volym, här endast med sjöar med area mindre än 10 km^2.



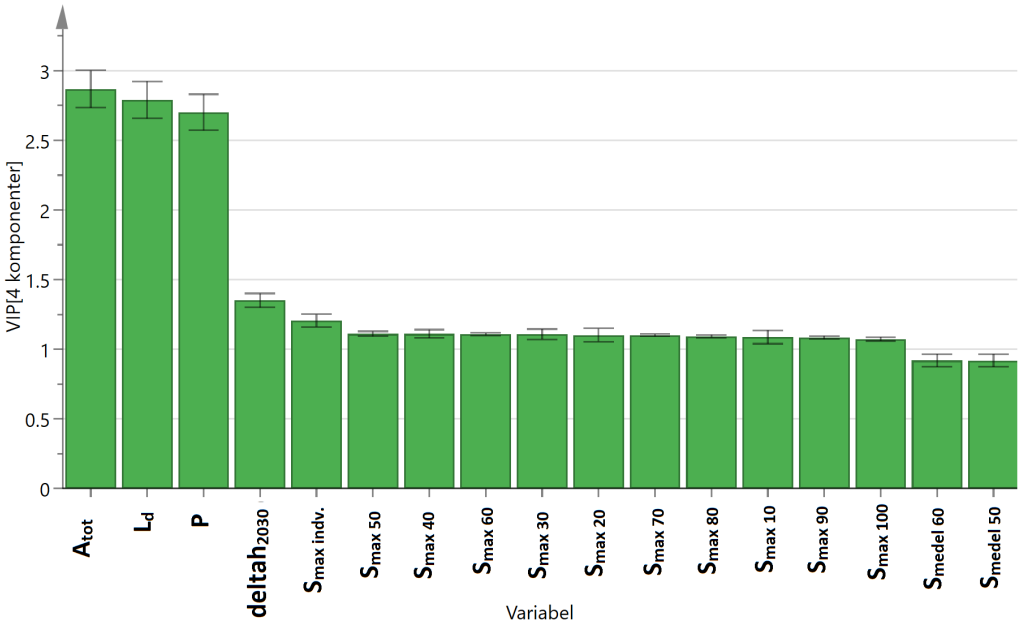
Figur 15: Graf med första objektvektorn i X-matrisen mot första objektvektorn i Y-matrisen för volymen. Studeras för att upptäcka icke-linearitet. Producerad med log10-transformerade data över volym, här endast med sjöar med area mindre än 10 km^2.

Då log10-transformering utfördes av alla variabler försvann en del prediktorvariabler automatiskt, de som är lika med 0, för en majoritet av sjöarna. Detta gällde främst en del markanvändningstyper och minimumlutningar inom olika buffertzoner. Efter log10-transformeringen uppnåddes ett bättre förhållande närmare linearitet (figur 15). Korre-

lationerna blev också något tydligare och en modell med bättre värden på R^2_Y och Q^2_Y erhölls (figur 16 & 17).



Figur 16: Variablers vikt för log10-transformerade data över volym, här endast med sjöar med area mindre än 10 km^2. Alla variabelförkortningar förklaras i tabell 4 och 5. Antal prediktorvariabler var 73.



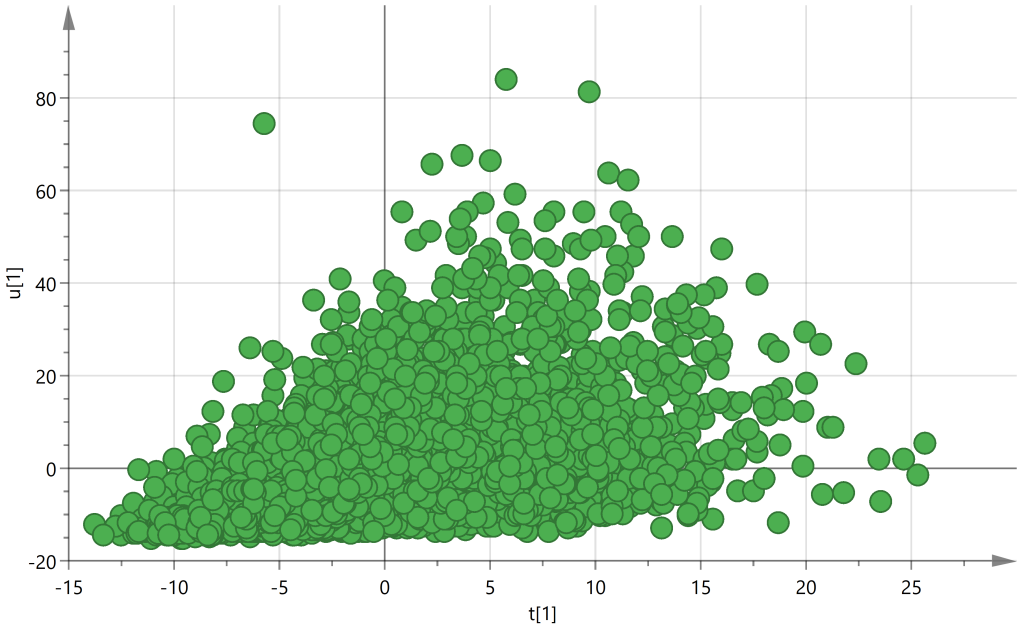
Figur 17: De viktigaste variablerna för att beskriva volymen då en modell med 4 PLS-komponenter används. Här för log10-transformerade data och sjöar med area mindre än 10 km^2. Ett VIP-värde > 1 anses beskriva y-variabeln (volymen) bra.

Skillnaden i medelhöjd mellan 10-20 och 20-30 m zonerna blev den fjärde viktigaste variabeln för att förklara volymen, framför maxlutningen i den individuella zonen (figur 17). Här erhölls en modell med 4 PLS-komponenter med R^2_X=0.817, R^2_Y=0.854 och Q^2_Y=0.853, vilket är en klar förbättring mot icke-transformerade data. Det erhållna p-värdet är < 0.05, d.v.s. en signifikant modell. Både för icke-transformerade data och log10-transformerade data utfördes en rensning av de variabler som hade minst påverkan på volymen främst för att se tydligare korrelationer. De variabler som rensades bort hade alla ett VIP-värde mindre än 0.5. Mönstrena mellan variablerna är desamma som tidigare.

Då PLS-regressionen utfördes för datamängden som inkluderade sjöar med areor större än 10 km^2 erhölls en modell för icke-transformerade data med 4 PLS-komponenter, R^2_X=0.620, R^2_Y=0.748 samt Q^2_Y=0.620. För log10-transformerade data erhölls en modell med 5 PLS-komponenter, R^2_X=0.712, R^2Y=0.889 och Q^2_Y=0.886, båda modellerna med p-värde $<$ 0.05. Samma variabler uppvisade korrelation med volymen, och främst modellen för icke-transformerade data blev bättre än för sjöar med area $<$ 10 km^2.

3.2.2 PLS - maxdjup

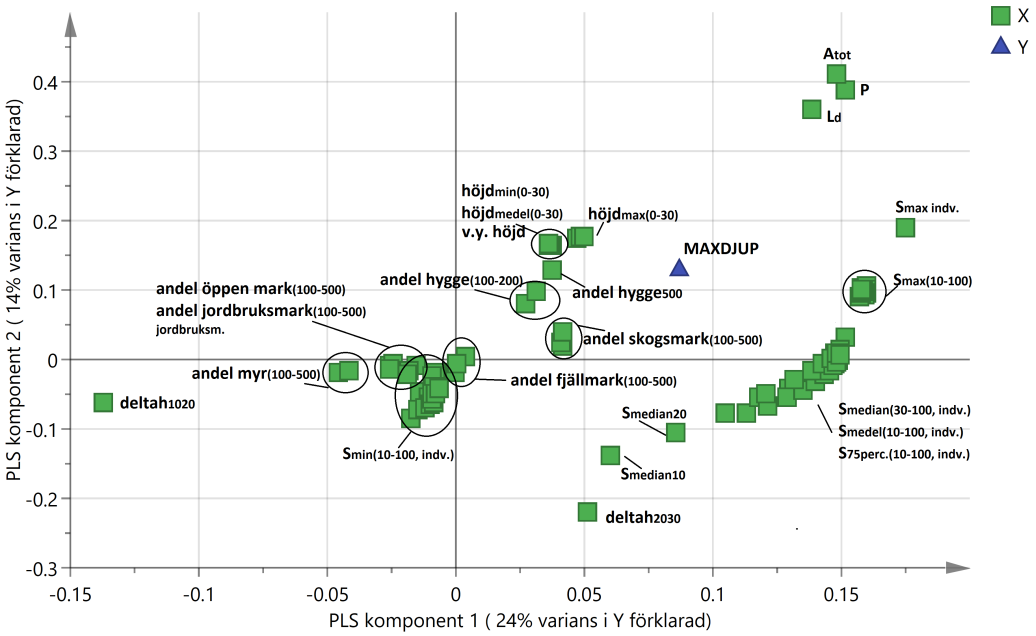
Inga tydliga extremvärden hittades vid PLS-regressionen av maxdjupsdata, men en punkt skulle kunna vara ett extremvärde. Dock uppvisade även denna datamängd tecken på att en transformering kan vara nödvändig (figur 18) då spridningen snarare är konformad än samlade kring en diagonal linje genom origo (heteroskedastisk).



Figur 18: Första objektvektorn i X-matrisen mot första objektvektorn i Y-matrisen för volymen. Studeras för att upptäcka icke-linearitet. Producerad med data över volym, här endast med sjöar med area mindre än 10 km^2.

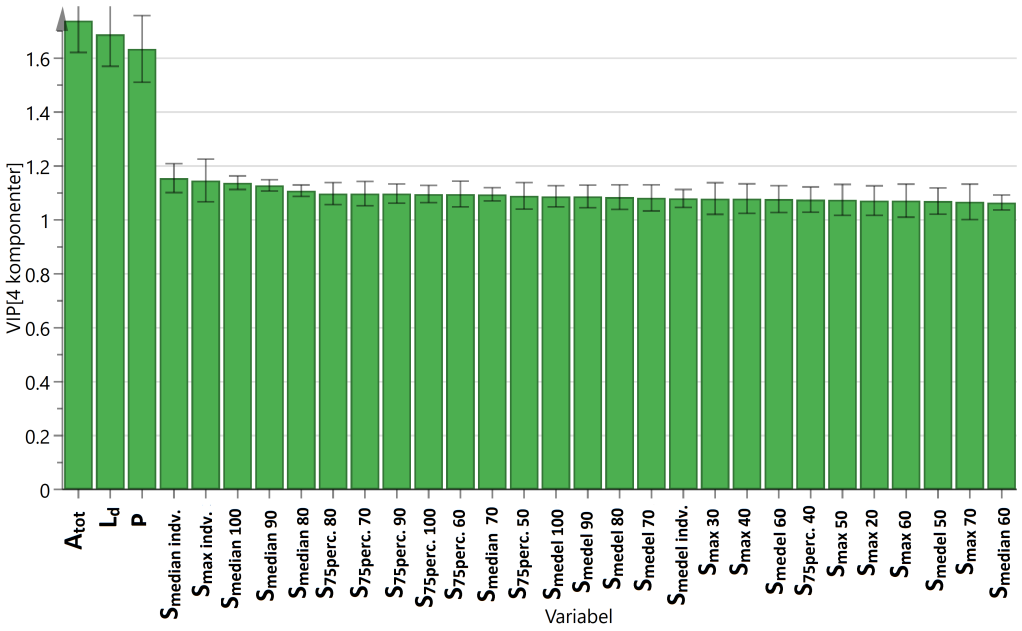
De variabler som starkast korrelerar positivt med maxdjupet i de två första komponenterna är sjöarea, perimeter, strandlinjeutveckling samt maxlutningarna i de olika buffertzonerna

(figur 19). Skillnaden i medelhöjd mellan 0-10 och 10-20 m zonerna visar negativ korrelation med maxdjupet i både första och andra komponenten och skillnaden i medelhöjd mellan 10-20 och 20-30 m zonerna visar positiv korrelation i första komponenten, men negativ i andra. Även medianlutningarna i flera buffertzoner visar en negativ korrelation med maxdjupet i andra komponenten (figur 19). Ingen korrelation med andel öppen mark och jordbruksmark hittades.

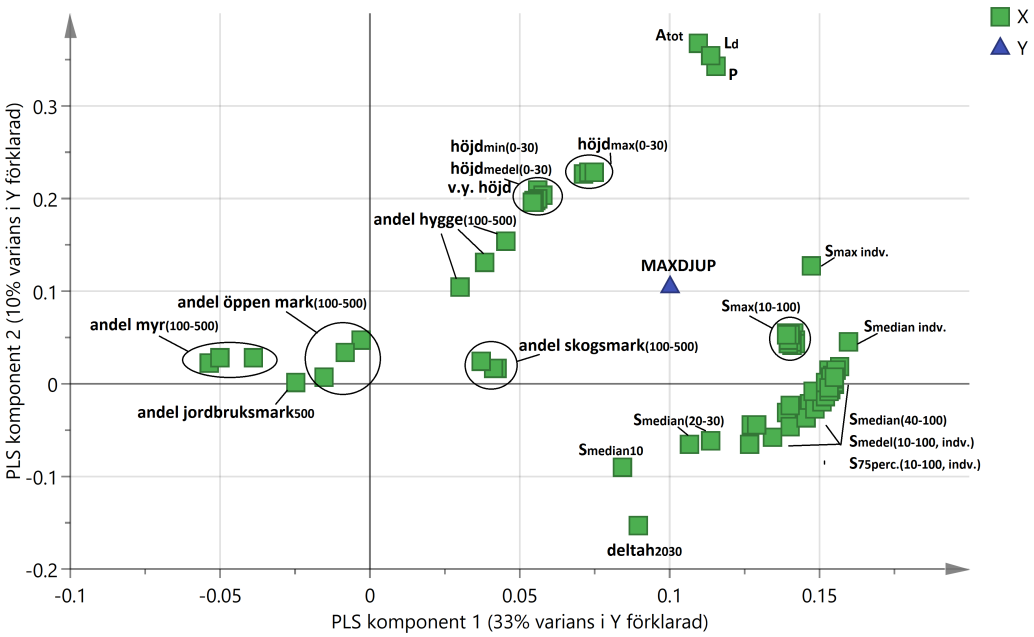


Figur 19: Vikter för alla variabler för data över maxdjup, här endast med sjöar med area mindre än 10 km^2. Alla variabelförkortningar förklaras i tabell 4 och 5. Antal prediktorvariabler var 93.

Här erhölls en modell med 7 PLS-komponenter, R^2_X=0.776, R^2_Y=0.476 samt Q^2_Y=0.457 och ett p-värde < 0.05. Datamängden rensades på variabler med svag korrelation med maxdjupet, lägre VIP-värde än 0.5, och PLS-regressionen gjordes om, utan en förhöjning i förklaringsgrad eller prediktionskraft. En log10-transformering av hela datamängden utfördes för att uppnå bättre linearitet. En marginell försämring av R^2_Y och Q^2_Y erhölls men bättre linearitet uppnåddes i datamängden, samt mindre antal komponenter. Mönstrena är desamma men tydligare (figur 21). Korrelationerna bekräftas även utifrån deras VIP-värden (figur 20), där sjöarean, strandlinjeutvecklingen och perimetern får högst VIP-värden.



Figur 20: De viktigaste variablerna för att beskriva maxdjupet då en modell med 4 PLS-komponenter används. Här för log10-transfomerade data med sjöar med area mindre än 10 km^2. Ett VIP-värde > 1 anses beskriva y-variabeln (maxdjupet) bra.



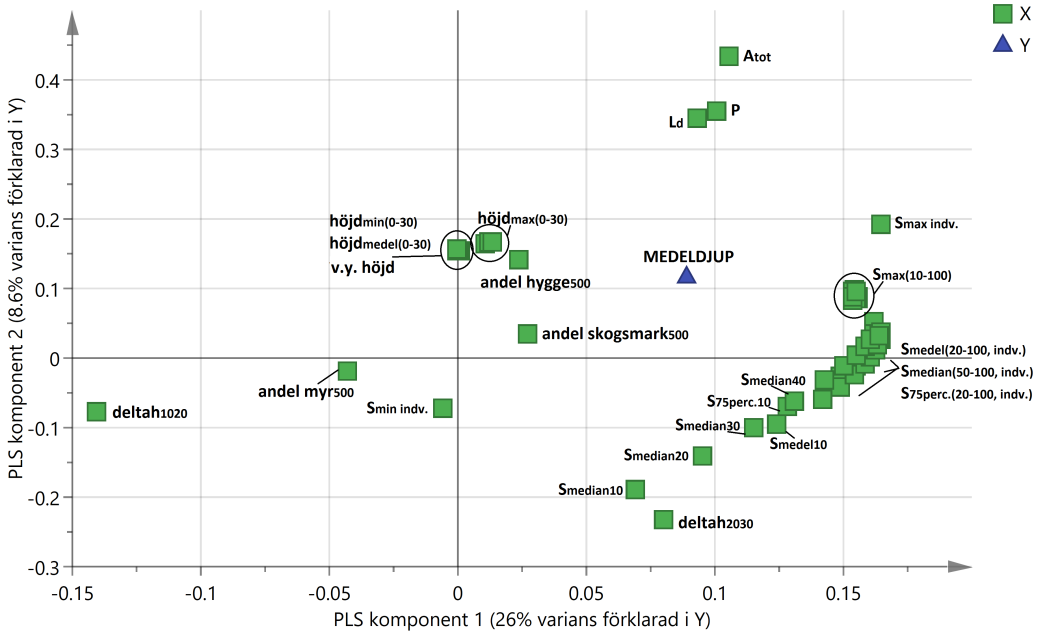
Figur 21: Vikter för variablerna för data över maxdjup, här endast med sjöar med area mindre än 10 km^2, log10-transformerade data. Alla variabelförkortningar förklaras i tabell 4 och 5. Antal prediktorvariabler var 73.

En modell med PLS-regression av log10-transformerade data innehöll 4 PLS-komponenter med R^2_X=0.76, R^2_Y=0.454, Q^2_Y=0.449 samt p-värde < 0.05.

Då PLS-regressionen av datamängden inkluderat de större sjöarna utfördes erhölls en modell med 7 PLS-komponenter, R^2_X=0.951, R^2_Y=0.568, Q^2_Y=0.543 och p-värde < 0.05, för icke-transformerade data.

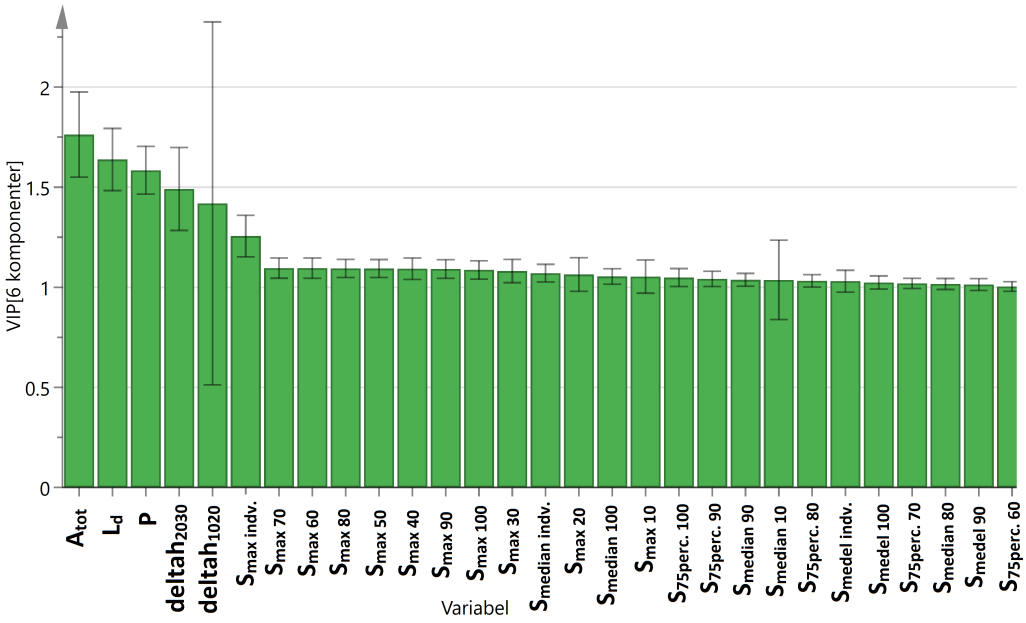
3.2.3 PLS - medeldjup

PLS-regression för medeldjup utfördes först med alla prediktorvariabler inkluderade (resultaten visas ej). En rensning utfördes sedan av de variabler som hade minst påverkan på medeldjupet genom VIP-värden lägre än 0.5 och regressionen gjordes om för att uppnå tydligare korrelationer och lyfta fram de variabler som visade starkast samband till medeldjupet (figur 22).



Figur 22: Vikter för alla variabler för data över medeldjup, här endast med sjöar med area mindre än 10 km^2. Alla variabelförkortningar förklaras i tabell 4 och 5. Antal prediktorvariabler var 63.

Maxlutningen i den individuella zonen, sjöarea, perimeter, strandlinjeutveckling samt skillnaden i medelhöjd mellan 0-10 och 10-20 m zonerna uppvisar korrelation med medeldjupet. Detta är även tydligt utifrån variablernas VIP-värden (figur 23), som alla får ett värde större än 1. Log10-transformering testades men gav ingen ökad förklaringsgrad eller starkare prediktionssamband utifrån graf med den första objektvektorn i X-matrisen mot den första objektvektorn i Y-matrisen. Den bästa modellen som erhölls för medeldjup bestod av 6 PLS-komponenter med R^2_X=0.911, R^2_Y=0.457 och Q^2_Y=0.445, p-värde mindre än 0.05.



Figur 23: De viktigaste variablerna för att beskriva medeldjupet då en modell med 6 PLS-komponenter används. Här för sjöar med area mindre än 10 km^2. Ett VIP-värde > 1 anses beskriva y-variabeln (medeldjupet) bra.

På samma sätt som vid PCA och utifrån samma typ av figur, visade sig Tarfalasjön här vara ett extremvärde. Sjön behölls ändå i analysen då den inte påverkade resultatet i övrigt.

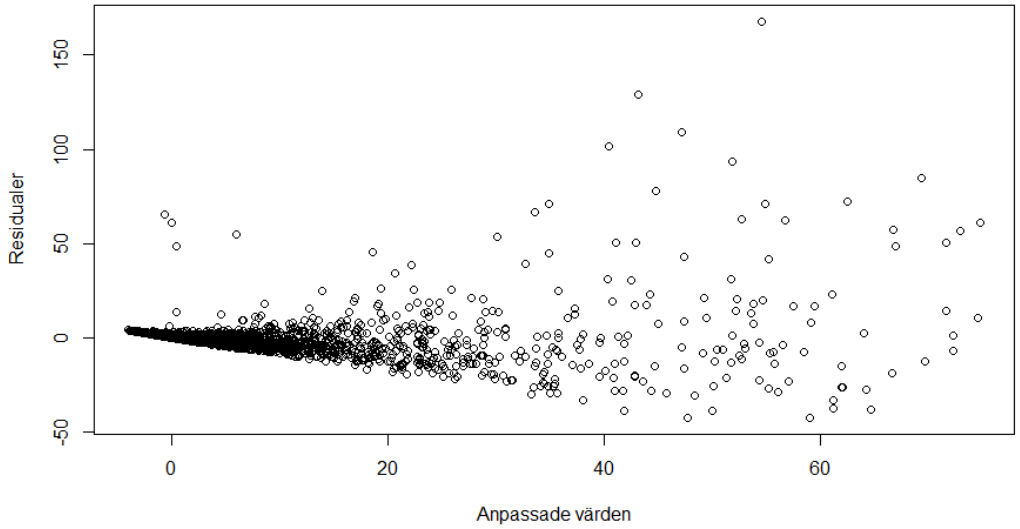
Då PLS-regression av medeldjup utfördes för alla sjöar, d.v.s. inkluderat även sjöar med större area erhölls en modell med 7 PLS-komponenter, R^2_X=0.771, R^2_Y=0.499 och Q^2_Y=0.462 samt p-värde < 0.05.

3.3 MULTIPEL LINJÄR REGRESSION

3.3.1 MLR - volym

Vid PLS-regressionen framkom att sjöarea, perimeter och strandlinjeutveckling alla korrelerade positivt med volymen. Dessa tre prediktorvariabler är alla samvarierande och beror delvis på varandra. Därför användes endast en av dem i modellen. Detta valdes till slut till sjöarea. Valet gjordes utifrån att sjöarean är den variabel som innehåller minst osäkerheter och som även gav högst förklaringsgrad. Höjdskillnaderna, deltah_{1020} och deltah_{2030}, uppvisade också viss samvariation, och därför valdes endast deltah_{2030} att ingå i modellen då den bidrog med högre förklaringsgrad.

Först anpassades en modell till volymen med sjöarea, deltah_{2030} samt maxlutningen i den individuella buffertzonen. Residualerna är heteroscedastiska (figur 24), vilket tyder på att transformering krävs. Detta framkom även under PLS-regressionen.

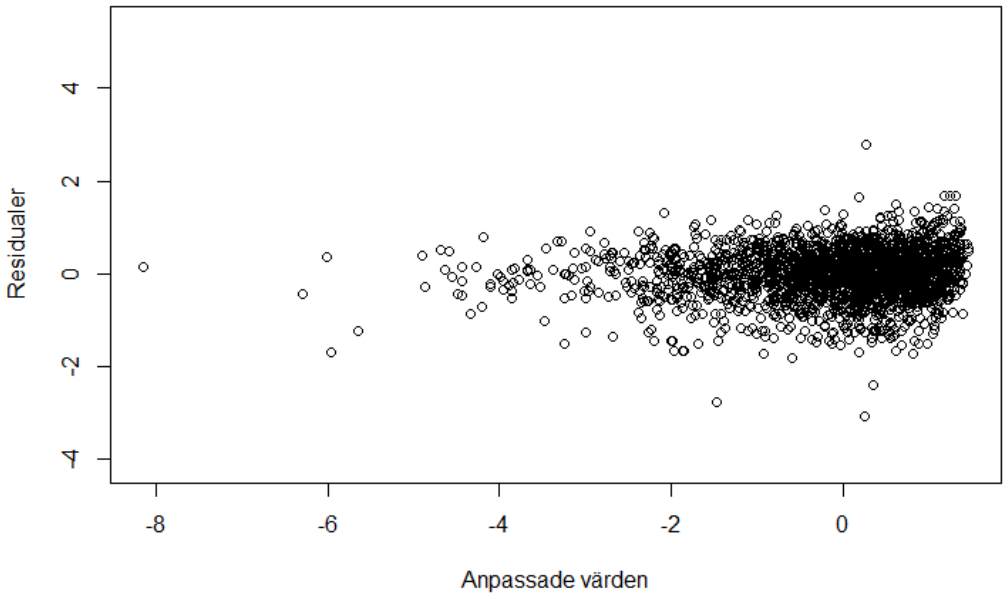


Figur 24: Residualer mot anpassade värden för multipel linjär regression av volym.

Den regressionsekvation som erhölls utifrån detta (ekvation 17) fick R^2=0.634, R_a^2=0.634, $F_{3,4664}$=2693, p-värde $< 2.2 \times 10^{-16}$. Variansen i residualerna blev s^2=0.5802.

$$V = -4.97 + 7.47 \times 10^{-6} A_{tot.} + 3.96 deltah_{2030} + 5.90 \times 10^{-2} s_{max\,indv.} \qquad (17)$$

En ln-transformering utfördes för att erhålla en bättre regression samt för att uppnå homoscedastiska residualer (figur 25). En bättre fördelning uppnåddes, men tecken på extremvärden finns. Modell enligt ekvation 17 gav även vissa negativa värden på volymen då den användes för prediktion.

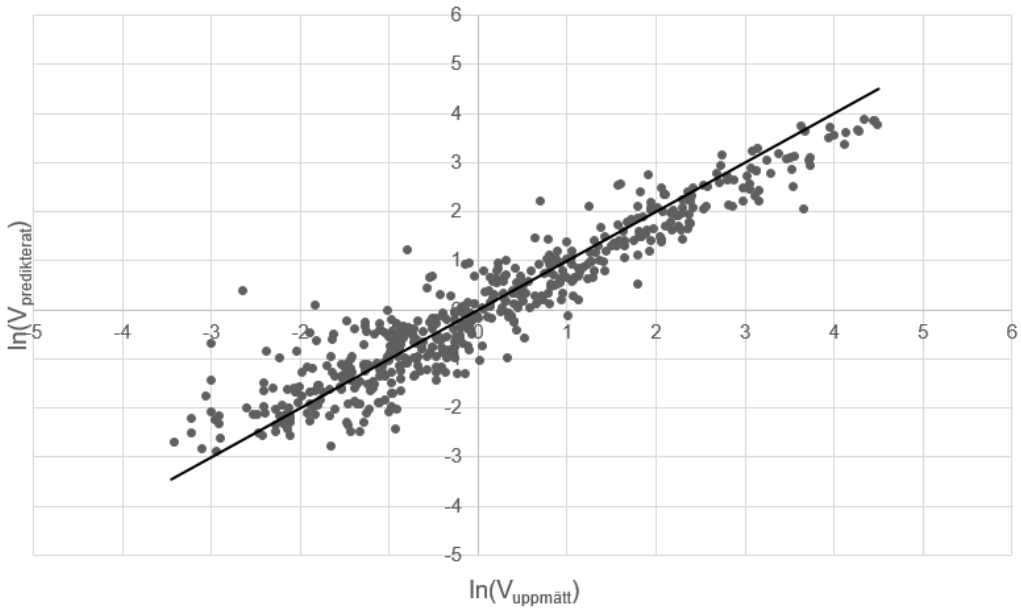


Figur 25: Residualer mot anpassade värden för multipel linjär regression av ln-transformerade volymdata.

En regression med ln-logaritmerade värden på volym och regressionsvariablerna sjöarea, deltah$_{2030}$ samt s$_{max\,indv.}$ fick R^2=0.861, R^2_a=0.861, F$_{3,4664}$=9600, p-värde $< 2.2\times 10^{-16}$. För att erhålla den bästa modellen testades även andra variabler som under PCA och PLS-regression visade viss korrelation med volymen. Den bästa modellen erhölls med sjöarea och medianlutningen i den individuella zonen (ekvation 18) med R^2=0.870, R^2_a=0.870, F$_{2,4665}$=1.567 $\times$ 10^4 samt p-värde $< 2.2 \times 10^{-16}$. Skillnaden i medelhöjd mellan 10-20 och 20-30 m zonerna tillförde ingen ökad förklaringsgrad till modellen. Variansen i residualerna i den erhållna modellen blev s^2=0.559.

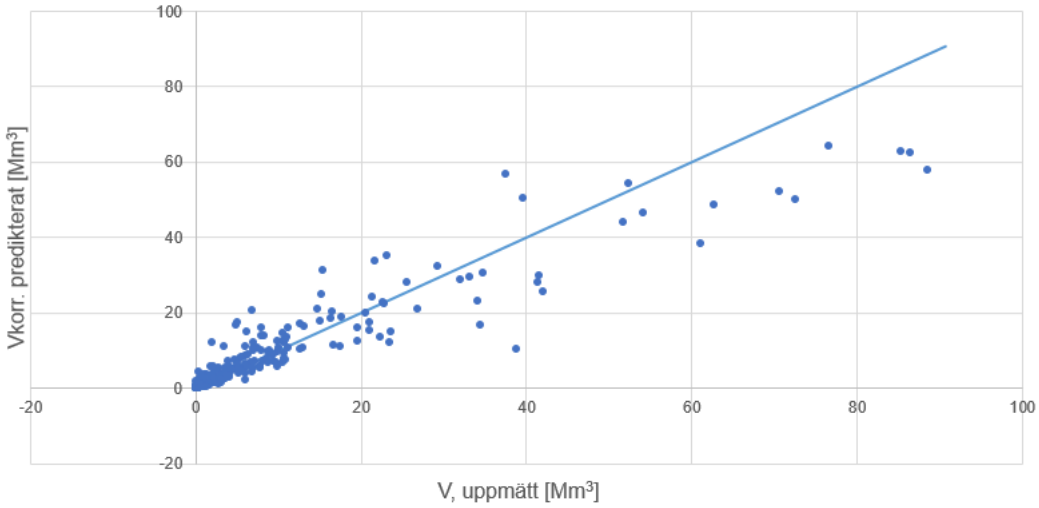
$$ln(V) = -15.628 + 1.161ln(A_{tot}) + 0.668ln(s_{median\,indv.}) \tag{18}$$

Logaritmerade (ln), uppmätta värden på volymen i valideringsmängden jämfördes med predikterade värden utifrån erhållen modell (ekvation 18, figur 26). Då prediktionerna jämfördes med uppmätta värden blev det absoluta medelfelet $\pm$ 0.421 och MSE=0.311, variansen i valideringsmängdens residualer blev s^2=0.132.



Figur 26: Uppmätta, logaritmerade värden på volymen mot predikterade värden. Den heldragna linjen beskriver ett 1:1-förhållande.

För att erhålla faktiska värden på volymen utfördes en tillbakatransformering av de predikterade värdena på volymen enligt ekvation 8 och 9. Högre prediktionskraft erhölls för sjöar med mindre volymer, < 10 Mm^3, än med större där det blev en större spridning (figur 27). Här blev det absoluta medelfelet $\pm$ 1.631 Mm^3 och MSE=17.257, variansen i valideringsmängdens residualer utifrån ekvation 9 blev 4.948. Den relativa standardavvikelsen för $V_{korr.}$ blev $\pm$ 45 %.



Figur 27: Uppmätta värden på volymen mot predikterade, korrigerade värden. Den heldragna linjen beskriver ett 1:1-förhållande.

För att se om endast det bättre kartmaterialet kan ge en säkrare modell än tidigare, togs en ekvation med lutning inom bestämd zon fram, istället för indivduell zon (ekvation 19). Här erhölls R^2=0.864, R_a^2=0.864, $F_{2,4665}$=1.483$\times 10^4$ samt p-värde$<2.2\times 10^{-16}$. Variansen i residualerna blev s^2=0.573.

$$ln(V) = -15.696 + 1.163ln(A_{tot}) + 0.626ln(s_{medel\,40}) \tag{19}$$

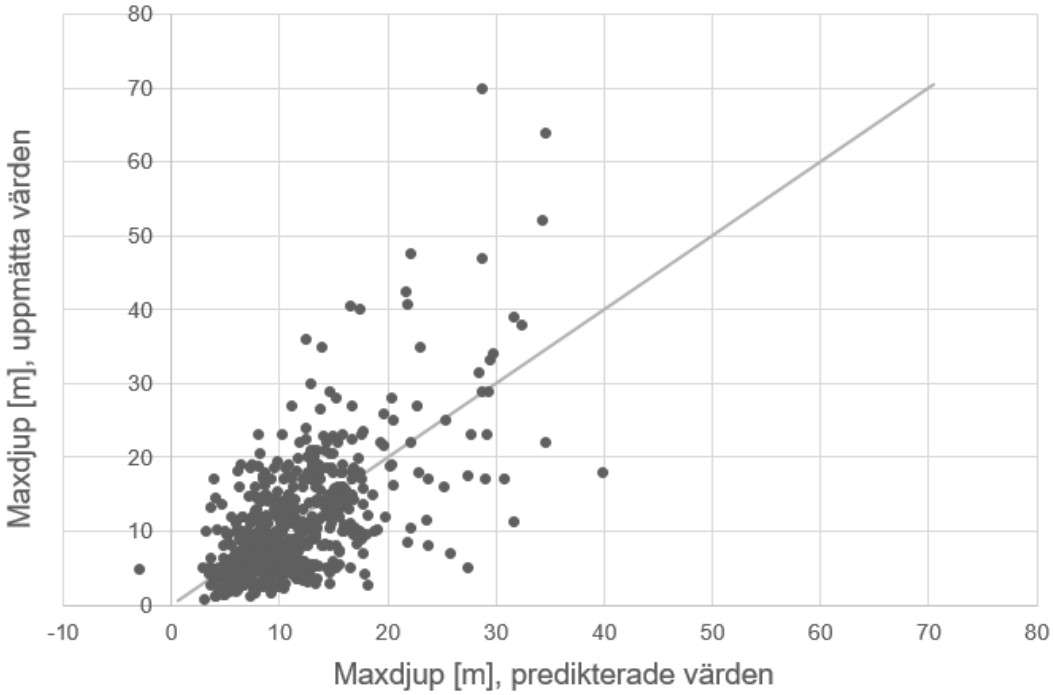
Logaritmerade (ln), uppmätta värden på volymen i valideringsmängden jämfördes med predikterade värden utifrån modellen. Detta gav det absoluta medelfelet $\pm$ 0.465, MSE=0.362 och variansen i valideringsmängdens residualer, s^2=0.144. Tillbakatransformering till faktiska värden på volymen utfördes av de predikterade värdena (ekvation 8 och 9). Det absoluta medelfelet blev då $\pm$7.340 Mm^3, MSE=25.924, variansen i valideringsmängdens residualer utifrån ekvation 9 blev 4.808. Den relativa standardavvikelsen för $V_{korr.}$ blev $\pm$ 47 %.

3.3.2 MLR - maxdjup

Vid PLS-regressionen kom strandlinjeutveckling, sjöarea och perimeter fram som korrelerande med maxdjupet. På samma sätt som för volymen valdes sjöarea som prediktorvariabel till den multipla regressionen. Även $deltah_{2030}$, $s_{median100}$, $s_{maxindv.}$ samt max- och medellutningar i övriga buffertzoner testades som prediktorvariabler. Den bästa modellen erhölls med sjöarea, $deltah_{2030}$ och $s_{median100}$ (ekvation 20). Denna modell fick R^2=0.402, R_a^2=0.402, $F_{3,5233}$=1174, p-värde $< 2.2\times 10^{-16}$ samt s^2=6.503.

$$D_{max} = 1.386 + 2.315 \times 10^{-6} A_{tot} - 17.85 deltah_{2030} + 2.409 s_{median\,100} \tag{20}$$

Den erhållna modellen testades mot valideringsmängden (tabell 1 och figur 28).

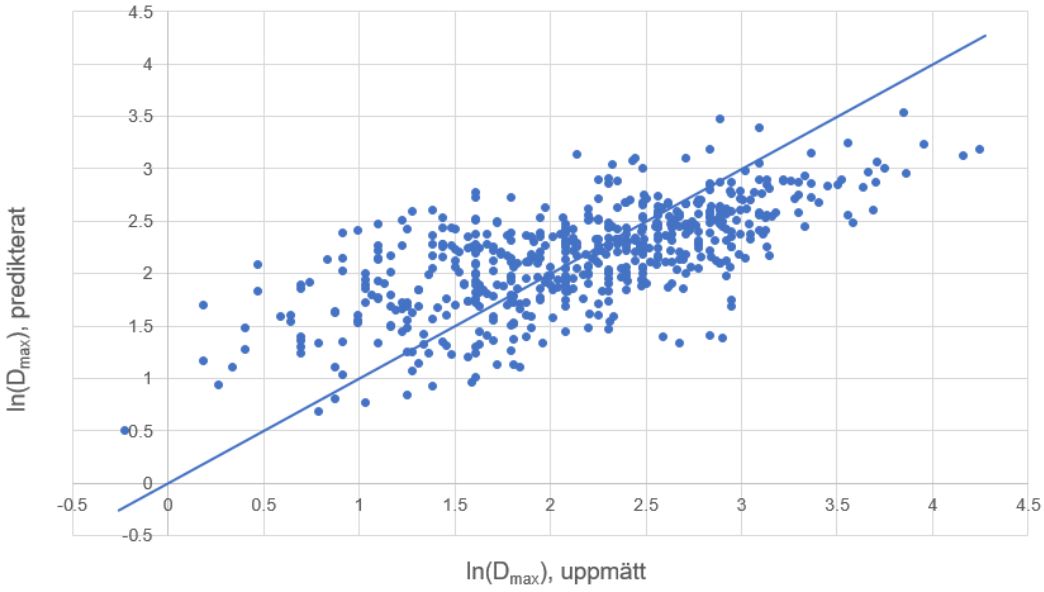


Figur 28: Uppmätta värden på maxdjupet mot predikterade värden. Den heldragna linjen beskriver ett 1:1-förhållande.

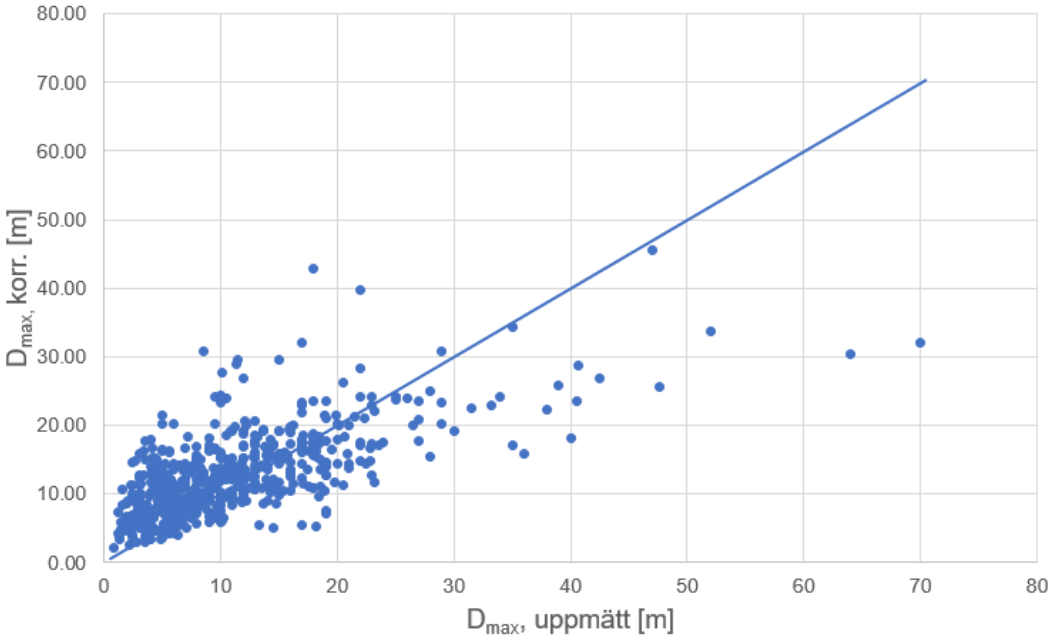
Det blir sämre prediktionskraft för större djup än för mindre. Ett negativt predikterat värde erhölls (figur 28). Det absoluta medelfelet blev $\pm$ 4.720 m, MSE=43.490 och variansen i residualerna hos valideringsmängden blev s^2=21.036. Studier av residualerna tydde på att transformering av data var nödvändig, vilket ledde till att en ln-transformering utfördes. Detta resulterade i bättre fördelning av residualerna och även en något högre förklaringsgrad. Här användes istället $s_{median\,indv.}$, då den gav bättre resultat än $s_{median\,100}$ (ekvation 21). Modellen fick R^2=0.422, R^2_a=0.422, $F_{3,5233}$=1273, p-värde $< 2.2 \times 10^{-16}$ samt s^2=0.569.

$$ln(D_{max}) = -1.817 + 0.249ln(A_{tot}) + 0.104ln(deltah_{2030}) + 0.697ln(s_{median\,indv.}) \quad (21)$$

Då predikterade värden jämfördes med uppmätta, ln-transformerade värden erhölls en relativt hög spridning (figur 29). Det absoluta medelfelet blev $\pm$ 0.435, MSE=0.295 och variansen i residualerna hos valideringsmängden blev s^2=0.105. Då tillbakatransformering utfördes (ekvation 8) undveks negativa värden på maxdjupet, men spridningen på prediktionerna blev ändå hög (figur 30).



Figur 29: Uppmätta, ln-transformerade värden på maxdjupet mot predikterade värden. Den heldragna linjen beskriver ett 1:1-förhållande.



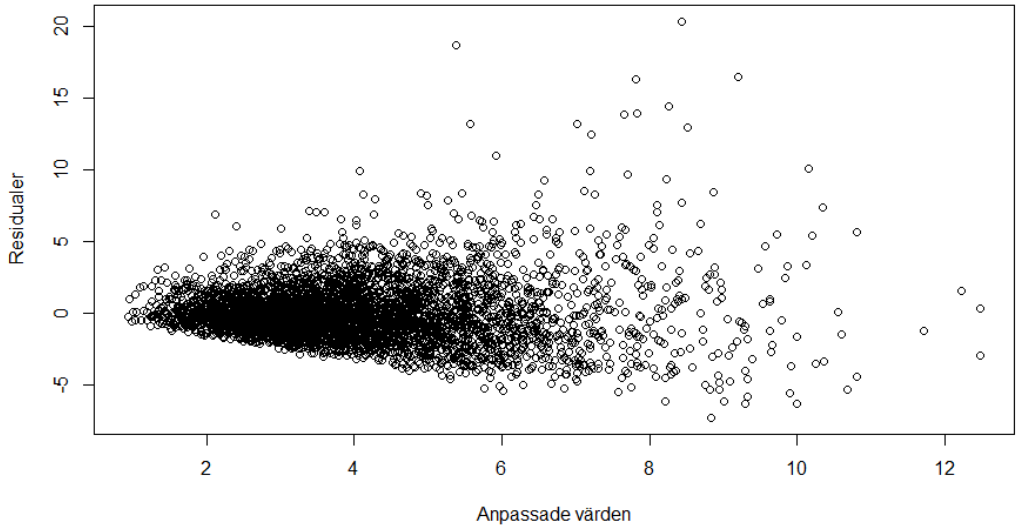
Figur 30: Uppmätta värden på maxdjupet mot tillbakatransformerade, predikterade värden. Den heldragna linjen beskriver ett 1:1-förhållande.

3.3.3 MLR - medeldjup

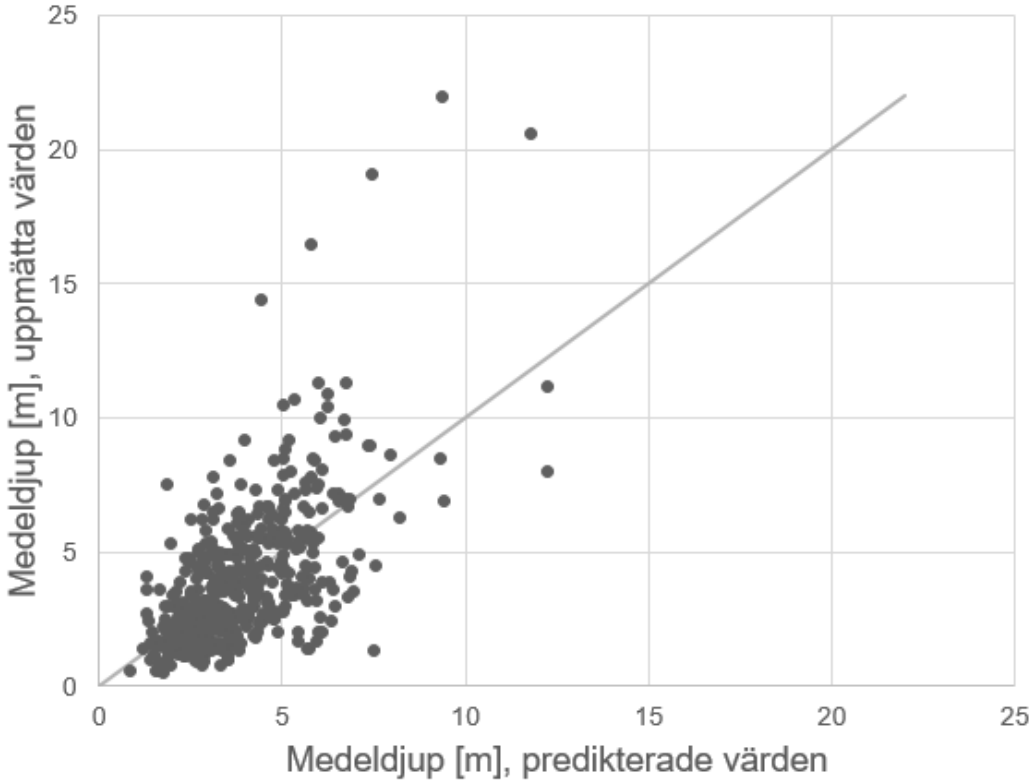
Utefter resultat från PLS-regressionen gjordes den multipla regressionen mellan medeldjup och sjöarea, deltah$_{1020}$, deltah$_{2030}$ samt max-, medel- och medianlutningar i de olika buffertzonerna. De variabler som gav bäst regressionsmodell blev sjöarea, deltah$_{1020}$ samt medellutningen i 100 m-buffertzonen (ekvation 22).

$$D_{medel} = 2.77\times10^{-1}+7.395\times10^{-7}A_{tot}-3.515\times10^{-1}deltah_{1020}+4.448\times10^{-1}s_{medel\,100} \quad (22)$$

Modellen fick R^2=0.372, R^2_a=0.372, $F_{3,4515}$=892.4, p-värde $< 2.2\times10^{-16}$ samt s^2=4.444. Residualerna uppvisade heteroscedastisk form (figur 31), men en ln-transformering av data ledde till ett p-värde > 0.05 samt R^2-värden som inte tyder på någon korrelation alls. Formen på residualerna indikerar dock att den erhållna modellen inte har prediktionskraft. De predikterade värdena jämfört med uppmätta värden på medeldjupet visar även de på dålig prediktionskraft (figur 32). Det absoluta medelfelet blev $\pm$ 1.404 m, MSE=4.125 och variansen i residualerna hos valideringsmängden blev 2.146.



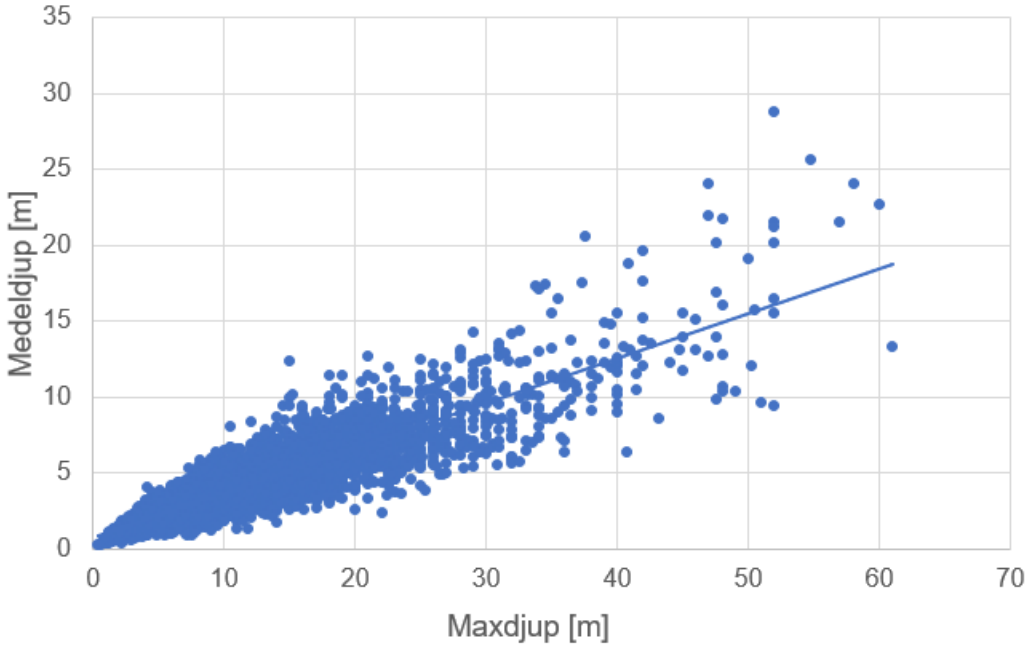
Figur 31: Residualer mot anpassade värden för multipel linjär regression av medeldjupsdata.



Figur 32: Predikterade värden på medeldjupet utifrån ekvation 22 mot uppmätta värden. Den heldragna linjen beskriver ett 1:1-förhållande.

Ett starkt samband mellan medeldjup och maxdjup fastställdes med enkel linjär regression (R^2=0.807, R_a^2=0.807, p-värde $< 2.2 \times 10^{-16}$, s^2=1.37, ekvation 23 och figur 33).

$$D_{medel} = 0.742 + 0.296 D_{max} \tag{23}$$



Figur 33: Uppmätta värden på medeldjupet mot uppmätta värden på maxdjupet. Den heldragna linjen är lutningen enligt regressionekvationen (ekvation 23).

4 DISKUSSION

4.1 DISKUSSION - VOLYM

Volymen kan modelleras med högre säkerhet än tidigare med variablerna sjöarea och $s_{median\,indv.}$ för en enskild sjö eftersom en lägre relativ standardavvikelse för $V_{korr.}$ erhölls ($\pm$ 45 %) än det Sobek m. fl. (2011) fann ($\pm$ 57 %). Då kartparametrar endast beroende på det nya kartmaterialet användes ($s_{medel\,40}$ och $A_{tot.}$) erhölls också lägre relativ standardavvikelse ($\pm$ 47 %). Hypotesen att lutningar inom den individuella zonen visar starkare korrelationer med volymen än lutningar i bestämda zoner kan styrkas, men också att det nya kartmaterialet bidrar till en säkrare modell. Övriga kartparametrar som visar hög, signifikant korrelation med volymen är perimeter, strandlinjeutveckling, $deltah_{2030}$, $s_{max\,indv.}$ samt $s_{max\,10}$. Hypotesen att volymen har signifikanta samband med variablerna perimeter, area, strandlinjeutveckling och maxlutning i en zon $<$ 100 m kring sjön kan därmed styrkas, medan sambandet mellan volym och minimumlutning förkastas.

De tre variabler som vid PLS-regressionen visade starkast korrelation med volymen var också tydligt samvarierande samt beror delvis på varandra, vilket MLR inte kan hantera. De tre variablerna var sjöarea, perimeter och strandlinjeutveckling, och där var sjöarean den variabel som bidrog med högst förklaringsgrad, och även den variabel som innehåller minst osäkerheter. Uträkningen av perimetern beror mycket på noggrannheten i skapandet av sjöpolygonerna. Strandlinjeutvecklingens osäkerheter följer av perimeterns osäkerhet då den beräknas utifrån perimetern. Även vid PLS-regressionen hittades denna korrelation mellan volym och perimeter, area och strandlinjeutveckling, där dessa tre variabler var de tre i hela datamängden som hade störst påverkan på volymen. De har också bekräftade samband med volymen sedan tidigare, bland annat av Sobek m. fl. (2011). Samma samband hittades vid analyser med hjälp av PCA, vilket ytterligare stödjer korrelationen.

De variabler som bäst förklarade volymen för icke-transformerade data var sjöarean, skillnaden i medelhöjd mellan 10-20 och 20-30 m zonerna och maxlutningen i den individuella zonen. För ln-transformerade data blev det sjöarean och medianlutningen i den individuella zonen. Medianlutningen valdes framför maxlutningen då den dels bidrog till högre förklaringsgrad (R^2) och dels resulterade i mindre varians i residualerna, till skillnad mot PLS-regressionen där maxlutningen bidrog med högre förklaringsgrad. Det som var framträdande vid MLR var att sjöarean är den variabel som bidrar mest till förklaringsgraden, men att lutningens bidrag ändå är betydande nog för att den ska behållas i ekvationen. Att sjöarean bidrar med hög förklaringsgrad stämmer även överens med studier av Cael m.fl. (2017) som byggde sin studie på ett mekanistiskt samband mellan volymen och arean. De beräknade dock medelvolymen för ett större antal sjöar, på nationell och global nivå, till skillnad mot denna studie där volymen för en individuell sjö predikterades. Sambandet mellan sjöarea och sjövolym är ändå tydligt.

Då regressionen utfördes kunde det konstateras att en transformering krävdes. Här syntes det framför allt under residualanalysen, där residualerna uppvisade heteroskedastisk form. Detta kan bidra till stor varians i prediktionerna, och därmed mindre säkra prediktioner och samband. Med en ln-transformering minimeras detta, vilket även bekräftades under analysens gång där en högre förklaringsgrad erhölls och även en mindre varians i residualerna.

Maxlutningen i de olika buffertzonerna visade under PLS-regressionen korrelation med volymen, där maxlutningen inom den minsta zonen (0-10 meter) var den som hade störst påverkan av de bestämda buffertzonerna. Även medellutningen i 40 m buffertzonen visade stark korrelation med volymen. Dock stack maxlutningen i den individuella zonen ut från de övriga och hade ännu starkare samband med volymen (figur 13). Detta innebär att delar av den metod som Hollister m. fl. (2011) använde kan bidra till en förbättrad prediktion av volym i svenska sjöar. Då det finns en spridning i arean hos de sjöar som är med i analysen (0.008-9.99 km^2), betyder resultatet att en större sjö påverkas av formen på ett större område kring sjön, medan en mindre sjö påverkas mer av formen på det allra närmsta området, vilket en buffertzon som anpassas efter sjöns area tar hänsyn till. Då bättre säkerhet erhölls med lutningen i den individuella zonen jämfört med lutningen i bestämda zoner ($\pm$ 45 $\%$ jämfört med $\pm$ 47 $\%$) innebär det även att denna metod ger säkrare modell än bestämda zoner. Anledningen till att maxlutningen i den minsta zonen förutom den individuella zonen var den som visade högst korrelation med volymen är förmodligen att datamängden består främst av mindre sjöar, som därmed blir mer påverkade av ett mindre område kring sjön och då mer framträdande i analysen. Vid PLS-regressionen och MLR kunde p-värden för de framtagna modellerna erhållas, vars värden visade på att de samband som fanns är signifikanta. Dock visar sig minimumlutningen i alla buffertzoner ha liten påverkan på volymen då de placeras kring origo (figur 13 och 16), till skillnad mot vad Sobek m. fl. (2011) hittade och här föreslagen hypotes. Detta kan bero på att det finns en viss förskjutning mellan sjöpolygonerna och kartan med höjddata, vilket kan leda till att delar av sjön tas med i buffertzonerna. När lutningsberäkningarna då utförs, får dessa värdet 0, och det spelar därmed ingen roll vilken buffertzon det är, alla har minimumlutningen 0, och variabeln visar inget samband. Om passningen mellan kartorna hade varit bättre hade detta kunnat undvikas och en möjlighet att minimumlutningen skulle ha större påverkan på volymen hade funnits. Om beräkningarna skulle

göras om, skulle ett alternativ vara att börja zonerna 5-10 meter från strandlinjen istället för direkt vid strandlinjen, och därmed kompensera för förskjutningen mellan kartorna. Utifrån detta gjordes även valet att inte ta med den allra minsta buffertzonen som var möjlig (0-2 meter), då det främst skulle vara för osäkert att den faktiskt innehåller strandområde, samt att lutningsberäkningarna blir mer osäkra i så små områden. Osäkerheten i lutningsberäkningarna är alltid närvarande och det finns risker för randeffekter, d.v.s. att beräkningarna blir felaktiga längs kanterna av buffertzonen. För att undvika detta utfördes lutningsberäkningarna först på större buffertzoner (0-1000 m), och sedan klipptes mindre zoner utifrån dessa och till sist beräknades statistik. På detta sätt kunde osäkerheterna minskas.

Höjdskillnaderna, deltah_{1020} och deltah_{2030}, kom fram tydligt som korrelerande med volymen vid PLS-regressionen. Det var främst deltah_{2030} som visade negativ korrelation, både i första och andra komponenten. Det innebär att ju större medelhöjden är i 20-30 m buffertzonen jämfört med i 10-20 m buffertzonen, ju mindre blir sjövolymen. Detta skulle kunna vara sammankopplat med landskapstyper då kuperade landskap, med tydliga höjdförändringar, ofta är sammankopplade med mindre sjöar medan slättlandskap ofta är sammankopplade med större sjöar. Detta indikerar att landskapstypen kan ha ett samband med volymen. Denna hypotes kunde tyvärr inte testas i denna studie.

De olika markanvändningstyperna, som var variabler som hade förväntats visa korrelation med volymen och även maxdjupet, och av dessa främst andel jordbruksmark i tre olika buffertzoner, visade sig inte ha samband med volymen. I första analyssteget med PCA kunde korrelation mellan andel jordbruksmark och volym varken bekräftas eller förkastas. De hamnade inte ortogonalt mot varandra (figur 8 och 6), och lade sig i närheten av varandra, vilket kan tyda på korrelation. Dock placerades de i olika kvadranter, men ändå nära i förhållande till varandra. Detta anses däremot inte som ett tillräckligt bevis för att bekräfta ett samband. Under PLS-regressionen framkom inte andel jordbruksmark eller andel öppen mark i någon av de testade buffertzonerna som variabler som har stor inverkan på volymen. Under alla analyser placerades de relativt nära origo vid studier av grafer med vikter (ex. figur 13). Utifrån dessa resultat förkastas hypotesen att andel jordbruksmark har en påverkan på volymen. Om korrelation ändå finns är den inte tillräckligt stark för att visas i analyserna. Det kan vara så att andra variabler i datamängden beskriver variansen i volymen bättre och därmed får andel jordbruksmark så pass låga vikter. En möjlighet är också att i de områden där det finns mycket jordbruk och öppen mark så spelar dessa variabler en större roll för volymen, men då denna uppdelning ej gjorts här, så kommer det inte fram. Ytterligare ett problem med dessa markanvändningsandelar är att för en stor andel av sjöarna är värdet 0. Detta leder till en skev fördelning av variabeln, där log10-transformering som använts i detta fall inte förbättrar fördelningen. Om markanvändningsandelar ska undersökas igen bör andra typer av transformationer testas för att uppnå bättre resultat.

4.2 DISKUSSION - MAXDJUP

Utifrån resultatet kan hypotesen att maxdjupet uppvisar signifikant korrelation med sjöarea, perimeter och strandlinjeutveckling samt maxlutningen inom en zon $<$ 100 m kring sjön stärkas. Dock visade sig $s_{median\,indv.}$ bidra med högre förklaringsgrad än $s_{max\,indv.}$. Även hypotesen att skillnader i medelhöjd i det närmaste området kring sjön har ett samband med djupet i sjön kan styrkas. Hypotesen att lutningar inom den individuella zonen visar starkare korrelation med maxdjupet än lutningar i bestämda zoner kan styrkas utifrån både PLS-regressionen och MLR.

Variansen i residualerna till den slutliga MLR-modellen för maxdjup (ekvation 20) blev relativt hög, och tyder på osäkerheter i modellen. Under PLS-regressionen tydde resultatet på att datamängden med maxdjup behövde transformeras (figur18) för att uppnå linearitet och högre prediktionskraft. Detta bekräftades under den multipla regressionen, där en ln-transformerad modell gav högre prediktionskraft och högre förklaringsgrad. Dock blev spridningen av de predikterade värdena fortfarande stor. Då tillbakatransformering utfördes undveks predikterade negativa värden som kunde erhållas då modellen för icke-transformerade data användes. Den högre prediktionskraften och den ökade linearitеten i datamängden, samt endast positiva predikterade värden gör modellen med ln-transformerade variabler till det bättre valet. Hollister m. fl. (2011) erhöll säkrare resultat utifrån sin modell för maxdjup med ett relativt lågt MSE värde på 5-6 m, där modellen i denna studie för otransformerade data erhöll ett så pass högt MSE värde som 43 m. Den rekommenderade modellen här gäller dock för ln-transformerade data, som fick ett betydligt lägre MSE värde. Hollister m. fl. (2011) hade också en helt annan metod som inte bygger på statistik som denna studie, samt utfördes för ett mindre antal sjöar inom ett visst område. Detta gör att modellerna och dess effektivitet inte är helt jämförbara.

På samma sätt som för volymen korrelerade sjöarean, strandlinjeutvecklingen och perimetern alla tre med maxdjupet. Detta blev främst tydligt under PLS-regressionen, då analys med PCA inte visade på tydliga korrelationer med någon variabel, utan endast antydningar. Ett exempel på en antydan till samband är mellan maxdjupet och maxlutningen i de olika buffertzonerna. Då PLS-regressionen utfördes kunde korrelationen med maxlutningen i de olika buffertzonerna bekräftas, och då främst med $s_{max\,indv.}$. Andra variabler som påverkade maxdjupet var skillnaderna i medelhöjd, $deltah_{1020}$ och $deltah_{2030}$, samt de olika medianlutningarna. Gällande korrelationer med höjdskillnader hittades liknande förhållanden mellan maxdjupet och skillnader i medelhöjd som mellan volymen och skillnader i medelhöjd. Här var det dock $deltah_{1020}$ som visade negativ korrelation med maxdjupet i både första och andra komponenten, medan $deltah_{2030}$ endast korrelerade negativt i andra komponenten, och positivt i första. Om höjdskillnaden har en positiv korrelation med djupet innebär det att sjön får ett större maxdjup ju större skillnad det är i höjd mellan de olika zonerna, d.v.s. ju brantare form närområdet har. Höjdskillnaden kan betraktas som en form av lutning, vilket gör det oväntat att den uppvisar negativ korrelation till skillnad mot maxlutningen i de olika buffertzonerna som visar positiv korrelation med både maxdjupet och volymen. Maxlutningens positiva korrelation med maxdjupet stämmer bra in på teorin att formen på närområdet även definierar formen på sjön, en större maxlutning leder till ett större maxdjup, vilket var det väntade resultatet.

4.3 DISKUSSION - MEDELDJUP

Ett tydligt, signifikant samband hittades mellan medeldjup och maxdjup med MLR, där medeldjupet uppvisar signifikant korrelation med maxdjupet med hög förklaringsgrad (R_a^2=0.807). Dock hittades inte tillräckligt tydliga och säkra samband med andra variabler. Detta bekräftar hypotesen att medeldjupet inte kan modelleras utifrån kartparametrar med hög säkerhet, men att ett starkt samband finns mellan medeldjupet och maxdjupet. För att denna modell ska vara användbar krävs uppmätta värden på maxdjupet, vilket minskar användbarheten hos modellen. Oftast om det finns uppmätta värden på maxdjupet, är även medeldjupet uppmätt.

Medeldjupet uppvisade antydan till liknande korrelationer som maxdjupet under PCA och PLS-regression, dock erhölls inte lika starka samband här. Vid den multipla regressionen av medeldjup och studier av residualerna syntes tydlig heteroskedasticitet. En regressionsekvation kunde trots detta tas fram med sjöarea, deltah_{1020} samt $s_{medel\,100}$ som prediktorvariabler, men med låg förklaringsgrad och hög varians. Prediktionskraften hos modellen kan även ifrågasättas utifrån residualernas utseende. En transformering av datamängden för att uppnå bättre residualer utfördes, men det resulterade i en icke-signifikant modell med höga p-värden och låga R^2-värden.

På samma sätt som för volym hade det för maxdjup och medeldjup förutspåtts ett samband med andel jordbruksmark. En antydan till negativ korrelation mellan maxdjupet och andel jordbruksmark och andel öppen mark kunde ses under första stegen i PCA (figur 9). Denna korrelation blev dock ännu mindre tydlig då datamängden rensades på vissa variabler och gjordes om. Maxdjupet placerades då i kvadranten bredvid andel jordbruksmark, och på i princip samma höjd, vilket snarare kan tolkas som att de inte korrelerar. Under PLS-regressionen kunde sedan denna korrelation helt bortses från, och hypotesen att andel jordbruksmark i en zon $<$ 500 m från strandlinjen har ett samband med djupet i sjön kan därmed förkastas.

4.4 DISKUSSION METOD

De tre analysmetoderna som användes är var och en vanliga metoder för att analysera datamängder och hitta korrelationer. I denna studie var kombinationen ett bra val, då datamängden var så pass stor och antalet prediktorvariabler så pass många att det inte hade varit varken effektivt eller möjligt att endast använda MLR.

Analyserna av datamängderna med hjälp av PCA visade att en stor del av variansen i hela datamängden kan förklaras med denna metod. En del strukturer och antydan till korrelationer påvisades, och det gick även att urskilja vilka variabler som har stor påverkan på variansen i datamängden i stort (bland annat sjöarnas placering över havet). För syftet i denna studie är det bra som ett första steg, men räcker inte som ensam analysmetod. Då PCA endast maximerar variansen i hela datamängden och inte delar upp variablerna i prediktor- och responsvariabler kan somliga variabler vara väldigt betydelsefulla för att förklara variansen, men inte alls ha någon påverkan på de variabler som i slutändan ska predikteras. I efterhand hade förmodligen samma resultat uppnåtts med endast PLS-regression och MLR, men PCA var ett bra första steg för att få en överblick över datamängden. PLS-regressionen fungerade bra för att hitta samband mellan responsvariablerna och prediktorvariablerna, och till skillnad från PCA är denna metod utformad för att

maximera kovariansen mellan responsvariabeln och prediktorvariablerna, vilket leder till att korrelationerna är lättare att se och det är även lättare att se vilka variabler som faktiskt har ett samband med och en påverkan på responsvariabeln. Som ett exempel här finns höjderna inom de minsta buffertzonerna samt sjöns höjd över havet, som kom fram som tydliga variabler som påverkade datamängderna under PCA. Under PLS-regressionen visade de knapp påverkan på någon av responsvariablerna. Detta styrker att de främst beskriver variansen i datamängderna i stort, men att de inte nödvändigtvis korrelerar med någon av responsvariablerna.

Fördelen med PLS-regression är även att den kan hantera samvarierande variabler, och på så sätt erhålla starkare prediktionssamband än det som erhålls vid MLR. Vid PLS-regressionen kan alla de tre variabler som påverkade både volymen och maxdjupet mest (sjöarea, perimeter och strandlinjeutveckling) behållas i modelleringen. Under MLR kunde endast en av dessa användas, vilket gör att den förklaringsgrad och prediktionskraft som de två övriga bidrar med förloras. Svårigheten med PLS-regression är dock att få ut en enkel ekvation som lätt går att implementera. I SIMCA produceras en mängd olika koefficienter för varje prediktorvariabel, och de representerar alla olika sätt att skriva en modellekvation på. Även ekvation 4 är mer svårtolkad än den ekvation som erhålls vid MLR. Om endast en metod ska användas bör den väljas utifrån tänkt tillämpningsområde, och hur slutresultatet ska se ut. Här var målet en enkel tillämpningsbar ekvation, vilket gjorde en kombination av PLS-regression och MLR till ett bra val. Ett problem med MLR och bristen på hantering av samvariation är dock att det är lätt att flera variabler som faktiskt samvarierar hamnar i ekvationen ändå och en hög förklaringsgrad erhålls. Men, prediktionskraften blir dålig och osäker. Det kräver att användaren har kunskaper kring variablerna som modelleras och hur de är relaterade till varandra. Fördelen med att använda metoderna i kombination är att under PLS-regressionen blir det tydligt vilka prediktorvariabler som samvarierar, vilket underlättar urskiljningen inför MLR.

Det extremvärde som hittades vid PCA behölls i PLS-regressionen och MLR. Detta då sjöns data inte var tillräckligt utstickande för en uteslutning. Om ett extremvärde ska tas bort ur en datamängd krävs noga övervägande och en tydlig motivering till varför datapunkten bör uteslutas. Risken är att extremvärden tas bort för lättvindigt för att förbättra resultatet. Vid PLS-regressionen kom inte heller denna sjö fram som ett lika tydligt extremvärde som det gjorde vid PCA, vilket också indikerar att valet att behålla sjön var korrekt. Detta gällde för både maxdjupet och volymen, men för medeldjupet lade sig Tarfalasjön som ett lika tydligt extremvärde som vid PCA. Förmodligen är det medeldjupet eller variabler som påverkar medeldjupet i denna sjö som gör att det blir ett extremvärde i detta fall.

För att sambanden som hittas vid PLS-regressionen ska vara prediktionskraftiga gäller det att de är linjära. Det blev tydligt vid analysen av volym att så var inte fallet. Dock när log10-transformering utfördes kunde linjära samband hittas, och därmed kunde hög prediktionskraft erhållas. Även för maxdjupet erhölls mer linjära samband vid log10-transformering. Att studera linearitеten mellan respons- och prediktorvariabler är en viktig del av analysen. Tydlig linearitet i datamängden visar på att det finns starka samband, även om exempelvis R^2_Y-värdet inte blir högre än 0.4 för den här typen av datamängd. Linearitеten, R^2_Y och Q^2_Y-värden behöver vägas ihop för att göra en bedömning av hur användbara sambanden faktiskt är. PLS-regressionen då även sjöar med större area än 10

km^2 togs med gav generellt högre värden främst på R^2_Y, men även på Q^2_Y. Detta beror antagligen på att de större sjöarna bidrar till linearitet i datamängden, och på så sätt förbättrar anpassningsgraden och den generella prediktionskraften, men resulterar i en försämrad prediktionskraft för de mindre sjöarna, som är de sjöar som är mest intressanta i detta fall. Problemet med att använda denna modell är även att modellen anpassar sig mer efter de större sjöarna än efter de mindre, och blir därför felaktig. Detta främst för otransformerade data.

4.5 FÖRSLAG TILL VIDARE FORSKNING

Resultaten visar på att med bättre underlag kan en bättre modell erhållas. En intressant variabel som det inte fanns möjlighet att ta fram under denna tidsperiod är ytan av varje sjös avrinningsområde. Då lutningen inom den individuella zonen var den som utmärkte sig för både volymen och djupet i jämförelse med de bestämda zonerna skulle det vara intressant att se vad avrinningsområdets storlek samt de olika lutningarna i avrinningsområdet skulle ha för påverkan. Detta eftersom den individuella zonen är framtagen utifrån varje sjös storlek, och precis på samma sätt som avrinningsområdet varierar storleken för varje sjö. Vid vidare studier rekommenderas därför att ta fram avrinningsområdenas storlek, exempelvis utifrån flödesackumulationskartor och använda både avrinningsområdets storlek samt de olika lutningarna som prediktorvariabler.

Det finns även ett antal sjöar som har uppmätta värden på max-, medeldjup och volym, men som inte har en motsvarande sjöpolygon i SVAR. En möjlighet är att vissa av dessa har motsvarade sjöpolygon i vägkartan, som inte användes här till skillnad mot Sobek m. fl. (2011) som använde sjöpolygoner från både vägkartan och SVAR. Förslag på vidare studier är att även använda de sjöar och dess polygoner som finns i vägkartan, samt att skapa nya sjöpolygoner för de sjöar som har data men saknar polygon. Detta skulle skapa ett större underlag till analyserna.

För att ytterligare exemplifiera modellerna samt illustrera osäkerheterna skulle liknande räkneexempel som Sobek m. fl. (2011) gjorde kunna utföras, där de beräknade medelvolymen hos en grupp av sjöar i ett avrinningsområde. Den slutgiltiga modellen för volym som de hittade gav mindre osäkerheter då den användes för att beräkna medelvolymen hos 15 sjöar. Liknande exempel skulle även kunna utföras här för att exemplifiera tillämpningsområden där osäkerheterna inte blir lika stora.

5 SLUTSATSER

Utifrån resultat och diskussion kunde följande slutsatser dras:

- Sjövolymen i svenska sjöar kan predikteras utifrån sjöarea och medianlutningen i den individuella zonen. De övriga kartparametrarna som visar starka samband med volymen är perimetern, strandlinjeutvecklingen, skillnaden i medelhöjd mellan 10-20 och 20-30 m buffertzonerna samt maxlutningen i den individuella zonen.
- Maxdjupet uppvisar signifikant korrelation med sjöarea, perimeter, strandlinjeutveckling, skillnaden i medelhöjd mellan 10-20 och 20-30 m buffertzonerna samt max- och medianlutningen i den individuella zonen. En något högre förklaringsgrad än tidigare studier erhålls med dessa variabler och det nya kartmaterialet.
- Medeldjupet kan inte modelleras utifrån endast kartparametrar, men visar starka samband med maxdjupet.
- Det nya kartmaterialet bidrar till modeller med högre säkerhet och förbättrade prediktioner än tidigare studier där den individuella zonen bidrar ytterligare.

De ekvationer som gav högst förklaringsgrad och minst osäkerheter (för maxdjupet ekvation 21, R^2=0.42 och för volymen ekvation 18, R^2=0.87) är de som rekommenderas för användning och vidare utveckling.

Referenser

Cael, B. B., Heathcote, A. J. & Seekell, D. A. (2017). *The volume and mean depth of Earth's lakes.* Geophys. Res. Lett., vol. 44, ss. 209–218. Tillgänglig: http://onlinelibrary.wiley.com/doi/10.1002/2016GL071378/full [2017-06-19]

Eriksson, L., Johansson, E., Kettaneh-Wold, N., Trygg, J., Wikström, C. & Wold, S. (2006). *Multi- and Megavariate Data Analysis. Part 1: Basic Principles and Applications.* 2 uppl. Umeå. Umetrics AB.

Gorham, E. & Boyce, F M. (1989). Influence of Lake Surface Area and Depth Upon Thermal Stratification and the Depth of the Summer Thermocline. *Journal of Great Lakes Research*, vol. 15 (2), ss. 233-245.

Havs- och vattenmyndigheten. (2013). *Ordbok*. Tillgänglig: https://www.havochvatten.se/funktioner/ordbok/ordbok.html [2017-01-18]

Helsel, D.R. & Hirsch, R.M. (2002). *Statistical Methods in Water Resources*. Chapter A3. U.S. Geological Survey. ss. 221-261 & 295-320.

Hollister, J. W., Milstead W. B. & Urrutia, M. A. (2011). Predicting Maximum Lake Depth from Surrounding Topography. *PLOS ONE*, vol. 6:9, e25764.

Håkanson, L. (2004). *Lakes: Form and function*. Caldwell (NJ): The Blackburn Press.

Joliffe, I.T. (1992). Principal component analysis and exploratory factor analysis. *Statistical Methods in Medical Research.* vol. 1 ss. 69-95.

Lantmäteriet. (2016). *Kvalitetsbeskrivning nationell höjdmodell*. Lantmäteriet, version 1.1. Tillgänglig: https://www.lantmateriet.se/globalassets/kartor-och-geografisk-information/hojddata/produktbeskrivningar/kvalitetsbeskrivning_nh.pdf [2017-01-26]

Lantmäteriet. (2017) *Mer fakta och metadata för GSD-Höjddata, grid 2+*. Tillgänglig: https://www.lantmateriet.se/sv/Kartor-och-geografisk-information/Hojddata/GSD-Hojddata-grid-2/Mer-fakta-och-metadata/ [2017-01-26]

Miljöstatistik (u.å.). *Miljöstatistik. PLS.* Tillgänglig: http://www.miljostatistik.se/PLS.html [2017-01-26]

Millennium Ecosystem Assessment. (2005). *Ecosystems and Human Well-being: Synthesis.* Island Press, Washington, DC. Tillgänglig: http://www.millenniumassessment.org/documents/document.356.aspx.pdf [2017-02-27]

Naturvårdsverket (2007). *Status, potential och kvalitetskrav för sjöar, vattendrag, kustvatten och vatten i övergångszon*. Stockholm: Naturvårdsverket. (Handbok 2007:4, utgåva 1) Tillgänglig: https://www.havochvatten.se/download/18.276e7ae81443563a750483d/1395245642661/nv-handbok-2007-4-status-potential-och-kvalitetskrav+620-0147-6.pdf [2017-01-19]

Persson, J., Fridell, K., Gustafsson, E-L., & Englund, J-E. (2014). *Att räkna på vatten – en formelsamling för landskapsingenjörer*. Alnarp: Sveriges Lantbruksuniversitet. (Landskapsarkitektur, Trädgård, Växtproduktionsvetenskap, Rapportserie: 2014:17) Tillgänglig: http://pub.epsilon.slu.se/11781/11/persson_j_etal_150203.pdf [2017-01-18]

Pilesjö, P. (u.å.). *Vad är GIS?* Tillgänglig: http://www.gis.lu.se/vadargis.htm [2017-01-19]

Rosipal R. & Krämer N. (2006) *Overview and Recent Advances in Partial Least Squares*. I: Saunders C., Grobelnik M., Gunn S., Shawe-Taylor J. (eds) Subspace, Latent Structure and Feature Selection. Lecture Notes in Computer Science, vol 3940. Springer, Berlin, Heidelberg. ss. 34-51.

Sawatsky, M. L., Clyde, M. & Meek, F. (2015). Partial Least Squares Regression in the Social Scienes. *The Quantitative Methods For Psychology.* vol. 11:2.

Shlens, J. (2005). *A Tutorial on Principal Component Analysis, CA.* Salk Institute for Biological Studies & University of California. Version 2.

SMHI (2008). *Faktablad om Sveriges sjöar.* Tillgänglig: http://www.smhi.se/polopoly_fs/1.504!Faktablad$%$252039_webb.pdf [2017-01-26]

SMHI (2015). *Sveriges sjöar.* Tillgänglig: http://www.smhi.se/kunskapsbanken/hydrologi/sveriges-sjoar-1.4221 [2017-01-18]

SMHI (u.å.). *Svenskt vattenarkiv.* Tillgänglig: http://www.smhi.se/klimatdata/hydrologi/svenskt-vattenarkiv [2017-02-21]

Sobek, S., Nisell, J. & Fölster J. (2011). Predicting the volume and depths of lakes from map-derived parameters. *Inland Waters*, vol. 1, ss. 177-184.

Widén-Nilsson, E., Djodjic, F., Englund, D., Hellgren, S., Liljeberg, M., Olshammar, M., Olsson, H., Orback, C. & Tengdelius Brunell, J. (2016). *Kartdata till PLC6*. Norrköping: Sveriges Meteorologiska och Hydrologiska Institut. (SMED Rapport Nr 186 2016).

Wold, H. (1982). Soft modelling, The basic design and some extensions, I: K.-G. Joreskog, H. Wold Eds, *Systems Under Indirect Observation*, vol. I och II,

North-Holland, Amsterdam.

Åmand, L. (2016) *Multivariat teknik och processoptimering.* Föreläsning (2016-09-06). IVL Svenska Miljöinstitutet, Processmodellering och IT.

A APPENDIX

A.1 PYTHONKOD

A.1.1 Skript för bestämda zoner

Steg 1

Kör först skriptet för att klippa ut DEM-rasterfiler från en stor rasterfil. Variant på detta finns under avsnitt A.1.3.

Steg 2

Beräknar lutningsrasters i de olika bestämda zonerna utifrån erhållna rasterfiler. Samma skript användes, något modifierat, för att beräkna slope till de individuella zonerna i avsnitt A.1.2.

```
# -*- coding: utf-8 -*-

import datetime
import os
import numpy
import shutil
import csv
import arcpy
from arcpy import env
from arcpy.sa import *
from shutil import copy

# Check out the ArcGIS Spatial Analyst extension license
arcpy.CheckOutExtension("Spatial")

#this script uses DEM-rasters to calculate new sloperasters

def main():
    label = "slope"
    import time

    #get workingdirectory
    workdir = os.getcwd()
    indata = """{}/clippedraster_dem/""".format(workdir) # Set
       indata folder, this is where the inrasters are
    make_dirifnotexist(indata)#create directory if it does not
       exist

    #create outdirectory, this is where the new calculated
       rasters should end up
    outdir = """{}/sloperasters/""".format(workdir)
    make_dirifnotexist(outdir)

    #  Make a list out of the inraster-files
```

```
    rasterlist = get_listoffiles(indata, "tif", spliton=".",
        isfullpath=False)

    #Start timer
    time.clock()
    counter=0
    #for all rasters in rasterlist
    for raster in rasterlist:
        counter=counter+1
        print(raster, counter)
        newraster = indata + raster
        # calculate a  slope-raster, with the slope in degrees
        outSlope = arcpy.sa.Slope(newraster, "DEGREE")
        utraster = get_newshallowfilename(outdir + raster, "", "
            _slope")
        # Save the out raster in output directory
        outSlope.save(outdir + utraster)

    # TIMER
    clocktiming = showtimeinfo(time.clock())
    print (clocktiming)
    with open("Timer.txt", "a") as f:
        f.write(clocktiming)

    print("The script finished")

# here follows all functions
def get_newshallowfilename(filnamn, stringtoaddbefore="",
    stringtoaddafter=""):
    """
        indata: a path to file, and two strings to add. e.g "c:\
            H\og.txt", "pre_", "_post"
        outdata:  a path to file with strings added. e.g. "
            pre_og_post.txt"
    """
    import os
    # Example: filnamn = c:/H/SLU/calc.xlsx
    filtomext, extension = os.path.splitext(filnamn)  # example:
        c:/H/SLU/calc, xlsx
    justfilnoext = os.path.basename(filtomext)  # example: calc
    nyttnamn = """{}{}{}{}""".format(stringtoaddbefore,
        justfilnoext, stringtoaddafter, extension)
    return nyttnamn

def showtimeinfo(t):
    s = int(t)
    m = int(s/60.0)
    h = int(m/60.0)
    sek = s - m*60
    minu = m - h*60
```

```
    timeinfo = "\nTime passed so far hour:min:sek = " + str(h).
       zfill(2) + ":" + str(minu).zfill(2) + ":" + str(sek).
       zfill(2)+"\n"
    return timeinfo

def make_dirifnotexist(dirname):
    import os
    if not os.path.isdir(dirname):
        os.mkdir(dirname)
        print "made a new dir : ", dirname

def get_listoffiles(thepath, extension, spliton=".", isfullpath=
   True):
    """
       Takes the path and extension (string).
       Lists all files that match extension in that path.
       Returns a list of file paths.
    """
    import os
    workSpace = thepath
    ResultFiles_list = []
    for fil in os.listdir(workSpace):
        if isfullpath:
            fil = os.path.join(thepath, fil)

        try:
            base, ext = fil.split(spliton)
            if ext == extension:
                ResultFiles_list.append(fil)
            elif extension == "allafiler":
                ResultFiles_list.append(fil)
        except:
            pass
    return ResultFiles_list
if __name__ == "__main__":
    main()
```

Steg 3

Beräknar statistik på de erhållna lutningsrasterfilerna. Kan användas för att beräkna statistik på alla rasterfiler, och användes även för att beräkna statistik för DEM-raster filer under avsnitt A.1.3 samt för de individuella zonerna.

```
# -*- coding: utf-8 -*-

import shutil
import csv
import os
import gdal
import fiona
```

```
import shapely
import numpy
import rasterio
import rasterstats
from rasterstats import zonal_stats
import geopandas as gpd

#this script calculates statistics from a rasterfile using a
    polygon that defines the area,
#and to write all calculated statistics to a csv file
#used for all raster statistics, altered to match specific
    purposes

def main():

    import time
    workdir = os.getcwd()
    indata1 = """{}/sloperasters/""".format(workdir) #where
       rasterfiles are
    rasterlist = get_listoffiles(indata1, "tif", spliton=".",
       isfullpath=False)

    #Start timer
    time.clock()

    #create csv-file to store stats in
    csvfile = "D:/sloperaster/slopestat.csv"
    with open(csvfile, "w") as output:
        writer = csv.writer(output, lineterminator='\n')
        headers = ["sjoid", "bufferdist", "statistics", "value"]
        writer.writerow(headers)

    raster_statistics(indata1, csvfile)

    # TIMER
    clocktiming = showtimeinfo(time.clock())
    print (clocktiming)
    with open("Timer.txt", "a") as f:
        f.write(clocktiming)

    print("The script finished")

def raster_statistics(indata1, csvfile):
    workdir = os.getcwd()
    tmpdir = """{}/tmpdir/""".format(workdir)
    make_dirifnotexist(tmpdir)

    #load shapepolygons as "bands"
    indata = """{}/polygonswithoutlake/""".format(workdir)
    make_dirifnotexist(indata)
```

```
    shapelist = get_listoffiles(indata, "shp", spliton=".",
        isfullpath=False)
    statprop = []
    raknare = 0
    for shape in shapelist:
        with open(csvfile, "a") as output:
            writer = csv.writer(output, lineterminator='\n')
            raknare+=1
            #read shapefile as gpd to work with zonal_stats
            shp = gpd.read_file("""{}/{}""".format(indata,shape)
                )
            name = shape[4:10]+ "-" + shape[11:17]
            sjoid = name
            dist = shape[18:21]
            #name with bufferdistance, match the names of
                sloperasters in indata1
            rasternamn = sjoid + "_dem_buf1000_slope.tif"
            inraster = indata1 + rasternamn
            print(sjoid)
            print("Start zonal_stats")
            mini = zonal_stats(shp, inraster, stats=['min'])
            maxi = zonal_stats(shp, inraster, stats=['max'])
            median = zonal_stats(shp, inraster, stats=['median
                '])
            medel = zonal_stats(shp, inraster, stats=['mean'])
            #sjuttiofemper = zonal_stats(shp, inraster, stats=['
                percentile_75'])

            statprop = [sjoid, dist, "min", mini[0]["min"]]
            writer.writerow(statprop)
            statprop = [sjoid, dist, "max", maxi[0]["max"]]
            writer.writerow(statprop)
            statprop = [sjoid, dist, "median", median[0]["median
                "]]
            writer.writerow(statprop)
            statprop = [sjoid, dist, "medel", medel[0]["mean"]]
            writer.writerow(statprop)
            #statprop = [sjoid, dist, "sjuttiofemper",
                sjuttiofemper[0]["percentile_75"]]
            #writer.writerow(statprop)

def showtimeinfo(t):
    s = int(t)
    m = int(s/60.0)
    h = int(m/60.0)
    sek = s - m*60
    minu = m - h*60
    timeinfo = "\nTime passed so far hour:min:sek = " + str(h).
        zfill(2) + ":" + str(minu).zfill(2) + ":" + str(sek).
        zfill(2)+"\n"
    return timeinfo
```

```
97
def make_dirifnotexist(dirname):
99      import os
        if not os.path.isdir(dirname):
101         os.mkdir(dirname)
            print "made a new dir : ", dirname
103
def get_listoffiles(thepath, extension, spliton=".", isfullpath=
    True):
105     """
            Takes the path and extension (string).
107         Lists all files that match extension in that path.
            Returns a list of file paths.
109     """
        import os
111     workSpace = thepath
        ResultFiles_list = []
113     for fil in os.listdir(workSpace):
            if isfullpath:
115             fil = os.path.join(thepath, fil)

117         try:
                base, ext = fil.split(spliton)
119             if ext == extension:
                    ResultFiles_list.append(fil)
121             elif extension == "allafiler":
                    ResultFiles_list.append(fil)
123         except:
                pass
125     return ResultFiles_list

127 if __name__ == '__main__':
        main()
```

A.1.2 Skript för individuella zoner

För att bestämma individuella zoner för alla sjöar, baserat på maxavståndet från strandlinjen kördes ett antal skript i följande ordning.

Steg 1

Alla sjöpolygoner gjordes om från polygoner till polylines för att kunna använda *Euclidian distance*-verktyget.

```
# -*- coding: utf-8 -*-
2
import shutil
4 import os
import arcpy
6 from arcpy.sa import *
```

```
#this script takes rasters to get lakeid and then identifies all
#lakes with same lakeid in shapefile. can use list aswell

arcpy.CheckOutExtension("Spatial")
#globals
keyInAttrTable = "SJOID"

def main():
    workdir = os.getcwd()
    indata1 = """{}""".format(workdir)
    rasterlist = get_listoffiles(indata1, "tif", spliton=".",
       isfullpath=False)

    for raster in rasterlist:
        sjoid = raster[0:13] #get lakeid from name
        infotext = """\Sjoid: {}""".format(sjoid)
        print(infotext)
        polygon_to_line(sjoid, raster)


def polygon_to_line(lakeid, raster):
    global keyInAttrTable
    workdir = os.getcwd()
    label = "poly"
    #indatafolder
    indata = """{}/indata_shp_{}/""".format(workdir, label) #
       This is where the shapefile is
    make_dirifnotexist(indata)

    outdir = """{}/polylines_{}/""".format(workdir, label)
    make_dirifnotexist(outdir)

    tmpdir = """{}/tmpdir/""".format(workdir)
    make_dirifnotexist(tmpdir)

    shapelist = get_listoffiles(indata, "shp", spliton=".",
       isfullpath=False)

    #load inputfeature, the shapefile with polygons
    inputfeature = get_newshallowfilename(shapelist[0])

    # Folder where the maps to clip from is placed (
       smhi_vattenforekomster_vattenytor_SVAR2012_2.shp)
    aro_y_shp = indata + inputfeature

    #  A function that dives into the shapefile and finds all
       polygons that will be used to clip with
    polygons = getinfopolygons(aro_y_shp, keyInAttrTable)

    # Loop over the ploygons
```

```
    nr_inprocessing = 0
    totpolys = len(polygons)
    for poly in polygons:
        nr_inprocessing += 1
        name = poly[keyInAttrTable]
        if lakeid == name:
            print("""Poly {nr}/{totnr}. Trying to create
               polyline from polygon {sjoid}""".format(
                       nr=nr_inprocessing, totnr=totpolys,
                          sjoid=name))

            #create outputline featureclass
            outputfeature = """{}{}_{}.shp""".format(tmpdir,
               label, name) #
            print(outputfeature)
            print("Start the polygon to polyline tool")
            arcpy.PolygonToLine_management(poly["feat"],
               outputfeature , "IGNORE_NEIGHBORS")
            shutil.copy(outputfeature, outdir)

                # Clean up in temp dirs, this removes everything
                    below too
            deleteallfilesindir(tmpdir)


def deleteallfilesindir(folder):
    """
    http://stackoverflow.com/questions/185936/delete-folder-
       contents-in-python
    """
    import os
    # Delete all files in dir
    for the_file in os.listdir(folder):
        file_path = os.path.join(folder, the_file)
        try:
            if os.path.isfile(file_path):
                os.unlink(file_path)
                # Uncomment below if also the dirs should be
                   removed
                # elif os.path.isdir(file_path): shutil.rmtree(
                   file_path)
        except Exception as e:
            print (e)


def get_newshallowfilename(filnamn, stringtoaddbefore="",
    stringtoaddafter=""):
    """
        indata: a path to file, and two strings to add. e.g "c:\
           H\og.txt", "pre_", "_post"
```

```
        outdata:  a path to file with strings added. e.g. "
            pre_og_post.txt"
94      """
        import os
96      # Example: filnamn = c:/H/SLU/calc.xlsx
        filtomext, extension = os.path.splitext(filnamn)  # example:
            c:/H/SLU/calc, xlsx
98      justfilnoext = os.path.basename(filtomext)  # example: calc
        nyttnamn = """{}{}{}{}""".format(stringtoaddbefore,
            justfilnoext, stringtoaddafter, extension)
100     return nyttnamn


102
  def make_dirifnotexist(dirname):
104     import os
        if not os.path.isdir(dirname):
106         os.mkdir(dirname)
            print "made a new dir : ", dirname
108
  def getinfopolygons(shapefile, keyInAttrTable):
110     """
        This function iterates over each polygon in the shapefile
            and pick shape extent and table content.
112     Headers in attribute table (we will only pick a few of these
            ):
            VYID,YTKOD,VYNAMN,SJOID,SJONAMN,EU_CD,MS_CD,VF,VDRID,
                DISTRICT,COMP_AUTH,COUNTRY,VYHOJD,VERSION,Shape_Leng,
114         Shape_Area

116     The function returns a list ([]) of dicts ({}) describing
            the polygons. Example:
            [{"boundaries": "232...743", "objectid": "655456-898899"
                }, {"boundaries": "456...644", "objectid": "
                7854687-879876"}, ...]
118
        """
120     shapeName = arcpy.Describe(shapefile).shapeFieldName
        rows = arcpy.SearchCursor(shapefile)
122     lst_of_dicts_polygons = []
        for row in rows:
124         feat = row.getValue(shapeName)  # Value in the attribute
                table of the shape
            extent = feat.extent
126         geom = row.Shape
            namn = row.getValue(keyInAttrTable)  # Value in the
                attribute table of the shape
128         boundaries = """{}_{}_{}_{}""".format(extent.XMin,
                extent.YMin, extent.XMax, extent.YMax)
            objectid = row.getValue("VYID")  # Uniqe value in the
                attribute table of the shape
```

```
130         dict_poly = {keyInAttrTable: namn, "boundaries":
               boundaries, "objectid": objectid, "geom": geom, "feat
               ": feat}
            lst_of_dicts_polygons.append(dict_poly)
132     return lst_of_dicts_polygons

134 def get_listoffiles(thepath, extension, spliton=".", isfullpath=
      True):
     """
136         Takes the path and extension (string).
            Lists all files that match extension in that path.
138         Returns a list of file paths.
     """
140     import os
     workSpace = thepath
142     ResultFiles_list = []
     for fil in os.listdir(workSpace):
144         if isfullpath:
                fil = os.path.join(thepath, fil)
146
            try:
148             base, ext = fil.split(spliton)
                if ext == extension:
150                 ResultFiles_list.append(fil)
                elif extension == "allafiler":
152                 ResultFiles_list.append(fil)
            except:
154             pass
     return ResultFiles_list
156
 if __name__ == "__main__":
158     main()
```

Steg 2

Använd polylines från steg 1 för att beräkna avstånd från strandlinje.

```
 # -*- coding: utf-8 -*-
2
 import datetime
4 import shutil
 import os
6 import arcpy
 from arcpy.sa import *
8 from arcpy import env

10 #this script uses polylines to create a distanceraster
 #use polygondistance script first to get polylines.
12
 arcpy.CheckOutExtension("Spatial")
```

```
#globals
keyInAttrTable = "SJOID"

def main():
    import time
    workdir = os.getcwd()
    indata = """{}/polylines_poly""".format(workdir) #load
       polylines created in 02_01polygondistance
    polylinelist = get_listoffiles(indata, "shp" , spliton=".",
       isfullpath=False)

    #Start timer
    time.clock()

    for line in polylinelist:
        print line
        newname = line[0:18]
        outdistraster = """{}/distancerasters/{}.tif""".format(
           workdir, newname)
        inputfeature = """{}/polylines_poly/{}""".format(workdir
           , line)
        outeudistance = EucDistance(inputfeature)
        outeudistance.save(outdistraster)

    #Timer
    clocktiming = showtimeinfo(time.clock())
    print(clocktiming)
    with open("Timer.txt", "a") as f:
        f.write(clocktiming)


def make_dirifnotexist(dirname):
    import os
    if not os.path.isdir(dirname):
        os.mkdir(dirname)
        print "made a new dir : ", dirname

def showtimeinfo(t):
    s = int(t)
    m = int(s/60.0)
    h = int(m/60.0)
    sek = s - m*60
    minu = m - h*60
    timeinfo = "\nTime passed so far hour:min:sek = " + str(h).
       zfill(2) + ":" + str(minu).zfill(2) + ":" + str(sek).
       zfill(2)+"\n"
    return timeinfo


def get_listoffiles(thepath, extension, spliton=".", isfullpath=
    True):
```

```
    """
        Takes the path and extension (string).
        Lists all files that match extension in that path.
        Returns a list of file paths.
    """
    import os
    workSpace = thepath
    ResultFiles_list = []
    for fil in os.listdir(workSpace):
        if isfullpath:
            fil = os.path.join(thepath, fil)

        try:
            base, ext = fil.split(spliton)
            if ext == extension:
                ResultFiles_list.append(fil)
            elif extension == "allafiler":
                ResultFiles_list.append(fil)
        except:
            pass
    return ResultFiles_list

if __name__ == "__main__":
    main()
```

Steg 3

Extrahera endast sjöpolygonens rasterfil för att få avståndet inne i sjön från strandlinjen.

```
# -*- coding: utf-8 -*-

import datetime
import shutil
import os
import arcpy
from arcpy.sa import *
from arcpy import env

arcpy.CheckOutExtension("Spatial")
#globals
keyInAttrTable = "SJOID"

#this script extracts only the lakeraster from distancerasters
   created in distance_poly
#this to get the distance from shoreline

def main():
    import time
    workdir = os.getcwd()
```

```
    indata = """{}/distancerasters/""".format(workdir) #load
        rasters created in 02_02distancepoly

    #Start timer
    time.clock()
    extract_mask(indata)

    # TIMER
    clocktiming = showtimeinfo(time.clock())
    print (clocktiming)
    with open("Timer.txt", "a") as f:
        f.write(clocktiming)

    print("The script finished")


def extract_mask(inrasterdir):

    workdir = os.getcwd()
    tmpdir = """{}/tmpdir/""".format(workdir)

    indata1 = """{}/indata_shp_poly/""".format(workdir) #
        shapefile with lakepolygons (SMHI)

    outdir = """{}/extracted_rasters/""".format(workdir)

    shapelist = get_listoffiles(indata1, "shp", spliton=".",
        isfullpath=False)

    #load inputfeature, the shapefile with polygons
    inputfeature = get_newshallowfilename(shapelist[0])

    # Folder where the maps to clip from is placed (
        smhi_vattenforekomster_vattenytor_SVAR2012_2.shp)
    aro_y_shp = indata1 + inputfeature

    #  A function that dives into the shapefile and finds all
        polygons that will be used
    polygons = getinfopolygons(aro_y_shp, keyInAttrTable)

    # Loop over the ploygons
    nr_inprocessing = 0
    totpolys = len(polygons)
    for poly in polygons:
        nr_inprocessing += 1
        name = poly[keyInAttrTable]
        f1= name[0:6]
        f2 = name[7:13]
        newname = f1 + "_" + f2
        raster = "line_" + newname + ".tif"

```

```
67          #create outputline featureclass
            #name cannot contain - so use _ instead when creating
               outputname
69
            outputraster= """{}extract_{}_{}.tif""".format(tmpdir,
               f1, f2)#
71
            print "Start extract by mask"
73          utraster = ExtractByMask(inrasterdir + raster, poly["
               feat"])
            utraster.save(outputraster)
75          movelist = get_listoffiles(tmpdir, "tif")
            del utraster
77          for afile in movelist:
                shutil.copy(afile, outdir)
79          print("moved all tif for:" + str(name) + "\n")

81          # Clean up in temp dirs
            deleteallfilesindir(tmpdir)
83

85  def deleteallfilesindir(folder):
        """
87      http://stackoverflow.com/questions/185936/delete-folder-
           contents-in-python
        """
89      import os
        # Delete all files in dir
91      for the_file in os.listdir(folder):
            file_path = os.path.join(folder, the_file)
93          try:
                if os.path.isfile(file_path):
95                  os.unlink(file_path)
                    # Uncomment below if also the dirs should be
                       removed
97                  # elif os.path.isdir(file_path): shutil.rmtree(
                       file_path)
            except Exception as e:
99              print (e)


101
    def get_newshallowfilename(filnamn, stringtoaddbefore="",
        stringtoaddafter=""):
103     """
            indata: a path to file, and two strings to add. e.g "c:\
               H\og.txt", "pre_", "_post"
105         outdata:  a path to file with strings added. e.g. "
               pre_og_post.txt"
        """
107     import os
```

```
    # Example: filnamn = c:/H/SLU/calc.xlsx
109 filtomext, extension = os.path.splitext(filnamn)  # example:
        c:/H/SLU/calc, xlsx
    justfilnoext = os.path.basename(filtomext)  # example: calc
111 nyttnamn = """{}{}{}{}""".format(stringtoaddbefore,
        justfilnoext, stringtoaddafter, extension)
    return nyttnamn
113
  def getinfopolygons(shapefile, keyInAttrTable):
115 """
    This function iterates over each polygon in the shapefile
        and pick shape extent and table content.
117 Headers in attribute table (we will only pick a few of these
        ):
         VYID,YTKOD,VYNAMN,SJOID,SJONAMN,EU_CD,MS_CD,VF,VDRID,
            DISTRICT,COMP_AUTH,COUNTRY,VYHOJD,VERSION,Shape_Leng,
119      Shape_Area

121 The function returns a list ([]) of dicts ({}) describing
        the polygons. Example:
         [{"boundaries": "232...743", "objectid": "655456-898899"
            }, {"boundaries": "456...644", "objectid": "
            7854687-879876"}, ...]
123
    """
125 shapeName = arcpy.Describe(shapefile).shapeFieldName
    rows = arcpy.SearchCursor(shapefile)
127 lst_of_dicts_polygons = []
    for row in rows:
129     feat = row.getValue(shapeName)  # Value in the attribute
            table of the shape
        extent = feat.extent
131     geom = row.Shape
        namn = row.getValue(keyInAttrTable)  # Value in the
            attribute table of the shape
133     boundaries = """{}_{}_{}_{}""".format(extent.XMin,
            extent.YMin, extent.XMax, extent.YMax)
        objectid = row.getValue("VYID")  # Uniqe value in the
            attribute table of the shape
135     dict_poly = {keyInAttrTable: namn, "boundaries":
            boundaries, "objectid": objectid, "geom": geom, "feat
            ": feat}
        lst_of_dicts_polygons.append(dict_poly)
137 return lst_of_dicts_polygons

139 def showtimeinfo(t):
    s = int(t)
141 m = int(s/60.0)
    h = int(m/60.0)
143 sek = s - m*60
    minu = m - h*60
```

```
145     timeinfo = "\nTime passed so far hour:min:sek = " + str(h).
           zfill(2) + ":" + str(minu).zfill(2) + ":" + str(sek).
           zfill(2)+"\n"
        return timeinfo
147
  def get_listoffiles(thepath, extension, spliton=".", isfullpath=
      True):
149     """
            Takes the path and extension (string).
151         Lists all files that match extension in that path.
            Returns a list of file paths.
153     """
        import os
155     workSpace = thepath
        ResultFiles_list = []
157     for fil in os.listdir(workSpace):
            if isfullpath:
159             fil = os.path.join(thepath, fil)

161         try:
                base, ext = fil.split(spliton)
163             if ext == extension:
                    ResultFiles_list.append(fil)
165             elif extension == "allafiler":
                    ResultFiles_list.append(fil)
167         except:
                pass
169     return ResultFiles_list

171 if __name__ == "__main__":
        main()
```

Steg 4

Beräkna maxvärdet för varje avståndsrasterfil och skriv över till csv-fil.

```
  # -*- coding: utf-8 -*-
2
  import datetime
4 import shutil
  import csv
6 import os
  import arcpy
8 from arcpy.sa import *
  import gdal
10 import fiona
  import shapely
12 import numpy
  import rasterio
14 import rasterstats
```

```
from rasterstats import zonal_stats
import geopandas as gpd
arcpy.CheckOutExtension("Spatial")
#globals
keyInAttrTable = "SJOID"

#this script calculates statistics from a rasterfile using a
   polygon that defines the area.

def main():

    import time
    workdir = os.getcwd()
    indata1 = """{}/extracted_rasters/""".format(workdir) #get
       rasters created in 02_03extractraster
    rasterlist = get_listoffiles(indata1, "tif", spliton=".",
       isfullpath=False)

    #Start timer
    time.clock()

    csvfile = "C:/Users/sara/Desktop/Rasterklipp/Maxdistance.csv
       "
    with open(csvfile, "w") as output:
        writer = csv.writer(output, lineterminator='\n')
        headers = ["raster", "max"]
        writer.writerow(headers)

    for raster in rasterlist:
        sjoid = raster[8:14] + "-" + raster[15:21]
        print(sjoid)
        raster_statistics(sjoid, indata1 + raster, csvfile)

    # TIMER
    clocktiming = showtimeinfo(time.clock())
    print (clocktiming)
    with open("Timer.txt", "a") as f:
        f.write(clocktiming)

    print("The script finished")

def raster_statistics(lakeid, raster, csvfile):
    from geopandas import GeoDataFrame

    workdir = os.getcwd()
    #load shapepolygons lake (SMHI)
    indata = """{}/indata_shp_nodupl/""".format(workdir)

    #get shapefile with all polygons, turn into geodataframe
    test = GeoDataFrame.from_file("C:/Users/sara/Desktop/
       Rasterklipp/indata_shp_nodupl/
```

```
        smhi_vattenforekomster_changed_noduplicates.shp")

    raknare = 0

    statprop = []
    with open(csvfile, "a") as output:
        writer = csv.writer(output, lineterminator='\n')
        #loop over polygons and sjoid in shapefile
        for geo, sjoid in zip(test["geometry"], test["SJOID"]):
            raknare+=1
            name = sjoid
            if lakeid == name:
                print("Start zonal_stats")
                maxi = zonal_stats(geo, raster, stats=['max'])
                statprop = [name, maxi[0]["max"]]
                #create csv-file to store stats in
                writer.writerow(statprop)

def showtimeinfo(t):
    s = int(t)
    m = int(s/60.0)
    h = int(m/60.0)
    sek = s - m*60
    minu = m - h*60
    timeinfo = "\nTime passed so far hour:min:sek = " + str(h).
       zfill(2) + ":" + str(minu).zfill(2) + ":" + str(sek).
       zfill(2)+"\n"
    return timeinfo

def get_listoffiles(thepath, extension, spliton=".", isfullpath=
    True):
    """
        Takes the path and extension (string).
        Lists all files that match extension in that path.
        Returns a list of file paths.
    """
    import os
    workSpace = thepath
    ResultFiles_list = []
    for fil in os.listdir(workSpace):
        if isfullpath:
            fil = os.path.join(thepath, fil)

        try:
            base, ext = fil.split(spliton)
            if ext == extension:
                ResultFiles_list.append(fil)
            elif extension == "allafiler":
                ResultFiles_list.append(fil)
        except:
            pass
```

```
108     return ResultFiles_list

110 if __name__ == '__main__':
        main()
```

Steg 5

Detta skript skapar bufferpolygoner och extraherar zonrasters utifrån beräknade maxavstånd i steg 4. Efter detta skript är kört, använd skript under A.1.1 för att beräkna slope med erhållna rasterfiler. Tillslut, använd skript under samma avsnitt för att beräkna statistik.

```
1 # -*- coding: utf-8 -*-

3 import datetime
  import shutil
5 import os
  import arcpy
7 from arcpy.sa import *
  from arcpy import env
9
  arcpy.CheckOutExtension("Spatial")
11 #globals
  keyInAttrTable = "SJOID"
13
  #this script creates buffer polygons and extracts zonerasters
     from a bigger raster based on polygons
15
  def main():
17
      import time
19    import csv
      from csv import DictReader
21    import itertools
      workdir = os.getcwd()
23    inraster = "" #set path to inraster to clip from (the big
         rasterfile)

25    #get csvfile with rasternames and bufferdistances
      csvfile = "C:/Users/sara/Desktop/Rasterklipp/maxdistmer1000.
         csv"
27    with open(csvfile) as input:
          sjoidlist = [row["raster"] for row in DictReader(input)]
29    with open(csvfile) as input:
          distance = [row["klippavst"] for row in DictReader(input
             )]
31
      label = "dem"
33    #Start timer
      time.clock()
```

```
35      # For each lake make the clipping in the "clip_mask"-
            function
        for (id, dist) in itertools.izip_longest(sjoidlist, distance
            ):
37          sjoid = id  # (tex 615365-134524).
            bufferdist = float(dist) #distance from csvfile, unique
                for each lake
39          clip_mask(inraster, sjoid, label, bufferdist)

41
        # TIMER
43      clocktiming = showtimeinfo(time.clock())
        print (clocktiming)
45      with open("Timer.txt", "a") as f:
            f.write(clocktiming)
47
        print("The script finished")
49

51  def clip_mask(inraster, lakeid, label, bufferdist):

53      global keyInAttrTable

55      workdir = os.getcwd()
        tmpdir = """{}/tmpdir/""".format(workdir)
57      make_dirifnotexist(tmpdir)


59
        outdir = """{}/dem_rasters/""".format(workdir)
61      make_dirifnotexist(outdir)

63      outdir2 = """{}/polygonswithoutlake/""".format(workdir)
        make_dirifnotexist(outdir2)
65
        indatashape = """{}/indata_shp_nodupl/""".format(workdir) #
            This is where the shapefile is to buffer from, the SMHI
            file
67      make_dirifnotexist(indatashape)

69      #  Get a list of shapefiles (but for now it is only 1 item
            in that list)
        shapelist = get_listoffiles(indatashape, "shp", spliton=".",
            isfullpath=True)  # Gets a list of all shapes in indata-
            dir
71
        #  Get shapefilename without the searchpath
73      # I only use 1 shape here (having many polygons),
        #   so the list of shapes contains only 1 element
75      #   (shapelist[0])

77      shapetoclipwith = get_newshallowfilename(shapelist[0])
```

```
    # Folder where the maps to clip from is placed (
       smhi_vattenforekomster_vattenytor_SVAR2012_2.shp)
79  aro_y_shp = indatashape + shapetoclipwith

81  #  A function that dives into the shapefile and finds all
       polygons that will be used to clip with
    polygons = getinfopolygons(aro_y_shp, keyInAttrTable)
83
    # Loop over the ploygons
85  nr_inprocessing = 0
    totpolys = len(polygons)
87  for poly in polygons:
        nr_inprocessing += 1
89      name = poly[keyInAttrTable]
        if lakeid == name:
91          print("""Poly {nr}/{totnr}. Trying to create polygon
                from {sjoid} with {colname}""".format(
                        nr=nr_inprocessing, totnr=totpolys,
                           sjoid=lakeid, colname=name))
93
            in_features = poly["feat"]  # OK
95          buffer_distance_or_field = bufferdist  # in meter
            newname = name[0:6] + "_" + name[7:13]
97          out_feature_class = """{}/poly{}""".format(outdir2,
               newname)
            line_side = "FULL"  # OK, FULL/LEFT/RIGHT/
               OUTSIDE_ONLY
99          line_end_type = "ROUND"  # OK, FLAT/ROUND
            dissolve_option = "ALL"  # OK, NONE/ALL/LIST
101         dissolve_field = None   # OK
            print("arcpy.Buffer_analysis(in_features,
               out_feature_class, buffer_distance_or_field")
103         arcpy.Buffer_analysis(in_features, out_feature_class
               , buffer_distance_or_field, line_side,
               line_end_type,
                                          dissolve_option,
                                             dissolve_field)
105         in_feat2 = out_feature_class + ".shp"

107     outputraster= """{}/{}_{}.tif""".format(outdir, sjoid,
           label)#

109     print "Start clipping"
        arcpy.Clip_management(inraster, "#", outputraster,
           in_feat2, "", "", "")
111

113 def showtimeinfo(t):
        s = int(t)
115     m = int(s/60.0)
        h = int(m/60.0)
```

```
117     sek = s - m*60
        minu = m - h*60
119     timeinfo = "\nTime passed so far hour:min:sek = " + str(h).
           zfill(2) + ":" + str(minu).zfill(2) + ":" + str(sek).
           zfill(2)+"\n"
        return timeinfo
121


123
  def get_newshallowfilename(filnamn, stringtoaddbefore="",
      stringtoaddafter=""):
125     """
            indata: a path to file, and two strings to add. e.g "c:\
               H\og.txt", "pre_", "_post"
127         outdata:  a path to file with strings added. e.g. "
               pre_og_post.txt"
        """
129     import os
        # Example: filnamn = c:/H/SLU/calc.xlsx
131     filtomext, extension = os.path.splitext(filnamn)  # example:
            c:/H/SLU/calc, xlsx
        justfilnoext = os.path.basename(filtomext)  # example: calc
133     nyttnamn = """{}{}{}{}""".format(stringtoaddbefore,
           justfilnoext, stringtoaddafter, extension)
        return nyttnamn
135

137 def make_dirifnotexist(dirname):
        import os
139     if not os.path.isdir(dirname):
            os.mkdir(dirname)
141         print "made a new dir : ", dirname

143 def getinfopolygons(shapefile, keyInAttrTable):
        """
145     This function iterates over each polygon in the shapefile
           and pick shape extent and table content.
        Headers in attribute table (we will only pick a few of these
           ):
147          VYID,YTKOD,VYNAMN,SJOID,SJONAMN,EU_CD,MS_CD,VF,VDRID,
                DISTRICT,COMP_AUTH,COMP_AUTH,COUNTRY,VYHOJD,VERSION,Shape_Leng,
             Shape_Area
149
        The function returns a list ([]) of dicts ({}) describing
           the polygons. Example:
151          [{"boundaries": "232...743", "objectid": "655456-898899"
                }, {"boundaries": "456...644", "objectid": "
                7854687-879876"}, ...]

153     """
        shapeName = arcpy.Describe(shapefile).shapeFieldName
```

```
155     rows = arcpy.SearchCursor(shapefile)
        lst_of_dicts_polygons = []
157     for row in rows:
            feat = row.getValue(shapeName)  # Value in the attribute
                table of the shape
159         extent = feat.extent
            geom = row.Shape
161         namn = row.getValue(keyInAttrTable)  # Value in the
               attribute table of the shape
            boundaries = """{}_{}_{}_{}""".format(extent.XMin,
               extent.YMin, extent.XMax, extent.YMax)
163         objectid = row.getValue("VYID")  # Uniqe value in the
               attribute table of the shape
            dict_poly = {keyInAttrTable: namn, "boundaries":
               boundaries, "objectid": objectid, "geom": geom, "feat
               ": feat}
165         lst_of_dicts_polygons.append(dict_poly)
        return lst_of_dicts_polygons
167
  def get_listoffiles(thepath, extension, spliton=".", isfullpath=
      True):
169     """
            Takes the path and extension (string).
171         Lists all files that match extension in that path.
            Returns a list of file paths.
173     """
        import os
175     workSpace = thepath
        ResultFiles_list = []
177     for fil in os.listdir(workSpace):
            if isfullpath:
179             fil = os.path.join(thepath, fil)

181         try:
                base, ext = fil.split(spliton)
183             if ext == extension:
                    ResultFiles_list.append(fil)
185             elif extension == "allafiler":
                    ResultFiles_list.append(fil)
187         except:
                pass
189     return ResultFiles_list

191 if __name__ == "__main__":
        main()
```

A.1.3 Skript för höjdskillnader

Detta skript skapar tre bufferzoner att beräkna höjdskillnader mellan. Använd sedan skript under avsnitt A.1.1, steg 3, för att beräkna statistik på de erhållna rasterfilerna.

```
# -*- coding: utf-8 -*-

import datetime
import os
import arcpy
import shutil
from arcpy.sa import *
# Check out the ArcGIS Spatial Analyst extension license
arcpy.CheckOutExtension("Spatial")

# this script uses a raster to identify correct lakepolygons, creates a buffer around each polygon
# using the lakepolygons, erase smaller buffered polygons from each bufferpolygon.
# end result is a buffered polygon, as a "band" around each lake.
# can be altered to just erase the lake polygon, to create bufferzones
# from shoreline to bufferdistance, this was done to create determined zones


# GLOBALS
keyInAttrTable = "SJOID"  # This should be a unique identifier found in the attribute table.

def main():
    """
    The script starts here after importing libraries, and reading Globals above.
    """
    import time
    workdir = os.getcwd()
    indata1 = """{}/inrasters""".format(workdir) #this is where the big raster files are, use to get sjoid
    rasterlist = get_listoffiles(indata1, "tif", spliton=".", isfullpath=False)

    #use this instead of rasterlist if list of sjoid is available
    #listoflist = []
    #listoflist = readfile2listoflist("lefttorun.csv", ",")
    #list_sjoid = [row[2] for row in listoflist]   # Pick the 3rd column in the list of lists (=sjoid)
    #listnohead = list_sjoid[1:]  # Cut away header
    # listnohead = ["615365-134524", "639134-132882", "639683-134896", "660054-130123", "660055-129701"]

    outdir = """{}/tmpdir/""".format(workdir)
    make_dirifnotexist(outdir)
```

```
    #the distances used to create bufferzones
    label = "dem"
    list_bufferdist = ["010","020", "030"]  # meter
    # Start timer
    time.clock()

    # For each lake make the clipping in the "
       clipraster_w_polygon"-function
    # Clip the raster for each bufferdistance in each lake

    #use this if listnohead is used above
    #for sjoid in listnohead:
        #print("""Buffering and cutting sjoid: {}""".format(
           sjoid)
        #polygonlist = {}
        #for bufferdist in list_bufferdist:
            #infotext = """\nSjoid: {}, Bufferdist: {}, Inraster
                name: {}""".format(sjoid, bufferdist, inraster)
            #print(infotext)
            #bufferpoly_w_polygon(label, sjoid, bufferdist,
               polygonlist, outdir)

    #use this if rasterlist is used
    for inraster in rasterlist:
        sjoid = inraster[0:13]  # Cut out position 0 to 13 from
           textstring (tex 615365-134524).
        polygonlist = {}
        for bufferdist in list_bufferdist:
            infotext = """\nSjoid: {}, Bufferdist: {}, Inraster
               name: {}""".format(sjoid, bufferdist, inraster)
            print(infotext)
            bufferpoly_w_polygon(label, sjoid, bufferdist,
               inraster, polygonlist, outdir)

    deleteallfilesindir(outdir)


    # TIMER
    clocktiming = showtimeinfo(time.clock())
    print (clocktiming)
    with open("Timer.txt", "a") as f:
        f.write(clocktiming)

    print("The script finished")

#def bufferpoly_w_polygon(label, lakeid, bufrdst, polygonlist,
    outdir):
 def bufferpoly_w_polygon(label, lakeid, bufrdst, inraster,
    polygonlist, outdir):
     global keyInAttrTable
     workdir = os.getcwd()
```

```
82
    indata = """{}/indata_shp/""".format(workdir) # This is
       where the shapefile from SMHI is
84  make_dirifnotexist(indata)

86  outdir2 = """{}/polygonzones/""".format(workdir)
    make_dirifnotexist(outdir2)
88
    #  Get a list of shapefiles (but for now it is only 1 item
       in that list)
90  shapelist = get_listoffiles(indata, "shp", spliton=".",
       isfullpath=True)  # Gets a list of all shapes in indata-
       dir

92  #  Get shapefilename without the searchpath
    # I only use 1 shape here (having many polygons),
                              #   so the list of shapes contains
       only 1 element
94  #   (shapelist[0])
    shapetoclipwith = get_newshallowfilename(shapelist[0])
96
    # Folder where the maps to clip from is placed (
       smhi_vattenforekomster_vattenytor_SVAR2012_2.shp)
98  aro_y_shp = indata + shapetoclipwith

100 #  A function that dives into the shapefile and finds all
       polygons that will be used to clip with
    polygons = getinfopolygons(aro_y_shp, keyInAttrTable)
102

104 # Loop over the ploygons
    nr_inprocessing = 0
106 totpolys = len(polygons)
    for poly in polygons:
108     nr_inprocessing += 1
        name = poly[keyInAttrTable]
110     if lakeid == name:
            print("""Poly {nr}/{totnr}. Trying to create polygon
                from {inraster} with {colname}""".format(
112                     nr=nr_inprocessing, totnr=totpolys,
                           inraster=inraster, colname=name))

114         in_features = poly["feat"]  # OK
            #out_feature_class = tmpdir + "polybuffered"  # OK
116         buffer_distance_or_field = bufrdst  # in meter
            newname = name[0:6] + "_" + name[7:13]
118         out_feature_class = """{}/poly{}_{}""".format(outdir
               , newname, buffer_distance_or_field)
            line_side = "FULL"  # OK, FULL/LEFT/RIGHT/
               OUTSIDE_ONLY
120         line_end_type = "ROUND"  # OK, FLAT/ROUND
```

```
            dissolve_option = "ALL"  # OK, NONE/ALL/LIST
            dissolve_field = None    # OK
            print("arcpy.Buffer_analysis(in_features,
               out_feature_class, buffer_distance_or_field)")
            arcpy.Buffer_analysis(in_features, out_feature_class
               , buffer_distance_or_field, line_side,
               line_end_type,
                                                dissolve_option,
                                                   dissolve_field)
            in_feat2 = out_feature_class + ".shp"
            bfdstint = int(buffer_distance_or_field)
            polygonlist["0"] = poly["feat"]
            if bfdstint == int(10):
                polygonlist["10"] = in_feat2

            if bfdstint == int(20):
                polygonlist["20"] = in_feat2

            if bfdstint == int(30):
                polygonlist["30"] = in_feat2

            final_out = """{}/poly{}_{}.shp""".format(outdir2,
               newname, buffer_distance_or_field)
            erase_w_polygon(final_out, in_feat2,
               buffer_distance_or_field, outdir2, outdir,
               polygonlist)

def erase_w_polygon(final_out, in_feat2, bufrdst, outdir2,
   outdir,  polygonlist):
    #function that loads buffered polygons and erases others,
       creating "bands" of polygons

    #create distances in integer form
    bufrdstint = int(bufrdst)
    print(bufrdstint)
    bfr1 = int(10)
    bfr2 = int(20)
    bfr3 = int(30)

    if bufrdstint == bfr1:
        #Erase lake from 10 m bufferpolygon
        print("Erase lakepolygon")
        arcpy.Erase_analysis(polygonlist["10"], polygonlist["0"
           ], final_out)

    if bufrdstint == bfr2:
        #Erase 10 m polygon from 20 m bufferpolygon
       print("Erase 10 m polygon")
       arcpy.Erase_analysis(polygonlist["20"], polygonlist["10"
          ], final_out)
```

```
    if bufrdstint == bfr3:
        #Erase 20 m polygon from 30 m bufferpolygon
        print("Erase 20 m polygon")
        arcpy.Erase_analysis(polygonlist["30"], polygonlist["20"
            ], final_out)


def showtimeinfo(t):
    s = int(t)
    m = int(s/60.0)
    h = int(m/60.0)
    sek = s - m*60
    minu = m - h*60
    timeinfo = "\nTime passed so far hour:min:sek = " + str(h).
        zfill(2) + ":" + str(minu).zfill(2) + ":" + str(sek).
        zfill(2)+"\n"
    return timeinfo

def deleteallfilesindir(folder):
    """
    http://stackoverflow.com/questions/185936/delete-folder-
        contents-in-python
    """
    import os
    # Delete all files in dir
    for the_file in os.listdir(folder):
        file_path = os.path.join(folder, the_file)
        try:
            if os.path.isfile(file_path):
                os.unlink(file_path)
                # Uncomment below if also the dirs should be
                    removed
                # elif os.path.isdir(file_path): shutil.rmtree(
                    file_path)
        except Exception as e:
            print (e)


def get_newshallowfilename(filnamn, stringtoaddbefore="",
    stringtoaddafter=""):
    """
        indata: a path to file, and two strings to add. e.g "c:\
            H\og.txt", "pre_", "_post"
        outdata:  a path to file with strings added. e.g. "
            pre_og_post.txt"
    """
    import os
    # Example: filnamn = c:/H/SLU/calc.xlsx
    filtomext, extension = os.path.splitext(filnamn)  # example:
        c:/H/SLU/calc, xlsx
```

```
202     justfilnoext = os.path.basename(filtomext)  # example: calc
        nyttnamn = """{}{}{}{}""".format(stringtoaddbefore,
           justfilnoext, stringtoaddafter, extension)
204     return nyttnamn


206
  def make_dirifnotexist(dirname):
208     import os
        if not os.path.isdir(dirname):
210         os.mkdir(dirname)
            print "made a new dir : ", dirname
212


214 def showtimeinfo(t):
        s = int(t)
216     m = int(s/60.0)
        h = int(m/60.0)
218     sek = s - m*60
        minu = m - h*60
220     timeinfo = "\nTime passed so far hour:min:sek = " + str(h).
           zfill(2) + ":" + str(minu).zfill(2) + ":" + str(sek).
           zfill(2)+"\n"
        return timeinfo
222


224 def getinfopolygons(shapefile, keyInAttrTable):
        """
226     This function iterates over each polygon in the shapefile
           and pick shape extent and table content.
        Headers in attribute table (we will only pick a few of these
           ):
228         VYID,YTKOD,VYNAMN,SJOID,SJONAMN,EU_CD,MS_CD,VF,VDRID,
               DISTRICT,COMP_AUTH,COUNTRY,VYHOJD,VERSION,Shape_Leng,
            Shape_Area
230
        The function returns a list ([]) of dicts ({}) describing
           the polygons. Example:
232         [{"boundaries": "232...743", "objectid": "655456-898899"
               }, {"boundaries": "456...644", "objectid": "
               7854687-879876"}, ...]

234     """
        shapeName = arcpy.Describe(shapefile).shapeFieldName
236     rows = arcpy.SearchCursor(shapefile)
        lst_of_dicts_polygons = []
238     for row in rows:
            feat = row.getValue(shapeName)  # Value in the attribute
                table of the shape
240         extent = feat.extent
            geom = row.Shape
```

```
242         namn = row.getValue(keyInAttrTable)  # Value in the
               attribute table of the shape
            boundaries = """{}_{}_{}_{}""".format(extent.XMin,
               extent.YMin, extent.XMax, extent.YMax)
244         objectid = row.getValue("VYID")  # Uniqe value in the
               attribute table of the shape
            dict_poly = {keyInAttrTable: namn, "boundaries":
               boundaries, "objectid": objectid, "geom": geom, "feat
               ": feat}
246         lst_of_dicts_polygons.append(dict_poly)
        return lst_of_dicts_polygons
248

250 def get_listoffiles(thepath, extension, spliton=".", isfullpath=
        True):
        """
252         Takes the path and extension (string).
            Lists all files that match extension in that path.
254         Returns a list of file paths.
        """
256     import os
        workSpace = thepath
258     ResultFiles_list = []
        for fil in os.listdir(workSpace):
260         if isfullpath:
                fil = os.path.join(thepath, fil)
262
            try:
264             base, ext = fil.split(spliton)
                if ext == extension:
266                 ResultFiles_list.append(fil)
                elif extension == "allafiler":
268                 ResultFiles_list.append(fil)
            except:
270             pass
        return ResultFiles_list
272
    if __name__ == "__main__":
274     main()
```

A.1.4 Skript för markanvändning

Steg 1

Skapa först bufferpolygoner på avstånden 100 m, 200 m samt 500 m på samma sätt som under avsnitt A.1.2, steg 5, fast ta endast bort själva sjöpolygonen ur varje bufferområde. Använd PLC6-kartan och verktyget textitintersect med de buffrade polygonerna för att erhålla tre zoner med markanvändning kring varje sjö.

Steg 2

```
# -*- coding: utf-8 -*-

import datetime
import os
import arcpy
import shutil
from arcpy.sa import *
# Check out the ArcGIS Spatial Analyst extension license
arcpy.CheckOutExtension("Spatial")

# this script loads buffered polygons (one for each sjoid) and
    intersect them with the right area in PLC6-map.
# result is a polygon shapefile for each lake that contains
    landuse

# GLOBALS
keyInAttrTable = "SJOID"  # This should be a unique identifier
    found in the attribute table.

def main():
    """
    The script starts here after importing libraries, and
       reading Globals above.
    """
    import time

    workdir = os.getcwd()
    indata1 = """{}/bufferedpolygons/""".format(workdir)
    polyshapelist = get_listoffiles(indata1, "shp", spliton=".",
        isfullpath=False)

    label = "mark"
    # Start timer
    time.clock()

    # For each lake make the intersect in the "
       intersect_w_polygon"-function
    counter=0
    for shape in polyshapelist:
        counter=counter+1
        sjoid = shape[4:10] + "_" + shape[11:21]
        print("Try intersect between bufferzone and markanv for:
            ", sjoid, " counter: ", counter)
        intersect_w_polygon(label, sjoid, indata1 + shape)


    # TIMER
    clocktiming = showtimeinfo(time.clock())
    print (clocktiming)
    with open("Timer.txt", "a") as f:
        f.write(clocktiming)
```

```
46      print("The script finished")

48
 def intersect_w_polygon(label, lakeid, shape):
50      global keyInAttrTable
        workdir = os.getcwd()
52
        arcpy.env.workspace = "//wms.vatten.slu.se/SMED/0_Ny
           mappstruktur/3_PLC6_Slutgiltiga_Produkter/PLC6_karta/
           PLC6_karta/PLC6_karta.gdb"
54      indata = "PLC6_karta" # This is where the shapefile with
           land use is

56      outdir = """{}/markshapes/""".format(workdir, label)

58      in_features = [indata, shape]  # OK
        outname = lakeid + "_" + label
60      out_feature = """{}/{}""".format(outdir, outname)
        arcpy.Intersect_analysis(in_features, out_feature, "ALL", ""
           , "INPUT")
62

64 def showtimeinfo(t):
        s = int(t)
66      m = int(s/60.0)
        h = int(m/60.0)
68      sek = s - m*60
        minu = m - h*60
70      timeinfo = "\nTime passed so far hour:min:sek = " + str(h).
           zfill(2) + ":" + str(minu).zfill(2) + ":" + str(sek).
           zfill(2)+"\n"
        return timeinfo
72

74 def get_listoffiles(thepath, extension, spliton=".", isfullpath=
       True):
        """
76          Takes the path and extension (string).
            Lists all files that match extension in that path.
78          Returns a list of file paths.
        """
80      import os
        workSpace = thepath
82      ResultFiles_list = []
        for fil in os.listdir(workSpace):
84          if isfullpath:
                fil = os.path.join(thepath, fil)
86
            try:
88              base, ext = fil.split(spliton)
```

```
            if ext == extension:
90                ResultFiles_list.append(fil)
            elif extension == "allafiler":
92                ResultFiles_list.append(fil)
        except:
94            pass
    return ResultFiles_list
96
if __name__ == "__main__":
98    main()
```

Steg 3

Ladda in skapade markanvändningspolygoner och beräkna hur stor andel av varje markanvändningstyp de olika zonerna innehåller. Skriv över till csv-fil.

```
# -*- coding: utf-8 -*-
2
import datetime
4import os
import arcpy
6import shutil
import numpy as np
8from arcpy.sa import *
import pandas as pd
10import csv
#from arcpy import env
12# Check out the ArcGIS Spatial Analyst extension license
arcpy.CheckOutExtension("Spatial")
14
# this script loads shapepolygons containing landuse, extracts
    the different landareas
16# from the attributetable and calculates how much of each
    landtype there is
# the result is written to a csv file
18
# GLOBALS
20keyInAttrTable = "SJOID"  # This should be a unique identifier
    found in the attribute table.

22def main():
    """
24    The script starts here after importing libraries, and
        reading Globals above.
    """
26    import time

28    workdir = os.getcwd()
    indata1 = """{}/markshapes/""".format(workdir)
30    shapelist = get_listoffiles(indata1, "shp", spliton=".",
        isfullpath=False)
```

```
    label = "mark"
    # Start timer
    time.clock()

    csvfile = "Markanv_3widht.csv"
    with open(csvfile, "w") as output:
        writer = csv.writer(output, lineterminator='\n')
        headers = ["buffer_dist", "sjoid", "area jordbruk", "
           andel jordbruk","area tatort", "andel tatort", "area
           skog",
                   "andel skog", "area hygge", "andel hygge", "
                      area fjall", "andel fjall",
                   "area myr", "andel myr", "area oppen mark", "
                      andel oppen mark", "total area", "total
                      area ex. vatten"]
        writer.writerow(headers)

    attr = []
    counter=0
    for shape in shapelist:
        counter=counter+1
        print(shape, counter)
        buffdist = shape[14:17]
        with open(csvfile, "a") as output:
            writer = csv.writer(output, lineterminator='\n')
            #extract all areas
            attribut1 = "Shape_area"
            attribut2 = "PLC6"
            arealista = getinfopolygons(indata1 + shape,
               attribut1)
            attrlista = getinfopolygons(indata1 + shape,
               attribut2)
            #put both attributes in a dataframe
            df = pd.DataFrame({"PLC6": attrlista, "Shape_area":
               arealista})
            #print(df)
            #sum all areas for the different attributes in PLC6
               column
            jordba = df[df["PLC6"] == 9].sum()["Shape_area"]
            tatorta = df[df["PLC6"] == 2].sum()["Shape_area"]
            skoga = df[df["PLC6"] == 3].sum()["Shape_area"]
            hyggea = df[df["PLC6"] == 10].sum()["Shape_area"]
            fjalla = df[df["PLC6"] == 5].sum()["Shape_area"]
            myra = df[df["PLC6"] == 8].sum()["Shape_area"]
            oppenmarka = df[df["PLC6"] == 4].sum()["Shape_area"]
            hava = df[df["PLC6"] == 7].sum()["Shape_area"]
            vattena = df[df["PLC6"] == 6].sum()["Shape_area"]
            vattentot = hava + vattena
            #calculate total area for each shape
            tota = df.sum()["Shape_area"]
            #tota = sum(arealista)
```

```
74              totaexv = tota - vattentot

76              #calculate how much there is of each attribute
                   compared to the total area
                andelj = jordba / totaexv
78              andelt = tatorta / totaexv
                andels = skoga / totaexv
80              andelh = hyggea / totaexv
                andelf = fjalla / totaexv
82              andelm = myra / totaexv
                andelo = oppenmarka / totaexv
84              sjoid = shape[0:6] + "-" + shape[7:13]

86              attr = [buffdist, sjoid, jordba, andelj, tatorta,
                   andelt, skoga, andels, hyggea, andelh, fjalla,
                   andelf, myra,
                        andelm, oppenmarka, andelo, tota, totaexv]
88              #write to csv
                writer.writerow(attr)
90
    # TIMER
92  clocktiming = showtimeinfo(time.clock())
    print (clocktiming)
94  with open("Timer.txt", "a") as f:
        f.write(clocktiming)
96  print("The script finished")

98 def showtimeinfo(t):
    s = int(t)
100 m = int(s/60.0)
    h = int(m/60.0)
102 sek = s - m*60
    minu = m - h*60
104 timeinfo = "\nTime passed so far hour:min:sek = " + str(h).
       zfill(2) + ":" + str(minu).zfill(2) + ":" + str(sek).
       zfill(2)+"\n"
    return timeinfo
106
  def get_listoffiles(thepath, extension, spliton=".", isfullpath=
     True):
108 """
        Takes the path and extension (string).
110     Lists all files that match extension in that path.
        Returns a list of file paths.
112 """
    import os
114 workSpace = thepath
    ResultFiles_list = []
116 for fil in os.listdir(workSpace):
        if isfullpath:
118         fil = os.path.join(thepath, fil)
```

```
        try:
            base, ext = fil.split(spliton)
            if ext == extension:
                ResultFiles_list.append(fil)
            elif extension == "allafiler":
                ResultFiles_list.append(fil)
        except:
            pass
    return ResultFiles_list

if __name__ == "__main__":
    main()
```

A.2 SKRIPT FÖR STATISTISKA ANALYSER

Skript för PCA i R

```
#set working directory
setwd("C:/Users/sara/Desktop/statistik_exjobb/stats_to_use/")
#load dataset
stat <- read.csv("merged_stat_volym_uarea.csv")
#remove the first column that contains sjoid,
#all columns must be numeric
nystat <- stat[,2:95]
#perform PCA analysis on the dataset with prcomp
# scale.= TRUE scales the data to unit variance
# mean centering is default
pcares <- prcomp(nystat, scale.=TRUE)
#get the loadings, in rotation
rot <- pcares$rotation
#transform rot from numerical matrix to dataframe
#to use variable names
rot_d <- as.data.frame(rot)
#plot the loadings
plot(rot)
text(rot, labels=row.names(rot_d), pos=4)
#calculate std and variance
std <- pcares$sdev
variance <- std^2
prop_var <- variance/sum(variance)
scores <- pcares$x
#make a barplot of explained variance,
#to see how many PCs that are necessary
barplot(prop_var, xlab="Principal component",
ylab = "Prop. variance explained")
biplot(pcares, scale=0)
plot(prop_var, xlab = "Principal component",
ylab ="Proportion of Variance explained", type = "b")
```

```
write.csv(pcares$rotation, file="loadings_volym_uarea.csv")
write.csv(pcares$x, file="scores_volym_uarea.csv")
```

Skript för PLS i R

```
#######################
#Load package and read data
#############################
#install.packages("pls")

#load training and validation sets
library(pls)
setwd( wd <- "C:/Users/sara/Desktop/statistik_exjobb/
stats_to_use/PLS/" )
lakes_training <- read.csv("merged_stat_meddjup_uarea_train.csv")
lakes_valid <- read.csv("merged_stat_meddjup_uarea_valid.csv")

#divide in predictor- and responsevariables
train<-as.matrix(lakes_training[,3:96])
resp<-lakes_training[,2]
valid <- as.matrix(lakes_valid[,3:96])
respv <- lakes_valid[,2]

#make the pls-regression and standardize the data,
#centering and scaling
plsfit <- plsr(resp~stdize(train),
data=lakes_training[,2:96], validation="CV")
plspred <- predict(plsfit, newdata=lakes_valid[,2:96],
type="response")

summary(plsfit)
#plot RMSEP
validationplot(plsfit, val.type="RMSEP")
pls.RMSEP <- RMSEP(plsfit, estimate="CV")
plot(pls.RMSEP, main="RMSEP PLS", xlab="Komponenter")
min <- which.min(pls.RMSEP$val)
points(min, min(pls.RMSEP$val), pch=1, col="red", cex=1.5)
min

plot(plsfit, ncomp=24, asp=1, line=TRUE)

#new prediction with ncomps from lowest RMSEP
plspred2 <- predict(plsfit, lakes_training[,2:96], ncomp=24)
plot(resp, plspred2)

#compute R2 och Q2
R2(plsfit, estimate="train")
#Q2
```

```
R2(plsfit)

plot(loading.weights(plsfit))

#plot correlation
labelnames <- colnames(lakes_training[2:96])
corrplot(plsfit, comps=1:2, labels=labelnames)
```