IT-verktyg för asfaltunderhållsplanering

Utveckling och test av prognosverktyg för spårdjup; för vidare integration i stödsystem för underhållsplanering

Peter Andrén
Björn Backgård

Linköping 2021

SBUF utvecklingsprojekt nummer 13483

FÖRORD

Denna rapport redovisar genomförandet och resultat av projektet *IT-verktyg för asfaltunderhållsplanering*. Projektet har delfinansierats av Svenska Byggbranschens utvecklingsfond, (SBUF) och genomförts som ett samarbete mellan VTI, Sweco, Trafikverket och NCC. Arbetsgruppen har bestått av Peter Andrén (VTI), Anders Lenngren (Sweco) och Olle Bergström (Trafikverket). Huvudförfattare till denna rapport har varit Björn Backgård, IT-konsult, och Peter Andrén på VTI.

Utvecklingsarbetet har till stor del genomförts av VTI:s samt Trafikverkets IT-konsulter. Studier, testning har huvudsakligen genomförts av NCC och Sweco.

Medverkande i projektets styrgrupp har varit Jonas Herbertsson, Sweco, Anders Gudmarsson, Peab, Roger Nilsson, Skanska, och Robert Lundström, NCC.

SAMMANFATTNING

Trafikverket upphandlar numera ofta vägar på totalentreprenad med funktionskrav där en huvudsaklig kravställning görs på tvärgående jämnhet (spårdjup). Denna kontraktstyp kan innebära relativt stora risker för entreprenörer eftersom man redan i anbudsskedet behöver kunna bedöma kostnader för felavhjälpande och underhållskostnader senare under garantitiden. För att kunna bedöma framtida underhållskostnader vid denna typ av kravställning behövs modeller som kan prognostisera spårdjupsutvecklingen. Sådan prognostisering kan antingen ske på olika sätt, t.ex. på övergripande nivå, inte sällan med manuell beräkning i populära kalkylprogram (t.ex. excel), baserat på olika nyckeltal från en eller flera vägar. Ett annat sätt, och som detta projekt utgår från är att på relativt detaljerad nivå utnyttja succesiva data från den aktuella entreprenaden, som skall prognostiseras, vilket kräver årlig data av hög kvalitet, d.v.s. data som filtrerats från förmodade felaktigheter och synkroniserats såväl geografiskt som kronologiskt. Detta projekt har syftat till att utveckla och utvärdera ett sådant verktyg.

Denna rapport går igenom utvecklingsprocessen av verktyget och hur man använder det samt förklarar dess uppbyggnad för de huvudsakliga delarna tillsammans med exempelkod i syfte att möjliggöra för aktörer som önskar implementera liknande varianter.

Utvecklingsarbetet har utförts i tre etapper.

1. Utformning och implementering av nytt utbytesformat från PMSv3 för svensk vägdata.
2. En webbaserad applikation har skapats där användaren kan lägga till enskilda entreprenader kopplade till kartor med Sveriges vägnät. Data från Trafikverkets utbytesformat samt data från användarens egna vägytemätningar indexeras och blir lättillgängligt för export.
3. Funktioner har skapats för att filtrera bort förmodad *felaktigt* data. Den kvarvarande således korrekta data tvättas och synkroniseras i längdled för att bland annat möjliggöra prognostisering av jämnhetsparametrar på 20 metersnivå. Det synkroniserade data kan sedan exporteras till ett Excelbaserat program där underhållsplaneringen sker.

Det framtagna verktyget har under utvecklingsprojektet även testats på sju totalentreprenader. Resultaten från studien indikerar att det med verktygets prognosmodell kan vara möjligt att bättre prediktera tvärgående ojämnhet (”spårdjup”) över tid för att bedöma framtida underhållsbehov jämfört med andra arbetssätt där man manuellt prognostiserar spårdjupsutvecklingen eller med den enklare form av prognostisering som kan göras på 100 m-nivå i Trafikverkets webbtjänst PMSv3. Utvärdering och verifiering av nyttor bör göras i fortsatt arbete.

INNEHÅLL

1. BAKGRUND

Trafikverkets tidigare uttalade mål att cirka 50 procent av alla större projekt skulle upphandlas som totalentreprenader entreprenader år 2018 har inneburit ett arbetssätt som konsulter och entreprenörer inte helt acklimatiserats till. Totalentreprenader innebär att entreprenörer upphandlas av Trafikverket för att både projektera och bygga en väg samtidigt som de sedan under en relativt lång garantitid (8-20 år) ansvarar för att funktionen klarar ställda krav. Sådan kravställning uttrycks normalt baserat på längs- och tvärgående (spårdjup) jämnhet, vilka utvärderas för varje längsgående 20-m sträcka. Behovet som gett upphov till detta projekt grundar sig i att planeringen av beläggningsåtgärder på vägar kan vara en stor utmaning då funktionen hos vägen utvecklas succesivt i och med att trafik- och miljörelaterade faktorer orsakar nedbrytning med tiden. Hos konsulter och entreprenörer finns idag behov av verktyg för att underlätta underhållsplaneringen i och med att allt fler upphandlingar skett på totalentreprenad och därför behovet att bedöma omfattningen av underhållet, lämpliga åtgärder och maskiner samt administration av analyser och redovisning.

UNDERHÅLLSPLANERING OCH UTVECKLINGSBEHOV

Programvara för planering av beläggningsunderhåll ingår i en bredare kategori som på engelska kallas för Pavement management system eller förkortat PMS. Enligt (AASHTO, 2012) definieras PMS som en samling verktyg eller metoder som assisterar beslutsfattare att finna optimerade strategier som ger, evaluerar och bibehåller en god vägstandard.

Intressenter som stater, kommuner och entreprenörer har idag använt PMS i årtionden. Så tidigt som 1987, men troligen ännu tidigare, har PMS använts till att dokumentera vägstatus, planera beläggningsunderhåll samt att uppskatta framtida kostnader (jämför Peterson 1987). Tidiga PMS använde sig av manuella metoder där man okulärt bedömde vägens tillstånd, antingen genom anteckningar fysiskt ute på plats eller att någon körde längs vägen och fotograferade för att i efterhand med hjälp av dessa foton ge vägens olika tillståndsmått en semi-kvantitativ poängbedömning, t.ex. från 1 till 10. Denna typ av bedömning används fortfarande i stor utsträckning även i Sverige. Litteratur indikerar dock en succesiv övergång från okulär bedömning till alltmer automatiserade metoder för såväl datainsamling som efterbearbetning av denna data i syfte att höja datakvaliteten (McGhee 2004; Flintsch and McGhee 2009). Nyttan av dessa system beror dock till stor del på hur realistiska och sofistikerade de verktyg och algoritmer är som bearbetar den insamlade data.

I dagsläget förmodas en vanlig metod för att planera beläggningsunderhållet för en given väg, utöver bedömningar i fält, vara att analysera data för spårdjup som finns tillgänglig via Trafikverkets webbtjänst PMSv3. Även om andra nedbrytningsmekanismer kan orsaka underhållsbehov utgör högt spårdjup en av de viktigaste och för vilket en succesiv förändring skulle kunna vara möjlig att prediktera. Årets data kan sedan jämföras med

tidigare års mätdata så att spårdjupsutvecklingen kan uppskattas. Att prognostisera framtida spårdjup på 20 meters nivå kan vara komplicerat och tidskrävande och resulterar inte sällan i opålitliga resultat. Detta beror delvis på att synkroniseringen mellan olika tidsserier för spårdjupsdata PMSv3 är begränsad. Många av de utmaningar som uppstår vid inläsning och automatiserad behandling av data från PMSv3 skulle exempelvis kunna mildras om fler parametrar från Trafikverkets årliga vägnätsmätningar inrapporterades i 1 m-intervall i stället för dagens 20 m-intervall. Detta har resulterat i att entreprenörer som vill skapa spårdjupsprognoser sannolikt använder sig av mer övergripande schablonberäkningar för spårdjupsutveckling för hela vägsträckan.

Det förefaller i dag finnas ett behov att på ett effektivare sätt prognostisera vägytans jämnhet i tvärled (spårdjup) med högre noggrannhet och kvalitet, och sedan föra detta resultat vidare som ett underlag till en framtida underhållsplan. Det bör utgöras av ett robust verktyg, eftersom ingående modeller och programmeringsspråk samt data från enskilda vägytemätningar såväl som data från PMSv3 bör kunna integreras, som kan prognostisera spårdjup med högre tillförlitlighet än mer manuella schablonmässiga metoder. Algoritmer som hittar homogena underhållssträckor med hjälp av ett glidande fönster kan med fördel skrivas i språket VBA för att tillåta alla som har Excel att utföra underhållsplaneringen utan att behöva installera ytterligare mjukvara på sin egen dator. Excel har dock en del begränsningar i prestanda och valmöjligheter hur algoritmerna kan implementeras jämfört med ett konventionellt programmeringsspråk, men den typ av algoritm som behövs i detta projekt är tidigare testad med gott resultat (se t.ex. Hajdin, 2014) och fördelarna man får av att manuellt kunna manipulera underhållet för enskilda vägsektioner direkt i kalkylarket överväger nackdelarna med de begränsningar man stöter på när man utvecklar och använder program skrivna i programspråket VBA. Resultatet från denna prognos bör kunna exporteras i Excelformat för vidare behandling och planering till berörda branschaktörer. Därefter kan prognostiserade data exporteras så att man på körfältsnivå kan planera underhållsåtgärderna under hela kontraktets garantitid.

Kontraktens komplexitet ger upphov till skillnader hur vi i Sverige planerar vår resursallokering för beläggningsunderhåll kontra andra länder. Den begränsade litteratur som finns inom området pekar på att automatiserade verktyg för underhållsplanering i de flesta länder ofta är avsedda för att underlätta för väghållaren att optimera fördelningen av underhåll på ett vägnät givet en fast budget (France-Mensah, O'Brien, 2018). Detta skiljer det sig något från hur vi till stor del arbetar i Sverige, där Trafikverket såväl som entreprenörer ofta startar utan fastställd budget. Om vi uttrycker oss förenklat, gäller det i stället att uppnå godkänd vägstandard på en väg eller ett vägnät till lägsta möjliga kostnad. Avvikelser från denna förenkling uppstår i regel alltid, exempelvis för kontrakt som omfattar bonus/vite för olika aspekter för vägens funktion. Där uppstår ofta en mer komplex relation mellan viten kopplade till funktionskraven kontra kostnaden för underhållet som krävs för att undvika dessa viten. Detta kompliceras ytterligare då det snarare är regel än undantag att krav och utvärdering av mätresultat ofta lämnar utrymme för tolkning samt huruvida ej uppfyllda funktionskrav kan förklaras av faktorer som ligger

bortom entreprenörens kontroll och därför kan undantas från åtgärd. Detta omöjliggör i praktiken en modell som kan skapa en underhållsplan från funktionstidens start till slut utan att avvikelser från verkligheten och den modellerade planen kommer uppstå. Detta innebär dock inte att en modell kan utvecklas som avsevärt underlättar och förbättrar underhållsplanering.

Även om det finns väsentliga skillnader vad gäller underhållsplanering så finns modeller och arbetssätt från andra länder som använder sig av liknande arbetssätt som vi tänker oss för totalentreprenader i Sverige, d.v.s. där vi söker en optimal strategi för underhåll av beläggningen för att uppnå en minsta acceptabla nivå till lägsta kostnad, (jämför AASHTO, 2012), men verktyg för att arbeta med sådan underhållsplanering i Sverige saknas.

Några av de största och mest funktionsrika programvaror som idag används runt omkring i världen är: ESRI Roads and Highways, TatukGIS, Heller-ig PMS Core, MtcPMS Streetsaver, Yotta Horizons, HDM Global HDM-4, Deighton dTIMS. Vid genomgång av dessa programvaror har slutsatsen dragits att funktionerna hos dessa programvaror skiljer för mycket för att de ska kunna anpassas till arbete med underhållsplanering av svenska vägar där fokus ligger på främst jämnhet i tvärled på 20 metersnivå.

SYFTE

Det övergripande syftet i detta utvecklingsprojekt är att utveckla en plattform, ett IT-verktyg, för planering av underhållsåtgärder, särskilt för totalentreprenader med funktionskrav. Verktyget förväntas, om och när det implementeras av intressenter, kunna öka förståelsen av och effektiviteten i underhållsplaneringen och bidra samhällsekonomiskt genom att kostnader minskar för underhåll.

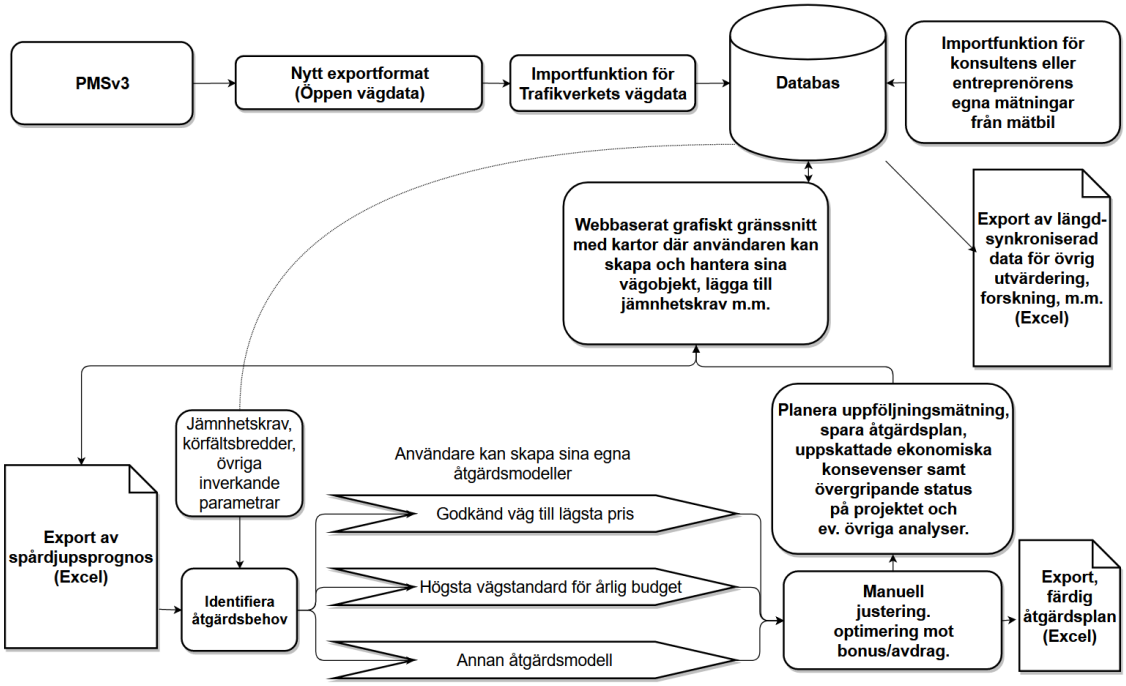
Verktyget består av flera delar varav vissa syftar till att samla in, strukturera och förmedla data. Eftersom underhållsarbeten och deras planering inte enbart beror på tekniska mätdata (jämför Zimmerman, 2017) utan t.ex. även utförarens resurser (t.ex. tillgängliga underhållsmetoder och ekonomiska förutsättningar) och tolkningar så har det förutsatts att det dels inte finns en objektiv optimal lösning för underhållsplanering och dels att användaren därför vill kunna använda data på ett flexibelt sätt, varför den slutliga analysen och resultatet erhålls i Excel-format.

Eftersom utvecklingsprojektet är begränsat till själva framtagandet verktyget, samtidigt som underhållsplanering görs av individer med givna resurser och egen know-how, har inte någon egentlig verifiering av verktyget gjorts annat än av funktionen hos själva mjukvaran exemplifiering av en handfull entreprenader.

2. GENOMFÖRANDE

I figur 1.1 illustreras schematiskt arbetsflödet och funktionen för det i denna rapport beskrivna och utvecklade verktyget. Data från såväl Trafikverkets databas med

vägytedata samt data från egna källor behöver kunna läsas in. Man ska kunna skapa vägprojekt i en grafisk miljö och sedan exportera längdsynkroniserad vägytedata till ett excelark där underhållsplaneringen sker. I detta Excelark behövs även programkod för att automatisera vissa delar av underhållsplaneringen.



Figur 1.1. Flödesschema som i grova drag beskriver programmets tänkta uppbyggnad och arbetsgång.

NYTT EXPORTFORMAT

Ett nytt exportformat för vägdata från PMSv3 har skapats av Trafikverket (se figur 1.1). Denna data innehåller information om tillståndet på statligt belagda vägar. Denna information har huvudsakligen inhämtats med mätbilar som skannat av vägytan och sedan simulerat fram de olika tillståndsparametrarna på den inskannade vägmodellen. För vissa vägar innehåller PMSv3 data för mätningar och beläggningar gjorda redan under första halvan av 1900-talet. De parametrar i denna data som används beskrivs i bilaga A. Denna bilaga innehåller även exempelkod för hur vi skapat databasen och konverterat trafikverkets data för att sedan kunna läsa in den i egen databas.

Trafikverkets data som internt hos Trafikverket lagras i en Microsoft SQL-databas, levereras i textform. Databasen består av åtta tabeller med benämningarna *Avsnitt, BelaggningsData, DelstrackaSegment, FiktivBelaggningData, MatData, MatDataSegment, ProgData samt VagData*. Antalet rader per tabell framgår av tabell 2.1

Tabell 2.1 Tabellen Avsnitt innehåller vad man kan kalla administrativ information, som län, löpande längd, vägnummer, roll och riktning, OID, etc.

Avsnitt	26,940,998
BelaggningsData	354,848
DelstrackaSegment	84,403,567
FiktivBelaggningData	83,495
MatData	22,776,005
MatDataSegment	186,614,769
ProgData	6,147,385
VagData	109,427

BelaggningsData innehåller en mängd information om beläggningen. Fälten BelaggningsDatum och BelaggningsTyp innehåller viktig information om maximal stenstorlek, beläggningens tjocklek, bindemedel etc. För äldre beläggningar saknas emellertid oftast denna information.

I tabellen DelstrackaSegment återfinns merparten av mätdata: IRI, spårdjup, MPD, tvärprofilen, etc. Data är uppdelat på så kallade termidentiteter (IRI höger heter 1287, maximalt spårdjup heter 1025 etc.). Vissa variabler i tabellen, som spårbredd och spårbottenavstånd, beräknas från tvärprofilen.

FiktivBelaggningData innehåller bara fälten BelaggningsDatum och BelaggningsTyp (som alltid är FIKT). Dessa fiktiva beläggningarna läggs in där man misstänker att en beläggningsåtgärd inte registrerats, eller med andra ord för att förklara vad som annars skulle vara spontana förbättringar av vägens tillstånd.

MatData är en slags övertabell till tabellen MatDataSegment, och innehåller egentligen ingen information.

Tabellen MatDataSegment är också en slags övertabell, denna gång till DelstrackaSegment. Här återfinns även information om "löpande längd" som används för att lokalisera delsträckor på en väg.

ProgData innehåller prognostiserade värden över 100 m (i både framtid och förfluten tid). Denna information används mest av TRV för att få ett heltäckande dataunderlag för en viss tidpunkt (oftast årsskiftet).

I tabellen VagData hittar man information om trafikmängd, vägbredd, vägtyp (motorväg etc.), hastighetsbegränsning, och så vidare.

Det har varit för omfattande att i detta projekt implementera funktioner som drar nytta av alla parametrar i Trafikverkets dataformat. Dessutom har vissa av Trafikverkets parametrar bedömts ha ingen eller begränsad nytta i detta projekt. Längdprofilerna, som Vägverket/Trafikverket beställt sedan 2001, ingår inte i detta dataformat.

Man kan ta ut data i två lägen beroende på om fältet StrackDataTyp är 1 eller 2. Med StrackDataTyp satt till 1 returneras homogena avsnitt och med 2 avses 20m-data.

För att ta ut data används lämpligen så kallade ”left joins” för att i praktiken samla all information i en stor tabell. Efter detta anger man en valfri mängd villkor för vad som

ska presenteras. Exempelkod nedan samt Tabell 2.2 illustrerar hur bl.a. IRI samt beläggningsinformation från väg nummer 140 på Gotland för beläggningsdatum 1967-01-01 och mätdatum 2010-09-03 väljs ut.

```
select
  avsnitt.LanKod as LAN,
  cast(avsnitt.VagNummer as decimal(6,2)) as VAGNR,
  matdatasegment.StartLopandeLangd as LL,
  vagdata.Hastighet as HAST,
  cast(vagdata.Vagbredd as decimal(3,1)) as VAGB,
  cast(matdata.RSTDatum as date) as MATDAT,
  cast(belaggningsdata.BelaggningsDatum as date) as BELDAT,
  cast(delstrackasegment.x1287 as decimal(10,2)) as IRI_H,
  belaggningsdata.BelaggningsTyp as BEL
  from delstrackasegment
  left join matdatasegment on delstrackasegment.DelstrackaSegmentId
    = matdatasegment.DelstrackaSegmentId
  left join matdata on matdata.MatDataId = matdatasegment.MatDataId
  left join avsnitt on avsnitt.MatDataId = matdata.MatDataId
  left join vagdata on avsnitt.VagDataId = vagdata.VagDataId
  left join belaggningsdata on belaggningsdata.BelaggningsDataId
    = avsnitt.BelaggningsDataId
  where
  avsnitt.DATF <= matdata.RSTDatum and avsnitt.DATT >=
matdata.RSTDatum
  and avsnitt.StrackDataTyp = 1
  and avsnitt.LanKod = 9
  and avsnitt.VagNummer = 140
  and avsnitt.Korfalt = 10
  and avsnitt.Riktning = 1
  and cast(matdata.RSTDatum as date) = '2010-09-03'
  and cast(belaggningsdata.BelaggningsDatum as date) = '1967-01-01'
  order by LL
  \g
```

Resultatet av exempeluttaget ovan framgår av Tabell 2.2.

Tabell 2.2 Exempel på hur ett datauttag från Trafikverkets dataformat kan se ut.

LAN	VAGNR	LL	HAST	VAGB	MATDAT	BELDAT	IRI_H	BEL
9	140.00	24906	80	6.5	2010-09-03	1967-01-01	2.29	MABT
9	140.00	24918	80	6.5	2010-09-03	1967-01-01	3.32	MABT
9	140.00	29107	80	6.5	2010-09-03	1967-01-01	1.28	MABT
9	140.00	29121	80	6.5	2010-09-03	1967-01-01	1.56	MABT
9	140.00	29141	80	6.5	2010-09-03	1967-01-01	1.18	MABT
9	140.00	29150	80	6.5	2010-09-03	1967-01-01	1.18	MABT
9	140.00	29161	80	6.5	2010-09-03	1967-01-01	1.59	MABT
9	140.00	29181	80	6.5	2010-09-03	1967-01-01	1.51	MABT
9	140.00	29201	80	6.5	2010-09-03	1967-01-01	1.41	MABT

Det bör även sägas att Trafikverkets data innehåller en stor mängd fel, vilket försvårat användandet. Trafikverkets data bör därför filtreras eller tvättas för att bli mer användbar.

Detta har gjorts för de mer kritiska parametrarna för prognostiseringsdelen av detta projekt.

INLEDANDE UTVECKLINGSARBETE WEBBAPPLIKATION

Innan verktyget började sättas samman så skrevs prototypfunktioner i Python, Pandas och Shapely i Jupyter notebooks. När dessa funktioner och datautbytet mellan dem börjat fungera tillfredställande så fanns en grundstruktur att basera hela systemets uppbyggnad på. Viss pytonkod kommunicerar med övriga delar av systemet via HTTP-anrop (så kallade microservices). Andra delar körs som subprocesser från PHP.

TEKNOLOGIER OCH UTVECKLINGSMILJÖ

Programspråk och verktyg vid utveckling av användargränssnitt och grafisk layout är Symfony backend, Angular frontend, Bootstrap 4 och CSS ramverk. Den underliggande väggeometrin som applikationen använder för positionering, vägens linjeföring, synkronisering av data från olika källor, t.ex. egen mätbil och PMSv3-data samt projicering på karta i webbapplikationen, kommer från Trafikverkets webbportal lastkajen.trafikverket.se. Datapaketet benämns Sverigepaket ”Drift Och Underhåll”. Geometrin i detta datapaket är en kopia av samma väggeometri som används i NVDB[1]. Kartmaterial är skapat med egenutvecklade tiles som använder denna väggeometri tillsammans med kartdata från OpenStreetMap. När användaren klickar ut start- och slutposition för att lägga till ett körfält i sitt vägprojekt, så sparas detta som en kedja bestående av en serie Objekt - ID[2] och löpande längd i NVDB, detta genereras med hjälp av programbiblioteket pgRouting. För att knyta mätfiler till väggeometrins koordinater utförs distansberäkningar med programbiblioteket st_dwithin och PostGIS som är ett databastillägg för geospatial data. Skript för import av data är skrivna i Bash och Python. Installation av PostgreSQL, Python och övriga programvaruberoenden har automatiserats i Docker.

ANVÄNDARGRÄNSSNITT

För att få en bra bild över vägprojekten och kopplingen de har till objektens garantitider avseende jämnhet anses ett intuitivt kartbaserat grafiskt gränssnitt som visar Sveriges vägar behövas. I denna grafiska miljö behöver användaren kunna skapa objekt[3] och lägga

[1] NVDB (Nationella vägdatabasen) är resultatet av ett regeringsuppdrag som Vägverket fick 1996. Databasen innehåller geometrin för Sveriges vägnät samt en stor mängd data kopplad till denna geometri, bland annat tillståndsmått för vägytan. Databasen används av mängd aktörer inom både privat och offentlig sektor.

[2] OID eller Objekt-id är ett av Trafikverket skapat geografiskt positionsbestämt och över tiden stabilt ID som unikt identifierar en viss vägsträcka.

[3] Med objekt menas ett av användaren definierat vägobjekt som har start/slut i koordinater eller i löpande längd på givet vägnummer. Dessa start/slut-positioner kommer i regel sammanfalla med start och slut på

till körfält till dessa. Varje körfält utgörs av en kedja av OID (Objekt – ID). All övrigt data som läggs till körfältet när användaren arbetar i applikationen kommer sedan knytas till denna geometri (kedja av OID). Multipla kedjor av OID kan grupperas i ett objekt. Varje körfält på ett vägprojekt får till exempel en egen OID-kedja då varje körfält har en egen unik linjeföring genom terrängen. När ett nytt körfält läggs till, anges startposition och slutposition på en karta likt som görs i Trafikverkets webbtjänst PMSv3 när man vill analysera en sträcka. Då detta körfält (OID-kedja) knyts till väggeometrin i NVDB får man bland annat fram löpande längd och längd på sträckan i meter. Mätdata som laddas upp av användaren behöver sedan kunna knytas till denna OID-kedja. För att möjliggöra detta beräknas serie av koordinater fram längs denna kedja. De framräknade koordinaterna i kedjan är nu redo att matchas med koordinater i användarens eget mätdata.

IMPORT AV TRAFIKVERKETS DATA

Som tredje steg i flödesschemat (se figur 1.1) importeras Trafikverkets data till en PostgreSQL relationsdatabas. Importen har automatiserats med script. En relationsdatabas är i detta fall lämplig då Trafikverkets data kommer i tabellform. Import av denna data till en relationsdatabas möjliggör snabb och parallell sökning i de olika relaterade tabellerna. Importskriptet fungerar även som en referens för oss så vi kan härleda varifrån de olika parametrarna kommer i Trafikverkets data. Då detta körs i en Dockerkontainer kan programvaran köras på en mängd olika system, och får alltid tillgång till rätt versioner av de olika programbibliotek som krävs. Detta möjliggör framtida integration med olika aktörers befintliga IT-system.

Trafikverkets korta databastabeller så som Region och FiktivBelaggningData importeras rad för rad. Först skapas en tom tabell i databasen. Data som läses in kontrolleras så den inte har saknade värden (NULL-värden). Datatypen för varje variabel sätts enligt vad som är dokumenterat i bilaga A. Sista av allt skapar vi ett index och efterbehandlar databasen för effektivare databassökning.

För längre tabeller behövs ytterligare optimeringar. Rader i Trafikverkets CSV-fil konverteras till ett binärformat lämpligt för att sedan kunna importeras via PostgreSQL COPY-funktion. Denna optimering gör att de stora tabellerna kan importeras flera gånger snabbare. Under importprocessen så inhämtas viss statistik, t.ex. hur lång tid det tog att processa 1000 rader. Denna statistik har använts för att optimera importskripten till PostgreSQL-databasen. Fsync och transaktionsloggar stängs av under importprocessen. Detta för att undvika korrupt data i det fall servern tappar ström eller kraschar under importprocessen.

av Trafikverket upphandlad vägsträcka i vägentreprenad eller upphandlad grupp som omfattas av samma kontrakt.

När en användare sedan gör en sökning på en sträcka så identifieras först vägsträckans sektioner i tabellen Avsnitt baserat på matchande OID, RelStart, RelSlut och Direction. Se bilaga A för förklaring av variabler.

IMPORT EGEN MÄTDATA

Ett viktigt delsyfte i detta utvecklingsprojekt är att man i systemet själv som entreprenör ska kunna importera eget mätdata. För att hålla omfattningen av utvecklingsarbetet nere har importen begränsats till mätdata i textformat. Data måste vara i CSV-format där första kolumnen i filen är distans, övriga kolumner innehåller de olika mätstorheter som samlats in under mätningen. En av dessa kolumner behöver vara koordinater. De flesta mätbilar som idag utför vägytemätning i Sverige, exporterar redan sin data i detta format. Alla mätdatafiler som av användaren laddats upp på servern indexeras i en databas. I denna databas sparas mätdatafilernas mätdatum, körfältsnummer, riktning, start- och slutkoordinater, längd samt en lista på vilka mätstorheter filen innehåller. För att kunna använda egna data i applikationen behöver användaren förenklat förklarat, bara ladda upp data till servern där övriga steg som passar data till vägnätet sker mer eller mindre automatiskt. När användaren sedan vill komma åt mätdata för ett valt körfält så filtreras all mätdata som är uppladdad till servern så endast relevanta data knyts till det körfält användaren arbetar med.

Den automatiserade processen som följer beskrivs härmed. Databasen som innehåller information om uppladdade mätfiler skannas. Alla filer som inte har en start eller slutposition närmare än 10 meter från körfältets OID-kedja sorteras bort. Kopplingen mellan egen koordinatsatt data och svenska vägnätets geometri är viktig att förstå då data som endast är positionsbestämd med koordinater (data från mätbilar) inte har någon direkt koppling till det referenssystem som Trafikverket använder sig av vid positionsbestämning på väg. Trafikverket använder sig av vägnummer och löpande längd från vägens start. I nationella vägdatabasen (NVDB) och för varje vägnummer anges även en kedja av OID, och det är denna kedja man måste koppla all externt data till om man vill använda sig av samma referenssystem för positionering som Trafikverket. Alla entreprenadkontrakt börjar och slutar, om man ser det ur Trafikverkets perspektiv, på en specifik position som anges med ett OID och en relativ distans inom det OID. Start och slutkoordinat för varje OID går dock plocka ut. När man vill ange en exakt position inom ett OID anger man den som en relativ position mellan starten (0) och slutet (1) på detta OID. Dvs. en sträcka som ligger en fjärdedel in från starten på ett OID får en relativ position om 0,25. Den totala längden i meter mellan starten (0) och slutet (1) på ett OID finns sparad tillsammans med detta OID i NVDB. Om man vill knyta ett uppmätt och koordinatsatt värde till en exakt position på svenska vägnätets geometri behöver det kopplas till denna relativa position på OID. Ett OID som är komplett startar på 0 och slutar på 1. Där är det förhållandevis enkelt att beräkna distanser mellan relativa positioner på OID. Alla OID i NVDB är dock inte kompletta.

När användaren skapar ett körfält och lägger det till sitt vägprojekt i webbgränssnittet så behöver bland annat den totala längden på körfältet (OID-kedjan) beräknas. När kedjans längd beräknas, summeras längden för alla OID i kedjan. Början på första och slutet på sista OID i kedjan sammanfaller dock ytterst sällan med starten och slutet på det körfältet användaren markerat ut på kartan när körfältet skapades och lades till vägprojektet. Därav behöver längder mellan relativa positioner inom ett OID också kunna beräknas. Längder till och från relativa positioner på OID som inte är kompletta och därav börjar på t.ex. relativ distans 0,465 och slutar på relativ distans 0,789 beräknas enligt följande formel:

MeterPerRelativEnhetFörOID = LängdFörOIDEnlNVDB / abs(RelativSlutDistans - RelativStartDistans).

För ytterligare öka kvaliteten på mätdatat filtreras skräpdata som eventuellt finns i början och slutet på en mätdatafil bort. Detta gör importprocessen genom kontroller som upptäcker om det finns någon utmarkerad start- eller slutposition i det inlästa mätdatat. Denna funktion identifierar i dagsläget endast start- och slutpositioner i data genererade med mätbilar från leverantören Greenwood Engineering, men kan enkelt anpassas för mätdata från andra utrustningsleverantörer.

Genom att jämföra koordinaterna för starten och slutet i mätdatafilerna med koordinaterna framräknade från OID-kedjan får vi fram den relativa position på OID-kedjan där våra mätfiler börjar och slutar. Genom att kontrollera att längden i det egna uppladdade mätdatat stämmer överens med längden mellan motsvarande start och slut i OID-kedjan kan vi anta att mätfilen hör till den väg vi arbetar med. Fortsatt filtrering sker genom en jämförelse av riktning, körfält och objektlängd mellan mätfiler och det körfält användaren skapat. Då dessa procedurer är av relativt teknisk karaktär redogörs de i mer detalj tillsammans med programkod i bilaga B.

Data utanför datumintervall angivna av användaren filtreras också bort. En ytterligare filtrering vi använt oss av vid tester av denna applikation har varit att vi filtrerat bort datafiler som har under 500 m data som matchar våra kriterier i filtret ovan. Denna gräns kan dock ändras vid behov. Slutligen får användaren en lista på de filer som automatiskt matchats till aktuellt körfält. Användaren kan då manuellt inkludera eller exkludera enskilda mätfiler genom att bocka för eller ur dem i en lista.

EXPORTERA EGEN MÄTDATA

I tidigare steg har vi lagt in vår eget mätdata och knutit den till vägnätet och sparat den i databasen. När man sedan vill göra ett uttag från denna databas på en viss sträcka så läses alla mätdata kopplad till denna vägsträcka in i en funktion som klipper bort eventuellt skräpdata i början och slutet på mätfilen. Då importfunktionen för mätdata i tidigare steg identifierat vilka filer som är riktning 2, inhämtas den informationen från databasen. Riktningsinformationen används till att avgöra om det exporterade data ska flippas runt,

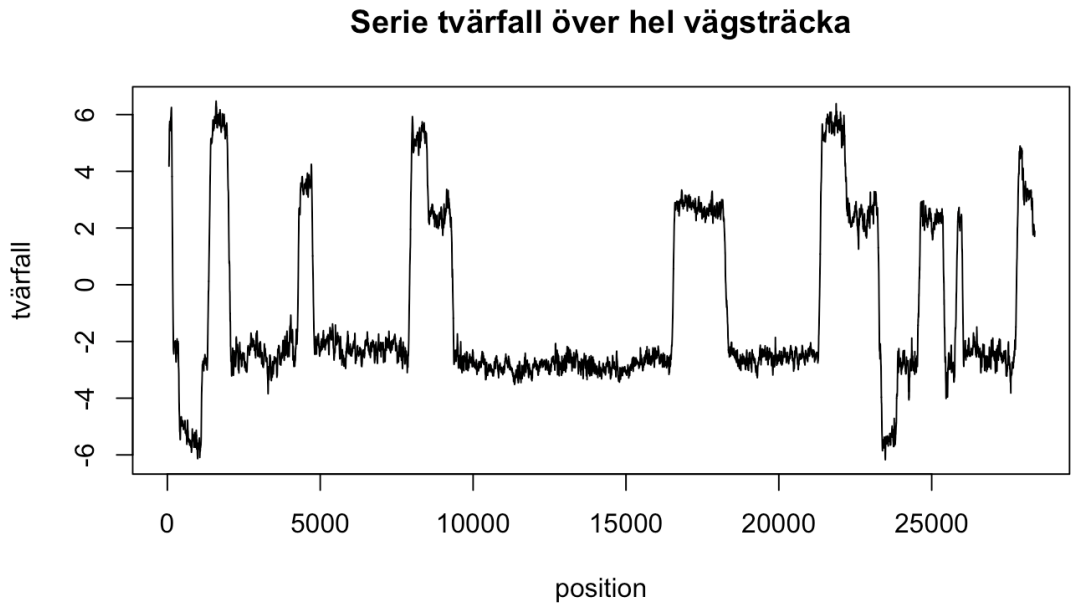
dvs redovisas i omvänd ordning. Slutligen dumpas all data i antingen CSV eller Excelformat så användaren kan ladda ner den.

EXPORTERA TRAFIKVERKETS DATA

Då all mätdata som kommer från Trafikverket redan använder OID som referens för positionering och de vägsträckor användaren skapat i webbapplikationen också gör det är exporten av Trafikdata relativt enkel. Se bilaga C för källkod. Slutligen dumpas all data i antingen CSV eller Excelformat så användaren kan ladda ner den.

SYNKRONISERING AV EGEN DATA

Olika metoder för justeringar i längdled av data har under projektets gång testats. Initialt antogs data kunna justeras i längdled med adekvat resultat genom att data delas in i 1000 m-sträckor som sedan justeras fram och tillbaka på sådant sätt att optimal korrelation mellan olika tidsserier uppnås. Resultat från dessa försök visar att denna typ av synkronisering i längdled inte varit god nog för att skapa prognoser på 20 m-nivå. Vidare försök där dessa 1000 m-sträckor stretchas och krympts för att på så sätt öka korrelationen i längdled mellan olika tidsserier har visat på tydligt ökad korrelation och bättre prognoser av 20 m-data. Men prognoser av 20 m-värden visade sig i enstaka fall fortfarande vara av ganska låg kvalitet om man jämför med prognoser av 100 meters medelvärden. Ett synkroniseringsfel på t.ex. 4 meter påverkar inte ett prognostiserat 100 meters värde nämnvärt medan det har stor påverkar på ett prognostiserat 20 meters värde. En ny metod där justeringar av mätvärden sker dynamiskt och genom förskjutning av serier relativt varandra längs hela mätseriens längd har sedan testats. En serie kan exempelvis beräknas vara felrapporterad med +5 meter relativt en referensserie under mätningens första 1000 meter, för att sedan beräknas vara felrapporterad med +6 meter under följande 400 meter, och sedan vara +7 meter fel. En förskjutning av serien justeras då dynamiskt längs sträckan i enlighet med den beräknade felrapporteringen. Denna typ av dynamisk justering har uppvisat bättre resultat än tidigare försök med mer strikta regler för justering. Denna typ av justering har även funktioner som hanterar specialfall där mätbilen vid vissa mättillfällen t.ex. kört om hinder, använt annat eller bytt till annat körfält på sätt som är avvikande från övriga mätserier. Därför bedömdes denna typ av längdjustering som mest lämplig. Figur 2.1 visar uppmätt tvärfall. Längs med en sträcka uppvisar tvärfallet dels små och till synes slumpmässiga förändringar, dels något större mer systematiska förändringar, och i vissa fall väldigt tydliga sådana. Dessa tydliga förändringar är tvärfallsövergångar där körfältets projekterade tvärfall går från positivt till negativt, eller tvärt om. Dessa tvärfallsövergångar användbara för att justera de olika dataseriernas längdmätning relativt varandra.

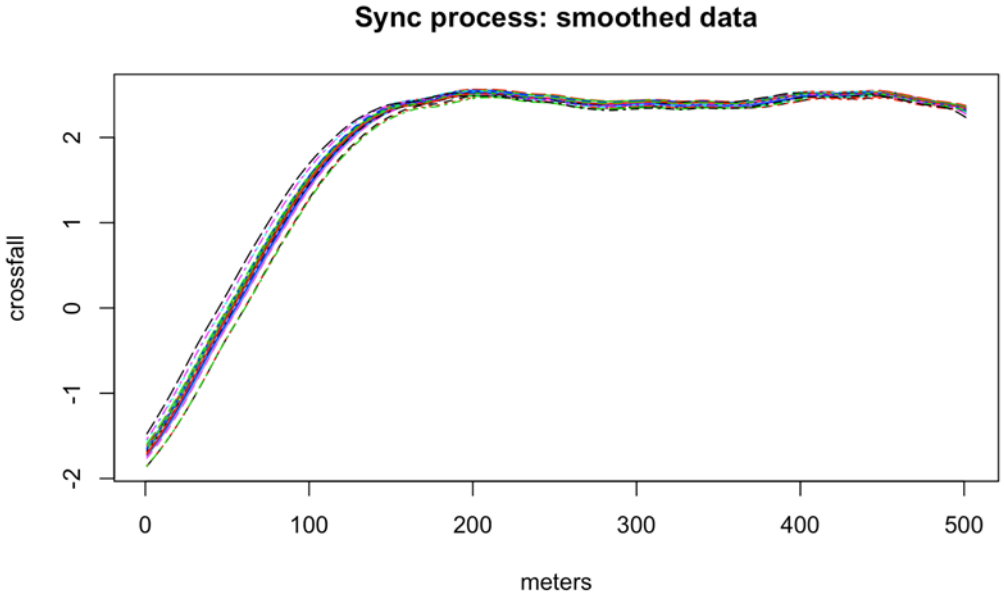


Figur 2.1. Uppmätt tvärfall över en sträcka på strax under 3 mil.

Dock är tvärfallsövergångar i många fall inte tillräckligt frekventa för att justera serierna tillräckligt väl för att datapunkterna i olika serier nära nog ska representera samma position i längdled. I grafen ovan finns exempelvis inga större förändringar mellan 10 000 och 15 000 meter och på en så lång sträcka kommer längdförskjutningar i längdmätningen vid en vägytemätning oundvikligen att uppstå. Justeringen görs genom att mäta skillnader i y-led vid en given position. Vid betraktelse av grafen ovan kan det tyckas mer naturligt att justera för skillnader i x-led. Kurvorna är dock inte kontinuerliga och saknar annat än i undantagsfall gemensamma värden i y-led. Däremot delar de värden längs x-axeln. Metoden försöker hitta den förskjutning av position som minimerar skillnader i y-led. Förskjutningar av värden i x-led ger tydliga utslag vad gäller differenser i y-led vid de distanser då tvärfallet kraftigt förändras. Figur 2.2 visar uppmätt tvärfall från flera separata vägytemätningar över en sträcka på 500 meter. Grafen i figuren är för bättre tydlighet något utslätad. De olika serierna uppvisar systematiska skillnader i y-led under de första 100 meterna. Från 100 till 500 meter växlar istället seriernas relativa skillnader i höjd.

Figur 2.2. Tvärfall på en sträcka om 500 meter, före synkronisering.

Synkroniseringen sker i flera iterationer. Först används endast de kraftiga förändringarna av tvärfallet. När serierna är bättre justerade blir mindre förändringar i tvärfallet mer informativa och kan användas för ytterligare justering. Först slätas serierna ut (Figur 2.3) så att mindre förändringar elimineras. Utslätningen görs för att de systematiska skillnaderna i höjdled då framträder tydligare. Grafen nedan visar hur den ursprungliga serien ser ut efter att så kallad *kernel smoothing* med lämpliga parametrar tillämpats.

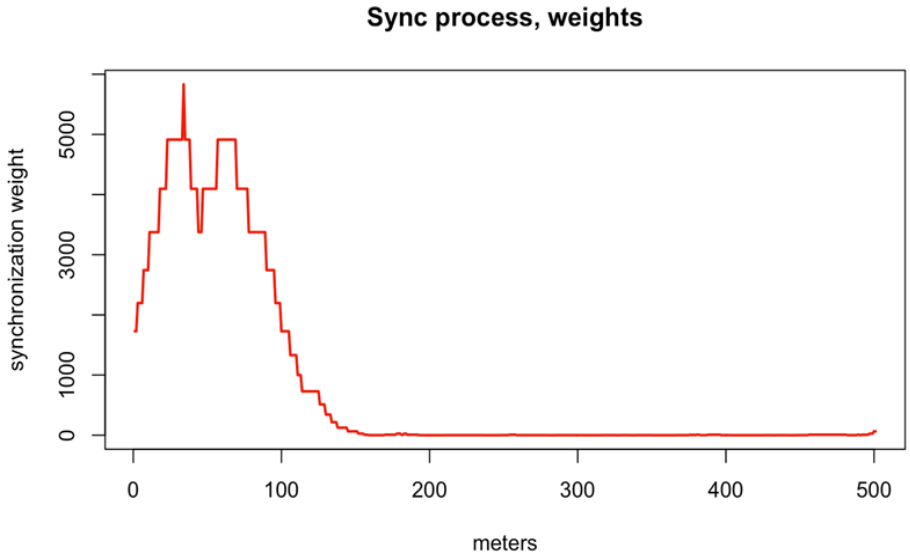


Figur 2.3. Tvärfall på en sträcka om 500 meter, utslätad.

I förekommande fall kan serier uppvisa relativt stora systematiska skillnader i y-led även där tvärfallet inte förändras på något tydligt sätt till följd av någon typ av mätfel eller

förändring som skett på vägsträckan. Användandet av vikter som är proportionerliga med förändringen av tvärfallet gör att betydelsen av skillnader i y-led som har sådana orsaker minskar.

Figur 2.4 visar de i första iterationen beräknade vikterna. I det illustrerade fallet sammanfaller vikterna ganska väl med den utslätade skillnaden i tvärfall, och förbättrar därför inte nödvändigtvis synkningen på något avgörande sätt i just detta fall.

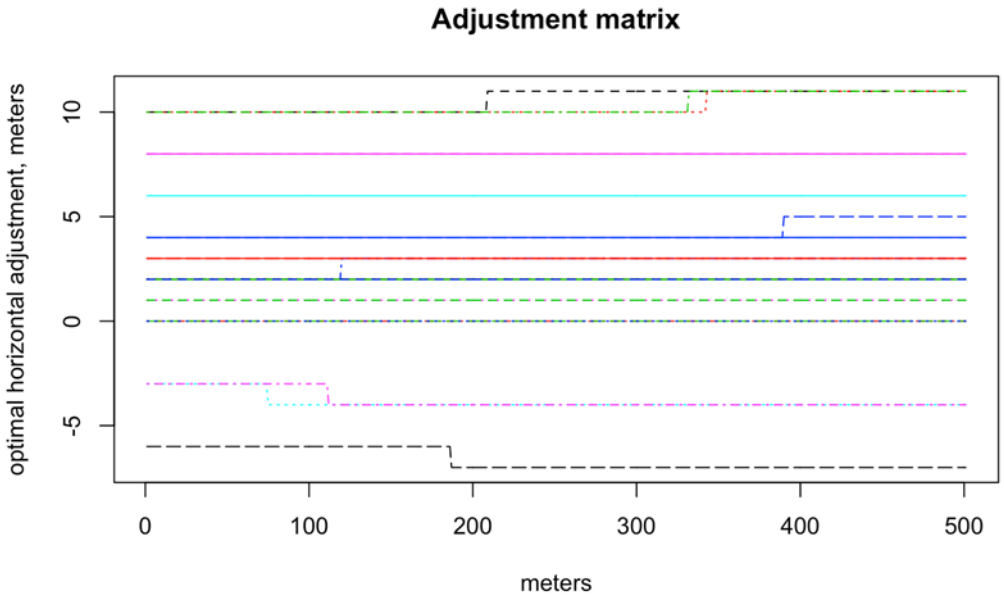


Figur 2.4. Visar hur stor vikt som ska läggas vid de olika tvärfallsförändringarna längs sträckan när data senare synkroniseras med hjälp av tvärfallet.

Därefter beräknas för varje individuell serie ett viktat glidande medelvärde för vilken förskjutning som minimerar skillnaden i y-led gentemot en referensserie, som exempelvis kan vara den senaste mätningen.[4] För exemplet i figur 2.4 innebär viktningen att endast observationerna från de första 100 metrarna i grafen används i synkningen. Genom att ett långt glidande medelvärde för bästa förskjutning beräknas, justeras sträckan från ca 200 till 500 meter i grafen utifrån den förskjutning som ger bäst resultat under de första 100 metrarna, eller i tvärfallsövergångar som kommer senare på vägsträckan och inte innefattas av de 500 meter som illustreras av figur 2.4.

Resultatet av processen blir en matris där varje kolumn representerar en serie och där varje rad representerar seriens ojusterade position i meter. Matrisens värden visar hur mycket seriens rapporterade position ska förskjutas för att förbättra överensstämmelsen i y-led med referensserien. Figur 2.5 visar denna matris beräknat på data från figur 2.4.

[4]Detta beräknas genom att genom att konstruera en matris med ”lags” och ”leads” för tvärfall och varje position beräkna vilken lag eller lead som minimerar skillnaden i y-led mellan en individuell serie och referensserien. Ett glidande medelvärde för bästa förskjutning tillämpas därefter.



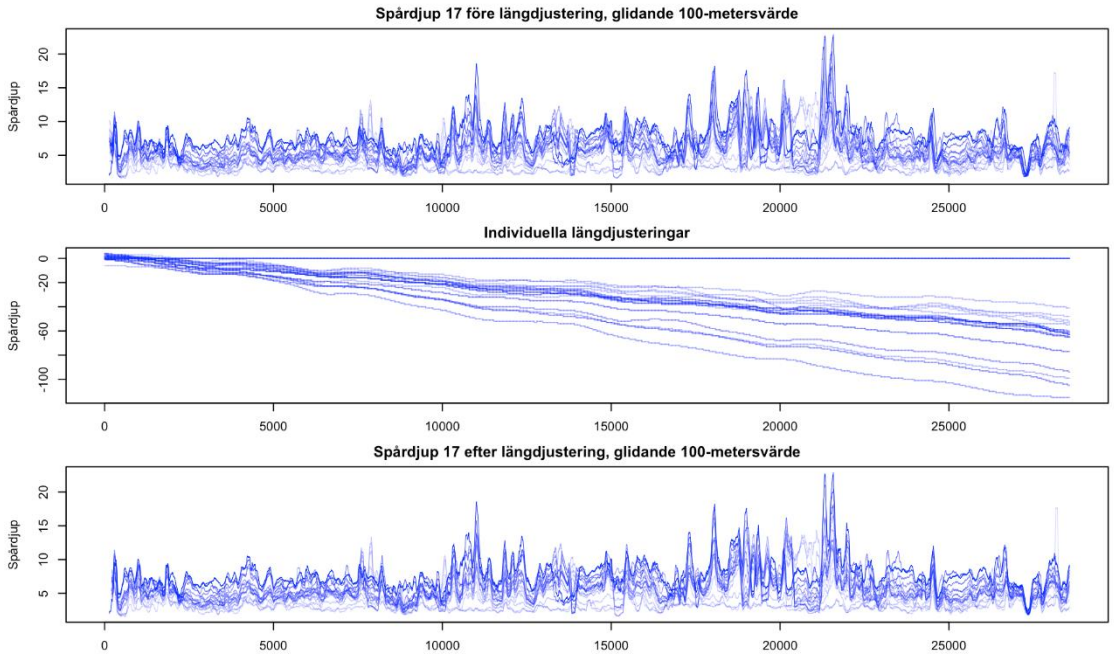
Figur 2.5. De färgade linjerna visar vägytemätningar över en mätt sträcka om 500 meter över ett flertal år. Y-axel visar hur många meter framåt eller bakåt i längdled data justeras när den sedan synkroniseras.

Efter att varje serie justerats i x-led i enlighet med matrisens värden så görs en ny iteration av synkning. Nästa iteration slätar ut serierna i mindre grad än den första, tillåter mindre maximala justeringar i x-led och lägger mindre relativ vikt vid stora förändringar i tvärfall. Dessa förändringar av parametrar medför tillsammans med det bättre utgångsläget i termer av synkning att de mindre förändringarna av tvärfallet kan användas för att ytterligare förbättra synkningen. Antalet iterationer som tillämpas är justerbart liksom de parametrar som används.[5] Figur 2.6 visar resultatet efter tre iterationer.

[5]Parametrarna är: 1. hur mycket serien slätas ut, 2. hur vikterna beror av förändringen av tvärfall, 3. hur många ”lags” och ”leads” som skapas innan bästa lag/lead beräknas och 4. hur långt glidande medelvärde som tillämpas för de individuella ”bästa” förskjutningarna.

Figur 2.6. Tvärfallsdata över 500 meter synkroniserat i tre iterationer.

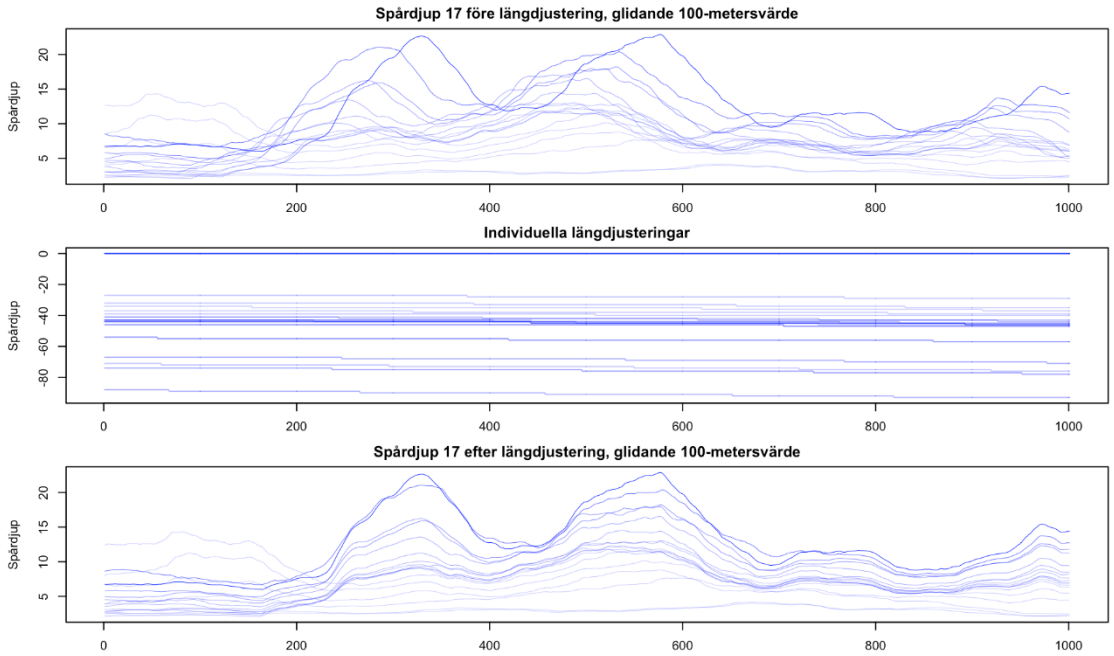
Samma justeringar som gör att serierna med tvärfall blir synkroniserade tillämpas sedan på motsvarande serier över spårdjup. Spårdjupsdata används alltså inte till att göra justeringar, justeringarna görs endast med hjälp av tvärfallsdata. Före justering innehåller framräknade förändringar av spårdjup för individuella 20 metersträckor till stor del effekter av felrapporterad position: att de mätpunkter som jämförs representerar olika faktiska positioner i längdled. Efter en lyckad synkronisering kommer de istället att till allra största del visa på verkliga förändringar av uppmätt spårdjup över tid för en och samma bit av en väg. Figur 2.7, där färgmättnaden indikerar seriernas inbördes kronologiska ordning, visar i den övre grafen, glidande 100-metersvärden för spårdjup före genomförd synkronisering. I mitten visas längdjusteringarna, och nederst visas spårdjupet efter justeringar.



Figur 2.7. Visar synkroniserade spårdjupsdata över en hel vägsträcka.

Som den mittersta figuren visar så ökar förskjutningen längs med sträckan. För sträckans första 5000 meter är förskjutningens effekt svår att se med blotta ögat, men efter ungefär 15000 meter, då vissa serier har en förskjutning som är större än 30 meter börjar effekten synas. Mot sträckans slut har vissa serier förskjutits med så mycket som 100 meter.

Figur 2.8 visar en närbild av slutet på samma sträcka som illustreras i figur 2.7 och tydliggör hur längdjusteringarna förbättrar rapporterat data. I den övre grafen i figur 2.8 minskar vid vissa distanser det rapporterade spårdjupet över tid till följd av felaktig längdangivelse. Efter att de positionsjusteringar som den mittersta figuren indikerar gjorts så blir serierna betydligt mer informativa. Under de första 200 metrarna kan en beläggningsåtgärd anas, eftersom de högsta spårdjupet vid efterföljande mätningar ligger lägre, och därefter ökar spårdjupet systematiskt över de kommande 300 metrarna.



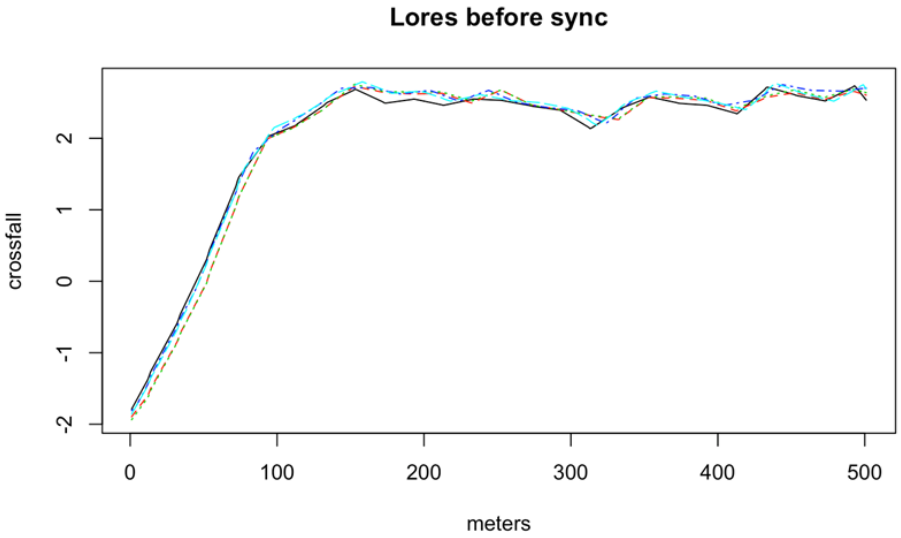
Figur 2.8. Visar effekten över en sträcka på 1000 meter.

Utan justeringar är omläggningen i figur 2.8 och dess utbredning relativt svår att identifiera varför den genomsnittliga spårdjupsförändringen hade underskattats. Med längdjusterade serier kan beläggningsåtgärder identifieras och påverkan från dessa kan rensas bort i prognostiseringen. Från de justerade serierna går det också att utläsa att spårdjupets ökning varierar längs sträckan.

Vissa serier kan dock bedömas innehålla avvikelser av en natur som gör justering svår att utföra och utesluts därför. Bedömning görs genom att jämföra summan av kovarienserna för en dataserie med den genomsnittliga summan av kovarianser för alla dataserier. Uteslutningen sker efter att justeringar av position gjorts. Algoritmen innehåller ett antal parametrar som kan justeras för att få bästa möjliga resultat givet förutsättningarna, som vägsträckans längd och kvaliteten på tvärfallsdata. Ett möjligt fortsatt arbete är att förbättra metoderna för att automatiskt välja optimala parametrar utifrån egenskaperna hos de serier som ska synkroniseras.

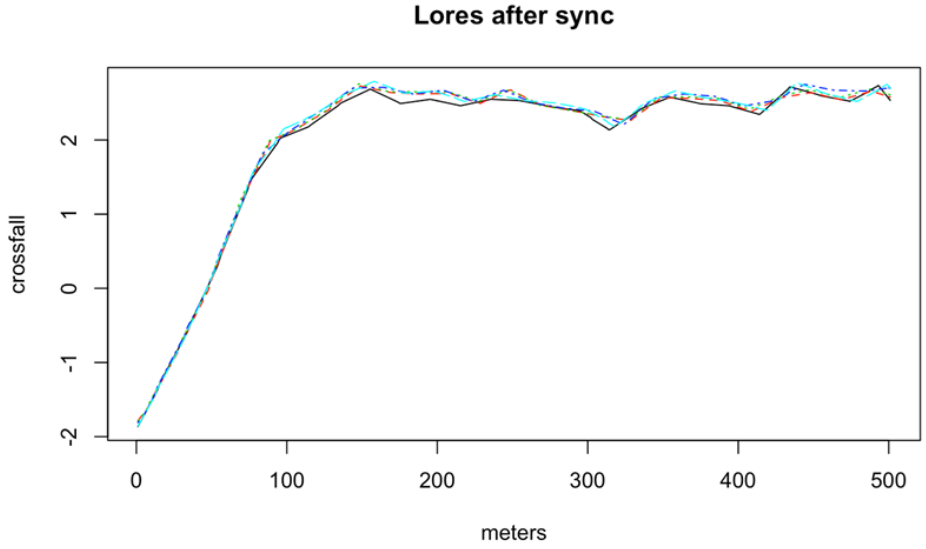
SYNKRONISERING AV TRAFIKVERKETS DATA

Trafikverkets data synkroniseras enligt samma metod som beskrivs ovan. Eftersom olika serier inte har mätvärden för samma angivna positioner genomförs dock som första steg en interpolering av mätvärden för tvärfall och spårdjup. Distanserna avrundas därefter till hela 1-metersvärden. Synkroniseringen av Trafikverkets relativt lågupplösta data blir sämre till följd av att de faktiska mätvärdena inte görs vid samma position vid varje mätning och att avstånden emellan mätpunkterna är längre. Figur 2.9 visar Trafikverkets PMSv3-data för samma delsträcka.



Figur 2.9. Trafikverkets tvärfallsdata före synkronisering.

Efter synkronisering syns det tydligt på de första 100 metrarna att en förbättring uppnåtts (Figur 2.10). I det här fallet har endast en två iterationer gjorts. Den del av sträckan som inte innehåller stora skiften av tvärfallet (200-500 m) kan dock inte användas till ytterligare förbättringar, vilket bör framgå av figur 2.9. Eftersom det på vissa vägsträckor kan vara så långt som 5000 meter mellan tydliga skiften av tvärfallet blir datapunkterna däremellan inte lika bra synkroniserade som med den högupplösta data.



Figur 2.10. Trafikverkets tvärfallsdata efter synkronisering.

Trafikverkets lågupplösta data kan också kombineras med egen högupplöst data. I detta fall görs först en synkning av varje dataset för sig (lågupplöst och högupplöst). Därefter

synkas de lågupplösta serierna till de högupplösta serierna genom att ett medianvärde för de lågupplösta serierna synkas med de referensserien för högupplöst data.

PROGNOSTISERING

Ett rimligt antagande är: förändring av uppmätt spårdjup är antingen positiv (slitage och deformation) eller negativ (utförd beläggningsåtgärd) och en kombination av dessa två typer av förändringar representerar genomsnittliga förändringar av spårdjupet över en hel sträcka. Skulle omfattningar av omläggningar och de spårdjup som var rådande då de utfördes vara kända skulle ett relativt bra mått på genomsnittlig förändring kunna skattas genom att använda någon percentil av förändringarna. Men eftersom dessa variabler till stor del är okända så är detta angreppsätt inte optimalt. Tidigare studier har med relativt gott resultat använt sig av percentilvärden för att uppskatta genomsnittliga förändringar på medelvärdesbildade 100 metersträckor för spårdjupet (Andren och Eriksson, 2019). Men data i detta projekt bedöms bli alltför brusigt då vi försöker prognostisera på 20 metersnivå. Spårdjupsförändringen varierar mellan olika delsträckor vilket innebär att det finns en vinst i att skatta förändringen över tid för kortare delsträckor. Vi har vid behandlingen av spårdjupsdata tagit i beaktning att data innehåller tre typer av felkällor: (1) det uppmätta spårdjupet skiljer sig från det faktiska spårdjupet, (2) mätningens position i längdled kan avvika från den faktiska positionen i olika tidsserier samt (3) att vissa dataserier kan anses vara helt felaktiga på grund av fel i datahanteringen; fel riktning, körfält, vägnummer registreras vid inläsning till databas etc. vilket gör att vissa data behöver sorteras bort. Helt felaktiga mätdata kan ganska enkelt i ett tidigt skede identifieras och förkastas. Variation i mätningen av spårdjup kan såvida mätutrustningen fungerar korrekt antas vara slumpmässig och relativt liten. Den är svår att göra någonting åt i efterhand men har visat sig ha relativt liten betydelse i sammanhanget. Avvikelser i position har visat sig utgöra ett betydligt större problem då olika serier för en enskild rapporterad position kan representera olika faktiska positioner, vilket leder till att beräknade förändringar över tid för vissa sträckor är nästan helt slumpmässiga. Problemet har dock till stor del åtgärdats genom den dynamiska längdsynkronisering som utförts på vår data.

Nästa steg i databehandlingen är identifieringen av utförda beläggningsåtgärder längs sträckan. Denna del är kritisk och en grundförutsättning för att skapa en träffsäker prognos av spårdjupsutvecklingen. Om föregående steg utförts väl så syns beläggningsåtgärder tydligt genom att spårdjupet för en sträcka kraftigt minskar från ett år till ett annat.

Spårdjupsdata kommer även efter synkronisering att innehålla vissa felkällor. Dels finns en mindre slumpmässig variation i uppmätt spårdjup givet positionen. Dels finns potentiellt en stor variation till följd av att serierna inte synkroniserats exakt, vilket gör att jämförda datapunkter därför inte representerar samma position. Att med exakthet

identifiera uteliggare och beläggningscykler är därför inte möjligt, utan blir en bästa uppskattning.

Synkroniseringens kvalitet kommer variera längs med en sträcka. Bäst blir den i närheten av de positioner där tvärfallet tydligt förändras och där serier kan synkroniseras med stor exakthet. Sämst blir de kring mitten av långa sträckor utan tydliga förändringar i tvärfallet.

Att behandla data från individuella 20-metersträckor som oberoende mätvärden är därför inte alltid en optimal metod. Med kortare serier fås bättre prognostiseringar när kvaliteten på data är bra, men med dålig kvalitet blir variansen skattningarnas kvalitet lidande. Vid dålig datakvalitet kan glidande medelvärden användas för att minska skattningarnas varians. Ett annat alternativ är att göra skattningen för en längre sträcka. Ett tredje alternativ är att beräkna ett viktat värde:

$$\alpha * \text{medelvärde} + (1 - \alpha) * \text{punktskattning}$$

Punkskattning anger den förändring som prognostiserats med data från en 20-metersträcka, och medelvärde kan representera en hel vägsträcka eller ett glidande medelvärde längs ett godtyckligt antal meter, och där viktningsparametern α kan bestämmas av hur väl tvärfallet synkroniserar vid mätpunkten. På så vis kan precisa prognostiseringar erhållas för sträckor med god datakvalitet samtidigt som robusta prognostiseringar fås för sträckor med sämre datakvalitet. För sträckor där ett flertal mätningar finns tillgängliga kan parametern väljas utifrån vilket värde som bäst prognostiserar en exkluderad ”testserie”.

Vårt synkroniserade data har visat sig användbart för att göra relativt granulära skattningar av spårdjupsförändringar, vilket för vissa delsträckor där spårdjupsökningen är särskilt hög eller låg kan innebära en väsentlig förbättring jämfört med tidigare manuella metoder där ett medelvärde för spårdjupsökningen över en längre sträcka använts.

För att utvärdera prognosmodellen analyserades spårdjupsvärden på 20 meters-nivå för en grupp utvalda totalentreprenader. De senast uppmätta 20 metersvärden för spårdjup uteslöts. Med endast de tidigare uppmätta värden så prognostiseras ett nytt värde fram. Sedan jämförs skillnaden mellan de prognostiserade värden med det värde som tidigare uteslöts. Medelvärdet av skillnaden mellan uteslutna värden och prognostiserade värden längs ett helt körfält har sedan använts som ett mått för att utvärdera hur väl prognosmodellen fungerar.

Eftersom spårdjupsprognostiseringen blir missvisande om inte beläggningsåtgärder identifieras, beaktas utgör identifieringen av beläggningsåtgärder ett viktigt delmoment i analysen. Metoden redovisas i bilaga D, men går i korthet ut på att identifiera negativa hopp i tidsserier, alltså där spårdjupet kraftigt minskar mellan två mätningar.

Beräkningen av förväntad spårdjupsförändring görs genom linjär regression, som skattar hur mycket spårdjupet ökar över tid. För en sträcka där varken beläggningsåtgärder eller uteliggare registrerats utgörs regressionen av endast två variabler:

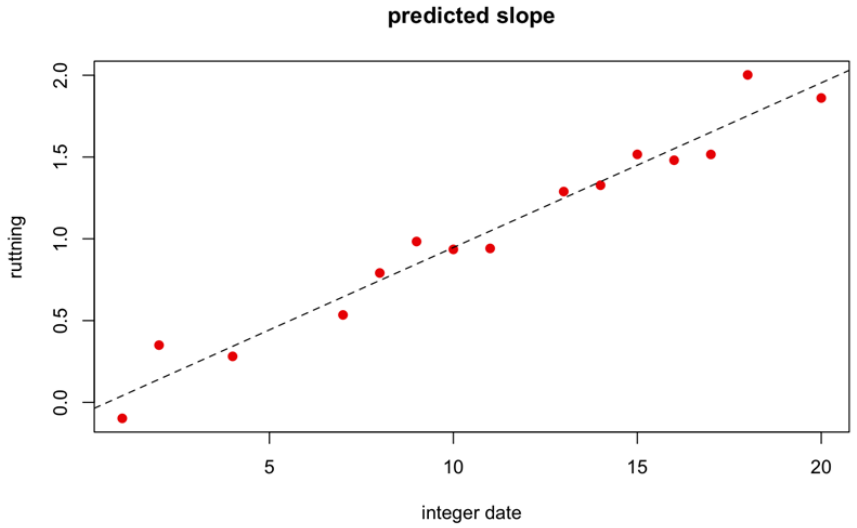
- x: seriens datum i integer-format, där varje enhet representerar en dag
- y: seriens spårdjup mätt i mm

Sambandet mellan tid och spårdjup antas i det enklaste fallet, då inga omläggningar gjorts, ta följande form:

$$y = bx + e$$

Ekvationen säger att det uppmätta spårdjupet i genomsnitt antags förändras med en konstant (b) för varje enhet som x-variabeln ökar, alltså en dag. Därtill antas feltermen (e) utgöras av summan av mätfel och slumpmässiga avvikelser från de den antagna förändringsfaktorn mellan mätningar.

I figur 2.11 representerar den streckade linjens lutning det skattade värdet för (b) medan de horisontella avstånden mellan skattningslinjen och de individuella punkterna de skattade feltermerna (e)[6]. Konstanten (b) skattas med minstakvadratmetoden, vilken minimerar summan av kvadrerade avvikelser från skattningslinjen.



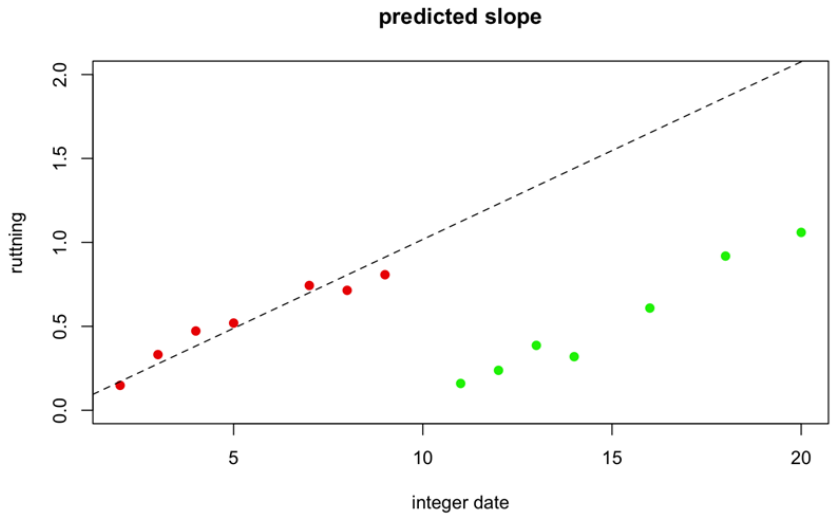
Figur 2.11. Illustrering av linjär regression utan omläggningscykler.

När omläggningar identifierats representeras dessa med indikatorvariabler för varje omläggningscykel:

[6] Inom den ekonometriska litteraturen antas ett sant (b) och en sann felterm existera. Modellen ger ett skattat (b) vars förväntade värde är det detsamma som det sanna (b), eftersom feltermen antas vara normalfördelad kring värdet noll och vara okorrelerad med modellens förklarande variabler. I ett mindre sample är residualen alltid i någon grad korrelerad med modellens förklarande variabler vilket medför att skattat (b) aldrig blir exakt detsamma som det underliggande sanna (b). Den skattade feltermens korrekta benämning är residual, och är en matematisk konstruktion som alltid har medelvärdet noll.

$$y = bx + \delta_1 c_1 + \ldots + \delta_n c_n + e$$

I figur 2.12. illustreras hur parametern (b) genom inkludering av en indikatorvariabel för beläggningsomgång blir ett vägt[7] genomsnitt av den skattade lutningen inom varje.



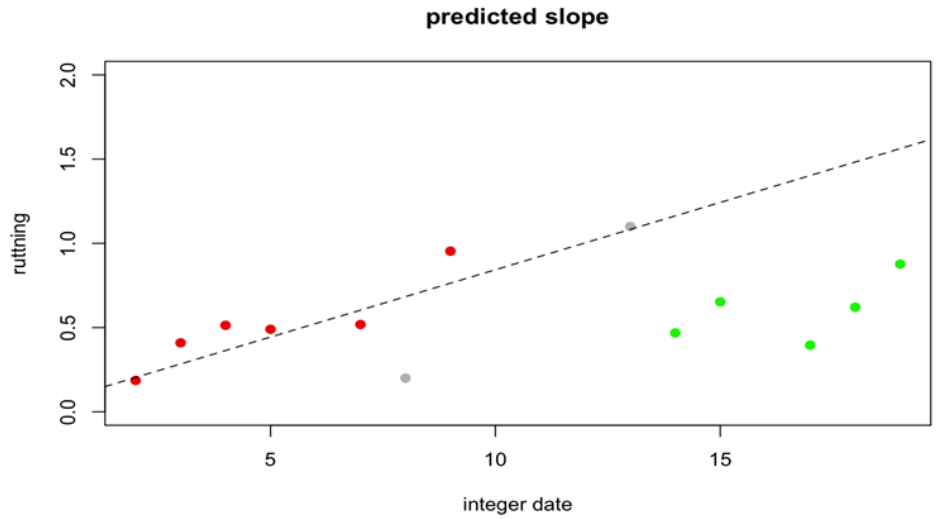
Figur 2.12. Illustrering av regression med två beläggningscykler.

Om en eller flera uteliggare identifierats så elimineras dessa från beräkningen genom att en dummy-variabel läggs till för varje sådan observation, vilket gör att dess inflytande på den skattade lutningen elimineras.

$$y = bx + \delta_1 c_1 + \ldots + \delta_n c_n + \lambda_1 d_1 + \ldots. + \lambda_n d_n + e$$

Skattningarnas kvalitet beror på kvaliteten hos den data som används. Denna beror i sin tur i hög grad på hur väl mätserierna synkroniserats. Figur 2.13 illustrerar ett fall med två identifierade beläggningscykler, två dummyvariabler och relativt stor felterm. I detta fall är det inte uppenbart att de röda och de gröna datapunkterna faktiskt representerar två olika cykler. Alternativt skulle den sista röda punkten kunna vara en uteliggare och samtliga datapunkter höra till samma beläggningscykel. Här råder således större osäkerhet gällande sträckans spårdjupsförändring över tid än i figur 2.12.

[7]Viktningen beror på variansen runt skattningslinjen och antalet datapunkter i respektive cykel



Figur 2.13. Illustration av regression med två beläggningscykler och skattning med relativt dålig data.

Skattningarnas osäkerhet kan kvantifieras genom att beräkna ett prediktionsintervall, vilket kombinerar två komponenter. Den första komponenten är konfidensintervallet för den skattade lutningen, vilket är en funktion av residualernas varians, den tidslängd som mätningarna sträcker sig över och antalet datapunkter. Den andra komponenten är även den, residualens varians, vilken är oberoende av antalet datapunkter och tidsspannet.

Tabell 2.3 visar ett utdrag från den CSV-fil där prognosresultatet dumpats. Senast uppmätta spårdjupsdata för både Spårdjup 17 och Spårdjup 15 skrivs ut tillsammans med beräknad årlig spårdjupsökning för dessa. Utöver detta skrivs även två kvalitetsmått för den prognostiserade spårdjupsökningen ut. Det första kvalitetsmåttet är standardavvikelsen för de punkter som använts i regressionsanalysen mot punkternas regressionslinje. Det andra måttet som vi döpt till Prognoskvalitet, anger hur många punkter som använts i regressionsanalysen.

Tabell 2.3 Resultat från prognostiseringen.

	Spårdjup (Rutting) 17				Spårdjup (Rutting) 15			
Distans	20210511	Årlig ökning	Stdev	Prognoskvalitet	20210511	Årlig ökning	Stdev	Prognoskvalitet
57	6,59	0,96		1	6,76	0,98		1
77	5,17	2,75		2	7,22	4,22		2
97	4,58	0,85	1,94	4	6,61	1,23	2,53	4
117	13,15	2,74	2,94	5	15,91	3,33	2,25	4
137	14,30	4,65	4,52	3	16,58	5,09	2,14	3
157	9,35	2,56	1,83	3	11,94	4,93	1,86	3
177	5,10	0,64	0,48	6	8,09	0,94	1,55	6
197	3,52	0,31	0,23	6	4,25	0,33	0,19	6
217	3,92	0,39	0,20	6	4,08	0,36	0,16	6
237	2,87	0,27	0,20	6	3,27	0,29	0,25	6
257	1,88	0,21	0,22	6	2,03	0,16	0,34	6
277	2,24	0,31	0,28	6	2,25	0,27	0,39	6
297	2,54	0,35	0,34	6	2,54	0,27	0,40	6
317	3,00	0,39	0,29	6	3,00	0,32	0,15	3

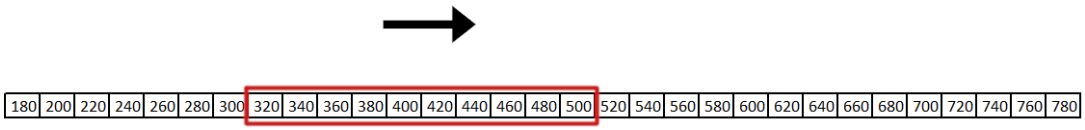
EXPORT AV UNDERHÅLLSPLANERING

CSV-filen med prognosresultatet exporteras i nästa steg vidare till en Excelfil där själva underhållsplaneringen sker. Till denna fil kommer även viss övrig information om körfältet exporteras, som t.ex. skyltad hastighet, körfältsbredd, spårdjupskrav m.m.

I Excelfilen har vi sedan lagt till formler som med hjälp av den prognostiserade årliga spårdjupsökningen och senast uppmätta spårdjup visar en matris som innehåller spårdjupet för varje år till garantitidens slut. I denna matris kan man sedan manuellt markera celler för de sträckor man önskar utföra beläggningsunderhåll på. Men det behövs även automatiserade funktioner för att lättare kunna hitta mer optimala åtgärdssträckor. Man kan tänkas behöva optimera underhållsplaneringen med hänsyn till t.ex. kontraktskrav, underhållskostnader eller resurstillgänglighet.

Det är vanligt förekommande att man vid underhållsplanering försöker homogenisera tillståndsdata på sträckor med liknande tillstånd för att på så sätt få en mer överblickbar vägsträcka. Detta medför dock att information förloras som skulle kunna vara viktig för prognostisering och underhållsplanering på mer detaljerad nivå. Den största fördelen med en sådan homogenisering av tillståndsdata består dock i att mängden data som behöver hanteras och lagras, minskar (Ping et al 1999). Detta anser vi inte vara något större problem idag tack vare moderna datorers beräknings- och lagringskapacitet. Av den anledningen önskar vi jobba med data och basera underhållningsplaneringen på 20 metersnivå i så stor utsträckning som möjligt.

Valet av teknik för att beräkna vilka sträckor som är lämpliga för omläggning i detta utvecklingsprojekt föll på det glidande fönstret (Figur 2.13) då det ger oss möjligheten att i förväg bestämma vilken längd, eller intervall av längder, vi föredrar för våra planerade beläggningsåtgärder.

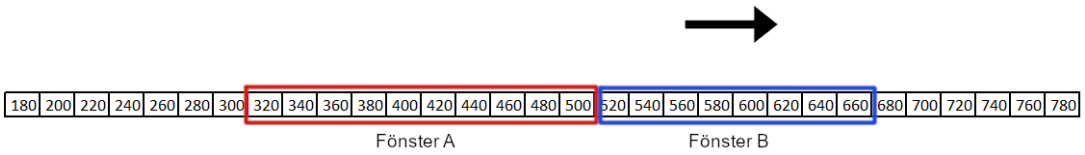


Figur 2.13. Exempel på glidande fönster som är 10 sektioner långt (200 m)

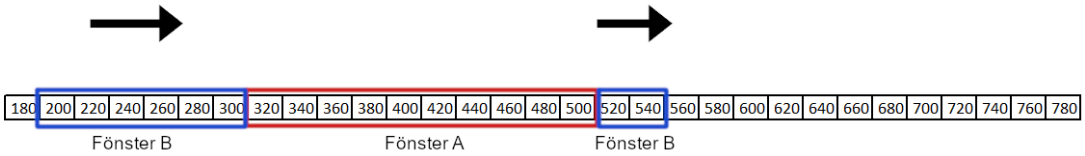
Det är ekonomiskt fördelaktigt att utföra vissa typer av åtgärder i vissa längdintervall. Exempelvis kan fräsning och läggning av nytt slitlager vara att föredra på relativt korta sträckor jämfört med att ta in en remixer-maskin för samma arbete eftersom etableringskostnaden för den senare är relativt hög och kräver således relativt långa åtgärdssträckor för att vara ekonomiskt lönsam. Detta för att passa in tidsåtgången att utföra en beläggningsåtgärd med längden på arbetsskift för yrkesarbetare samt transport och etableringskostnader i relation till timkostnader för yrkesarbetare och maskiner. Om

man bortser från denna optimering bör det planerade underhållet i teori, förstås skapas med så få, men samtidigt långa oavbrutna sträckor, som möjligt.

Förfarandet vid beräkning av det glidande fönstret går förenklat till på detta sätt. Ett fönster på t.ex. 1 km glider längs vägsträckan med steg om 20 m. För varje steg fönstret flyttas så beräknas ett statistiskt mått för de 20-meterssträckor som är inom fönstret i det steget. Implementeringen av det glidande fönstret i detta projekt använder sig av en del olika metoder för att finna den optimala positionen för fönstret. För de värden som är inom fönstret, och för varje steg som fönstret tar så beräknas medianvärdet, medelvärdet, beläggningens restlevnadstid, och bl.a. antalet värden som överstiger kontraktskrav. Detta gör att användaren själv kan välja vilket nyckeltal vägunderhållet ska optimeras mot. Det glidande fönstret behöver även ta hänsyn till specialfall. När t.ex. fönstret stöter på vägsträckor som har andra eller saknar funktionskrav. På sträckor där det angivna körfältet saknas (t.ex. körfält 2 på en 2+1-väg) tillämpas s.k. *överhopp* (*Figur 2.14)* av fönster. Om det glidande fönstret stöter på ett redan sparat glidande fönster där man tidigare bestämt att en beläggningsåtgärd ska utföras med samma beläggningstyp så tillämpas s.k. *tunnling* av fönster (*Figur 2.15).*

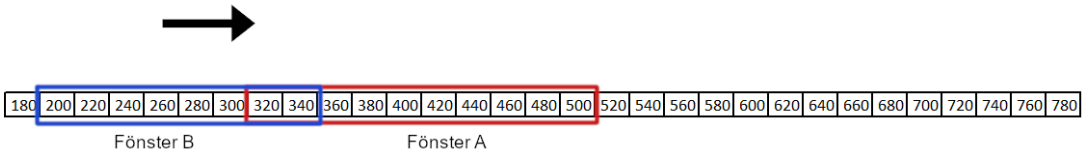


Figur 2.14. Exempel på överhopp. Fönster B glider mot en sträcka där aktuellt körfält saknas som här representeras av Fönster A. Då upphör fönster B för att sedan uppstå i sin helhet på högra sidan av fönster A.



Figur 2.15. Exempel på tunnling.

I de fall fönstret bör tillämpa överhopp och delsträckan som fönstret glider över är kortare än själva fönstret, så tillämpas överlapp. (Figur 2.16). Det finns flera andra mindre vanliga fall som måste hanteras ur ut ett programmeringstekniskt perspektiv men som inte dokumenterats här.



Figur 2.16. Exempel på överlapp.

Den data undersökts i utvecklings- och testfasen kommer från 7 utvalda entreprenader som utförts under de senaste 13 åren (se tabell 2.3). Entreprenaderna är av olika vägtyper (länsväg, 2+1- och 2+2 vägar) omfattar såväl nybyggnadsobjekt som underhållsobjekt och har individuella garantitider och längder. Anledningen till att dessa objekt valdes var att mätdata för dessa var lättillgänglig samt att dessa objekt är några av de som mätts minst en gång årligen med NCC:s profilografbil, vilket gett ett större dataunderlag för att testa mjukvaran.

Tabell 2.3. Funktionsentreprenader som ingått i test och analysfasen av mjukvaran.

Objekt	Kategori	Trafiköppning (år)	Garantitid (år)	Vägtyp	Längd (km)
Totalentreprenad 1	NB	2008	15	2+1	7
Totalentreprenad 2	NB	2011	10	LV	20
Totalentreprenad 3	NB/B	2012	10	2+2	5,4
Totalentreprenad 4	NB/B	2012	10	2+2	9,7
Totalentreprenad 5	NB	2013	20	2+2/2+1	28,6
Totalentreprenad 6	U	2015	17	2+2	19
Totalentreprenad 7	U	2009	10	2+2	11

UTMANINGAR

Den största utmaningen under utvecklingsprojektet var utan tvekan att kunna nyttja Trafikverkets spårdjupsdata till att prognostisera tillförlitliga spårdjupsvärden på 20-metersnivå, då tillståndsdata levereras från Trafikverket redan på 20-metersnivå. Succesiva mätningar mellan olika år är som tidigare nämnts inte synkroniserade i längdled, vilket visat sig nästintill omöjliggöra en tillförlitlig prognostisering på 20-metersnivån. Efter flera olika försök att passa denna data i längdled för att lösa detta problem så återstod problemet att de längdsynkroniserade 20-meterssträckorna inte sammanfaller med startpunkten på körfältet som användaren angivit. Ett beslut togs att interpolera Trafikverkets data till nya 20 m sträckor så de sammanfaller med 20-meterssträckorna som användaren valt till sitt vägprojekt.

Generering av vägsträckor utifrån en utplacerad start- och slutpunkt på karta blev en ganska komplicerad process då det underlag vi använde oss av inte hade direkta kopplingar mellan OID och anslutningspunkterna i väggeometrin. Vi använde oss av en ruttgenereringsmodell som räknade sig fram i en 2D-geometri, detta kan leda till en automatiskt genererad rutt som är felaktig. Exempelvis kan rutten gå mot enkelriktat, fel

riktning genom cirkulationsplatser osv. Detta kringgår vi dock genom att man kan definiera en rutt genom att placera ut fler än två punkter på kartan, samt att man kan välja att ignorera information om tillåtna riktningar i vägnätet.

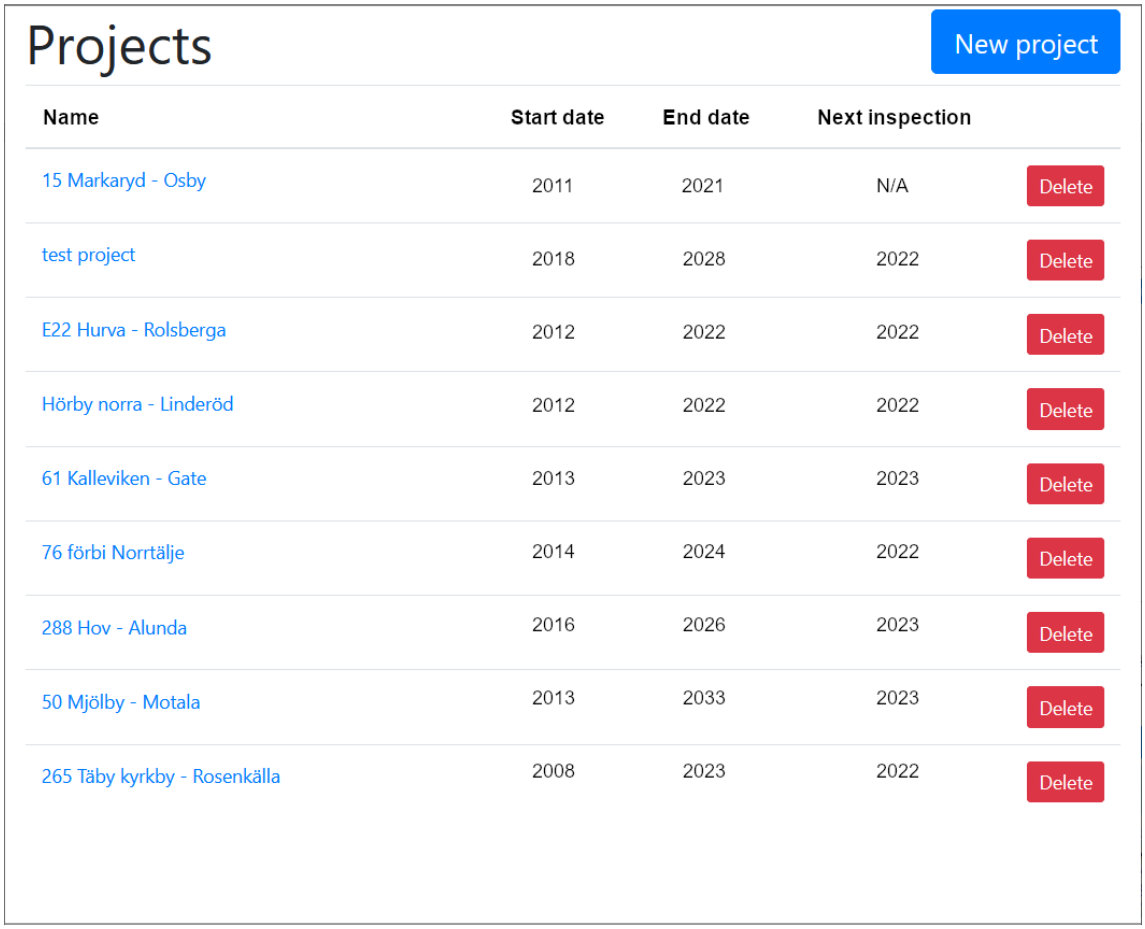
Längdsynkronisering mellan egen uppladdad data och väggeometrin var ett svårlöst moment och har krävt ett stort arbete med flertalet algoritmer av varierande komplexitet. Relativt mycket arbete lades ned på denna del, och det finns fortfarande möjligheter till förbättring.

3. RESULTAT

I detta avsnitt beskrivs resultatet från projektet närmare.

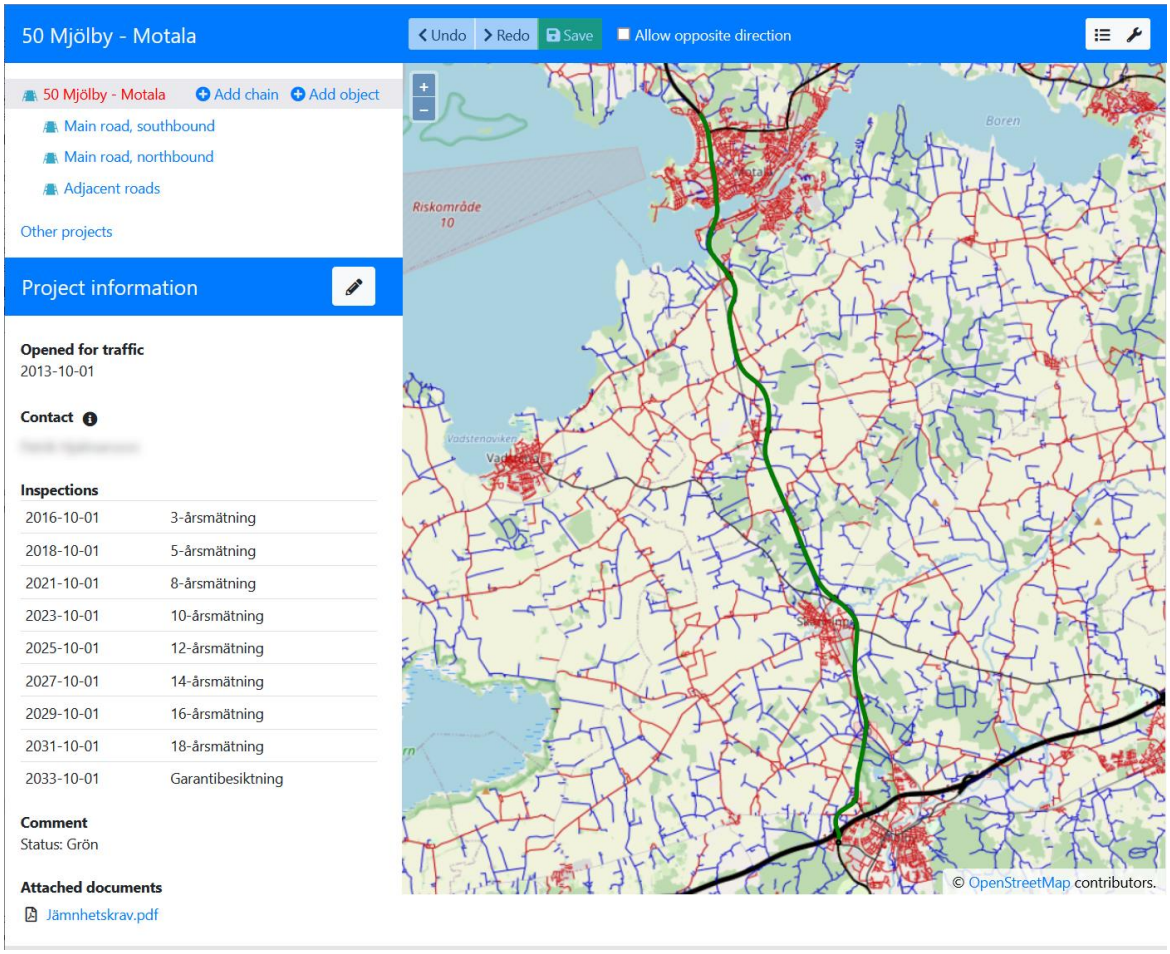
ANVÄNDARGUIDE

Huvudmenyn i det framtagna verktyget (figur 3.1.) ger en överblick över projekt, deras löptid samt nästa besiktningsdatum. När nytt objekt skapas anger användaren objektnamn, trafiköppningsdatum, kontaktinformation till ansvarig för projektet, längd på kravställda sektioner, t.ex. 20 m, besiktningsdatum med mera.



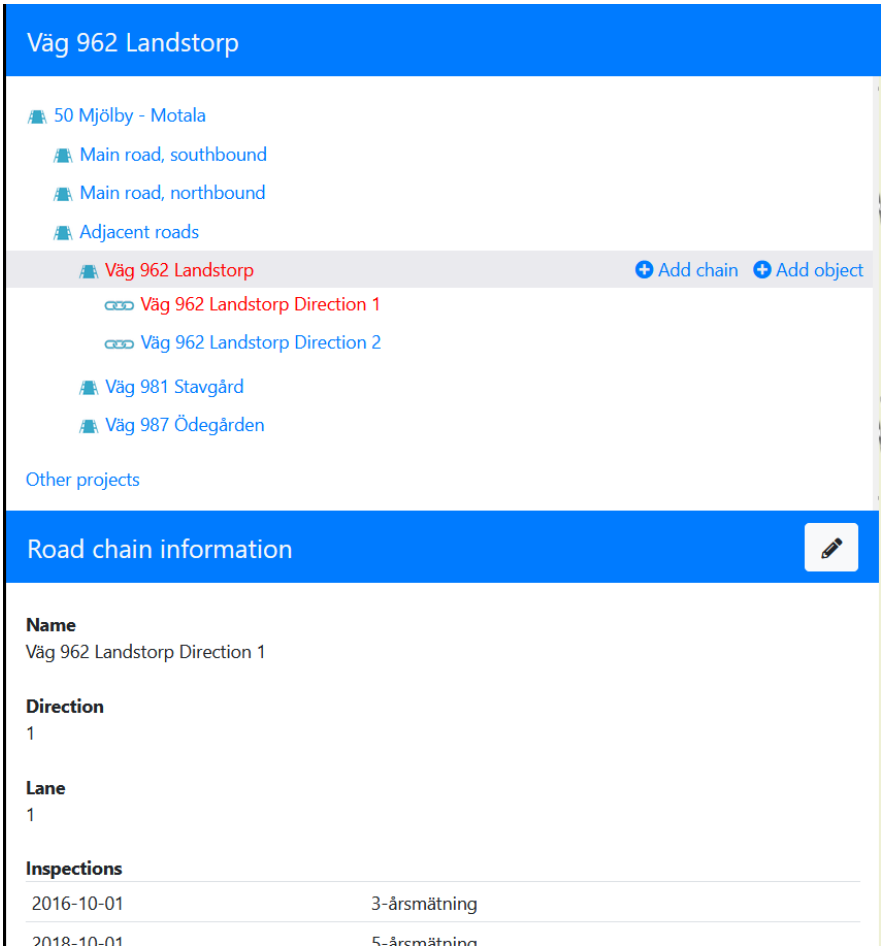
Figur 3.1. Projektlista med exempel på entreprenader som användaren lagt till.

När man öppnar ett givet projekt kommer man till applikationens huvudgränssnitt (figur 3.2.) där man dels kan se ytterligare relevant information om projektet och dels projektets utsträckning projicerad på karta.



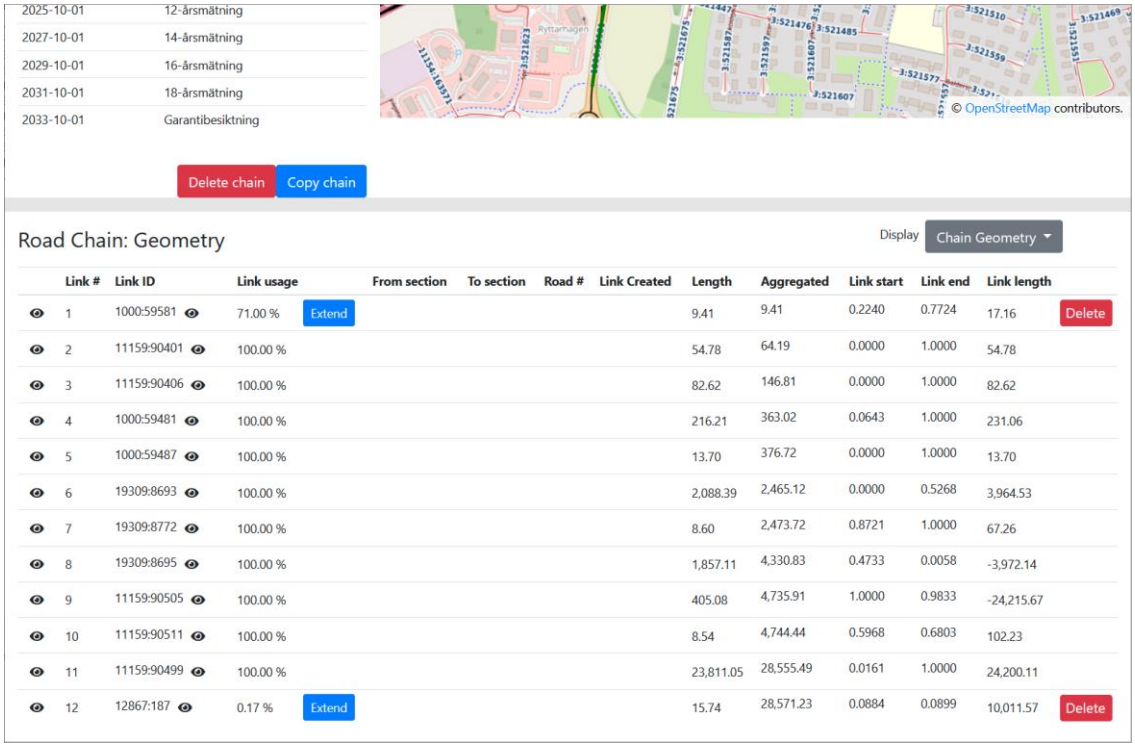
Figur 3.2. applikationens gränssnitt.

Vid stora kontrakt som innehåller flera vägar kan man skapa en projekt-hierarki för att gruppera t.ex. huvudväg, ramper, mindre anslutande vägar osv. Varje väg i listan behöver en definierad geometri för varje körfält. Denna geometri markeras ut på karta av användaren när denna skapar körfältet. Körfältet sparas i samma referenssystem som Trafikverket använder, dvs. en kedja med Objekt-ID (OID). Därav föreställer ikonen för körfältet i figur 3.3. en kedja. Grupperingar av underprojekt underlättar då man kan skapa bl.a. kravgränser och besiktningsdatum som sedan kopieras till en hel väggrupp.



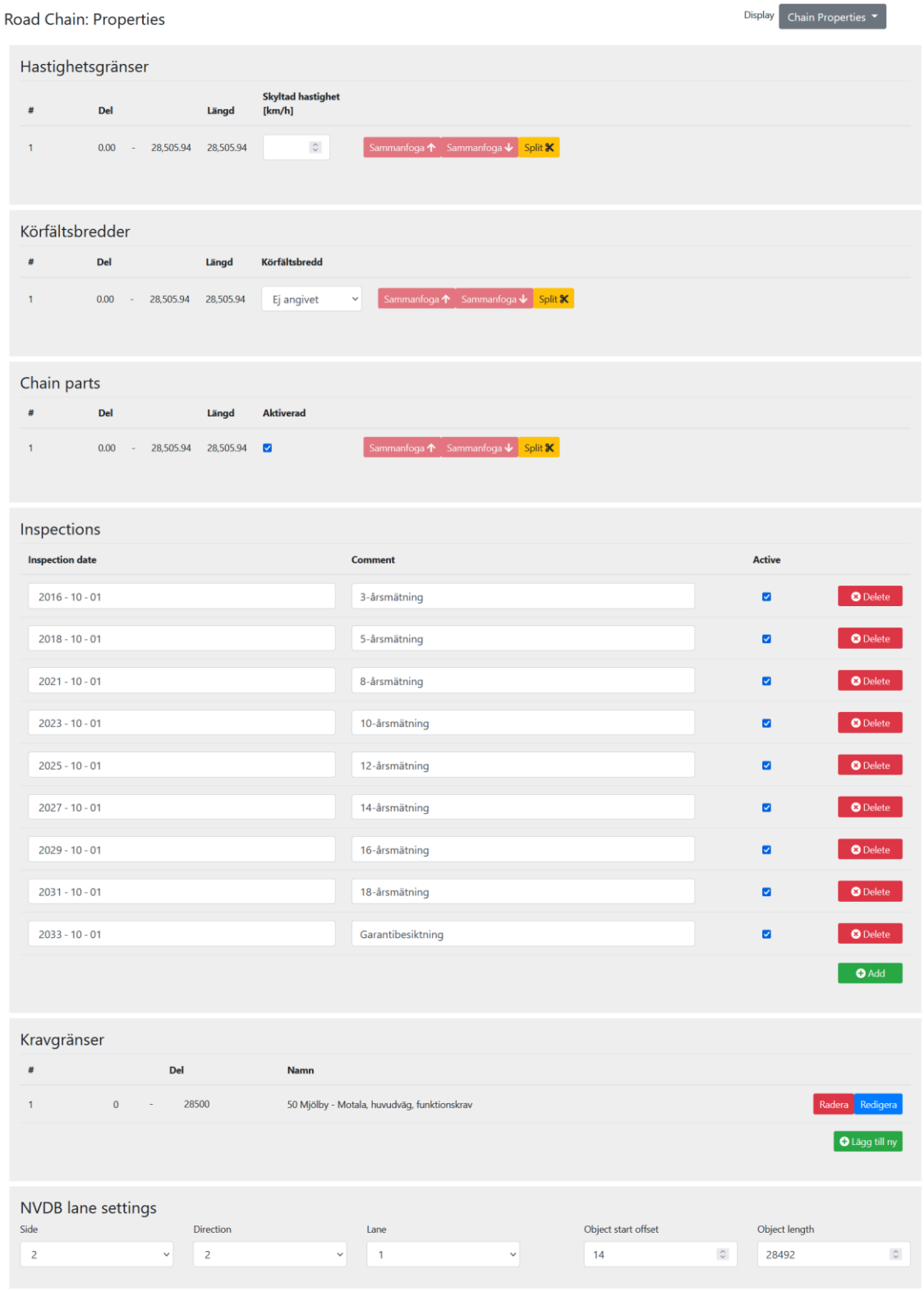
Figur 3.3. Faksimil från verktyget som visar hur användaren skapat ett huvudprojekt för en väg (50 Mjölby – Motala) där underprojekt skapats för huvudvägen i varje riktning samt ett separat underprojekt för anslutande vägar som också omfattas av entreprenadkontraktet.

Den underliggande väggeometrin kan granskas och OID-kedjan kan redigeras så start och slutposition stämmer med omfattningen i kontraktet (Figur 3.4.).



Figur 3.4. Visar analyssidan för väggeometrin där man också kan redigera sin OID-kedja.

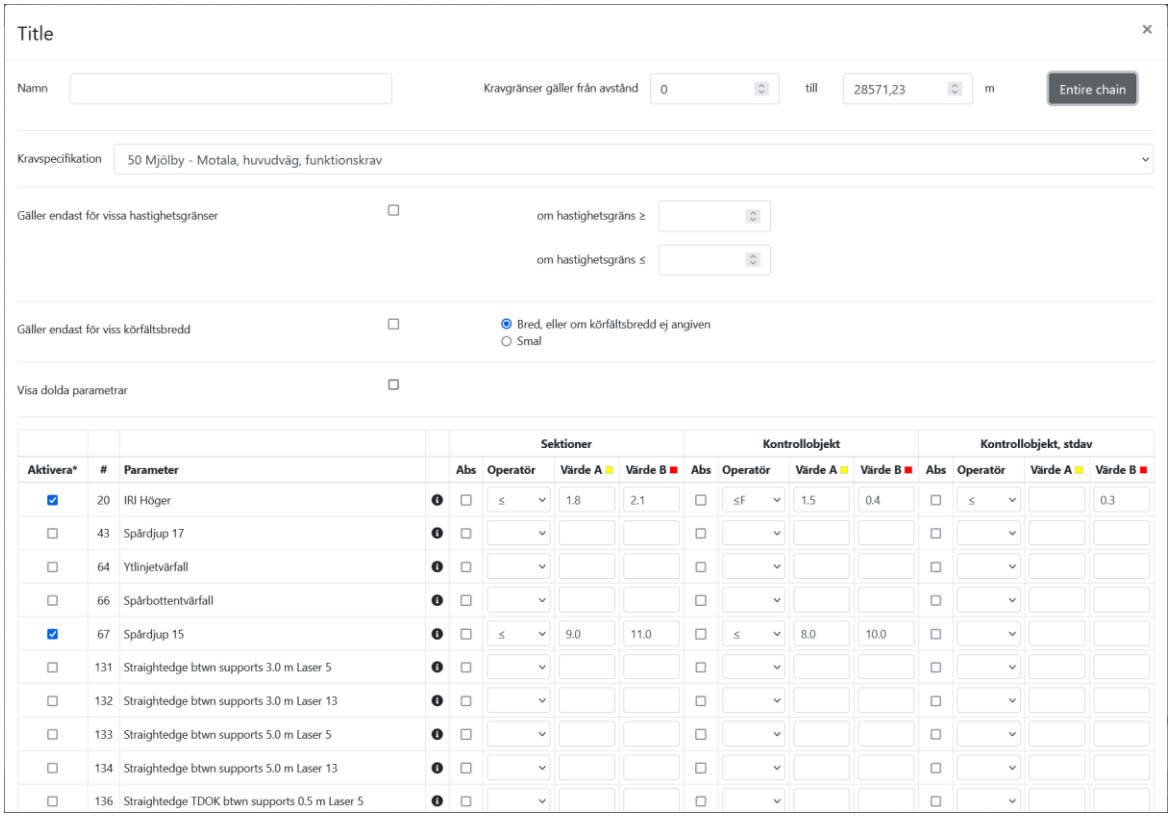
Hela väggrupper, enstaka vägar, enskilda körfält eller t.o.m. enskilda sträckor på ett körfält kan tilldelas kravgränser för vägytans beskaffenhet. I figur 3.5. illustreras gränssnittet där ett körfält tilldelas olika egenskaper. Man kan ange hastighetsgränser, körfältsbredder eller olika objektdelar baserade på definierade distanser.



Figur 3.5. Faksimil från verktyget var man anger egenskaperna för ett enskilt körfält.

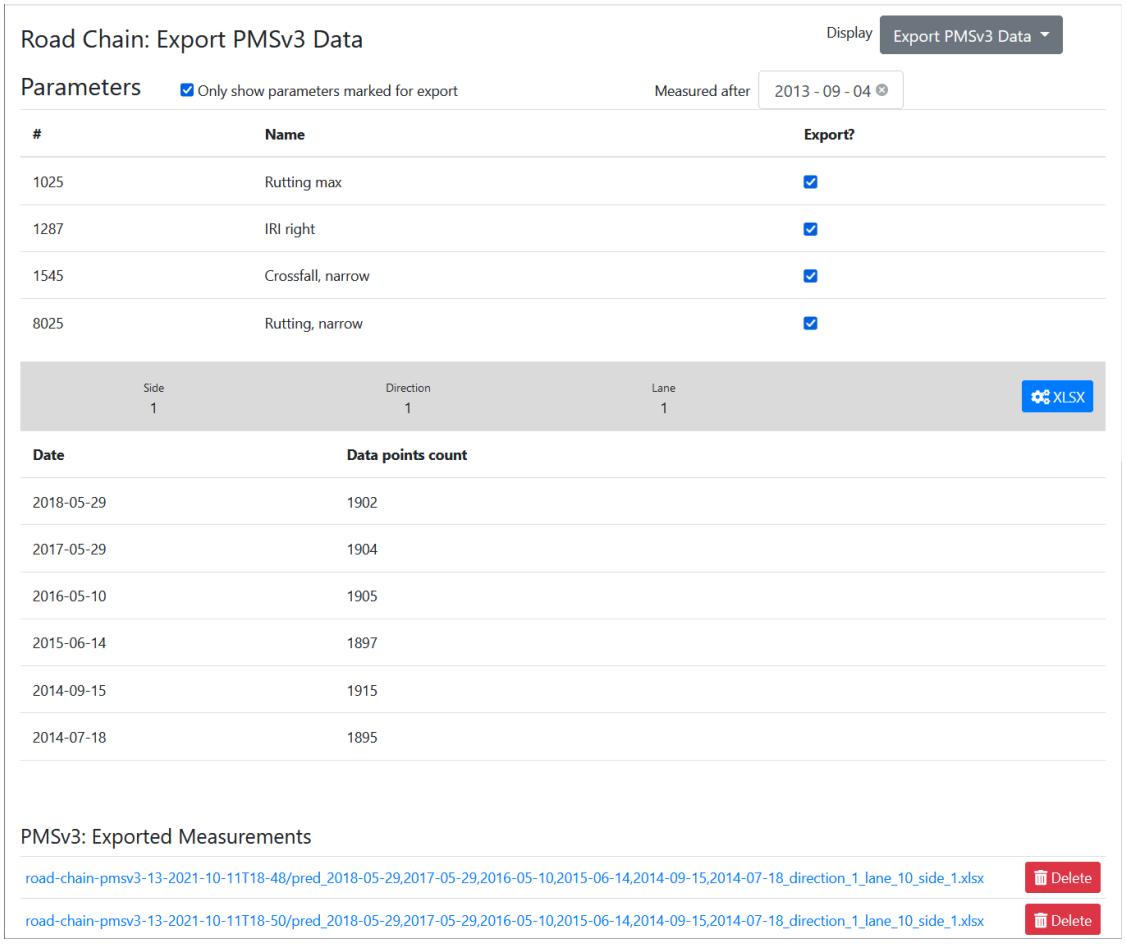
Utöver uppgifterna ovan kan man även tilldela kravgränser med olika kravnivåer beroende på körfältets tidigare tilldelade egenskaper (figur 3.6). Där sträckor varierar mellan nybyggnads- och underhållsentreprenad eller där körfältsbredden varierar är det

då möjligt att då sätta olika kravnivåer. Kraven kan skapas separat och i efterhand tilldelas olika grupper, vägar eller körfält.



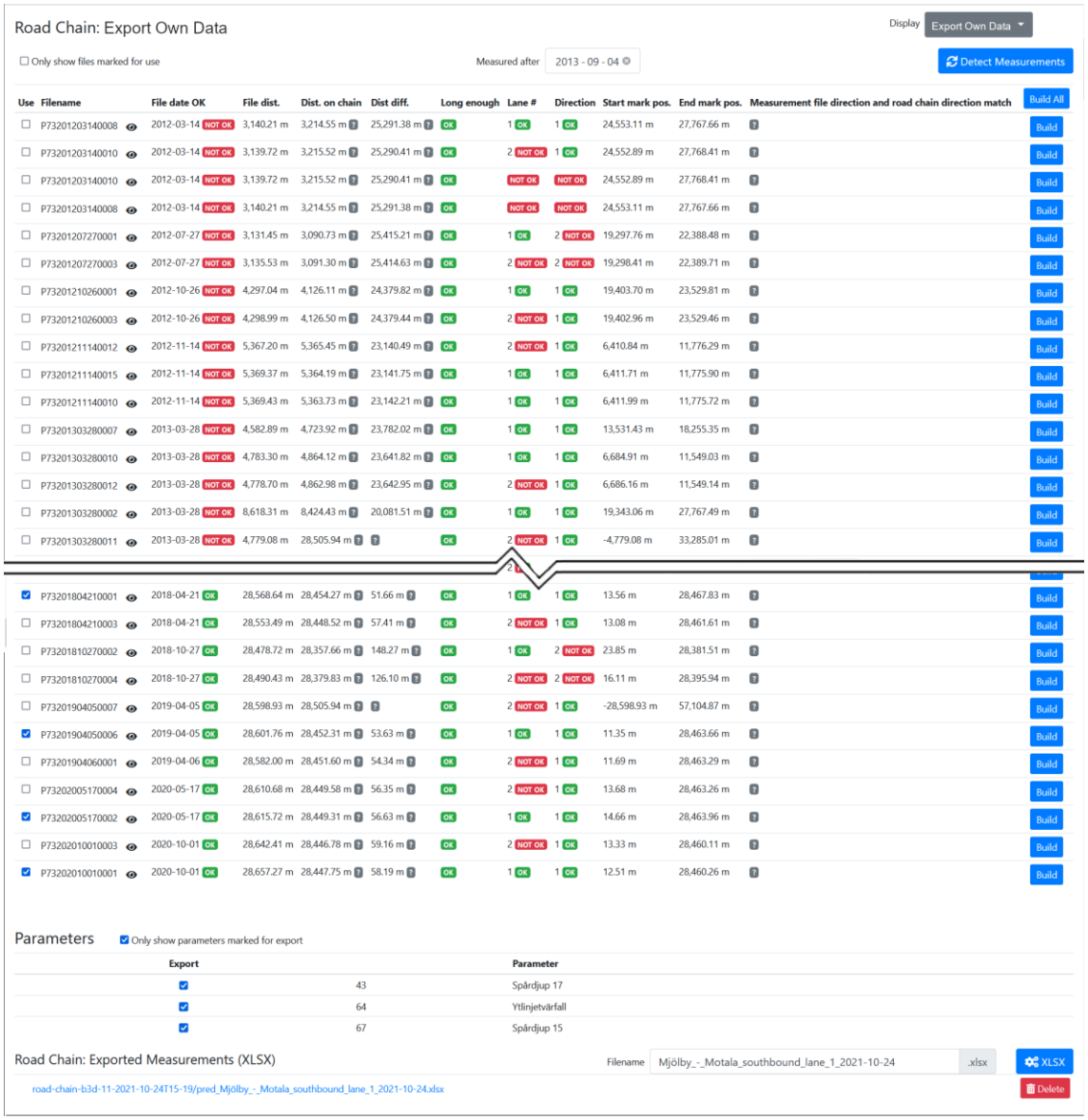
Figur 3.6.visar sidan där krav för vägen anges.

Figur 3.7. visar sidan där Trafikverkets PMSv3 data exporteras. Matchande data från Trafikverkets mätningar inhämtas automatiskt när man väljer att exportera en sträcka. I figur 3.7. framgår att 6 stycken mätningar vilka innehåller mellan 1895 och 1915 uppmätta 20 m-sträckor hittats mellan åren 2014 och 2018 för vägen i exemplet. Användaren väljer sedan vilka parametrar som ska exporteras vidare till nästa steg.



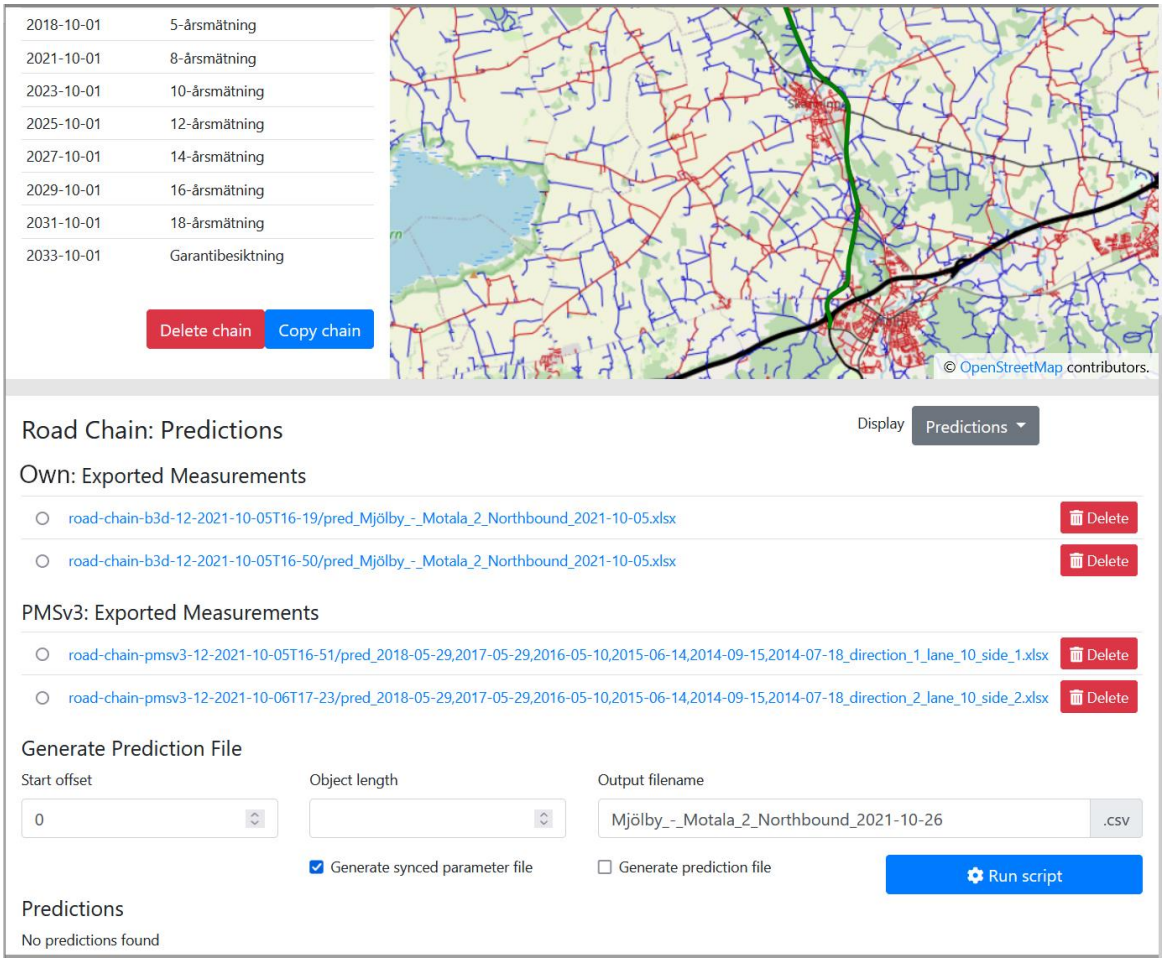
Figur 3.7. visar sidan där man kan exportera längdsynkroniserad PMSv3 data.

Motsvarande exportprocess för eget uppladdat data från vägytemätningar görs på sidan som illustreras i Figur 3.8. Filer som matchar vald vägsträcka inhämtas automatiskt och filtreras så som beskrivet i avsnittet ”Import eget mätdata” i denna rapport. Listan på matchande filer kan bli väldigt lång så listan i Figur 3.8 är nerklippt och visar endast ett urval av de mätningar som identifierats som lämpliga för export.



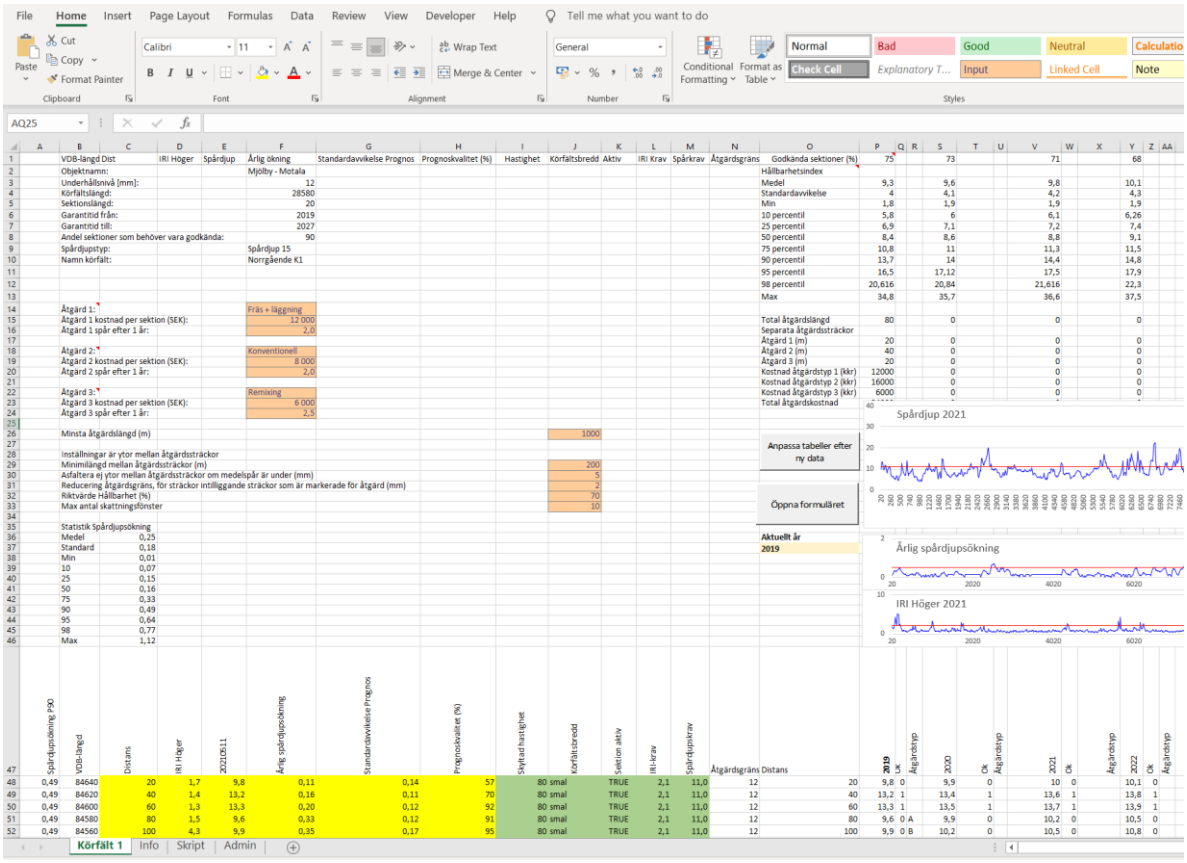
Figur 3.8. Sidan där eget uppladdat data kan exporteras.

I nästa steg, där vår tidigare exporterade data ytterligare en gång kommer läsas in och exporteras till en ny fil, kommer data synkroniseras i längdled med hög noggrannhet sedan kommer en prognos för spårdjupsökningen på 20-metersnivå skapas (jämför figur 3.9.). Den fil som skapas i detta steg kallar vi för prognosfil. Användaren kan välja att skapa en prognosfil baserad på endast Trafikverkets data, eget data, eller både och.



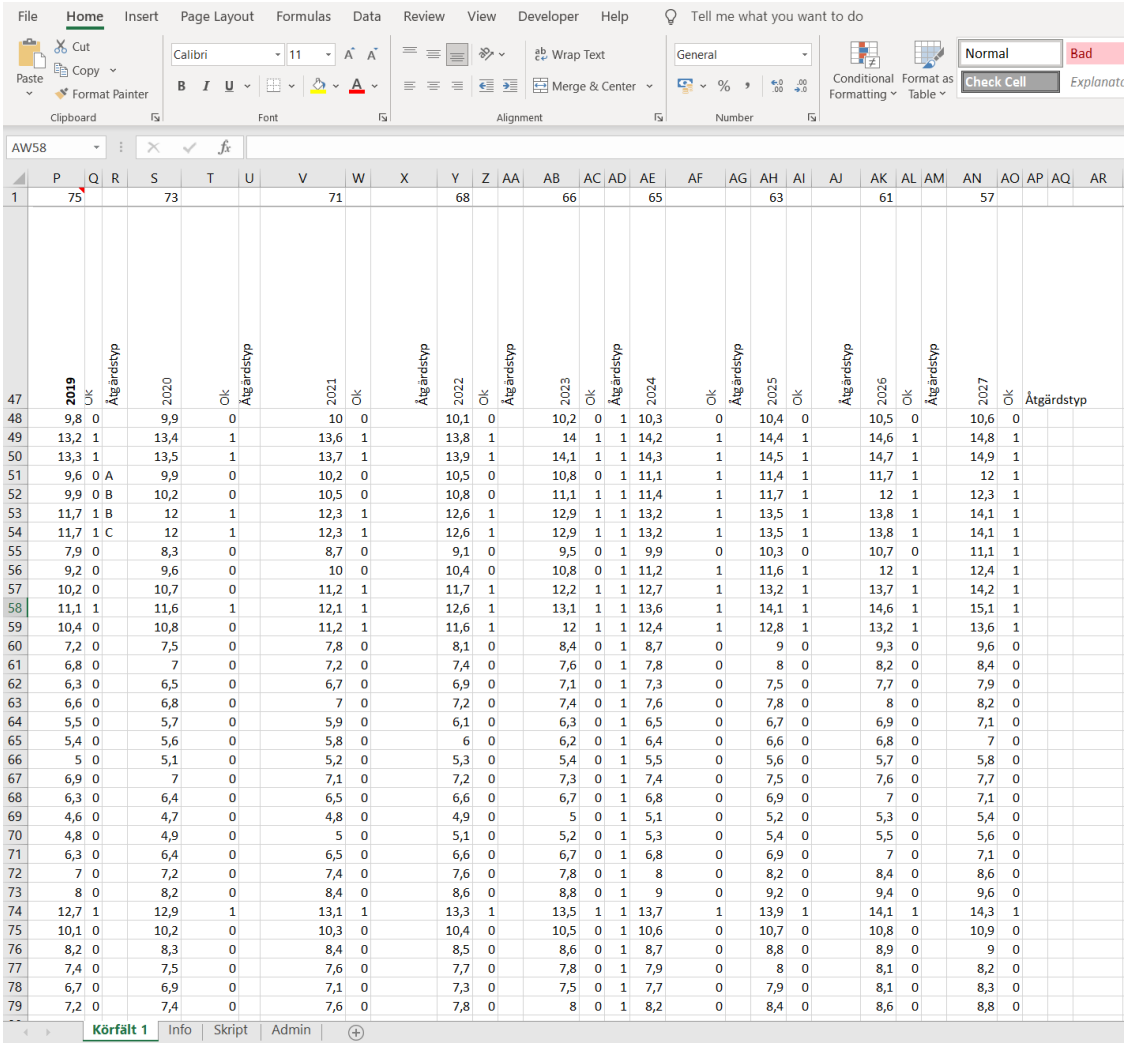
Figur 3.9. visar sidan där data från Trafikverkets PMSv3 och eller eget mätdata kan användas för att skapa en fil med prognostiserade värden för spårdjupet.

Skapade prognosfiler hamnar sedan i en lista på sidan där användaren kan klicka på dem och ladda ner den som Excelfil (figur 3.10) där själva underhållsplaneringen kan ske.



Figur 3.10 Excelfil där underhållet planeras.

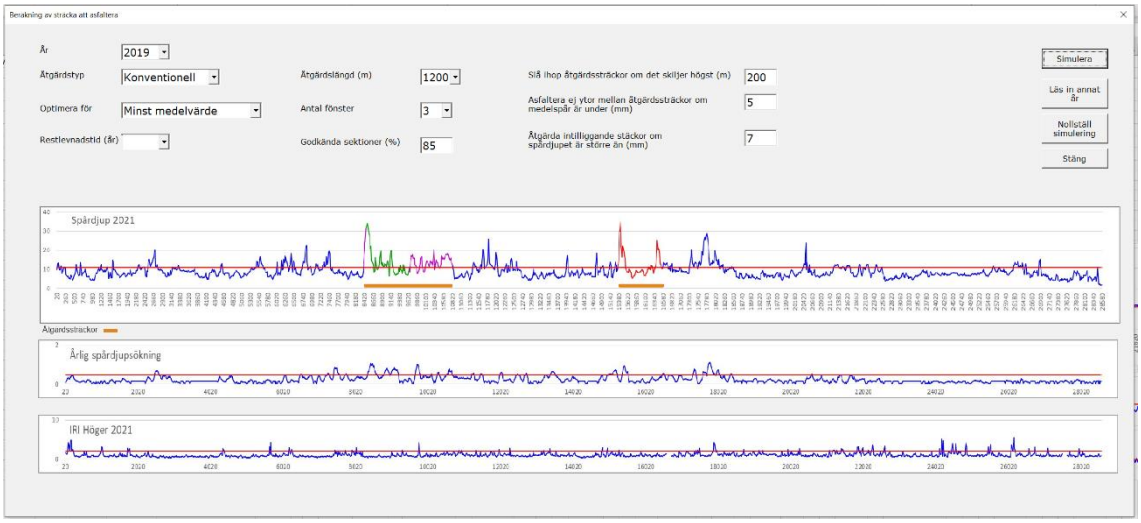
Användaren kan manuellt i tabellen i Figur 3.11 ange vilken åtgärdstyp samt vilken distans som ska åtgärdas för alla de år som kontraktet omfattar. Statistik över vägsträckan uppdateras automatiskt då åtgärder skapas, vilket gör att det blir lätt att optimera mot kontraktskrav. Man får även en överblick över kostnader för planerade beläggningsåtgärder



	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR
1	75			73			71			68			66			65			63			61			57				
47	**2019**	Ok	Åtgärdstyp	2020	Ok	Åtgärdstyp	2021	Ok	Åtgärdstyp	2022	Ok	Åtgärdstyp	2023	Ok	Åtgärdstyp	2024	Ok	Åtgärdstyp	2025	Ok	Åtgärdstyp	2026	Ok	Åtgärdstyp	2027	Ok	Åtgärdstyp		
48	9,8	0		9,9	0		10	0		10,1	0		10,2	0	1	10,3	0		10,4	0		10,5	0		10,6	0			
49	13,2	1		13,4	1		13,6	1		13,8	1		14	1	1	14,2	1		14,4	1		14,6	1		14,8	1			
50	13,3	1		13,5	1		13,7	1		13,9	1		14,1	1	1	14,3	1		14,5	1		14,7	1		14,9	1			
51	9,6	0	A	9,9	0		10,2	0		10,5	0		10,8	0	1	11,1	1		11,4	1		11,7	1		12	1			
52	9,9	0	B	10,2	0		10,5	0		10,8	0		11,1	1	1	11,4	1		11,7	1		12	1		12,3	1			
53	11,7	1	B	12	1		12,3	1		12,6	1		12,9	1	1	13,2	1		13,5	1		13,8	1		14,1	1			
54	11,7	1	C	12	1		12,3	1		12,6	1		12,9	1	1	13,2	1		13,5	1		13,8	1		14,1	1			
55	7,9	0		8,3	0		8,7	0		9,1	0		9,5	0	1	9,9	0		10,3	0		10,7	0		11,1	1			
56	9,2	0		9,6	0		10	0		10,4	0		10,8	0	1	11,2	1		11,6	1		12	1		12,4	1			
57	10,2	0		10,7	0		11,2	1		11,7	1		12,2	1	1	12,7	1		13,2	1		13,7	1		14,2	1			
58	11,1	1		11,6	1		12,1	1		12,6	1		13,1	1	1	13,6	1		14,1	1		14,6	1		15,1	1			
59	10,4	0		10,8	0		11,2	1		11,6	1		12	1	1	12,4	1		12,8	1		13,2	1		13,6	1			
60	7,2	0		7,5	0		7,8	0		8,1	0		8,4	0	1	8,7	0		9	0		9,3	0		9,6	0			
61	6,8	0		7	0		7,2	0		7,4	0		7,6	0	1	7,8	0		8	0		8,2	0		8,4	0			
62	6,3	0		6,5	0		6,7	0		6,9	0		7,1	0	1	7,3	0		7,5	0		7,7	0		7,9	0			
63	6,6	0		6,8	0		7	0		7,2	0		7,4	0	1	7,6	0		7,8	0		8	0		8,2	0			
64	5,5	0		5,7	0		5,9	0		6,1	0		6,3	0	1	6,5	0		6,7	0		6,9	0		7,1	0			
65	5,4	0		5,6	0		5,8	0		6	0		6,2	0	1	6,4	0		6,6	0		6,8	0		7	0			
66	5	0		5,1	0		5,2	0		5,3	0		5,4	0	1	5,5	0		5,6	0		5,7	0		5,8	0			
67	6,9	0		7	0		7,1	0		7,2	0		7,3	0	1	7,4	0		7,5	0		7,6	0		7,7	0			
68	6,3	0		6,4	0		6,5	0		6,6	0		6,7	0	1	6,8	0		6,9	0		7	0		7,1	0			
69	4,6	0		4,7	0		4,8	0		4,9	0		5	0	1	5,1	0		5,2	0		5,3	0		5,4	0			
70	4,8	0		4,9	0		5	0		5,1	0		5,2	0	1	5,3	0		5,4	0		5,5	0		5,6	0			
71	6,3	0		6,4	0		6,5	0		6,6	0		6,7	0	1	6,8	0		6,9	0		7	0		7,1	0			
72	7	0		7,2	0		7,4	0		7,6	0		7,8	0	1	8	0		8,2	0		8,4	0		8,6	0			
73	8	0		8,2	0		8,4	0		8,6	0		8,8	0	1	9	0		9,2	0		9,4	0		9,6	0			
74	12,7	1		12,9	1		13,1	1		13,3	1		13,5	1	1	13,7	1		13,9	1		14,1	1		14,3	1			
75	10,1	0		10,2	0		10,3	0		10,4	0		10,5	0	1	10,6	0		10,7	0		10,8	0		10,9	0			
76	8,2	0		8,3	0		8,4	0		8,5	0		8,6	0	1	8,7	0		8,8	0		8,9	0		9	0			
77	7,4	0		7,5	0		7,6	0		7,7	0		7,8	0	1	7,9	0		8	0		8,1	0		8,2	0			
78	6,7	0		6,9	0		7,1	0		7,3	0		7,5	0	1	7,7	0		7,9	0		8,1	0		8,3	0			
79	7,2	0		7,4	0		7,6	0		7,8	0		8	0	1	8,2	0		8,4	0		8,6	0		8,8	0			

Figur 3.11. Tabell över prognostiserat spårdjup samt planerade beläggningsåtgärder för varje år.

Ett VBA-program (figur 3.12.) har tagits fram med funktioner som identifierar vilka sträckor som bör läggas om. Glidande fönster simuleras över sträckan och användaren kan välja bland olika algoritmer som optimerar beläggningsunderhållet mot olika parametrar, om lägsta medelspårdjup längs hela vägsträckan eftersträvas, eller lägst antal underkända sträckor. Användaren kan även taktiskt leta efter och åtgärda sträckor som har en viss restlevnadstid.



Figur 3.12. Exempel på underhållsprogram där användaren kan simulera olika beläggningsåtgärder. Figuren visar resultatet av en simulering där användaren valt att hitta optimala distanser för 3 stycken ca 1200 m långa beläggningsåtgärder (indikeras av grön, rosa respektive röd färg) där minsta möjliga medelspårdjup på hela sträckan efter åtgärd eftersträvas.

Om de simulerade åtgärderna godkänns så sparas de till åtgärdsmatrisen i figur 3.11. Planerade åtgärder kan sedan vidarebefordras till arbetsledning eller lag som utför själva beläggningsarbetet.

SLUTSATS OCH DISKUSSION

I och med detta utvecklingsprojekt så finns en grund för nya möjligheter för konsulter och entreprenörer att förstå, analysera och skapa underhållsplaner inom entreprenadkontrakt, inte minst inom totalentreprenader med funktionskrav. En potentiell nytta med det framtagna verktyget, vilket även utgjort det största arbetsmomentet i projektet, är den geografiska databasen med mätdata, där man via ett webinterface, förutsatt att eget data är koordinatsatt, enkelt kan plocka ut data från såväl Trafikverkets PMSv3 samt eget mätdata, så att den automatiskt kan knytas till vägnätet. Detta har tidigare varit en svår och tidskrävande del vid arbeten med att ta fram fleråriga och detaljerade underhållsplaner för vägar med funktionskrav.

Vi har tagit vara på möjligheten för denna typ av system att ytterligare mogna genom att vi i detta projekt läst in vägens tillståndsmått i hög upplösning (1 m) och sedan använt modeller som historiskt troligen ansetts vara orealistiska då de är väldigt beräkningsintensiva, för att synkronisera mätningar från olika succesiva år med varandra i längdled. Denna data har högre kvalitet och ger oss ett bättre utgångsläge än tidigare när vi sedan vill skapa en underhållsplan.

En viktig fråga för oss är vilken nytta användare har av detta underhållsplaneringsverktyg och den data som genereras. Vi kallar denna nytta för effektivitet, d.v.s. till hur stor del av den genererade underhållsplanen som man i slutändan håller sig till. Om man med

verktyget tidigt i projektet genererar en underhållsplan för ett visst antal sträckor och man efter garantitidens slut inte avvikit någonting från den genererade planen så skulle effektiviteten teoretiskt ha varit 100%. För de sju exemplifierade totalentreprenaderna i detta projekt (se tabell 2.3) har effektiviteten av åtgärdsplanerna inte utvärderats men borde ligga någonstans kring 60 - 70 % för att vara jämförbart med resultat från utvärderingar av olika PMS (se Zimmerman, 2017). Vissa av entreprenaderna är t.ex. så pass nya (ny beläggning vid slutbesiktningen/garantitidens start) så det har inte behövts några beläggningsåtgärder ännu varför det naturligtvis inte går utvärdera effektiviteten. Avvikelser mellan underhållsplaner som genererats och faktiskt utförda beläggningsåtgärderna kan även beroende på externa faktorer som ofta är svåra eller till och med omöjliga att ta med i en beräkning. Denna typ av avvikelser kan in många fall förklaras av, i förväg oförutsägbara politiska, ekonomiska eller klimatrelaterade faktorer (Zimmerman, 2017). Vår bedömning är att vi delvis på grund av detta har liten eller ingen vinning i en mer automatiserad och detaljerad underhållsplan. En automatiserad och relativt detaljerad underhållsplan kan dock vara till stor hjälp för användare som snabbt och enkelt vill uppskatta mänger och kostnader för kommande år samt att den i många fall kan utgöra en bra start för fortsatt arbete med en manuellt justerad underhållsplan. Detta projekts Excelbaserade och inte allt för komplex metod med glidande fönster anses därmed ge en väl anpassad detaljnivå i underhållsplaneringen. Att göra en uppskattning av effektiviteten är som indikeras inte helt lätt och vår bedömning av den skulle definitivt kunna göras mer grundlig. Även om en sådan analys ligger bortom detta utvecklingsprojekt kan detta komma att utgöra ett kommande arbete hos enskilda intressenter.

4. REFERENSER

American Association of State Highway and Transportation Officials (AASHTO). (2012). Pavement Management Guide, Second Edition. American Association of State Highway and Transportation Officials, Washington, DC.

Hajdin, R., Botzen, M. (2014). Sectioning technique of Road Data for Pavement Management. *Forschungsprojekt VSS 2009/705 auf Antrag des Schweizerischen Verbandes der Strassen- und Verkehrsfachleute (VSS).*

Peterson D., E. (1987), Pavement management practices. National Cooperative Highway Research Program. *Transportation Research Board. Issue number 135*.

McGhee, K. (2004). Automated Pavement Distress Collection Techniques. *Academies of Sciences, Engineering, and Medicine. 2004. Washington, DC: https://doi.org/10.17226/23348.*

Flintsch, Gerardo W, McGhee, Kenneth, (2009), Quality Management of Pavement Condition Data Collection. *NCHRP Synthesis of Highway Practice. Issue number 401*.

Andrén, P., Eriksson, O. (2019), *Prognosmetoder för spårdjup och IRI*. VTI rapport 1010.

Zimmerman, K.A., Pierce, L.M. , Krstulovich. J. (2010). Pavement Management Roadmap. *Federal Highway Administration, Washington, DC. Report number FHWA-HIF-11-011.*

Zimmerman, K. A., (2017). Putting Data to Work. *National Cooperative Highway Research Program, Pavement Management Systems:. Washington, DC: The National Academies Press. https://doi.org/10.17226/24682.*

France-Mensah, J., O'Brien, W. J. 2018. Budget Allocation Models for Pavement. *Journal of Management in Engineering Vol. 34, Issue 2 (March 2018).*

Ping, W.V., Yang, Z., Gan, L., Dietrich, B (1999). Development of procedure for automated segmentation of pavement rut data. *Journal of the Transportation Research board. Issue 1 1999.*

Något att tänka på.

Om vi inte kan mäta det vi bedömer som viktigt så mäter vi vad vi kan och bedömer det som viktigt.

James Willbanks, Militär rådgivare U.S Army, 2017

5. BILAGOR

Bilaga A. Exempelkod för att skapa databaser för import av Trafikverkets data. Samt beskrivning av de olika variablerna i dataformatet.

Bilaga B. Pythonskript som matchar koordinatsatt mätdata med vägens geometri.

Bilaga C. Pythonscript som matchar och exporterar PMSv3 data i antingen CSV- eller Excelformat.

Bilaga D. Kod skrivet i programspråket R, som beskriver den automatiska identifieringen av vägsträckornas olika beläggningscykler.

BILAGA A

Exempelkod för att skapa databas

Filnamn; make_pms.sql

```
drop database if exists pms_2018;
create database pms_2018 character set latin1 collate latin1_swedish_ci;
grant all on pms_2018.* to 'andren';
grant file on *.* to 'andren';
show warnings;
```

Exempelkod för att skapa tabellerna i databasen

Filnamn; use_pms.sql

```
set sql_mode = 'TRADITIONAL';

drop table if exists avsnitt;
create table if not exists avsnitt
(
	PMSV3Id char(36) not null,			-- [01] Koppling till 'progdata'. (Varför är det inte 'ProgDataId' som används?)
	/**/GenereringsId char(36) not null,		-- [02] Fristående UUID
	AvsnittId char(36) not null,			-- [03] En UUID per "avsnitt", homogen eller generaliserad sträcka
	VagDataId char(36) null,		-- [04] Koppling till 'vagdata'
	BelaggningsDataId char(36) null,		-- [05] Koppling till 'belaggningsdata'
	FiktivBelaggningsDataId char(36) null,		-- [06] Koppling till 'fiktivbelaggningsdata'
	MatDataId char(36) null,		-- [07] Koppling till 'matdata' och 'matdatasegment'
	StrackDataTyp int null,			-- [08] Typ av avsnitt.  1: homogent avsnitt och 2: 100m-avsnitt
	/**/KommunKod int null,			-- [09] Kommunkod
	LanKod int null,			-- [10] Länskod
	StartLopandeLangd int null,			-- [11] Startposition på löpande längd för vägen
	SlutLopandeLangd int null,			-- [12] Slutposition på löpande längd för vägen
	Langd int null,				-- [13] Längd på avsnittet
	VagNummer int null,			-- [14] 4 för E4:an
	VagUndernummer int null,			-- [15] 8 för E4.08
	LankRoll int null,			-- [16] Länkroll.  1: 'Normal', 2: 'Syskon fram' 3: 'Syskon bak', 4: 'Gren'
	VagRoll int null,			-- [17] Vägroll.  1: 'Värd', 2: 'Gäst fram', (3: 'Gäst bak')
	Riktning int null,			-- [18] Mätriktning 1: 'med', 2: 'mot'
	Sida int null,				-- [19] Mätriktning 1: 'med', 2: 'mot'
	/**/Lager int null,			-- [20] X
	Korfalt int null,			-- [21] Körfält
	/**/ArRepresentativt int null,			-- [22] X
	OID nvarchar(50) null,			-- [23] Objektsidentitet
	RelStart float null,			-- [24] Relativ startposition på OID för avsnittet
	RelSlut float null,			-- [25] Relativ slutposition på OID för avsnittet
	DATF int null,				-- [26] Frândatum
	DATT int null,				-- [27] Tilldatum
	ProgHarTackningsGrad int null,			-- [28] Boolskt
	ProgHarPrognos int null,			-- [29] Boolskt
	ProgHarTrafikUppgift int null,			-- [30] Boolskt
	ProgGallandeBelaggningsDatum char(20) null,		-- [31] Datum på formen "Aug  1 1988 12:00AM"
	GeomRelStart float null,			-- [32] Flyttal [0-1)
	GeomRelSlut float null,			-- [33] Flyttal (0-1]
	Direction int null,			-- [34] Heltal 1|2
	TotalAtgardskostnadBetraktelseAr int null,	-- [35] Alltid NULL
	Vagyta float null,			-- [36] Heltal 4-1500
	Avvattningsbrist int null,			-- [37] Heltal 0|1|2
	AvvattningsbristLangd int null,			-- [38] Heltal 0-99
	FordeladAtgardYta float null,			-- [39] Flyttal
	FordeladAtgardKostnad float null,		-- [40] Flyttal
	/**/TidsversionId char(36) not null,		-- [41] Fristående UUID
	AtgardsYtaBeraknad float null,			-- [42] Flyttal
	AtgardsYtaUppmatt float null			-- [43] Flyttal
); show warnings;
select 'Loading avsnitt' as 'Loading avsnitt';
```

```
load data local infile 'R:/PeterA/DATA/TRV/PMSv3/2018_03_14/T_Avsnitt.txt' into table avsnitt fields terminated by ';' lines terminated by '\r\n' ignore 1 lines (PMSV3Id, GenereringsId, AvsnittId, VagdataId, BelaggningsDataId, FiktivBelaggningsDataId, MatDataId, StrackDataTyp, KommunKod, LanKod, StartLopandeLangd, SlutLopandeLangd, Langd, VagNummer, VagUndernummer, LankRoll, VagRoll, Riktning, Sida, Lager, Korfalt, ArRepresentativt, OID, RelStart, RelSlut, DATF, DATT, ProgHarTackningsGrad, ProgHarPrognos, ProgHarTrafikUppgift, ProgGallandeBelaggningsDatum, GeomRelStart, GeomRelSlut, Direction, TotalAtgardskostnadBetraktelseAr, Vagyta, Avvattningsbrist, AvvattningsbristLangd, FordeladAtgardYta, FordeladAtgardKostnad, TidsversionId, AtgardsYtaBeraknad, AtgardsYtaUppmatt); show warnings;
select 'Building index' as 'Building index';
alter table avsnitt add index (PMSV3Id); show warnings;
alter table avsnitt add index (VagDataId); show warnings;
alter table avsnitt add index (BelaggningsDataId); show warnings;
alter table avsnitt add index (FiktivBelaggningsDataId); show warnings;
alter table avsnitt add index (MatDataId); show warnings;
alter table avsnitt add index (VagNummer); show warnings;
alter table avsnitt add index (LanKod); show warnings;
alter table avsnitt add index (OID); show warnings;
select 'Done' as 'Done';


drop table if exists belaggningsdata;
create table if not exists belaggningsdata
(
        BelaggningsDataId char(36) not null,                -- [01] Koppling till 'avsnitt'
        /**/GenereringsId char(36) not null,                -- [02] Fristående UUID
        BelaggningsDatum date null,                         -- [03] Beläggningsdatum
        BelaggningsTyp nvarchar(50) null,           -- [04] Beläggningstyp
        AvtagTyp nvarchar(50) null,                         -- [05] Avtagning, hel eller delvis ('AH' och 'AD')
        /**/JusteradVikt nvarchar(50) null,                 -- [06] Alltid 'NULL'
        MaxStenStorlek nvarchar(50) null,           -- [07] Maximal stenstorlek
        /**/JusteradKostnad nvarchar(50) null,              -- [08] Alltid 'NULL'
        Tjocklek nvarchar(50) null,                         -- [09] Tjocklek [millimeter] på översta lagret?
        Entreprenor nvarchar(50) null,                      -- [10] Entrepenör
        SparLagning nvarchar(50) null,                      -- [11] Typ av spårlagning
        Bindemedel nvarchar(18) null,                       -- [12] Bindemedelstyp
        TillverkningsMetod nvarchar(50) null,               -- [13] X ny 2018
        UtlaggningsMetod nvarchar(50) null,                 -- [14] X ny 2018
        Kulkvarnsvarde float null,                  -- [15] Kulkvarnsvärde
        /**/Entreprenadform nvarchar(50) null,              -- [16] Alltid 'NULL'
        /**/Justering int null,                             -- [17] X
        /**/Mangd int null,                                 -- [18] X
        /**/UppmattYta int null,                    -- [19] X
        /**/Garantitid int null,                    -- [20] Garantitid [år]
        SlutbesiktningsDatum char(20) null,                 -- [21] Datum på formen "Aug  1 1988 12:00AM"
        GarantiForfaller char(20) null,                     -- [22] Datum på formen "Aug  1 1988 12:00AM"
        /**/GarantiKommentar nvarchar(50) null,             -- [23] Alltid 'NULL'
        Objektnummer nvarchar(30) null,                     -- [24]
        Atgardskostnad float null,                  -- [25]
        Finansieringskod nvarchar(34) null,                 -- [26]
        TotalObjektkostnad float null                       -- [27]
); show warnings;
select 'Loading belaggningsdata' as 'Loading belaggningsdata';
load data local infile 'R:/PeterA/DATA/TRV/PMSv3/2018_03_14/T_BelaggningsData.txt' into table belaggningsdata fields terminated by ';' lines terminated by '\n' ignore 1 lines (BelaggningsDataId, GenereringsId, BelaggningsDatum, BelaggningsTyp, AvtagTyp, JusteradVikt, MaxStenStorlek, JusteradKostnad, Tjocklek, Entreprenor, SparLagning, Bindemedel, TillverkningsMetod, UtlaggningsMetod, Kulkvarnsvarde, Entreprenadform, Justering, Mangd, UppmattYta, Garantitid, SlutbesiktningsDatum, GarantiForfaller, GarantiKommentar, Objektnummer, Atgardskostnad, Finansieringskod, TotalObjektkostnad); show warnings;
select 'Building index' as 'Building index';
alter table belaggningsdata add index (BelaggningsDataId); show warnings;
select 'Done' as 'Done';


drop table if exists fiktivbelaggningsdata;
create table if not exists fiktivbelaggningsdata
(
        FiktivBelaggningsDataId char(36) not null,  -- [01] Koppling till 'avsnitt'
        /**/GenereringsId char(36) not null,                -- [02] Fristående UUID
        BelaggningsDatum date null,                         -- [03] Fiktivt beläggningsdatum
        BelaggningsTyp char(3)                              -- [04] Alltid 'FIK'
```

```
); show warnings;
select 'Loading fiktivbelaggningsdata' as 'Loading fiktivbelaggningsdata';
load data local infile 'R:/PeterA/DATA/TRV/PMSv3/2018_03_14/T_FiktivBelaggningData.txt' into table fiktivbelaggningsdata fields terminated by ';' lines terminated by '\r\n' ignore 1 lines (FiktivBelaggningsDataId, GenereringsId, BelaggningsDatum, BelaggningsTyp); show warnings;
select 'Building index' as 'Building index';
alter table fiktivbelaggningsdata add index (FiktivBelaggningsDataId); show warnings;
select 'Done' as 'Done';


drop table if exists matdata;
create table if not exists matdata
(
	MatDataId char(36) not null,			-- [01] Koppling till 'avsnitt' och 'matdatasegment'
	/**/GenereringsId char(36) not null,		-- [02] Fristående UUID
	/**/MatDataTyp int not null,			-- [03] Alltid '1' (för profilometer?)
	RST_MASK int null,				-- [04] Alltid NULL
	/**/TKVAL int null,				-- [05] Kontrollkod för mätning (alltid 'NULL')
	RSTDatum date null,				-- [06] Mätdatum
	/**/MatSystem int null,				-- [07] Alltid '0'
	/**/MatKategori nvarchar(255) null,		-- [08] Mätkategori. År samt mätområde, t.ex.\ \texttt{98VM04} (stämmer EJ för 2015)
	/**/PeriodDatum date null,			-- [09] Perioddatum. Sista mätdatum för mätområde
	ProgIngarIPrognos int				-- [10] Boolskt
) character set latin1 collate latin1_swedish_ci; show warnings;
select 'Loading matdata' as 'Loading matdata';
load data local infile 'R:/PeterA/DATA/TRV/PMSv3/2018_03_14/T_MatData.txt' into table matdata fields terminated by ';' lines terminated by '\r\n' ignore 1 lines (MatDataId, GenereringsId, MatDataTyp, RST_MASK, TKVAL, RSTDatum, MatSystem, MatKategori, PeriodDatum, ProgIngarIPrognos); show warnings;
select 'Building index' as 'Building index';
alter table matdata add index (MatDataId); show warnings;
select 'Done' as 'Done';


drop table if exists matdatasegment;
create table if not exists matdatasegment
(
	MatDataSegmentId char(36) not null,		-- [01] Fristående UUID
	/**/GenereringsId char(36) not null,		-- [02] Fristående UUID
	MatDataId char(36) not null,			-- [03] Koppling till 'avsnitt' och 'matdata'
	DelstrackaSegmentId char(36) not null,		-- [04] Koppling till 'delstrackasegment'
	RSTSektA int null,				-- [05] Startposition i "mätning" för 20m-segmentet
	RSTSektB int null,			-- [06] Slutposition i "mätning" för 20m-segmentet
	RSTLangd int null,				-- [07] Längd på 20m-segmentet (oftast 20m)
	StartLopandeLangd int null,			-- [08] Startposition i löpande längd för 20m-segmentet
	SlutLopandeLangd int null,			-- [09] Slutposition i löpande längd för 20m-segmentet
	AvvattningsBrist smallint null,			-- [10] 0-5 (0: Ingen indikation; 1: Ej beräknad eller grusväg; 2: Snabb nedbrytning Spår; 3: Snabb nedbrytning IRI; 4: Höga värden Spår/IRI; 5: Höga värden kantdjup)
	AvvattningsBristDetalj smallint null		-- [11] X
); show warnings;
select 'Loading matsegmentdata' as 'Loading matsegmentdata';
load data local infile 'R:/PeterA/DATA/TRV/PMSv3/2018_03_14/T_MatDataSegment.txt' into table matdatasegment fields terminated by ';' lines terminated by '\r\n' ignore 1 lines (MatDataSegmentId, GenereringsId, MatDataId, DelstrackaSegmentId, RSTSektA, RSTSektB, RSTLangd, StartLopandeLangd, SlutLopandeLangd, AvvattningsBrist, AvvattningsBristDetalj); show warnings;
select 'Building index' as 'Building index';
alter table matdatasegment add index (MatDataId); show warnings;
alter table matdatasegment add index (DelstrackaSegmentId); show warnings;
select 'Done' as 'Done';


drop table if exists delstrackasegment;
create table if not exists delstrackasegment
(
	DelstrackaSegmentId char(36) not null,		-- [01] Koppling till 'matdatasegment'. UUID per 20m-segment
	DelstrackaId char(36) not null,			-- [02] Fristående UUID. UUID per "mätning"?
	StartRelAvstand float not null,			-- [03] Relativ startposition på OID för 20m-segmentet
	SlutRelAvstand float not null,			-- [04] Relativ slutposition på OID för 20m-segmentet
	StartRelGeomAvstand float not null,		-- [05] Relativ startposition på OID för 20m-segmentet
```

```
	SlutRelGeomAvstand float not null,		-- [06] Relativ slutposition på OID för 20m-segmentet
	SegmentLangd float not null,		-- [07] Längd på segmentet.  Normalt 20m
	SegmentIndex int null,		-- [08] Räknare på segmentet.  Startar med '0'.  Ett-till-ett-relation med
'DelstrackaSegmentId'
	x0257 int null,		-- [09] Segmentets längd
	x0258 int null,		-- [10] Distans
	x0260 int null,		-- [11] Mäthastighet
	x0859 float null,		-- [12] MPD, höger
	x0860 float null,		-- [13] MPD, mitt
	x0863 float null,		-- [14] MPD, vänster
	x1025 float null,		-- [15] Spårdjup, max (17 lasrar)
	x1026 float null,		-- [16] Spårdjup, vänster (17 lasrar)
	x1036 float null,		-- [17] Tvärprofil, laser 02
	x1037 float null,		-- [18] Tvärprofil, laser 03
	x1038 float null,		-- [19] Tvärprofil, laser 04
	x1039 float null,		-- [20] Tvärprofil, laser 05
	x1040 float null,		-- [21] Tvärprofil, laser 06
	x1041 float null,		-- [22] Tvärprofil, laser 07
	x1042 float null,		-- [23] Tvärprofil, laser 08
	x1043 float null,		-- [24] Tvärprofil, laser 09
	x1044 float null,		-- [25] Tvärprofil, laser 10
	x1045 float null,		-- [26] Tvärprofil, laser 11
	x1046 float null,		-- [27] Tvärprofil, laser 12
	x1047 float null,		-- [28] Tvärprofil, laser 13
	x1048 float null,		-- [29] Tvärprofil, laser 14
	x1049 float null,		-- [30] Tvärprofil, laser 15
	x1050 float null,		-- [31] Tvärprofil, laser 16
	x1105 float null,		-- [32] Spårdjup, höger (17 lasrar)
	x1287 float null,		-- [33] IRI, höger (mm/m)
	x1310 float null,		-- [34] IRI, vänster
	x1541 int null,		-- [35] Kurvatur (10000/m)
	x1545 float null,		-- [36] Ytlinjetvärfall (\%)
	x1547 float null,		-- [37] Backighet (\%)
	x3000 float null,		-- [38] Regressionstvärfall
	x3020 double null,		-- [39] Koordinat, N (SWEREF99TM)
	x3021 double null,		-- [40] Koordinat, E (SWEREF99TM)
	x3022 float null,		-- [41] Koordinat, A (RH70)
	x3023 float null,		-- [42] Koordinat, felterm (Radiellt medelfel för position)
	x8025 float null,		-- [43] Spårdjup, max (15 lasrar)
	x8026 float null,		-- [44] Spårdjup, vänster (15 lasrar)
	x8000 float null,		-- [45] Kantdjup
	x8101 float null,		-- [46] Spårarea
	x8102 float null,		-- [47] Stödpunktsavstånd (mm)
	x8103 float null,		-- [48] Lutningsförändring (\%)
	x8104 float null,		-- [49] Vattenarea (dm\textsuperscript{2})
	x8110 float null,		-- [50] Spårbredd, vänster (mm)???
	x8106 float null,		-- [51] Spårbredd, höger (mm)
	x8107 float null,		-- [52] Spårbottenavstånd (mm)
	x8201 float null,		-- [53] Lokal ojämnhet
	x8010 float null,		-- [54] Megatextur vänster (20m)
	x8011 float null,		-- [55] Megatextur höger (20m)
	x8105 float null,		-- [56] Spårdjup, höger (15 lasrar) ???
	x8301 float null,		-- [57]
	x8302 float null,		-- [58]
	x8310 float null,		-- [59]
	x8311 float null,		-- [60]
	x8312 float null,		-- [61]
	x8320 float null,		-- [62]
	x8321 float null		-- [63]
); show warnings;
select 'Loading delstrackasegment' as 'Loading delstrackasegment';
load data local infile 'R:/PeterA/DATA/TRV/PMSv3/2018_03_14/T_VymDelstrackaSegment.txt' into table delstrackasegment fields
terminated by ';' lines terminated by '\r\n' ignore 1 lines (DelstrackaSegmentId, DelstrackaId, StartRelAvstand, SlutRelAvstand,
StartRelGeomAvstand, SlutRelGeomAvstand, SegmentLangd, SegmentIndex, x0257, x0258, x0260, x0859, x0860, x0863, x1025,
x1026, x1036, x1037, x1038, x1039, x1040, x1041, x1042, x1043, x1044, x1045, x1046, x1047, x1048, x1049, x1050, x1105, x1287,
x1310, x1541, x1545, x1547, x3000, x3020, x3021, x3022, x3023, x8025, x8026, x8000, x8101, x8102, x8103, x8104, x8110, x8106,
x8107, x8201, x8010, x8011, x8105, x8301, x8302, x8310, x8311, x8312, x8320, x8321); show warnings;
select 'Building index' as 'Building index';
```

```
alter table delstrackasegment add index (DelstrackaSegmentId); show warnings;
select 'Done' as 'Done';


drop table if exists progdata;
create table if not exists progdata
(
	ProgDataId char(36) not null,			-- [01] X
	PMSV3Id char(36) not null,			-- [02] Koppling till 'avsnitt'
	GenereringsId char(36) not null,		-- [03] Fristående UUID
	PrognosDatum date not null,			-- [04] Datum för prognostiserat värde
	PrognosTidpunkt int not null,			-- [05] 1 för 2016-07-01, 2 för 2017-07-01, etc
	IRI_MV float null,				-- [06] Prognostiserat IRI-värde
	Spardjup_MV float null,				-- [07] Prognostiserat spårdjupsvärde
	Kantdjup_MV float null,				-- [08] Prognostiserat kantdjupsvärde
	AvvikerFranUs tinyint(1) null,			-- [09] X (Us = underhållstandard)
	AvvikerFranUsIRI tinyint(1) null,		-- [10] X
	AvvikerFranUsSparDjup tinyint(1) null,		-- [11] X
	AvvikerFranUsKantDjup tinyint(1) null,		-- [12] X
	ProgIRIMetod int null,				-- [13] Antal punkter i prognos.
	ProgIRINedbrytningsTakt float null,		-- [14] Förändring per år.
	ProgSpardjupMetod int null,			-- [15] Antal punkter i prognos.
	ProgSpardjupNedbrytningsTakt float null,	-- [16] Förändring per år.
	ProgKantdjupMetod int null,			-- [17] Antal punkter i prognos.
	ProgKantdjupNedbrytningsTakt float null		-- [18] Förändring per år.
); show warnings;
select 'Loading progdata' as 'Loading progdata';
load data local infile 'R:/PeterA/DATA/TRV/PMSv3/2018_03_14/T_ProgData.txt' into table progdata fields terminated by ';' lines terminated by '\r\n' ignore 1 lines (ProgDataId, PMSV3Id, GenereringsId, PrognosDatum, PrognosTidpunkt, IRI_MV, Spardjup_MV, Kantdjup_MV, AvvikerFranUs, AvvikerFranUsIRI, AvvikerFranUsSparDjup, AvvikerFranUsKantDjup, ProgIRIMetod, ProgIRINedbrytningsTakt, ProgSpardjupMetod, ProgSpardjupNedbrytningsTakt, ProgKantdjupMetod, ProgKantdjupNedbrytningsTakt); show warnings;
select 'Building index' as 'Building index';
alter table progdata add index (PMSV3Id); show warnings;
select 'Done' as 'Done';


drop table if exists vagdata;
create table if not exists vagdata
(
	VagDataId char(36) not null,			-- [01] Koppling till 'avsnitt'
	/**/GenereringsId char(36) not null,		-- [02] Fristående UUID
	Barighet int null,				-- [03] Bärighetsklass: 1, 2, eller 3
	DriftOmrade int null,				-- [04] Nummerkod för driftomrade
	Hastighet int null,				-- [05] Hastighetsbegränsning [km/h]
	/**/HuvudVagGods nvarchar(50) null,		-- [06] 'NULL'
	LeveransKvalitetDoU nvarchar(50) null,		-- [07] Leveranskvalitet DoU.  1: 'Storstadsvägar', 2: 'Övriga stamvägar', 3: 'Pendlings servicevägar', 4: 'Övriga NRL vägar', 5: 'Övriga lågtrafikerade vägar'
	NRLKod int null,				-- [08] Näringslivets ... 1--12
	Slitlager int null,				-- [09] Slitlager: 1: 'Bitumen', 2: 'Oljegrus', 3: 'Grus', 4: 'Sten', 5: 'Betong', 6: 'Y1G', 7: 'Försegling'
	/**/TERNLankId nvarchar(50) null,		-- [10] 'NULL'
	/**/TjalRestriktionFrom nvarchar(50) null,	-- [11] 'NULL'
	/**/TjalRestriktionTom nvarchar(50) null,	-- [12] 'NULL'
	Trafik int null,				-- [13] ÅDT
	TrafikTung int null,				-- [14] ÅDT, tung trafik
	VagBredd float null,				-- [15] Vägbredd [m]
	/**/VagHallare int null,			-- [16] Alltid '1'
	VagKategori int null,				-- [17] Vägkategori. 1: 'Europaväg', 2: 'Riksväg', 3: 'Primär länsväg', 4: 'Sekundär länsväg', 5: 'Tertiär länsväg', 6: 'SoT'
	VagTyp int null,				-- [18] Vägtyp.  1: 'Motorväg', 2: 'Motortrafikled', 3: 'Motortrled mfri', 4: '4-fältsväg', 5: 'Vanlig väg', 6: 'Vanlig väg mfri'
	Vinter2003 int null,				-- [19] Standardklass enligt ATB-Vinter 2003
	VagNybyggnad2009 int null,			-- [20] Nybyggnadsår (2009?)
	Korfaltsbeskrivning int null,			-- [21] 1--4: '1+1'; '2+1'; '2+2'; 'Ingen beskrivning'
	TrafikMatar int null,				-- [22] 'YYYYMM'
	TrafikMatmetod int null				-- [23] 1--4
); show warnings;
```

```
select 'Loading vagdata' as 'Loading vagdata';
load data local infile 'R:/PeterA/DATA/TRV/PMSv3/2018_03_14/T_Vagdata.txt' into table vagdata fields terminated by ';' lines terminated by '\r\n' ignore 1 lines (VagDataId, GenereringsId, Barighet, DriftOmrade, Hastighet, HuvudVagGods, LeveransKvalitetDoU, NRLKod, Slitlager, TERNLankId, TjalRestriktionFrom, TjalRestriktionTom, Trafik, TrafikTung, VagBredd, VagHallare, VagKategori, VagTyp, Vinter2003, VagNybyggnad2009, Korfaltsbeskrivning, TrafikMatar, TrafikMatmetod); show warnings;
select 'Building index' as 'Building index';
alter table vagdata add index (VagDataId); show warnings;
select 'Done' as 'Done';
```

BILAGA B

In [1]:

```
import math
import re
import ipyleaflet
import pyproj
import shapely
import shapely.geometry
import shapely.ops
import shapely.wkt

calculations_crs = pyproj.CRS('EPSG:3006')
wgs84 = pyproj.CRS('EPSG:4326')
```

In [2]:

```
wgs84_to_sweref = pyproj.Transformer.from_crs(wgs84, calculations_crs, always_xy=True).transform
def ewkt_to_linestring(ewkt: str) -> shapely.geometry.linestring.LineString:
    m = re.search('^SRID=(?P<epsg_id>\d+);(?P<wkt>.*)$', ewkt, flags=re.DOTALL)
    assert m
    input_crs = pyproj.CRS(f'EPSG:{m.group("epsg_id")}')
    linestring = shapely.wkt.loads(m.group('wkt'))
    project = pyproj.Transformer.from_crs(input_crs, wgs84, always_xy=True).transform
    return shapely.ops.transform(project, linestring)
```

In [3]:

```
ewkt_measurement = "SRID=4326;LINESTRING(
55.7896258260738,13.4356604675448 55.7892298070083,13.4346402850548 55.7889338096532,13.4335602244034
55.7886204337222,13.4325911463932 55.7883537192146,13.4314376311079 55.7880362245609,13.4304352993493
55.7877749906264,13.4292935993877 55.7874774326781,13.4282214778518 55.7872133027255,13.4278619015792
55.7871297313753,13.4271397670478 55.7869619003024,13.4260446619186 55.7867325313941,13.425481617476
55.7866258225413,13.4254541233029 55.7866202672017,13.425388298204 55.7866070755114,13.425388298204
55.7866070755114,13.4253855347239 55.7866065216947,13.4253855347239 55.7866065216947,13.4252860783671
55.7865865900399,13.4251171928859 55.7865528030262,13.4249480475849 55.7865191382603,13.4247785750877
55.7864852893367,13.4246071745143 55.7864510110756,13.4244345396107 55.7864168964831,13.4242875908577
55.7863884144423)"
```

Större delen av positionsdatat ovan är bortklippt för att göra bilagan mer överblickbar

```
ls_measurement = ewkt_to_linestring(ewkt_measurement)
ls_roadchain = ewkt_to_linestring(ewkt_roadchain)
```

In [4]:

```
m = ipyleaflet.Map(basemap=ipyleaflet.basemaps.OpenStreetMap.Mapnik, center=ls_roadchain.centroid.coords[0][::-1], zoom=13)
m.add_layer(ipyleaflet.WKTLayer(wkt_string=ls_roadchain.wkt, style={'color':'#00FF00'})) # green
m.add_layer(ipyleaflet.WKTLayer(wkt_string=ls_measurement.wkt, style={'color':'#0000FF'})) # blue
display(m)
```

In [5]:

```
def relate_point(linestring: shapely.geometry.linestring.LineString, reference_linestring: shapely.geometry.linestring.LineString, rel_point_position_on_linestring_metres):
    d = 1.0
    x0 = max(0.0, rel_point_position_on_linestring_metres - d/2)
    x2 = min(linestring.length, rel_point_position_on_linestring_metres + d/2)
    x1 = x0/2 + x2/2
```

```
    point0 = linestring.interpolate(x0)
    point1 = linestring.interpolate(x1)
    point2 = linestring.interpolate(x2)
    projected0 = reference_linestring.project(point0)
    projected1 = reference_linestring.project(point1)
    projected2 = reference_linestring.project(point2)

    path_relationship_coefficient = (projected2 - projected0)/(x2-x0)
    # If close to 1, paths go in the same direction
    # If close to -1, paths go in opposite direction
    # If between -0.95 and 0.95, paths don't match

    threshold = 0.95

    if path_relationship_coefficient > threshold:
        simplified_path_relationship_coefficient = 1
    elif path_relationship_coefficient < (-1) * threshold:
        simplified_path_relationship_coefficient = -1
    else:
        simplified_path_relationship_coefficient = 0

    return (x1, projected1, simplified_path_relationship_coefficient, path_relationship_coefficient)

In [6]:
ls_measurement_sweref = shapely.ops.transform(wgs84_to_sweref, ls_measurement)
ls_roadchain_sweref = shapely.ops.transform(wgs84_to_sweref, ls_roadchain)

In [13]:
l = list(map(lambda x: relate_point(ls_measurement_sweref, ls_roadchain_sweref, x), range(0,
math.floor(ls_measurement_sweref.length))))

def to_intervals(xs_and_projected_and_path_relationship_coefficients):
    intervals = []
    current = None
    for (x, projected, simplified_path_relationship_coefficient, path_relationship_coefficient) in
xs_and_projected_and_path_relationship_coefficients:
        if current and current['path_relationship_coefficient'] == path_relationship_coefficient:
            current['end_x'] = x
            current['end_projected'] = projected
        else:
            if current:
                intervals.append(current)
            current = {
                'path_relationship_coefficient': path_relationship_coefficient,
                'start_x': x,
                'end_x': x,
                'start_projected': projected,
                'end_projected': projected
            }
    if current:
        intervals.append(current)
    return intervals
intervals = to_intervals(l)

In [14]:
def filter_intervals(intervals):
    result = []
    i = 0
    while i < len(intervals):
```

```
    if i > 0 and i+1 < len(intervals) and intervals[i]['path_relationship_coefficient'] == 0 and intervals[i-1]['path_relationship_coefficient'] == intervals[i+1]['path_relationship_coefficient'] and abs(intervals[i]['start_x'] - intervals[i]['end_x']) < 10:
        result[-1]['end_x'] = intervals[i+1]['end_x']
        result[-1]['end_projected'] = intervals[i+1]['end_projected']
        i = i + 2
      else:
       result.append(intervals[i])
       i = i + 1
  return result


# 1. pick longest interval
# 2. calculate:
#   - file start relative to chain start = ? = x of first point where abs(relation) > 0.99
#   - file end relative to chain end = ? = x of last point where abs(relation) > 0.99
#   - distance on chain = ? = abs ( file start relative to chain start - file end relative to chain end )
#   - distance diff = road chain length - distance on chain

def calculate_relative_positions(intervals, road_chain_length):
  longest_interval = None
  longest_interval_length = 0
  for interval in intervals:
    length = abs(interval['start_projected'] - interval['end_projected'])
    if length > longest_interval_length and interval['path_relationship_coefficient'] != 0:
      longest_interval_length = length
      longest_interval = interval
  file_length = intervals[-1]['end_x']
  if longest_interval['start_projected'] < longest_interval['end_projected']: # Forward direction
    file_start_relative_to_chain_start = longest_interval['start_projected'] - longest_interval['start_x']
    file_end_relative_to_chain_start = longest_interval['end_projected'] + file_length - longest_interval['end_x']
  else: # Reversed
    file_start_relative_to_chain_start = longest_interval['start_projected'] + longest_interval['start_x']
    file_end_relative_to_chain_start = longest_interval['end_projected'] - file_length + longest_interval['end_x']
  distance_on_chain = abs(longest_interval['start_projected'] - longest_interval['end_projected'])
  distance_diff = abs(road_chain_length - distance_on_chain)

  return {'file_start_relative_to_chain_start': file_start_relative_to_chain_start, 'file_end_relative_to_chain_start': file_end_relative_to_chain_start, 'distance_on_chain': distance_on_chain, 'distance_diff': distance_diff}
calculate_relative_positions(filter_intervals(intervals), ls_roadchain_sweref.length)
filter_intervals(intervals)

Out[14]:
[{'path_relationship_coefficient': 0.9976349661938002,
  'start_x': 0.25,
  'end_x': 0.25,
  'start_projected': 62.68460887603007,
  'end_projected': 62.68460887603007},
 {'path_relationship_coefficient': 0.9977360007703311,
  'start_x': 1.0,
  'end_x': 1.0,
  'start_projected': 63.43283510095853,
  'end_projected': 63.43283510095853},
{'path_relationship_coefficient': 0.9999997422235083,
  'start_x': 997.0,
  'end_x': 997.0,
  'start_projected': 1059.171529265861,
  'end_projected': 1059.171529265861},
 {'path_relationship_coefficient': 0.9999996257097337,
```

```
 'start_x': 998.0,
 'end_x': 998.0,
 'start_projected': 1060.1715289880333,
 'end_projected': 1060.1715289880333},
{'path_relationship_coefficient': 0.9999993926267052,
 'start_x': 999.0,
 'end_x': 999.0,
 'start_projected': 1061.1715284874333,
 'end_projected': 1061.1715284874333},
...]
```

Merparten av utdatat ovan är bortklippt för att göra bilagan mer överblickbar

In [15]:

```
def v2(ls_roadchain_sweref, ls_measurement_sweref):
    start_point = relate_point(ls_measurement_sweref, ls_roadchain_sweref, 0)
    end_point = relate_point(ls_measurement_sweref, ls_roadchain_sweref, ls_measurement_sweref.length)
    print(start_point)
    print(end_point)
    length_over_road_chain = start_point[1] - end_point[1]
    length_over_measurement = start_point[0] - end_point[0]
    print(length_over_road_chain, length_over_measurement, (length_over_measurement-
length_over_road_chain)/length_over_road_chain)

v2(ls_roadchain_sweref, ls_measurement_sweref)
```

```
(0.25, 62.68460887603007, 1, 0.9976349661938002)
(5466.063408216515, 5519.782362398962, 1, 0.999800012776177)
-5457.097753522931 -5465.813408216515 0.001597122699141201
```

BILAGA C

```
import pandas as pd
import psycopg2
import psycopg2.extras
import re
from typing import Dict, List, Tuple

from .util import dict_extract_keys
from .util import is_valid_date
from .parameters import parameters

def get_measurements_from_db(dbconn, oids, direction, lane, side, dates, parameters):
  available_parameters = set([
    257,
    258,
    260,
    859,
    860,
    863,
    1025,
    1026,
    1036,
    1037,
    1038,
    1039,
    1040,
    1041,
    1042,
    1043,
    1044,
    1045,
    1046,
    1047,
    1048,
    1049,
    1050,
    1105,
    1287,
    1310,
    1541,
    1545,
    1547,
    3000,
    3020,
    3021,
    3022,
    3023,
    8025,
    8026,
    8000,
    8101,
    8102,
    8103,
    8104,
    8110,
    8106,
    8107,
    8201,
    8010,
```

```
        8011,
        8105,
        8301,
        8302,
        8310,
        8311,
        8312,
        8320,
        8321
    ])
    parameters = list(available_parameters.intersection([257] + list(map(int, parameters))))

    query = """SELECT DISTINCT ON(mds.part_stretch_id)
      s.p_oid AS oid,
      s.section_id,
      s.measurement_data_id,
      s.direction,
      s.side,
      s.lane,
      md.measurement_category,
      md.period_date,
      vps.relative_geom_distance_start,
      vps.relative_geom_distance_end,"""
    query = query + ",\n".join(map(lambda param: "vps.param_{0} as param_{0}".format(param), parameters))
    query = query + """
      FROM pmsv3_section s
      JOIN pmsv3_measurement_data md ON md.measurement_data_id = s.measurement_data_id
      RIGHT JOIN pmsv3_measurement_data_segment mds ON mds.measurement_data_id = s.measurement_data_id
      LEFT JOIN pmsv3_vym_part_stretch vps ON vps.vym_part_stretch_id = mds.part_stretch_id
      """

    query_params_for_conditions = ()
    query_conditions = []

    query_conditions = query_conditions + ['s.p_oid = ANY(%s)']
    query_params_for_conditions = query_params_for_conditions + (oids,)

    if direction is not None:
      query_conditions = query_conditions + ['s.direction = %s']
      query_params_for_conditions = query_params_for_conditions + (direction,)
    if lane is not None:
      query_conditions = query_conditions + ['s.lane = %s']
      query_params_for_conditions = query_params_for_conditions + (lane,)
    if side is not None:
      query_conditions = query_conditions + ['s.side = %s']
      query_params_for_conditions = query_params_for_conditions + (side,)
    if dates is not None and len(dates) > 0:
      query_conditions = query_conditions + ['md.period_date = ANY(%s :: date[])']
      query_params_for_conditions = query_params_for_conditions + (dates,)


    query = query + """ WHERE """ + (" AND ".join(query_conditions))

    with dbconn.cursor(cursor_factory = psycopg2.extras.RealDictCursor) as cur:
      cur.execute(query, query_params_for_conditions)
      return cur.fetchall()

def inside_interval(a0, a1, b0, b1):
    """Check whether [b0, b1] is inside [a0, a1]"""
```

```
    if a0 > a1:
        c = a0
        a0 = a1
        a1 = c

    if b0 > b1:
        c = b0
        b0 = b1
        b1 = c

    return b0 >= a0 and b1 <= a1

def position_measurement_values_relative_to_road_chain(road_links, measurement_values):
    metres_per_rel_dist = list(map(lambda road_link: abs(road_link['chainEnd'] - road_link['chainStart']) / abs(road_link['oidRelEnd']
- road_link['oidRelStart']), road_links))

    for mv in measurement_values:
        m_link_idx = None
        m_start = None
        m_end = None
        for link_idx in range(0, len(road_links)):
            road_link = road_links[link_idx]
            link_oid = road_link['oid']
            link_rel_start = road_link['oidRelStart']
            link_rel_end = road_link['oidRelEnd']
            if str(link_oid) == mv['oid'] and inside_interval(link_rel_start, link_rel_end, mv['relative_geom_distance_start'],
mv['relative_geom_distance_end']):
                m_link_idx = link_idx
                m_start = road_link['chainStart'] + abs(mv['relative_geom_distance_start'] - link_rel_start) * metres_per_rel_dist[link_idx]
                m_end = road_link['chainStart'] + abs(mv['relative_geom_distance_end'] - link_rel_start) * metres_per_rel_dist[link_idx]
                continue
        mv['linkIdx'] = m_link_idx
        mv['chainStart'] = m_start
        mv['chainEnd'] = m_end
    return filter(lambda mv: mv['linkIdx'] is not None, measurement_values)

def validate_and_filter_dates(dates):
    dates = str(dates)
    if len(dates) < 8:
        return []

    dates = dates.split(',')
    dates = list(filter(is_valid_date, dates))
    return dates


def measurement_data_for_road_chain(dbconn, road_links, direction, lane, side, dates, parameters):
    dates = validate_and_filter_dates(dates)
    oids = list(set(map(lambda road_link: road_link['oid'], road_links)))
    data_from_db = get_measurements_from_db(dbconn, oids, direction, lane, side, dates, parameters)
    measurement_values = position_measurement_values_relative_to_road_chain(road_links, data_from_db)
    measurement_values = sorted(measurement_values, key=lambda mv: mv['chainStart'])

    keys_for_output = ('oid', 'chainStart', 'chainEnd', 'direction', 'lane',
        'side', 'direction', 'linkIdx') + tuple('param_{0}'.format(param) for param in parameters)

    return list(map(lambda mv: {**dict_extract_keys(mv, keys_for_output), **{
        'measurementCategory': mv['measurement_category'],
        'measurementDataId': mv['measurement_data_id'],
```

```
      'periodDate': mv['period_date'].isoformat(),
      'sectionId': mv['section_id']
    }}, measurement_values))



def measurement_data_for_road_chain_to_xlsx(data, xlsx_path: str):
    """Converts `List[{"chainStart": 123, "chainEnd": 456, "periodDate": '2020-10-20', param_xyz: 123, param_uvw: 456}]` to an XLSX file."""
    def _extract_param_indices_and_names(df: pd.DataFrame) -> List[Tuple[int, str]]:
        param_indices = []
        param_names = []
        for col in df.columns:
            m = re.match(r'param_(?P<param_index>\d+)', col)
            if m is not None:
                param_index = int(m['param_index'])
                param_indices.append(param_index)
                param_names.append(parameters()[param_index]['name'] if param_index in parameters() else f"param_{param_index}")

        return list(zip(param_indices, param_names))

    df = pd.DataFrame(data)
    with pd.ExcelWriter(xlsx_path, engine='openpyxl') as writer:
        for (param_index, param_name) in _extract_param_indices_and_names(df):
            df.pivot_table(index='chainStart', columns='periodDate', values=f"param_{param_index}").round(2).to_excel(writer, sheet_name=param_name)
```

BILAGA D

```
calc_cycle2 <- function(depth, bounds = 2.5, days, int_date, tau = 0.6,
                        print_plot = FALSE, fallback_change_rate = 0.0015){

 # To be able to manually check how well the function works
 if (is.numeric(print_plot)){

   print_plot <- sample(c(TRUE, FALSE), 1,
                        prob  = c(print_plot, 1 - print_plot))
 }

 depth_change <- diff(depth)
 days_passed <- diff(int_date)

 # Should below be estimated for each 20m section?

 # change_rate <-  tryCatch({
 #   qpred(
 #     depth_change ~ days_passed - 1,
 #     tau = 0.6,
 #     days_ahead = 1,
 #     weight_age = FALSE,
 #     return_range = FALSE)
 # }, error = function (e) fallback_change_rate)
 change_rate <- fallback_change_rate

 if (print_plot) {
   plot(int_date, depth, col = 2, pch = 19)
 }

 outlier <- rep(FALSE, length(depth))
 observations <- rev(1:length(depth))
 cycle <- rep(0, length(depth))
 prev_in_band_of_next <- FALSE

 for(i in observations[-1]) { # skip last in loop

   n_obs <- max(observations)
   current_depth <- depth[i]
   depth_next <- depth[i + 1]
   days_to_next <- days[i + 1]
   days_since_previous <- days[i]


   outside_band_next <- abs(current_depth + days_to_next * change_rate - depth_next) > bounds

   if (n_obs - i >= 2){
     days_to_next_next <- days[i + 1] + days[i + 2]
     depth_next_next <- depth[i + 2]
     outside_band_next_next <- abs(current_depth + days_to_next_next * change_rate - depth_next_next) > bounds
   } else {
     outside_band_next_next <- FALSE
   }

   if (i !=1){
     depth_previous <- depth[i - 1]
     outside_band_prev <- abs(current_depth - change_rate * days_since_previous - depth_previous) > bounds
     prev_in_band_of_next <- abs(
       depth_next - change_rate * (days[i] + days[i - 1]) - depth[i - 1]
     ) < bounds
   } else {
     # Not possible to check, set for things to work
     outside_band_prev <- TRUE
     prev_in_band_of_next <- FALSE
   }
   if (!outside_band_next_next & outlier[i + 1]){
```

```
    cycle[i] <- cycle[i + 1]
  }else if (outside_band_next & (outside_band_next_next | outlier[i + 1]) & outside_band_prev){
    outlier[i] <- TRUE
    cycle[i] <- cycle[i + 1]
    if (print_plot) points(int_date[i], current_depth, pch = 21, col = 1, bg = 0)
  } else if (outside_band_next & (outside_band_next_next | outlier[i + 1]) & !outside_band_prev){
    cycle[i] <- cycle[i + 1] + 1
    if (print_plot) points(int_date[i], current_depth, col = 2 + cycle[i], bg = 2 + cycle[i], pch = 21)
  } else {
    cycle[i] <- cycle[i + 1]
    if (print_plot) points(int_date[i], current_depth, col = 2 + cycle[i], bg = 2 + cycle[i], pch = 21)
  }

 }

 outlier <- outlier * cumsum(outlier)
 return(list(cycle, outlier))
}
```