Utvärdering av kollektivtrafik - PM av det genomförda arbetet och skriptguide

Nikos Papakatsikas, WSP Sverige

2021-08-18

Följande dokument är en kort sammanfattning av uppdraget samt en användarhandledning av R-skript som utvärderar Region Uppsalas kollektivtrafik genom att analysera utbudet enligt UL:s API.

Projektbeskrivning

I ett projekt genomfört under våren 2021 har WSP kvalitetssäkrat och vidareutvecklat de programmeringsprocesserna som används i kollektivtrafikens utvärdering. Mer specifikt har följande aktiviteter genomförts:

- Genomgång av alla 4 separata skript som har levererats från Region Uppsala och en kvalitetssäkring i sammanställningsprocesserna. Varje skript har körts rad för rad och mellanresultat har kontrollerats så att de stämmer mot förväntningarna.
- Omstrukturering så att alla skript för de olika indikatorerna kan köras från ett huvudskript *main.R* och därmed förenkla kommande körningar. Huvudskriptet innehåller även alla beroenden (R packages samt underskript) så att det går att köra på alla datorer som har R installerat.
- Parametrisering av de olika funktioner i huvudskriptet så att användaren kan ange specifika argument som kan ändra förutsättningar för bl.a. API-hämtningen (t.ex. definition av rusningstid, datum för hämtning, exkluderande linjer). Detta hjälper användaren att kunna specificera analysen enligt önskade eller uppdaterade förutsättningar.
- En omstrukturering av mappstruktur så att det lättare kan hittas och lagras relevant indata och utdata.
- En lagring av relevant shapefil som används för lokalisering av hållplatserna i de olika tätorten i mappstrukturen, så att det inte krävs en nedladdning med varje körning, eftersom det orsakade problem på vissa datorer där användaren inte är administratör.
- Ett tillägg i skript som hämtar API-data från UL så att linjerna som inte har lästs på grund av t.ex. uppkopplingsproblem kan lättare identifieras. Datat hämtas per 100 linjer, och i fall något inte fungerar under hämtningen, skrivs ut vilket intervall har misslyckats, så att användaren kan hämta datat bara för de linjerna.
- Ett argument i skript som hämtar API-data från UL har lagts till, vilket möjliggör att skriptet använder redan nedladdade data från tidigare körning.
- Den här guiden har kompilerats så att skriptet kan köras av andra.

Slutsatser om projektet

Skriptet möjliggör en kontinuerlig utvärdering av kollektivtrafikutbudet genom att använda API data och jämföra det mot Trafikförsörjningprogrammets mål. Detta är ett viktigt verktyg för Region Uppsala som kan nu genomföra analyser på egen hand, med väldigt lite manuellt arbete och flexibelt mot olika mål och frågeställningar. Analyserna kan göras för flera punkter i framtiden så att det kan tas fram olika resultat för olika planerade trafikutbud, exempelvis sommartrafikering eller helgutbud.

Det kvarstår några punkter som kan förbättra skriptets funktionalitet och användarvänlighet som inte fick utrymme inom det befintliga projektets budget och ambition. Dessa punkter påverkar inte resultatet.

- Lagrat API-data kan användas men det finns inte inbyggd funktionalitet för att lagra API-data som hämtas under en körning, utan detta måste fortfarande göras manuellt.
- API hämtningen kan skrivas i ett separat skript även för indikatorer 4 och 5, i stället av att vara inbyggd i dataanvändningen. Det skulle betyda att API-data hämtas en gång och det används sen i flera funktioner.
- Mappstrukturen är fast och det är inte möjligt att använda filer som ligger någon annanstans än i skriptets mapp. Det kan läggas till ett steg där användaren väljer var inputfilerna ligger och var uttagsfilerna ska lagras.
- Skriptet kan ha långa körtider pga API-hämtningen. Användaren kan se om programmet fortfarande är igång, eftersom det skrivs ut vilka linjer som har hämtats under körningen. Denna funktion kan utökas i flera delar i koden, så att man vet vilka steg som har körts.

Skriptet har körts och resultat för flera indikatörer har redovisats för Region Uppsala. Efter redovisningen upptäcktes att resultat för tider utanför högtrafik inte stämde mot manuella kontroller för vissa relationer. I en skriptkontroll upptäcktes att anledningen till detta var en parameter i api data sammanställningen som var statisk och inte hade uppdaterats så att den är dynamisk mot om man analyserar högtrafik eller lågtrafik. Skriptet åtgärdades och ytterligare en kontrolprocess inleddes under sommaren 2021.

Guidens Innehåll

1. Syfte
2. Skriptstruktur
3. Filstruktur

Syfte

Skript används för att läsa in data från UL:s API angående linjer, hållplatser och tidtabellsinformation. Datat analyseras utifrån flera perspektiv för att utvärdera kollektivtrafikutbudet enligt krav som ställs i Trafikförsörjningsprogrammet.

Skriptstruktur

Kravspecifikation

Skript har strukturerats så att användaren kan enkelt ädnra några grundförutsättningar, såsom API inläsningsdatum och definition av högtrafik.

Huvudskriptet *main.R* används för att köra de olika stödjande skript. Alla stödjande skript har skrivits i form av en *R-funktion*, det vill säga en funktion som kan anropas från huvudskriptet genom att ange något värde för alla argument. Alla funktioner har default argument, men standardanvändning är att ange argumentvärden när funktionerna anropas i *main.R*. Standardanvändningen kräver inte att editera något annat skript än *main.R*.

Huvudskriptet säkerställer att samtliga stödjande skript samt R-paket är installerade och laddade. Det kräver att stödjande skript ligger i samma mapp som *main.R* (mer om det i nästa sektion).

Krav på paket:

- httr

- rlist
- jsonlite
- dplyr
- sp
- rgdal
- raster
- readxl
- tidyr
- stringr
- lubridate
- rgeos
- spatialEco

Stödjande skript:

- *get_stop_and_line_data.R*
- *create_api_input_dataframe.R*
- *get_api_data.R*
- *get_indicator_pendling.R*
- *get_indicator_aktiviteter.R*
- *get_indicator_skolor.R*

Beskrivning av huvudskriptet *main.R*

I huvudskriptet kan man anropa de olika stödjande skripten genom att köra bara den delen i koden. Man kan också köra igenom hela processen om man köra hela huvudskriptet. Nedan beskrivs processen i högre detalj.

Inläsning av hållplats- och linjedata

Genom skriptet *get_stop_and_line_data.R* kan man uppdatera utdatafiler *HplData_Datum.csv*, *HplIdKoordinat_Datum*, *AllaLinjerPerHplID_Datum*. Skriptet använder tre argument; en vektor med linjer som ska exkluderas från analysen, en vektor med linjetyper som ska exkluderas från analysen och en vektor med tätortsnamn som ska betraktas som storregionala kärnor. I funktionen nämns de två argument *excluded_lines*, *excluded_types* och *storregionala_tatorter*.

Linjetyper kan exkluderas enligt följande lista: 1 - Uppsala stad, 2 - tätortstrafik, 3 - Expressbussar, 4 - SL bussar, 5 - färja, 6 - interregionala bussar, 7 - Uppsala stad linjer 30-32, 8 - SL pendeln, 9 - Upptåget, 11 - skoltrafik och förstärkningstrafik

HplIdKoordinat_Datum: En tabell med alla hållplatsers ID samt koordinater, vilka hämtas från API:n *AllaLinjerPerHplID_Datum*: En tabell med alla linjer som trafikerar (stannar vid) varje hållplats *HplData_Datum.csv*: En tabell med geografisk information för varje hållplats. Koordinaterna används för att allokera hållplatsen till en specifik kommun, län och tätort. Tätortens befolkning skrivs också ut, och även en typ mellan följande:

- Storregional kärn, om tätortens namn finns i argument *storreg_tat*
- Större tätort, om tätorten har flera än 7000 invånare
- Medelstor tätort, (mellan 1000 och 7000 invånare)
- Mindre tätort, (mellan 200 och 1000 invånare)
- Småort, (mellan 50 och 200 invånare)
- Utanför tätort, mindre än 50 invånare

Indikatorer 1.1, 1.2, 2.1, 2.2, 3.1, 3.2

Indikatorerna hanteras med två skript, *create_api_input_dataframe.R* och *get_api_data.R*.

I *create_api_input_dataframe.R*, en excelfil med krav på antal turer mellan olika tätorter och hållplatsdatafiler används för att skapa en dataframe som ska användas för utvärderingen. I dataframe ligger alla krav på tätortsrelationer, samt vilka hållplatser som ska kontrolleras per relation. Dataframe exporteras i en csv fil *AnalysPlanInputForAPI.csv*.

I *get_api_data.R* hämtas data för de angivna relationer (mellan hållplatser i *AnalysPlanInputForAPI.csv*) för flera timmar, både rusning och övrig tid. Datat bearbetas så att den formuleras till inedx-värden som beskriver hur väl stämmer utbudet med kravet. Om det exempelvis finns 1.5 turer per timme i snitt, medan målet ligger på 2 turer per timme, då uppfylls målet på 75% och indexvärdet för just denna indkatorn är 0.75.

I funktionen matas in flera argument som kan hanteras i *main.R*. Anges inte något värde till dessa, kommer defaultvärden att användas.

- *vector_peak_hrs*: en vektor med strings, som definierar vilka tider ska betraktas som rusningstrafik, default är 6-8 samt 15-18.
- *vector_off_peak_hrs*: en vektor med strings, som definierar vilka tider ska betraktas som övrig trafik, default är 5, 9-14, 19-23.
- *existing_api_data*: en string som är filnamn av redan nedladdade API-data som ett RData objekt. Default är NA, vilket betyder att datat ska hämtas på nytt från API:n.
- *test_run*: en boolean. Om TRUE då kommer bara en del av indikatorer och en del av tider utvärderas, vilket kan vara hjälpsamt i provkörningar. Default är FALSE.
- *alternatives*: en integer som definierar hur många alternativ från varje sökning ska utvärderas. Maxvärdet är 6, eftersom det är 6 sökningar som API tillåter.
- *api_date*: en string med datum i form "ÅÅÅÅ-MM-DD". Sökningar ska göras för det datumet. Datumet får inte vara i dåtid. Mellanfiler "API_data" och "API_distinct_data" lagras med detta datum i namnet. Default är körningsdatum.

För att underlätta kontrollen, efter varje 100:e API-hämtning skrivs ut ett meddelande att inläsningen av just dessa linjer har lyckats. I fall av et misslyckande har det funnits problem i minst en av dessa linjer. Då kan man komplettera genom att endast hämta data för den del av linjerna.

Den förväntade körtiden för API hämtningen ligger mellan 9 och 12 timmar, beroende på hur snabbt servern svarar på requests.

I framtiden kommer API hämtningen kodas som ett separat skript, för att förenkla hämtning av specifika kopplingar samt ge möjlighet att exportera resultatet av en API hämtning så att den inte behöver göras om.

Resultatet sparas i en csv fil med indikatorer, tätortsrelationer och indexvärdet. Filen heter *Resultat_Datum.csv*.

Indikator 4.1 (Pendling)

På liknande sätt hanteras indikator 4.1, med skript *get_indicator_pendling.R*. Här görs både dataframe förberedning för indikatorn och api hämtning i samma skript.

I funktionen matas in flera argument som kan hanteras i *main.R*. Anges inte något värde till dessa, kommer defaultvärden att användas.

- *alternatives*: en integer som definierar hur många alternativ från varje sökning ska utvärderas. Maxvärdet är 6, eftersom det är 6 säkningar som API tillåter.
- *api_date*: en string med datum i form "ÅÅÅÅ-MM-DD". Sökningar ska göras för det datumet. Datumet får inte vara i dåtid. Default är körningsdatum.

Resultatet sparas i tre csv filer, som resultat mellan tätorter (*resultat_pendling_Datum.csv*) och grupperat på startort och slutort (*resultat_pendling_starttatort_Datum.csv*, *resultat_pendling_stoptatort_Datum.csv*).

Indikator 4.2 (Aktiviteter)

På liknande sätt hanteras indikator 4.2, med skript *get_indicator_aktiviteter.R*. Här görs både dataframe förberedning för indikatorn och api hämtning i samma skript.

I funktionen matas in flera argument som kan hanteras i *main.R*. Anges inte något värde till dessa, kommer defaultvärden att användas.

- *alternatives*: en integer som definierar hur många alternativ från varje sökning ska utvärderas. Maxvärdet är 6, eftersom det är 6 säkningar som API tillåter.
- *hpl_file_date*: en string med datum i form "ÅÅÅÅ-MM-DD". Det hjälper i att få rätt hållplatsindatafil. Default är körningsdatum.
- *api_date_vd*: en string med datum i form "ÅÅÅÅ-MM-DD". Sökningar ska göras för det datumet för vardagar. Datumet får inte vara i dåtid och syftet är att det är en vardag. Default är körningsdatum.
- *api_date_h*: en string med datum i form "ÅÅÅÅ-MM-DD". Sökningar ska göras för det datumet. Datumet får inte vara i dåtid och syftet är att det är en helgdag. Default är körningsdatum.

Resultatet sparas i en csv fil, *resultat_aktiviteter_Datum.csv*.

Indikator 5.1 (Skolor)

På liknande sätt hanteras indikator 5.1, med skript *get_indicator_skolor.R*. Här görs både dataframe förberedning för indikatorn och api hämtning i samma skript.

I funktionen matas in flera argument som kan hanteras i *main.R*. Anges inte något värde till dessa, kommer defaultvärden att användas.

- *alternatives*: en integer som definierar hur många alternativ från varje sökning ska utvärderas. Maxvärdet är 6, eftersom det är 6 säkningar som API tillåter.
- *hpl_file_date*: en string med datum i form "ÅÅÅÅ-MM-DD". Det hjälper i att få rätt hållplatsindatafil. Default är körningsdatum.
- *api_date_vd*: en string med datum i form "ÅÅÅÅ-MM-DD". Sökningar ska göras för det datumet för vardagar. Datumet får inte vara i dåtid och syftet är att det är en vardag. Default är körningsdatum.
- *api_date_h*: en string med datum i form "ÅÅÅÅ-MM-DD". Sökningar ska göras för det datumet. Datumet får inte vara i dåtid och syftet är att det är en helgdag. Default är körningsdatum.

Resultatet sparas i en csv fil, *skolor_ResultatKort.csv*.

Filstruktur

Alla filer ligger i en mapp som heter *00_skript*. Under mappen finns det en mapp som heter *src* (kort för *source code*) och innehåller alla skript, samt en mapp som heter *data*. Under mapp *data*, finns det tre mappr, *input*, *interim*, *output*. Under *input* ligger alla indata, såsom shapefiler (i en mapp *shp*), excelfil med indikatorernas kopplingar och csv fil med tätort-hållplats kopplingar. Under *interim* ligger alla filer som genereras i processen och sen återanvänds som indatafiler i ett annat steg i processen. Under *output* ligger alla slutliga uttag från skript, exempelvis indikatorernas tabeller.