[Handläggare]
[Avdelning]

Automatändring av misstänkta fel i Triton

Inledning

I granskningslitteraturen delas misstänkta fel upp i två grupper, systematiska fel och slumpmässiga fel. Systematiska fel påverkar inkomna svar i samma riktning för alla objekt och leder till bias i de slutliga skattningarna. Den här typen av fel uppkommer t.ex. genom att uppgiftslämnaren tolkar en fråga fel och t.ex. lämnar efterfrågade uppgifter i kronor istället för efterfrågade 1000-tals kronor. Slumpmässiga fel inträffar utan systematik, ett exempel skulle kunna vara att en uppgiftslämnare råkar lägga till en siffra för mycket i ett svar.

För systematiska fel där felorsaken är känd finns ofta möjlighet att göra deduktiva ändringar som kan automatiseras. Den här typen av fel hanteras med fördel tidigt i granskningsprocessen för att undvika att de fastnar i senare mer resurskrävande delar. Automatisk hantering av slumpmässiga fel förenklas och efterföljande selektiv granskning effektiviseras i och med att mängden misstänkta fel minskar.

I mikrogranskningen ska misstänkta systematiska fel hanteras automatiskt där så är möjligt. Det finns metoder och verktyg för att hantera även slumpmässiga fel med automatik, men dessa används inte i så hög grad på SCB idag. För undersökningar som ligger i Triton hanteras automatiska ändringar genom att svaren skickas till SAS, i SAS identifieras misstänkta fel och identifierade misstänkta fel ändras för att sedan skickas tillbaka till Triton.

Arbetet med att införa automatändringar omfattar att:

- Skapa en struktur för kopplingen mellan SAS och Triton
- Genomföra test och utvärdera effekt av automatändringar på tidigare produktionsomgångar
- Implementering i produktion
- Skapa rutin för utvärdering.

Granskning och automatiska ändringar

För att få en uppfattning om förekomsten av systematiska fel bör granskningskontroller och deras flaggningar av data undersökas. Om en granskningskontroll flaggar fel på ett liknande sätt för inkomna svar är det ett tecken på ett systematiskt fel. Ett första steg i arbetet med att införa automatändringar är att analysera felsignalerade värden och granskningskontroller för tidigare produktionsomgångar.

Ett komplement till deduktiva ändringar av inkomna svar är att försöka förebygga fel via ändringar i frågeformuläret. Förstår ul vilka uppgifter det är vi efterfrågar? Om det finns möjlighet att förebygga fel via ändring av frågeformuläret ska det normalt väljas framför automatändring.

Enhetsfel

I många undersökningar är tanken att ul ska lämna efterfrågade uppgifter i överenstämmelse med en gemensam bas, exempelvis 1000-tal. Om ul inte tar del av instruktioner kan det hända att de svar ul rapporterar avviker med en multipel från det förväntade. Den här typen av fel benämns enhetsfel och ett vanligt enhetsfel är tusenfel. Tusenfel uppkommer t.ex. då uppgiftslämnare rapporterar en variabel i kronor istället för efterfrågade tusentalskronor. Tusenfel har stor påverkan på den publicerade statistiken och skiljer sig från andra typer av fel genom att ul, i de flesta fall, gör samma fel för alla efterfrågade variabler.

Det finns olika metoder för identifiering av tusenfel. Gemensamt är att de ofta baseras på en jämförelse mellan det rapporterade värdet och ett referensvärde. Referensvärdet kan vara granskat värde från föregående produktionsomgång, stratummedelvärde eller en registervariabel. På samma sätt som kvarvarande tusenfel har stor påverkan på statistikens kvalitet har också falska ändringar stor påverkan. En rekommenderad metod för att hålla hög precision i ändringar är att undersöka differensen mellan x_{ogr} och x_{ref} via testvariabeln

$$diff = \left|\left\lceil log_{10} x_{ogr} \right\rceil - \left\lceil log_{10} x_{ref} \right\rceil\right|$$

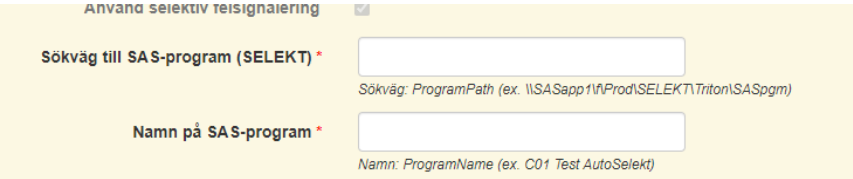
Där $\lceil a \rceil$ betecknar det minsta heltal större än eller lika med a. Om testvariabeln tar värdet tre dras slutsatsen att det finns ett tusenfel. En fördel med den här metoden är att det bara är antalet positioner som påverkar utfallet, det räcker till exempel med att känna till förväntat antal positioner för det värde som undersöks.

Här finns värden och situationer att vara uppmärksam på när x_{ogr} eller x_{ref} tar värdet noll går inte testvariabeln att använda. Värdet ett är också ett problem då logaritmen av ett är noll, på liknande sätt är även 10, 100 osv. problematiska, ett sätt att komma förbi de här problemen är att addera 1 till x_{ogr}.

I vissa situationer kan värdet fyra på testvariabeln leda till att tusenfel inte identifieras. Om exempelvis variabeln är omsättning och ett företag uppvisar en omsättningsökning och samtidigt gör ett tusenfel skulle testvariabeln kunna ta värdet fyra. Om det med ämneskunskap går att öka precisionen i ändringarna så kan ändringar också göras för andra värden på testvariabeln.

Förberedande arbete

Automatändringar kräver funktionalitet som inte finns inbyggd i Triton. Vanligtvis används program i SAS för hanteringen så det behöver finnas en koppling där data kan skickas från Triton till SAS och sedan tillbaka till Triton. En generell sådan koppling finns på plats. Kopplingen utgår från den lösning SCB använder för selektiv granskning via Selekt. På samma sätt som Triton behöver veta vilken mapp och vilket skript som ska användas för Selekt så behövs mapp och sökväg för skript för automatändringar. I ett första steg sätts informationen upp i test vid implementering av insamlingsomgång i undersökningens processyta.



Här ska sökväg peka på den mapp där SAS program finns och Namn på SAS-program pekar på det första program som ska exekveras, mer information finns i avsnittet om konfiguration av den undersökningsspecifika delen.

Med sökvägarna specificerad kommer inkomna svar och bakgrundsinformation gå vidare till SAS efter att svaren passerat Signal.

När metoden för automatändringar implementerats och kontrollerats i test görs motsvarande arbete inför implementering i produktionsmiljön.

Automatändringar i Triton

En förutsättning för implementering av automatändringar är att det finns möjlighet att ta fram referensvärden av god kvalitet att jämföra nya rapporterade svar med.

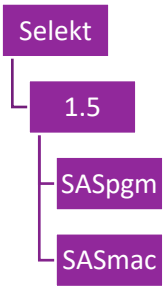
En struktur för automatisk hantering av misstänkta fel

Nedan följer en beskrivning av struktur och kod för automatändringar. En demostruktur med program finns här.

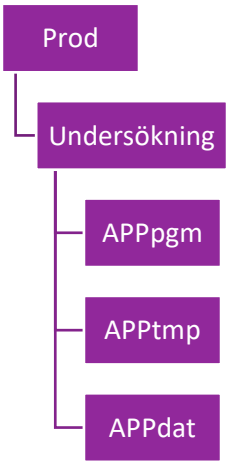
Mappstruktur

En del av programmen är menade att vara generiska. En annan del måste anpassas till varje undersökning eller undersökningsomgång. Därför lagras program på två ställen, där den ena är generisk och ej ska ändras. Den andra delen har en mappstruktur som bör efterliknas när vid implementering i en ny undersökning.

Generisk del



Applikation



Överst i hierarkin i den generiska delen finns mappen Selekt och direkt under finns versioner lagrade. Under Prod finns de produktspecifika tillämpningar efter undersökningsnamn.

Den generiska delen

Den generiska delen är delad i två delar, SASpgm och SASmac, i SASpgm ligger det program som sätter upp libnames, macron och format, I SASmac finns de specifika macron som används.

Program: *A00 Init Session*

Sätter upp libnames och kompilerar de macron som ligger i SASmac

Macron:

TritonGetAÄ: Läser in data från Triton och transformerar det.

Följande parametrar skickas in från Tritonanropet:

RequestCallID	Temporärt ID för objekt som ska felsignaleras
Server	Där aktuellt data ligger
Databas	Där aktuellt data ligger
_METAUSER	Användare, för att kunna komma åt andra databaser för jämförande data (t.ex. Tritonshard)
COLLECTIONROUNDID	Insamlingsomgångens id i Triton
REFERENCESTARTDATE	Startdatum för insamlingsomgångens referensperiod
REFERENCESTOPDATE	Slutdatum för insamlingsomgångens referensperiod
SHARDDATABASE	Databasen där undersökningsomgången ligger i Triton
SHARDSERVER	Serven där undersökningsomgången ligger i Triton

Macrot *TritonGetAÄ* tar in denna data och lägger ihop den till en tabell `Hotdeck`. Denna tabell innehåller transponerade variabler med en rad per rapporteringsobjekt(SCBID) och en kolumn per direktinsamlad variabel(om värde finns för denna i indata). Används dynamiska variabler i undersökningen skapas en rad per sådan. Makrovariabler skapas av ShardServer, ShardDatabase, CollectionRoundId, ReferenceStartDate och ReferenceStopDate för att enklare kunna användas i den undersökningsspecifika bearbetningskoden.

TritonSetAÄ behöver tabellen `Error_list`. `Error_list` måste skapas av den undersökningsspecifika bearbetningskoden och innehålla en rad per SCBID och felsignal (Felkod/ErrorCode) samt information om felkontrollen har släckts efter automatändring eller inte. Två metadatatabeller måste finnas för den specifika undersökningen för att TritonSetAÄ ska fungera,

antingen i work eller i APPdat. Dessa är `TritonOutError` och `TritonOutErrorVariable`.

`TritonSetAÄ` fyller `OutError` , `OutErrorVariable`, `OutSelektor` och `OutAnswer` i Triton med följande variabler:

`OutError`:

`Id`	Id för felsignalen	`Int`
`fkRequestCallId`	Temporärt id för blanketten	`Int`
`ErrorCode`	Felkodens namn	`$50`
`Description`	Beskrivning av felkoden	`$1024.`
`Instruction`	Instruktion till granskning	`$1024.`
`Expression`	Formel för felkontroll	`$1024.`
`Hard`	Huruvida det är en mjuk eller hård kontroll, Hard=1 är hård Hard=0 är mjuk	`bit`

`OutErrorVariable`:

`Id`	Id för felsignalen och variabeln	`Int`
`fkOutErrorId`	Id för felsignalen	`Int`
`VariableName`	Den variabel som ska felsignaleras av ErrorCode	`$200`
`VariableIndex`	Om dynamisk variabel	`Int`

Tabellerna förhåller sig till varandra så att `OutError` innehåller unika felsignaler för ett SCBID och CollectionRoundId, medan `OutErrorVariable` innehåller unika rader per felsignal och variabel (och dynamisk variabel) som ska felflaggas. Detta styr markeringarna i Edit och vilken information som visas om felkontroller.

OutAnswer:

Id	Id för felsignalen och variabeln	Int
fkRequestCallId	Temporärt id för blanketten	Int
VariableName	Den variabel som ska felsignaleras av ErrorCode	$200
VariableIndex	Om dynamisk variabel	Int
Value	Värde efter automatändring	$1024

Ska innehålla information om vilka variabler som har automatändrats och värdet efter automatändring.

OutSelektor:

Id	Id för felsignalen och variabeln	Int
fkRequestCallId	Id för felsignalen	Int
Value	Värde Selektor	Int

Om det inte finns några återstående fel för objektet efter automatändring så ska selektorn sättas till 101. Om det finns återstående fel ska selektorn sättas till 1.

Den undersökningsspecifika delen

Det finns en koppling mellan Triton och Selekt för att skicka in inkomna data från Signal till SAS. Triton startar upp en session av SAS och exekverar skript med hjälp de parametrar som satts vid implementering av insamlingsomgång i undersökningens processyta (Sökväg till SAS-program, Namn på SAS-program). Dessa parametrar styr alltså vilket produktspecifikt program som ska exekveras.

Undersökningsspefika SAS-program

Tritonanropet startar upp det SAS-program som är angivet under Namn på SAS-program i formuläret för insamlingsomgången.

I det här programmet ska två katalogvägar sättas, en som pekar på den generiska delen och en som pekar på den undersökningsspecifika. Det här görs lämpligast i ett initierande program som läggs i den undersökningsspecifika mappen under AppPGM. Namnet bör innehålla

undersökningens kortnamn (eller undersökningsomgången). Detta program anropar sedan resterande SAS-program, generiska och undersökningsspecifika.

Följande kod bör finnas med:

```
%let PATH_sys=\\Sasapp1\F\Prod\SELEKT\1.5;

%let PATH_app=\\Sasapp1\F\Prod\SELEKT\Undersökning;

%include "&PATH_sys.\SASpgm\A00 Init Session.sas";

%GLOBAL ReferenceStartDate ReferenceStopDate
CollectionRoundId ShardServer ShardDatabase;

/*##################################################
######################*/

*ProcessBody;

/*##################################################
######################*/

%let    Server=&SERVER;

%let    Databas=&DATABAS;

libname Triton odbc schema=SASReview bulkload=yes
dbcommit=0
           complete="driver=SQL Server;
            server=&Server;

           Trusted_Connection=Yes;
            database=&DATABAS";
%put _ALL_;
```

När data lästs in och bearbetats behöver undersökningsspecifik kod köras där felkontroller finns som skapar de automatändringar man vill använda sig av. Det kan också finnas ett behov att ytterligare bearbeta indata. Till exempel kan det vara nödvändigt att upprepa granskningskontroller från produktionsgrankningen då variabelvärden kan ha justerats efter automatändring och kontrollen kanske inte längre ska leda till en felsignal.

Output från den produktspefika programmen behöver heta Error_list och bestå av en rad per SCBID och felsignal (Kolumn=ErrorCode) och information om ifall felkontrollen ska släckas efter automatändring. Rader utan värde på ErrorCode kommer inte läsas tillbaka till Triton.

Utöver Error_list måste också en tabell, motsvarande OutAnswer, skapas som innehåller variabler som har automatändrats och automatändrade värden.

Utvärdering

Inför införande av automatändringar i produktionsmiljön ska implementeringen utvärderas i testmiljön. Normalt finns en granskad produktionsomgång att utgå ifrån och utvärderingen görs både på mikro- och makronivå.

På mikronivå analyseras genomförda automatändringar genom att undersöka hur automatändrade värden förhåller sig till produktionsdata för testomgången. Här är det intressant att både undersöka de värden metoden ändrat och värden metoden inte identifierat men som ändrats i produktionsdata.

På makronivå kan skattningar för automatändrade data jämföras med motsvarande skattningar baserade på granskade data. I de fall granskad data innehåller andra typer av fel och där dessa har stor påverkan kan en anpassning behöva göras. En möjlighet är att endast jämföra automatändrade data med ändrade enhetsfel i produktionsdata. Inför införande i produktionsmiljön är kvaliten i skattningarna styrande. Ett sätt att utvärdera skattningarna är att studera biaskvoten

$$BR(\hat{\theta}) = \frac{B(\hat{\theta})}{\left[\hat{V}(\hat{\theta})\right]^{1/2}}$$

Om biaskvoten är mindre än 0,3, kommer täckningsgraden för konfidensintervall fortfarande vara god.

Automatändringar ska också utvärderas löpande när metoden implementerats i produktionsmiljön. Antalet genomförda automatändringar följs över tid för att säkerställa att ändringarnas tillförlitlighet.