Miniprojektuppgift i TSRT04: Mandatfördelning i val

21 mars 2017

1 Uppgift

I parlamentsval så röstar medborgarna på olika partier som ska företräda dem. När alla rösterna har räknats så kan man se hur de har fördelat sig över de olika partierna. Eftersom varje parlament har ett begränsat antal ledamöter/mandat (t.ex. 349 i Sveriges riksdag och 20 svenska ledamöter i EU-parlamentet) så måste man ha en metod för att avgöra hur många mandat som varje parti ska få. Detta kallas för ”mandatfördelning”.

Sverige och många andra länder har proportionella val, vilket innebär att mandatfördelningen ska efterlikna röstandelarna så väl som möjligt.[1] Eftersom mandaten är mycket färre än antalet röster så är det oundvikligt att det blir avrundningsfel. Det finns flera kända metoder för att fördela mandat och dessa är designade för att ge olika typer av avrundningsfel. Vissa metoder premierar partier som har en stor röstandel, medan andra metoder lyfter fram mindre partier.

I det här miniprojektet ska ni jämföra tre olika metoder för mandatfördelning: D’Hondt-metoden, uddatalsmetoden och jämkade uddatalsmetoden. Målet är att skriva en funktion som, givet en viss röstfördelning och antal mandat, räknar ut mandatfördelningarna och plottar dessa tillsammans med röstfördelningen. Ni kommer att få testa er funktion på det senaste EU-valresultatet och även importera resultat från något annat val.

1.1 Redovisning

Detta är ett av miniprojekten i kursen TSRT04. För att bli godkänd på kursen måste ni lösa ett av miniprojekten enligt anvisningarna och redovisa det för en lärare på något av examinationstillfällena. Redovisningen sker på engelska så skriv er kod och era kodkommentarer på engelska (det är en bra övningen för framtiden då engelsk kodning är standard på företag). Uppgiften ska lösas i grupper om två, eller individuellt. Eftersom det rör sig om ett examinationsmoment är det inte tillåtet att dela eller visa MATLAB-kod/anteckningar för andra studenter. Det är däremot okej att diskutera uppgiften muntligen med andra grupper, exempelvis för att dela med sig av goda råd!

[1] Alla valsystem är inte proportionella. USA och flera andra länder har *majoritetsval* baserat på ett antal valkretsar. Det parti som får flest röster i en valkrets får alla mandat/ledamöter som är kopplade till den valkretsen. I andra sammanhang, såsom Eurovision Song Contest, används *Bordaräkning* där de röstande rangordnar kandidaterna och vinnaren är den med flest totalpoäng.

- Skriv en MATLAB-funktion `electionresults` som beräknar mandatfördelningen med tre olika kända metoder (se nedan). Funktionen ska även rita diagram över resultatet och därmed illustrera hur avrundningsfelen ser ut hur de olika metoderna. Funktionen `electionresults` ska anropa andra funktioner som löser vissa delproblem.
- Lösningen ska demonstreras och koden ska visas upp för lärarna på laboration 3 eller 5. Vi kommer kontrollera att ni följt anvisningarna, det finns en rimlig mängd kommentarer i koden, att plottarna är lättförståeliga och att ni kan förklara vad olika delar av koden gör. Det finns en stilguide på kurshemsidan som berättar mer om hur en bra lösningen ska se ut. Se till att läsa den och följa rekommendationerna!
- När lärarna sagt att ni blivit godkända på projektet så ska ni även skicka koden till Urkund för plagiatskontroll. Ni skickar ett e-brev till `hakan.johansson.liu@analys.urkund.se` med era fullständiga namn i textfältet. Koden bifogas i ett text-dokument döpt till kurskod_år_studentid1_studentid2.txt (t.ex. TSRT04_2017_helan11_halvan22.txt). Detta dokument ska innehålla alla funktioner och skript som examinerats, inklusive ett kort exempel på hur man kan anropa dessa funktioner för att lösa projektet.

2 Förslag på arbetsgång

Börja med att läsa hela dokumentet för att sätta dig in i problemställningen och vårt förslag till hur man kan dela upp den i delproblem.

Nedan har vi gett ett förslag på hur den stora programmeringsdelen av uppgiften kan lösas bit för bit. Vi har delat upp problemet så att det finns möjlighet att kontrollera att varje bit fungerar innan man tar sig an nästa bit. Utnyttja dessa möjligheter! Trots att arbetsgången föreslår att ni ska skriva små funktioner som löser delproblem innebär detta inte att det nödvändigtvis är bra/effektivt/snyggt att använda sig av (anropa) alla de små funktionerna när man går över till att lösa större delproblem. Vissa (alla?) funktioner löser helt enkelt så små bitar att det kan vara bättre att kopiera programkoden från den lilla funktionen till den större, istället för att låta den större funktionen anropa den mindre.

Observera att den föreslagna arbetsgången är anpassad för studenter med ingen eller liten tidigare programmeringserfarenhet. En erfaren programmerare skulle troligen dela upp problemet på ett annat sätt baserat på tidigare erfarenheter. Om du känner dig erfaren och har ett förslag till en bättre lösning så är det inget som hindrar dig att använda den istället, men om du behöver hjälp så har assistenterna mer erfarenhet av den av oss föreslagna lösningsgången. Ifall ni frångår den föreslagna arbetsgången så måste ni se till att lösningen ger åtminstone samma funktionalitet.

2.1 Teori: Tre metoder för mandatfördelning

Ni ska testa tre metoder för mandatfördelning som alla är sekventiella, vilket innebär att de fördelar ett mandat i taget. I detta avsnitt beskrivs hur det går till. När ett mandat ska delas ut så ges det till det parti som har det största *jämförelsetalet*. Från början så är jämförelsetalet detsamma som antalet röster (eller proportionellt mot det), men varje gång som ett parti får ett mandat så minskar partiets jämförelsetal så att nästa mandat troligtvis går till ett annat parti. Hur denna minskning beräknas är det som skiljer olika metoder åt.

Vi behöver lite matematisk notation för att kunna beskriva metoderna. Antalet partier betecknas med n. Antalet röster som parti $1, 2, \ldots, n$ har fått betecknas med $r_1, r_2, \ldots, r_n$. Låt $m_1, m_2, \ldots, m_n$ vara antalet mandat som ett parti har fått hittills i den sekventiella metoden. Från början är alla dessa variabler satta till noll ($m_1 = m_2 = \ldots = m_n = 0$), men efterhand som mandaten delas ut så ökar dessa m-variabler.

När ett mandat ska delas ut så beräknas jämförelsetalen som

$$z_i = \frac{r_i}{f(m_i)} \quad i = 1, 2, \ldots, n, \tag{1}$$

där funktionen $f(\cdot)$ skiljer mellan olika metoder. I *D'Hondt-metoden* är

$$f_{\text{hondt}}(m_i) = m_i + 1, \tag{2}$$

vilket innebär att vi kommer dela antalet röster med antalet mandat som partiet fått hittills plus ett. I *uddatalsmetoden* (även kallad Sainte-Laguës metod) är

$$f_{\text{udda}}(m_i) = 2m_i + 1, \tag{3}$$

så vi delar med dubbla antalet mandat plus ett. I Sverige används den *jämkade uddatalsmetoden* där jämförelsetalet för partier som är utan mandat har minskats genom att dela med 1.4 istället för 1:

$$f_{\text{jmk}}(m_i) = \begin{cases} 1.4, & \text{om } m_i = 0, \\ 2m_i + 1, & \text{om } m_i > 0. \end{cases} \tag{4}$$

Det parti som har det största jämförelsetalet i (1) får nästa mandat. Ifall detta är parti j så ökar man då deras mandat: m_j ersätts med $m_j + 1$. Jämförelsetalet räknas om enligt (1) och därefter delar man ut nästa mandat på samma sätt, till det parti som nu har det största jämförelsetalet. Denna sekventiella process pågår tills alla mandat har delats ut.

Ofta finns det även en ”spärr” där ett parti behöver få en viss procent av alla rösterna för att ens få ett mandat. I Sverige är spärren 4% i val till riksdagen och EU-parlamentet. I val till landsting så spärren på 3% och till kommunfullmäktige finns ingen spärr.

2.2 Visualisera valresultat

Börja med att skriva en funktion som skapar ett diagram över ett valresultat. Låt funktionen ta en matris med röstantal som inargument. Varje kolumn i

matrisen motsvarar ett parti och varje rad motsvarar ett visst valdistrikt (t.ex. Linköping i Östergötlands län). Varje element är antalet röster på ett givet parti i ett givet valdistrikt. Er funktion kan, exempelvis, redovisa valresultatet som ett cirkeldiagram och/eller ett stapeldiagram, gärna i procentform.

Testa din funktion på valresultatet från EU-parlamentsvalet från 2014, som finns i filen `euvalet2014.mat` på kursens hemsida. Kolumnerna motsvarar i tur och ordning partierna M, C, FP, KD, PP, MP, SD, FI, S, V och Övriga. Stämmer ert diagram med det slutliga valresultatet? Detta kan ni hitta hos Valmyndigheten:

`http://www.val.se/val/ep2014/slutresultat/E/rike/index.html`

För att göra resultatet mer läsbart så kan funktionen ha ett inargument med partinamn som kan användas i plottarna. Det går att lagra textsträngar i matriser i MATLAB, men det är lite omständligt. Det är bättre att skriva det som en så kallad cell-struktur:

```
>> parties = {'M','C','FP','KD','PP','MP','SD','FI','S','V','Other'};
```

När partinamnen är lagrade på detta sätt så är det ganska enkelt att använda det ihop med cirkel- och stapeldiagram, även om tillvägagångssättet är lite olika. Se dokumentationen för vardera funktionen.

2.3 Dela ut ett mandat med D'Hondt-metoden

Skriv en funktion som delar ut ett mandat med den enklaste av de tre metoderna: D'Hondt. Funktionen tar antalet röster per parti ($[r_1 \ \ldots \ r_n]$) och antalet mandat som partierna fått hittills ($[m_1 \ \ldots \ m_n]$) som inargument. Funktionen ska returnera indexet (ett tal från 1 till n) för det parti som har störst jämförelsetal enligt (1) och (2).

Exempel:

- Röstfördelningen [6 19 5 10 4] och mandatfördelningen [0 2 0 1 0] ger jämförelsetalen [6 6.33 5 5 4]. Parti 2 har störst jämförelsetal och därför ska funktionen returnera indexet 2.

2.4 Dela ut t mandat med D'Hondt-metoden

Skriv en ny funktion som delar ut ett visst antal mandat, t, med D'Hondt-metoden. Funktionen tar antalet röster per parti ($[r_1 \ \ldots \ r_n]$) och t som inargument. Funktionen ska returnera en vektor med antalet mandat som varje parti får. Ni kan antingen dela ut mandaten, en efter en, genom att anropa funktionen från föregående avsnitt. Eller så kan ni skapa en ny funktion som sköter hela mandatfördelningen. Ledning: Vilken typ av programmeringsstruktur använder man för sekventiella metoder? Kontrollera att elementen i den returnerade vektorn summerar till t.

Exempel:

- Röstfördelningen [6 19 5 10 4] och $t = 10$ ska ge mandatfördelningen [1 5 1 2 1]. Jämförelsetalen inför ett elfte mandat ska vara [3 3.17 2.5 3.33 2]. Verifiera också att $1 + 5 + 1 + 2 + 1 = 10$ vilket motsvarar värdet på t.

2.5 Dela även ut mandat med uddatalsmetoden och jämkade uddatalsmetoden

Nästa steg är att utveckla funktionen från föregående avsnitt så att den även hanterar uddatalsmetoden och jämkade uddatalsmetoden. Förslagsvis så kan funktionen få ett nytt inargument där man väljer vilken av metoderna som ska användas. Sedan är det bara att byta ut de rader av koden där jämförelsetalen räknas ut så att formeln beror på vilken metoder som blivit vald (se (1)–(4)).

Exempel:

- Röstfördelningen [6 19 5 10 4] och $t = 5$ ska ge olika mandatfördelningar och jämförelsetal inför ett sjätte mandat.
  D'Hondt-metoden ska ge mandatfördelningen [1 3 0 1 0] och jämförelsetalen [3 4.75 5 5 4].
  Uddatalsmetoden ska ge mandatfördelningen [1 2 1 1 0] och jämförelsetalen [2 3.8 1.67 3.33 4].
  Jämkade uddatalsmetoden ska ge mandatfördelningen [1 3 0 1 0] och jämförelsetalen [2 2.71 3.57 3.33 2.86].

2.6 Spärr för småpartier

I många fall finns det en särskild spärr mot småpartier, som måste uppnå ett visst antal procent av rösterna för att kunna få mandat. Lägg till denna procentspärr som inargument där det behövs. Ledning: Ett enkelt sätt att lägga in spärren är att kolla vilka partier som ligger under spärren och sätta deras jämförelsetal till noll. Fundera på hur du lättast kontrollerar om spärren fungerar som den ska.

2.7 Visualisera mandatfördelningen

Utöka nu funktionen som ni skrev i avsnitt 2.2 så att den löser hela uppgiften, som den beskrevs i början av dokumentet (gå tillbaka och läs avsnitt 1 igen!).

De tilltänkta inargumenten till din slutgiltiga funktion är valresultatet (`votes`), partinamnen (`parties`), procentspärren (`minpercent`) och antalet mandat som ska fördelas (`totalseats`). Ni kan döpa funktionen och utargumenten på valfritt sätt som beskriver dess innehåll för en potentiell användare (den ska alltså inte heta `uppgift2.7.m` eller liknande). Ett förslag är att döpa och definiera funktionen på följande vis:

```
>> [seats,quotients] = electionresults(votes,parties,minpercent,totalseats)
```

där `seats` är den slutliga mandatfördelningen med vardera metoden och `quotients` är de slutliga jämförelsetalen.

Er slutgiltiga funktion i projektet ska alltså räkna ut mandatfördelningen med de tre metoderna och plotta resultatet i diagramform. Resultatet ska bestå av en bild med stapeldiagram och en bild med cirkeldiagram. För varje diagramtyp ska du använda subplottar för att demonstrera alla metoderna i samma bild. Glöm inte att lägga in partinamnen i bilderna och att skriva förklarande texter på axlarna och med `legend`. Målet är, som alltid, att bilderna ska vara lätt att läsa och förstå!

Ni kan fundera över hur de olika metoderna hanterar stora och små partier. Ni kan utgå från resultatet i EU-parlamentsvalet och förändra antalet mandat (som ursprungligen är 20) och/eller ta bort procentspärren (som normalt är 4%). En annan intressant fråga är: hur många mandat verkar behövas för att avrundningsfelen ska bli försumbara?

2.8 Testa funktionen för ett annat val

Slutligen ska ni importera resultat från något annat val för att testa er funktion. Ni kan hitta sådan statistik på `http://www.val.se/valresultat/`

Det kan förslagsvis vara från tidigare EU-val eller från din hemkommun. Ta reda på hur många mandat som finns att fördela eller testa resultatet med olika antal. Ni ska vara beredda att visa upp detta för assistenten när ni redovisar projektet.

2.9 Förberedelse inför redovisning

Som en förberedelse inför att presentera projektet bör ni läsa igenom avsnittet ”Redovisning” i början av detta dokument samt läsa examinations- och kodningsstil-guiden som ni hittar på kurshemsidan. På dessa ställen framgår saker som att ni måste ha minst två funktioner i er kod, funktioner och variabler måste ha beskrivande namn, det ska finnas en rimligt mängd kommentarer i koden, alla bilder ska vara självförklarande, kommentarer ska vara på engelska, etc. Om det är något av kraven i examinations- och kodningsstil-guiden som ni *inte* uppfyller så bör ni åtgärda detta *innan* ni redovisar projektet, annars kanske ni får vänta länge på att få en andra redovisningschans.

Slutligen: Glöm inte att skicka in koden till Urkund, när ni blivit godkända. Instruktionerna finns i avsnittet ”Redovisning” i början av detta dokument.

3 Frivillig extrauppgift

I det här avsnittet beskrivs en frivillig extrauppgift. Ifall ni är vana programmerare och känner att projektet var för enkelt så rekommenderar vi att ni även gör denna uppgift, så att ni lär känna MATLAB ordentligt.

Valtekniskt samarbete: Avrundningseffekterna i mandatfördelningen kan ibland få avgörande konsekvenser. Två småpartier kan vardera få 3% av rösterna och inte nå över procentspärren. Om partierna däremot hade samarbetat och kunde tillgodoräkna sig alla 6% så skulle de (kanske) blivit tilldelade ett mandat.

Er extrauppgift är att skriva en funktion som undersöker alla kombinationer där två av partierna samarbetar. Ett samarbete innebär att ni summerar partiernas röster och hanterar dem som ett parti. Under parollen "tillsammans är vi starkare" ska ni undersöka vilka samarbeten som leder till att de gemensamma mandaten blir fler än partierna skulle haft individuellt.

Ett naturligt ytterligare steg är att leta efter kombinationer där det är tre eller fyra partier som samarbetar för att öka det gemensamma antalet mandat. Ni kan även skriva en funktion som hittar dåliga kombinationer där partierna skulle få färre mandat ifall de samarbetade. Slutligen kan ni hålla utkik efter kritiska situationer där ett antal partier bara skulle kapa åt sig en majoritet av mandaten ifall de samarbetade – eller skulle förlora sina majoritet av mandaten om de samarbetade.

Välja ut några intressanta exempel, gärna från verkligheten, och plotta dessa på lämpligt sätt.