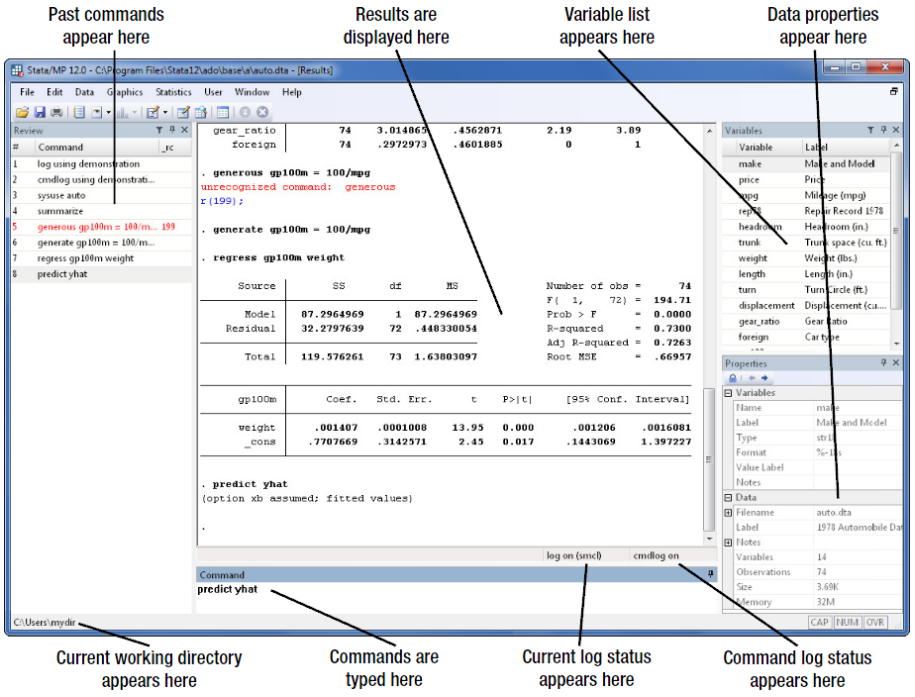
Kom igång med Stata

Introduktion

Stata är det vanligaste statistikprogrammet bland de på institutionen som bedriver mycket kvantitativ forskning. Det är relativt enkelt att lära sig, samtidigt som det finns inbyggt stöd för de flesta modeller vi använder och goda möjligheter att arbeta mer effektivt i takt med att man lär sig programmet och dess programmeringsspråk. Tyvärr är det ett dyrt program, vilket gör det svårt att använda i metodundervisningen. Det här är en kort introduktion till Stata, anpassad för studenter och andra som vill lära sig de absoluta grunderna.



Figur 1: Statafönstret

Det finns i grunden två olika sätt att använda Stata. Antingen använder man menyerna i verktygsfältet eller skriver man alla kommandon i Statafönstret *Command*. Oavsett om man använder menyerna eller kommandofönstret kommer alla kommandon att dyka upp som kod i fönstrena *Review* och *Results*. Reviewfönstret spar alla utförda kommandon, så att man kan utföra dem på nytt genom att klicka på dem. Resultsfönstret visar kommandot följt av vilket resultat det gav.

Många tycker att det här med kod är krångligt och börjar med att använda menyerna. Det har också fördelen att man kan bläddra bland de olika alternativen för att få en bättre känsla för vilka valmöjligheter som finns. Att skriva kod är dock både snabbare och mer flexibelt, så det är en bra idé att successivt försöka lämna menysystemet så gott det går. Du kommer också märka att all dokumentation och alla diskussioner på internet utgår från koden, snarare än var i menyerna man ska klicka. Den här introduktionen lär därför ut hur man skriver sina första rader kod, men det går alldeles utmärkt att kombinera de kunskaperna med att rota i menysystemet.

Få hjälp

Varje Statakommando har en dokumentation som du når genom att skriva `help` följt av kommandots namn. Om man bara skriver `help` kommer man till en innehållsförteckning för dokumentationen. Statas hjälpfunktion är en ovärderlig källa till information, men i början kan det vara svårt att ta till sig vad som står där. Dessutom är det svårt att hitta rätt om man inte vet namnet på kommandot man vill lära sig.

Det finns gott om forum att ställa frågor på, men snabbast är ofta att googla fram svaren på någon annans fråga. Bland de vanligaste och mest pålitliga träffarna är Statas FAQ (stata.com/support/faqs/), Statas forum (statalist.org) och forumet Stackoverflow. UCLA har också en omfattande resurssida med bra FAQ (www.ats.ucla.edu/stat/stata).

I Stata ingår det flera övningsdataset som ofta används för att ge tips och råd. Det vanligaste av dem innehåller information om 74 sålda bilar av årsmodell 1978. Alla exempel i den här texten förutsätter att du har öppnat detta dataset i Stata, vilket du gör genom att skriva:

```
sysuse auto, replace
```

Statas syntax

De flesta Statakommandon är inte svårare än att man skriver kommandots namn följt av namnet på den eller de variabler som kommandot ska utföras på. Såhär kan vi exempelvis skriva för att ta fram beskrivande statistik på försäljningspriser:

```
summarize price
```

Vill vi i stället utföra en regression av hur priset påverkas av hur långt bilen gått och om det är ett amerikanskt eller utländskt märke skriver vi:

```
reg price mpg foreign
```

För att kunna göra lite mer avancerade saker är det dock bra att bekanta sig med Statas syntax, alltså de regler som anger hur man i ett givet programmeringsspråk sammanfogar olika kommandon och symboler till meningsfulla satser. De flesta Statakommandon följer nedanstående struktur:[1]

```
[prefix:] command [varlist] [= exp] [if] [using] [, options]
```

Det framstår lätt som onödigt omständligt att börja med en komplicerad syntax, men om man förstår hur de olika beståndsdelarna hänger ihop blir det enklare att utvecklas och ta till sig nya saker. Nedan följer en genomgång av de olika delarna. I regel används bara några av dem i samma sats.

`prefix`	Det finns många prefix som anger hur kommandot ska köras. Det vanligaste är `by varlist`, vilket anger att kommandot ska upprepas för varje delmängd observationer såsom definieras av värdena på `varlist`. Det kräver att datasetet är sorterat efter `varlist`, vilket enklast åstadkoms genom att ersätta `by` med `bysort`. Skriv `help prefix` för att se fler prefix.
`varlist`	En variabellista är oftast bara en eller flera variabler separerade med blanksteg. Skriv `help varlist` för fler möjligheter.
`command`	Den enda beståndsdel som alltid måste ingå är det kommando som ska användas.
`= exp`	Många kommandon följs av ett uttryck (expression). Exempelvis kan vi skapa en variabel som anger det kvadrerade priset genom att skriva `generate weight2 = weight^2`, där `weight^2` utgör ett uttryck.
`if`	Genom att ange ett eller flera villkor som måste vara uppfyllda kan vi exempelvis begränsa vilken del av ett dataset som ska inkluderas. Använd de logiska operatorerna `&`, `|` och `!` samt relationsoperatorerna `>`, `<`, `>=`, `<=`, `==` och `!=`. För att få beskrivande statistik av priset på endast amerikanska bilar kan vi skriva `summarize price if foreign == 0`.
`using`	Om kommandot refererar till en fil på datorn så anges den efter `using`.
`options`	Varje kommando kommer med en uppsättning valmöjligheter, vilka anges efter ett kommatecken i slutet av kommandot. Använd `help command` för att se vilka de är.

[1] Nu uteslöt jag `[in]` och `[weight]`, vilket är de delar jag själv använder mest sällan.

De flesta kommandon och variabelnamn kan också förkortas. Exempelvis kan vi skriva `reg` i stället för `regress`, så länge det inte finns något annat kommando som börjar på `reg`.

Klicka på ett tidigare kommando eller en variabel för att klistra in det i kommandofönstret. CTRL+R hämtar automatiskt det senaste kommandot, medan PageUp och PageDown låter dig bläddra bland dem.

Bra kommandon

Här har jag samlat de av Statas kommandon som jag tror att nya användare har störst användning för. Namnet på kommandot är fetstilat och det följs av ett förenklat exempel på hur det kan användas. Använd Statas hjälpfunktion för att läsa mer om kommandot.

regress var1 var2 var3	Regression av var1 på var2 och var3.
summarize var1 var2	Beskrivande statistik för var1 och var2.
replace var1 = var2/var3	Ersätt variabelvärdena för var1 med var2 genom var3.
generate var1_sq = var1^2	Skapa en variabel var1_sq som är var1 i kvadrat.
list var1if var3 == 1	Visa var1 för alla observationer där var3 är 1.
sort var1 var2	Sortera observationerna efter var1 och därefter var2.
order var1 var2	Sortera variablerna med var1 först och var2 därefter.
tabulate var1 var2	Tabell över hur värdena på var1 fördelar sig över värdena på var2.
line var1 var2	Linjediagram med var1 på y-axeln och var2 på x-axeln. Sortera först på var2.
scatter var1 var2	Punktdiagram med var1 på y-axeln och var2 på x-axeln.
histogram var1	Histogram över var1.
gr bar var1, over(var2)	Stapeldiagram för medelvärden av var1 fördelat på var2.
use mindata, replace	Öppna filen mindata.dta i arbetsfoldern och ersätt data i Stataminnet.

`save mindata, replace`	Spara filen mindata.dta och ersätt fil med samma namn.
`egen var1 = mean(var2)`	Skapa en variabel var1 som är medelvärdet av var2.
`drop var1 var2`	Ta bort var1 och var2.
`drop if var1 == 1`	Ta bort alla observationer där var1 är 1.
`destring var1, replace`	Gör om strängvariabeln var1 till en numerisk variabel (var1 måste bestå av siffror).
`encode var1, gen(var1enc)`	Skapa en numerisk variabel var1enc av strängvariabeln var1 (var1 bör inte bestå av siffror).
`rename var1 var1new`	Byt namn på var1 till var1new.
`label var var1 "Variabel 1"`	Byt label på var1 till Variabel 1.
`edit`	Öppna dataeditorn.
`inspect var1 var2`	Beskrivande statistik för var1 och var2 i syfte att hitta misstänkta fel.
`cd D:/Dropbox`	Ange arbetsmapp.
`cond gen var1 = cond(var2<5,1,0,.)`	Skapa var1 med värdet 1 om var2 < 5.

Missing values

I flera avseenden hanterar Stata saknade värden som om de var oändligt stora värden, vilket ofta orsakar problem för mindre erfarna användare. Låt oss anta att vi analyserar enkätdata och bara vill ha kvar personer över 50 år. Om vi skriver `keep if age > 50` kommer vi även att inkludera personer vars ålder vi inte känner till, vilket förmodligen inte är vad vi avsåg. I stället måste vi skriva `keep if age > 50 & age <.` för att även exkludera personer vars ålder vi inte vet.

Anledningen till Statas beteende är att om missing values varken var små eller stora värden, utan en tredje kategori, skulle ett kommando som `keep if age > 50` inte längre ge samma resultat som `drop if age <= 50`. Det är således logiskt omöjligt att skapa ett program som är konsekvent samtidigt som det i båda dessa situationer gör vad användaren förväntar sig.

Min databas

Jag har en stor samling av variabler som man söka i på parnyman.com/data. Varje variabel består av en .dta-fil, vilket gör det enkelt att importera dem till Stata. Alla variabler med land-år (country-years) som analysenheter finns sparade på adressen paneldata.se/cy/. Om vi arbetar i ett dataset som innehåller variablerna year och id, vilken anger land enligt den numeriska standarden ISO 3166-1, kan vi använda Statas kommando `merge`.

```
merge m:m id year using http://paneldata.se/cy/varname.dta, nogen
```

Jag har skrivit ett enkelt script som först skapar ett tomt dataset enligt riktlinjerna ovan och därefter skapar ett program som heter get. Programmet används för att med så lite kod som möjligt importera variabler från min databas. För att köra detta skript, skriv följande i Stata (observera att allt du har i Stataminnet försvinner!).

```
do http://www.paneldata.se/scripts/cy.do
```

Hela koden som utgör scriptet ser ut såhär.

```
use http://paneldata.se/cy/cy.dta, clear
tsset id year
expand 71
sort id year
replace year = year[_n-1] +1 if id == id[_n-1]

capture program drop get
program get
local importvars `*'
foreach x in `importvars' {
clear mata
merge m:m id year using http://paneldata.se/cy/`x'.dta, nogen
}
sort id year
end
```