Bra att veta om matematik med Sage

Anders Martinsson

14 januari 2016

1 För att skriva uttryck

1.1 Definiera variabler och funktioner

Sage vet att x är en variabel i matematiska sammanhang men övriga variabler behöver definieras som variabler för att Sage ska tolka dem rätt.

Funktioner där det definierande uttrycket är känt definieras som vanligt i matematik, alltså exempelvis `f(x) = 3*x + 5`. Ibland behöver dock en funktion med okänt uttryck definieras, som till exempel före lösning av differentialfunktioner.

var("text") Funktionen används för att definiera nya variabler. Det som ska representera variabeln skrivs inom citationstecken (enkla eller dubbla spelar ingen roll). Exempelvis definierar `var("y")` y som en variabel som sedan kan användas i matematiska uttryck och beräkningar. (Om definitionen gör med en tilldelning: `y=var("y")` blir det ingen utskrift i resultatet av variabeldefinitionen.)

Sage utgår från att variabeln är ett komplext tal om inget annat anges. För att bestämma domänen för variabeln anges detta med `var("namn", domain="domän")`. Ex: `var("y", domain="real")` anger att y är ett reellt tal. Möjliga domäner är: `complex, real, positive` eller `integer`.

function("namn")(var) För att ange nya funktioner med okänt uttryck används `fnk = function("namn")(var)`. Exempelvis för att ange y(x) skrivs `y=function("y")(x)`.

1.1.1 Vektorer

En tvådimensionell vektor v definieras med `v=vector([x,y])` där x och y är komposanterna. Skalärmultiplikation av två vektorer u och v görs genom `u.dot_product(v)`.

1.2 = och ==

Sage använder vanliga = som tilldelningstecken. (Detta förekommer i många programmeringsspråk.) Alltså betyder `g=10` att variabeln g tilldelas värdet 10. För matematisk ekvivalens används == , så ekvationen $5x + 3 = 28$ skrivs i Sage `5*x+3==28`. De båda kan kombineras på en rad:

```
ekv=5*x+3==28
```

Ovanstående tilldelar variabeln ekv ekvationen 5x+3=28. Notera att variabeln kommer att innehålla hela ekvationen!

1.3 Funktioner, operatorer med mer

Nedan listas hur några vanliga matematiska funktioner, operationer och andra symboler skrivs i Sage.[1] Notera att Sage skiljer på stor och liten bokstav och att vinklar anges i radianer.

$\sin x \ldots$	sin(x) ...	$\int_a^b f(x)dx$	integral(f(x), x, a, b)
$\arcsin x \ldots$	asin(x) ...	$\int f(x)dx$	integral(f(x), x)
$\sqrt{(x)}$	sqrt(x)	$\sum_{k=1}^{n} f(k)dx$	sum(f(k), k, 1, n)
x^y	x^y	$n \pmod m$	mod(n, m)
$\lg x$	log(x, 10)	sgd(n, m)	gcd(n,m)
$\ln x$	log(x)	mgm(n, m)	lcm(n,m)
$n!$	factorial(n)	$\binom{n}{k}$	binomial(n,k)
$f'(x)$	diff(f(x), x)	π	pi
$f^n(x)$	diff(f(x), x, n)	$x \cdot 10^n$	xen (ex: 5e-3)
e	e	i	I eller i
∞	oo	$\|x\|$	abs(x)

1.3.1 Komplexa tal

Komplexa tal kan anges i rektangulär form, `(a+bI)`, polär form eller exponentiell. Nedan följer några vanliga matematiska operationer med komplexa tal och hur de skrivs i Sage.

Re(z)	real(z)	Im(z)	imag(z)
$\|z\|$	abs(z)	arg(z)	arg(z)
$\overline{z}$	conjugate(z)	$\|z\|^2$	norm(z)

1.3.2 Begränsa variabelintervall

För att beränsa intervallet för en variabel kan `assume()` användas. I nedanstående exempel beränsas k till $-1 < k < 1$ för att finna den geometriska summan då $n \to \infty$.

```
var("a, k, n")
assume(abs(k)<1)
sum(a*k^n, n, 0, oo)
```

[1] Många funktioner kan skrivas på flera sätt. Här anges ett.

1.3.3 Ange värden på konstanter i funktioner

Numeriska värden på konstanter i funktioner kan anges enligt följande exempel:

```
var("k,m")
y=cos(sqrt(k/m)*x)
plot(y(k=2, m=1), x, (0, 2*pi))
```

2 Lösa ekvationer och göra beräkningar

Enkla beräkningar av typen $125 \cdot 456$ eller $\int_0^3 (x^2 - 5)dx$ görs genom att skriva in uttrycket i beräkningsrutan och klicka på Run (eller Evaluate). För lösning av olika ekvationer finns speciella funktioner.

Sage arbetar med symbolisk algebra. Det betyder att uttryck, resultat av beräkningar med mer hanteras som matematisk text. Det skiljer Sage från miniräknare som normalt endast hanterar värden från beräkningar. Resultat som kan skrivas exakt kommer att presenteras exakt. Ex:

$$sin(\frac{\pi}{3}) = \frac{\sqrt{3}}{2}$$

n(uttryck) Funktionen ger det numeriska resultatet av en beräkning. Med `n(sin(pi/3))` fås resultatet `0.866025403784439`. Antal värdesiffror kan anges med `n(värde, digits=antal)`. Ex: `n(pi, digits=100)` ger π med 100 siffror.

2.1 Finna lösningar till ekvationer

Funktionen `solve(ekv, x)` används för vanliga algebraiska ekvationslösningar. I solve() anges ekvationen som ska lösas och för vilken variabel. Ex: `solve(5*x+3 == 28, x)` ger resultatet `[ x == 5]`. (Notera att resultatet är hela uttrycket x==5.)

Ibland ger standardinställningarna av `solve()` inte de resultat som förväntas. Till exempel ger användning av `solve(sin(2*x)==1/2, x)` endast en av lösningarna, nämligen $\pi/12$. För att få alla lösningar till ekvationen måste solve instrueras att använda en annan lösningsmotor än standardmotorn. Det görs med argumentet `to_poly_solve ='force'`.[2] Kommandot blir då

```
solve(sin(2*x)==1/2, x, to_poly_solve='force').
```

Vilket ger resultatet:

```
[x == 1/12*pi + pi*z37, x == 5/12*pi + pi*z39]
```

där z37 och z39 är okända heltal.

[2] Notera att poly_solve ofta är betydligt långsammare än standardmetoden samt ger något annorlunda formulerade resultat.

2.2 Arbeta med uttryck

expand(uttyck) Funktionen skriver om faktoriserade uttryck som termer så långt det går. Ex: `expand(x^3*(5+x))` ger $x^4 + 5x^3$. Funktionen är även användbar för att snygga till lösningar till ekvationer eller differentialekvationer.

factor(uttryck) Funktionen faktoriserar uttryck så långt det går. Exempelvis `factor(x^2+2*x+1` ger $(x+1)^2$.

divisors(uttryck) Funktionen finner alla delare till uttrycket. Fungerar både för tal och polynom.

limit(uttryck, var=limit) Funktionen finner gränsvärdet då $var \to limit$. Ex: `limit(3*x/(x+5), x=oo)` ger resultatet 3.

2.2.1 Numeriskt bestämma rötter och integraler

find_root(f(x), xmin, xmax) Funktionen finner roten $f(x) = 0$ i ett intervall $x_{min} \leq x \leq x_{max}$ med numerisk metod.

numerical_integral(f(x), a, b) En numerisk lösning av integraler görs och resultatet ges som ett värde och ett beräknat fel.

2.3 Lösning av differentialfunktioner

För symbolisk lösning av differentialekvationer används `desolve(ekv, funk, randvillkor)`. Funktionen kan användas med och utan randvillkor. Se exempel nedan. Notera att y måste deklareras som en funktion av x före lösning av ekvationerna, se ovan.
Utan randvillkor:

```
desolve(diff(y,x)+y==0, y)
```

Med randvillkoret $y(0) = 4$:

```
desolve(diff(y,x)+y==0, y, [0,4])
```

Med randvillkoren $y(2) = 3$ och $y'(2) = 4$:

```
desolve(diff(y,x,2)+diff(y,x)+y==0, y, [2,3,4])
```

Om differentialfunktionen innehåller konstanter som inte är tal behöver även den oberoende variabeln anges med argumentet `ivar`. Ex:

```
desolve(diff(U,t)+k*U==0, U, ivar=t)
```

2.3.1 Numerisk lösning av differentialekvationer

Det finns flertalet numeriska metoder att välja på för lösning av differentialekvationer. För första ordningens differentialekvation används ofta Runge-Kutta metoden. Följande funktion löser en differentialekvation med Runge-Kutta[3]:

[3]Det finns fler argument som kan användas än de som presenteras här. Se dokumentationen för Sage

`desolve_rk4(de, y, [x0,y0], end_points= xslut, output= ut)`
de – differentialekvationen
y – funktionen som ekvationen ska lösas för
[x0,y0] – startvärdena $y_0 = y(x_0)$
xslut – stoppvärdet för x. Lösningen ges för intervallet x_0 till x_{slut}
ut – Anger hur resultatet ska presenteras: "plot" skapar en graf, "list" en lista med koordinater och "slope_field" visar kurvan i ett riktningsfält.

Exempel:
`desolve_rk4(diff(y,x)==sqrt(y),y,[0,1],end_points=4, output="plot")`
Ovanstående ritar graf över lösningen till $y' = sqrt(y)$ då $y(0) = 1$ i intervallet x=0 till x=4.

3 För att arbeta med statistik och kombinatorik

Sage sparar data i listor. Listor är kommaseparerade element inom hakparenteser. Ex: `res=[1,3,5,7]` lagrar listan med 1, 3, 5, 7 i variabeln res. (Elementen i listan kan i sin tur vara vektorer eller listor.) Elementen i en lista är numrerade från och med 0 så res[1]=3. Några vanliga statistikoperationer på listor är:

mean(lista) Funktionen beräknar aritmetiska medelvärdet av elementen i listan.

median(lista) Funktionen beräknar medianen av elementen i listan.

std(l), variance(l) Funktionerna beräknar standardavvikelsen respektive variansen i listan l.

min(l), max(l) Funktionerna finner minsta respektive största värdet i listan l.

3.1 Slumptal

random() Funktionen genererar ett slumptal i intervallet 0 till 1.

randint(a, b) Funktionen genererar ett slumpat heltal i intervallet a till och med b.

shuffle(lista) Funktionen blandar elementen i listan.

3.2 Kombinatorik

Förutom beräkningar av fakultet och binomialkoefficienter, se ovan, finns flera andra verktyg för kombinatorik. Utifrån en lista kan till exmpel följande objekt skapas:

Arrangements(lista, k) Objektet innehåller alla ordnade urval (permutationer) som kan göras med k element ur listan.

Combiations(lista, k) Objektet innehåller alla oordnade urval (kombinationer) som kan göras med k element ur listan.

Tuples(lista, k) Objektet innehåller alla ordnade urval som kan göras av k element ur listan då upprepning av ett element är tillåtet.

UnorderedTuples(lista, k) Objektet innehåller alla oordnade urval som kan göras av k element ur listan då upprepning av ett element är tillåtet.

För att visa innehållet i något av ovanstående element används metoden `list()`. Ex: `Combinations([1, 2, 3],2).list()` ger resultatet:

```
[[1, 2], [1, 3], [2, 3]]
```

För att visa antalet element något av ovanstående element används metoden `cardinality()`. Ex: `Combinations([1, 2, 3],2).cardinality()` ger resultatet 3.

Ett element kan väljas i något av ovanstående objekt på samma sätt som i andra listor med [n]. Ex: `Combinations([1, 2, 3],2)[2]` ger resultatet `[2, 3]`.

3.3 Anpassa en linje eller kurva

Data från till exempel en mätning kan hanteras i form av en lista punkter som `data=[[x0, y0], [x1, y1], ..., [xn, yn]]`. En matematisk modell, till exempel en linje, kan även anpassas till datan. Se nedastående exempel på anpassning och plottning av data och regressionslinje:

```
data = [[0, 7.1], [1, 5.2], [2, 2.9], [3, 1.05], [4, -0.9]]
var("a b")
model(x) = a*x+b
fit=find_fit(data,model)
scatter_plot(data)+plot(model.subs(fit),(x,0,5))
```

Påsista raden plottas punkterna och den anpassade linjen i en graf (se mer om detta nedan). Med `model.subs(fit)` substitueras a och b i model med resultatet från anpassningen som finns lagrad i variabeln fit.

3.4 Läsa data från en fil

Sage kan läsa in en lista från textfiler, till exempel CSV-formaterade filer[4]. Nedastående kommandon läser in data från test.csv som som finns i samma katalog som Sage-filen och lagrar den inlästa datan i variabeln data.

```
import csv
```

[4]CSV står för kommaseparerade variabler där variabler står i tabellform med komma (eller annat separationstecken) mellan varje kolumn.

```
data=list( csv.reader(open('test.csv','rU')) )
```

4 För att skriva text och skapa snygga resultat

4.1 Prydligare matematik

show() Funktionen show() används för att få resultatet presenterat formaterat med LaTeX. Exempelvis `show(5*x+3==28)` ger resultatet:

$$5x + 3 = 28$$

\# Brädgård inleder en kommentar i beräkningsrutor. Ex:

```
# Detta är en kommentar
```

print() För att skriva texter i resultatet kan funktionen print() användas. Texten som ska skrivas ut anges inom citationstecken. Ex: `print("Maxmala värdet:")` ger en rad i resultatet med texten `Maximala värdet:`.

4.2 Textrutor

För att skapa en ny textruta används Shift+Click på en inmatningsrad. För att redigera en tidigare skriven text används dubbelklick. Textrutor inleds alltid med %html på första raden så låt detta stå kvar. (En textruta kan också skapas genom att skriva %html ensamt på första raden i inmatningsrutan.)

I textrutor används HTML-kod för formatering. Ex: `<br>` för en radbrytning och ett stycke skrivs mellan `<p>` och `<
p>`. För att skriva snygg matematik används formatering med LaTeX. Läs mer om matematik i LaTeXhär: https://en.wikibooks.org/wiki/LaTeX/Mathematics

5 Om att rita grafer

Sage har många olika funktioner för grafritning. Här är ett litet urval.

5.1 Rita funktioner

Vanliga funktionsgrafer ritas med funktionen `plot()`. Det finns många olika sätt att ange hur grafen ska ritas. Här är några exempel.

Ritar funktionen f(x) för $-5 \leq x \leq 5$:

```
plot(f(x), -5, 5)
```

Ritar funktionen f(x) för $-5 \leq x \leq 5$ med röd kurva:

```
plot(f(x), -5, 5, color="red")
```

Ritar funktionen f(x) för $-5 \leq x \leq 5$ med streckad kurva:

```
plot(f(x), -5, 5, linestyle= "--")
```

Ritar funktionen f(x) för $-5 \leq x \leq 5$ med fasta gränser för y-axeln $-5 \leq y \leq 5$:

```
plot(f(x), xmin=-5, xmax=5, ymin=-5, ymax=5)
```

Ritar funktionen f(x) för $-5 \leq x \leq 5$ med tätt rutnät:
`plot(f(x), -5, 5, gridlines="minor")`

Flera funktioner kan kombineras i en graf genom att addera plotfunktioner. Till exempel ritar denna funktionen och dess derivata i samma graf:
`plot(f(x),-2,2, color="blue")+plot(diff(f(x)),-2,2, color="red")`

5.2 Rita vektorer, punkter mm

Vektorer ritas med funktionen `plot(vektor)`. Vektorer ritas då med utgångspukt i origo. För att rita vektorer med en annan utgånspunkt än origo används `plot(vektor2), start=startvektor`. Till exempel kan två vektorer v och u ritas efter varandra med:
`plot(v)+plot(u, start=v)`

arrow() Funktionen `arrow((xstart, ystart), (xslut, yslut))` ritar en pil i ett koordinatsystem. Liksom `plot()` går `arrow()` att addera med andra grafritningskommandon.

point() Funktionen `arrow((x,y))` ritar en punkt i ett koordinatsystem. Ex: `point((1,2), color="red")` ger en röd punkt i x=1 och y=2. Även `point()` att adderas med andra grafritningskommandon.

text() Text kan skrivas i en graf med `text("text", (x,y))`

5.3 Plotta data

5.3.1 Endimensionell lista

Ett histogram över en lista med data kan plottas med `histogram(lista, bins=n, range=[min, max])`. Se nedanstående exempel:
`histogram([1.5, 2.2, 3.3, 3.0, 4.2], bins=5, range=[0,5])`

5.3.2 Lista med punkter

För att plotta datapunkter lagrade i en lista med x- och y-koordinater kan antingen `scatter_plot(lista)` användas som ger möjlighet att med ytterligare argument även definiera punkternas utseende.

Alternativt kan `list_plot(lista)` användas. Ett linjediagram kan ritas med `list_plot(lista, plotjoined=True)`.

5.4 3D-grafer

5.4.1 Rita rotationskroppar

Rotationskroppar kan ritas med i en 3D-graf med kommandot
`revolution_plot3d(f(x),(x,xmin,xmax), parallel_axis="axel")`.

Rotationsaxeln anges som x, y eller z. För att även se funktionen f(x) används argumentet `show_curve=True`. Exempelvis visar nedanstående kurvan för $\sqrt(x)$ och rotationskroppen roterad kring x-axeln:

```
revolution_plot3d(sqrt(x),(x,0,2), parallel_axis="x", show_curve=True)
```

6 Hantera objekt och inställningar

6.1 Metoder för hantering av uttryck

6.1.1 Att få ut talet från resultat av typ [x==12]

Resultatet `[x==12]` är bokstavligen hela x==12 inte bara talet 12. För att komma åt själva talet används metoden rhs(), rhs är förkortning för right hand side. Notera att rhs() är en metod[5] och skrivs .rhs() efter det resultat där högersidan ska extraheras. Exempel:

```
res=solve(x+5==0, x)
res[0].rhs()*4
```

Ovanstående ger resultatet -20. Resultaten, i detta fall ett, lagras i res som en lista, res[0] innehåller den första lösningen i listan av lösningar (alltså `x== -5`) och .rhs() extraherar talet -5 vilket multipliceras med 4.

För att få ut vänsterledet ur en ekvivalens (==) används på samma sätt `.lhs()`.

6.1.2 Substituering av variabler vid symbolisk beräkning

Metoden `substitute()` används för att byta ut en variabel mot en annan eller mot ett uttryck. Ex: `(x+5==0).substitute(x=sin(x))` ger resultatet $sinx + 5 = 0$.

6.2 Inställningar

Det går att med olika kommandon förändra hur Sage arbetar. Dessa kommandon kan till exempel köras i början av en arbetsbok för att ställa in arbetssättet vid de följande beräkningarna.

6.2.1 Symbolisk integration av $\frac{1}{x}$

Integralen av $\frac{1}{x}$ är $\ln|x|$. Dock ger Sage, liksom många andra matematikprogram, svaret `log(x)` vid denna integration, att det ska vara absolutvärdet lämnas till användaren att förstå. Dock finns det en inställning för att få svaret inklusive absolutvärdesfunktionen. Kommandot är `maxima_calculus.eval(" logabs:true")`.

[5] För att vara riktigt korrekt är de flesta funktioner som nämnts här metoder och kan skrivas med punkt efter objekt men för dig som användare är det viktiga att tex. rhs() endast fungerar för att få ut högersidan på resultat och måste skrivas efter resultatobjektet med .rhs()