Using Macaulay2 from within R: the m2r package

Christopher O’Neill

University of California Davis
coneill@math.ucdavis.edu

Joint with David Kahle and Jeff Sommars
Mathematics Research Communities on Algebraic Statistics

August 3, 2017
R: a statistician’s best friend

Data storage and manipulation, array calculations, data analysis, ...
R and Macaulay2

R: a statistician’s best friend

Data storage and manipulation, array calculations, data analysis, . . .

Macaulay2: an algebraic geometer’s best friend

Polynomial ideals, Gröbner bases, Hilbert functions, . . .
R: a statistician’s best friend

Data storage and manipulation, array calculations, data analysis, …

Macaulay2: an algebraic geometer’s best friend

Polynomial ideals, Gröbner bases, Hilbert functions, …

Algebraic statisticians: best of both worlds
Running Macaulay2 from R the old way

R version 3.3.0
...
>

```r
library("algstat")

code <- "R = QQ[x,y,z]
I = ideal(x^2, x*y, x^3*y^2)
gens gb I"

m2(code)
[1] "R"
[1] "ideal(x^2,x*y,x^3*y^2)"
[1] "matrix {{x*y, x^2}}"

m2Code.m2
f = "m2Out" << ""

f << toString( R = QQ[x,y,z] ) << endl
f << toString( I = ideal(x^2, x*y, x^3*y^2) ) << endl
f << toString( gens gb I ) << endl
f << close
```

Christopher O'Neill (UC Davis)
The m2r package
August 3, 2017 3 / 20
Running Macaulay2 from R the old way

R version 3.3.0
...

> library("algstat")

>

Christopher O'Neill (UC Davis)
The m2r package
August 3, 2017
Running Macaulay2 from R the old way

R version 3.3.0
...

> library("algstat")

> code <- "R = QQ[x,y,z]
  I = ideal(x^2, x*y, x^3*y^2)
  gens gb I"

>
Running Macaulay2 from R the old way

R version 3.3.0

... 

> library("algstat")

> code <- "R = QQ[x,y,z]
I = ideal(x^2, x*y, x^3*y^2)
gens gb I"

> m2(code)
[1] "R"
[1] "ideal(x^2,x*y,x^3*y^2)"
[1] "matrix {{x*y, x^2}}"
Running Macaulay2 from R the old way

R version 3.3.0
...

> library("algstat")

> code <- "R = QQ[x,y,z]
   I = ideal(x^2, x*y, x^3*y^2)
   gens gb I"

> m2(code)
[1] "R"
[1] "ideal(x^2,x*y,x^3*y^2)"
[1] "matrix {{x*y, x^2}}"

m2Code.m2

f = "m2Out" << ""
f << toString( R = QQ[x,y,z] ) << endl
f << toString( I = ideal(x^2, x*y, x^3*y^2) ) << endl
f << toString( gens gb I ) << endl
f << close
The m2r package in action

R version 3.3.0
...
>

library("m2r")
Loading required package: mpoly
Loading required package: stringr
please cite mpoly if you use it; see citation("mpoly")
M2 found in /usr/local/macaulay2/bin

> start_m2()
Starting M2... done.

> m2("1+1")
[1] "2"

> m2("a = 5")
[1] "5"

> m2("a")
[1] "5"
The m2r package in action

R version 3.3.0
...

> library("m2r")
Loading required package: mpoly
Loading required package: stringr
please cite mpoly if you use it; see citation("mpoly")
M2 found in /usr/local/macaulay2/bin

>
The m2r package in action

R version 3.3.0
...

> library("m2r")
Loading required package: mpoly
Loading required package: stringr
please cite mpoly if you use it; see citation("mpoly")
M2 found in /usr/local/macaulay2/bin

> start_m2()
Starting M2... done.

>
The m2r package in action

R version 3.3.0
...

> library("m2r")
Loading required package: mpoly
Loading required package: stringr
please cite mpoly if you use it; see citation("mpoly")
M2 found in /usr/local/macaulay2/bin

> start_m2()
Starting M2... done.

> m2("1+1")
[1] "2"

>
The m2r package in action

R version 3.3.0
...

> library("m2r")
Loading required package: mpoly
Loading required package: stringr
please cite mpoly if you use it; see citation("mpoly")
M2 found in /usr/local/macaulay2/bin

> start_m2()
Starting M2... done.

> m2("1+1")
[1] "2"

> m2("a = 5")
[1] "5"

>
The m2r package in action

R version 3.3.0
...

> library("m2r")
Loading required package: mpoly
Loading required package: stringr
please cite mpoly if you use it; see citation("mpoly")
M2 found in /usr/local/macaulay2/bin

> start_m2()
Starting M2... done.

> m2("1+1")
[1] "2"

> m2("a = 5")
[1] "5"

> m2("a")
[1] "5"
Under the hood: sockets

```
R
```

Christopher O'Neill  (UC Davis)  
The m2r package  
August 3, 2017  
5 / 20
Under the hood: sockets

- `m2_start()`
Under the hood: sockets

- m2_start()
- launch M2 process
Under the hood: sockets

- m2_start()
- launch M2 process
- wait for available connection
Under the hood: sockets

- `m2_start()`
- launch M2 process
- wait for available connection

- create server socket

Christopher O'Neill (UC Davis)
Under the hood: sockets

- m2_start()
- launch M2 process
- wait for available connection

- create server socket
- wait for client connection
Under the hood: sockets

- `m2_start()`
- launch M2 process
- wait for available connection

- create server socket
- wait for client connection

- connect to socket

Christopher O'Neill (UC Davis)
Under the hood: sockets

- `m2_start()`
- launch M2 process
- wait for available connection

- create server socket
- wait for client connection

- connect to socket
Under the hood: sockets

- `m2_start()`
- launch M2 process
- wait for available connection

- connect to socket
- wait for message from server

- create server socket
- wait for client connection
Under the hood: sockets

- m2_start()
- launch M2 process
- wait for available connection

- create server socket
- wait for client connection

- connect to socket
- wait for message from server

- send "1.0.0"
Under the hood: sockets

- m2_start()
- launch M2 process
- wait for available connection

- create server socket
- wait for client connection

- connect to socket
- wait for message from server

- send "1.0.0"
- wait for input from client
Under the hood: sockets

- `m2_start()`
- launch M2 process
- wait for available connection

- connect to socket
- wait for message from server

- receive "1.0.0"

- create server socket
- wait for client connection

- send "1.0.0"
- wait for input from client
Under the hood: sockets

- `m2_start()`
- launch M2 process
- wait for available connection
  - create server socket
  - wait for client connection
  - send "1.0.0"
  - wait for input from client
- connect to socket
- wait for message from server
  - receive "1.0.0"
  - verify version match
Under the hood: sockets

- \texttt{m2\_start()}
- launch \texttt{M2} process
- wait for available connection
- create server socket
- wait for client connection
- connect to socket
- wait for message from server
- send "1.0.0"
- wait for input from client
- receive "1.0.0"
- verify version match
- return from \texttt{m2\_start()}
Under the hood: sockets

wait for input from client
Under the hood: sockets

- m2("1+1")
  - send "1+1" to server
  - wait for response
  - receive "2" from server
  - return "2" from m2()

- wait for input from client

Christopher O'Neill (UC Davis)
Under the hood: sockets

R \rightarrow \text{Socket} \rightarrow \text{M2}

- \text{m2("1+1")}
- \text{send "1+1" to server}

\text{wait for input from client}
Under the hood: sockets

- `m2("1+1")`
- send "1+1" to server
- wait for response

- wait for input from client

Christopher O'Neill (UC Davis)
Under the hood: sockets

- \( m2("1+1") \)
- send "1+1" to server
- wait for response

- receive "1+1" from client
- wait for input from client

\( m2r \) package

Christopher O'Neill (UC Davis)
Under the hood: sockets

- m2("1+1")
- send "1+1" to server
- wait for response

- wait for input from client
- receive "1+1" from client
- evaluate "1+1" to "2"
Under the hood: sockets

- \texttt{m2("1+1")}
- send "1+1" to server
- wait for response

- receive "1+1" from client
- evaluate "1+1" to "2"
- send "2" to client

wait for input from client
Under the hood: sockets

- \texttt{m2("1+1")}
- send \texttt{"1+1"} to server
- wait for response

- receive \texttt{"1+1"} from client
- evaluate \texttt{"1+1"} to \texttt{"2"}
- send \texttt{"2"} to client
- wait for input from client
Under the hood: sockets

- m2("1+1")
- send "1+1" to server
- wait for response
- receive "2" from server
- receive "2" from server

- wait for input from client
- receive "1+1" from client
- evaluate "1+1" to "2"
- send "2" to client
- wait for input from client
Under the hood: sockets

- \texttt{m2("1+1")}
- send "1+1" to server
- wait for response

- receive "1+1" from client
- evaluate "1+1" to "2"
- send "2" to client
- wait for input from client

- receive "2" from server
- return "2" from \texttt{m2()}
Under the hood: sockets

R ➔ Socket ➔ M2

m2("a = 5")
[1] "5"
m2("a")
[1] "5"
m2("1+")
Error: Macaulay2 Error!
m2("2+1")
[1] "3"
Under the hood: sockets

\[ \text{m2}("a = 5") \]
\[ \text{m2}("a") \]
\[ \text{m2}("1+") \]
Error: Macaulay2 Error!
\[ \text{m2}("2+1") \]

Christopher O’Neill (UC Davis)
Under the hood: sockets

> m2("a = 5")
[1] "5"

> m2("a")
[1] "5"

>
> m2("a = 5")
[1] "5"

> m2("a")
[1] "5"

> m2("1+")
Error: Macaulay2 Error!

>
> m2("a = 5")
[1] "5"

> m2("a")
[1] "5"

> m2("1+")
Error: Macaulay2 Error!

> m2("2+1")
[1] "3"
Under the hood: sockets

> start_m2()
Starting M2... done.
> m2("1+1")
[1] "2"

So. . . now what?

New features since creation:
- Lots of convenience functions
- High-level parser
- Cloud computing

Christopher O'Neill (UC Davis)
> start_m2()
Starting M2... done.

> m2("1+1")
[1] "2"

>
Under the hood: sockets

> start_m2()
Starting M2... done.

> m2("1+1")
[1] "2"

> So... now what?
> start_m2()
Starting M2... done.

> m2("1+1")
[1] "2"

So... now what?

New features since creation:
- Lots of convenience functions
- High-level parser
- Cloud computing
Convenience functions

> m2("a = 5")
[1] "5"

> m2("a")
[1] "5"

>
Convenience functions

> m2("a = 5")
[1] "5"

> m2("a")
[1] "5"

> m2("R = QQ[x,y,z]"
[1] "QQ(monoid[x..z, Degrees => {3:1}, Heft => {1}, MonomialOrder
=> VerticalList{MonomialSize => 32, GRevLex => {3:1}, Position =>
Up}, DegreeRank => 1])"

>
Convenience functions

```r
> m2("a = 5")
[1] "5"

> m2("a")
[1] "5"

> m2("R = QQ[x,y,z]")
[1] "QQ(monoid[x..z, Degrees => {3:1}, Heft => {1}, MonomialOrder => VerticalList{MonomialSize => 32, GRevLex => {3:1}, Position => Up}, DegreeRank => 1])"

> m2("I = ideal(x^2, x*y, x^3*y^2)"
[1] "ideal map((R)^1,(R)^{{-2},{-2},{-5}},{x^2, x*y, x^3*y^2})"
```
Convenience functions

> m2("a = 5")
[1] "5"

> m2("a")
[1] "5"

> m2("R = QQ[x,y,z]")
[1] "QQ(monoid[x..z, Degrees => {3:1}, Heft => {1}, MonomialOrder => VerticalList{MonomialSize => 32, GRevLex => {3:1}, Position => Up}, DegreeRank => 1])"

> m2("I = ideal(x^2, x*y, x^3*y^2)")
[1] "ideal map((R)^1,(R)^{{-2},{-2},{-5}},{{x^2, x*y, x^3*y^2}})"

> m2("gens gb I")
[1] "map((R)^1,(R)^{{-2},{-2}},{x*y, x^2})"
Convenience functions

```r
> m2("a = 5")
[1] "5"

> m2("a")
[1] "5"

> m2("R = QQ[x,y,z]"
[1] "QQ(monoid[x..z, Degrees => {3:1}, Heft => {1}, MonomialOrder => VerticalList{MonomialSize => 32, GRevLex => {3:1}, Position => Up}, DegreeRank => 1])"

> m2("I = ideal(x^2, x*y, x^3*y^2)"
[1] "ideal map((R)ˆ1,(R)ˆ{{-2},{-2},{-5}},{x^2, x*y, x^3*y^2})"

> m2("gens gb I")
[1] "map((R)ˆ1,(R)ˆ{{-2},{-2}},{x*y, x^2})"
```
Convenience functions

> m2("a = 5")
[1] "5"

> m2("a")
[1] "5"

>
Convenience functions

```r
> m2("a = 5")
[1] "5"

> m2("a")
[1] "5"

> (R <- ring("x", "y", "z", coefring = "QQ"))
M2 Ring: QQ[x,y,z], grevlex order

> (I <- ideal("x^2", "x*y", "x^3*y^2"))
M2 Ideal of ring QQ[x,y,z] (grevlex) with generators:
  x^2, x*y, x^3*y^2

> (mygens <- gb(I))
x*y
x^2

> mygens[[2]]
x^2
```
Convenience functions

> m2("a = 5")
[1] "5"

> m2("a")
[1] "5"

> (R <- ring("x", "y", "z", coefring = "QQ"))
M2 Ring: QQ[x,y,z], grevlex order

> (I <- ideal("x^2", "x*y", "x^3*y^2"))
M2 Ideal of ring QQ[x,y,z] (grevlex) with generators:
< x^2, x y, x^3 y^2 >
Convenience functions

```r
> m2("a = 5")
[1] "5"

> m2("a")
[1] "5"

> (R <- ring("x", "y", "z", coefring = "QQ"))
M2 Ring: QQ[x, y, z], grevlex order

> (I <- ideal("x^2", "x*y", "x^3*y^2"))
M2 Ideal of ring QQ[x, y, z] (grevlex) with generators:
< x^2, x*y, x^3*y^2 >

> (mygens <- gb(I))

x y
x^2

> 
```
Convenience functions

```r
> m2("a = 5")
[1] "5"

> m2("a")
[1] "5"

> (R <- ring("x", "y", "z", coefring = "QQ"))
M2 Ring: QQ[x,y,z], grevlex order

> (I <- ideal("x^2", "x*y", "x^3*y^2"))
M2 Ideal of ring QQ[x,y,z] (grevlex) with generators:
< x^2, x y, x^3 y^2 >

> (mygens <- gb(I))

  x y
  x^2

> mygens[[2]]

  x^2
```
Convenience functions

```r
> m2("a = 5")
[1] "5"

> m2("a")
[1] "5"

(R <- ring("x", "y", "z", coefring = "QQ"))
M2 Ring: QQ[x,y,z], grevlex order

> (I <- ideal("x^2", "x*y", "x^3*y^2"))
M2 Ideal of ring QQ[x,y,z] (grevlex) with generators:
< x^2, x y, x^3 y^2 >

> (mygens <- gb(I))
x y
x^2

> mygens[[2]] ←− mpoly
x^2
```
Convenience functions

> m2("a = 5")
[1] "5"

> m2("a")
[1] "5"

(R <- ring("x", "y", "z", coefring = "QQ"))
M2 Ring: QQ[x,y,z], grevlex order

(I <- ideal("x^2", "x*y", "x^3*y^2"))
M2 Ideal of ring QQ[x,y,z] (grevlex) with generators :
< x^2, x y, x^3 y^2 >

(mygens <- gb(I)) ← mpolylist
x y
x^2

> mygens[[2]] ← mpoly
x^2
Convenience functions

\[
\begin{align*}
> & (I \leftarrow \text{ideal}("x^2", "x*y", "x^3*y^2")) \\
M2 \text{ Ideal of ring } \mathbb{Q}[x,y,z] \text{ (grevlex) with generators :} \\
& < x^2, x*y, x^3*y^2 > \\
> & 
\end{align*}
\]
Convenience functions

\[
> (I <- \text{ideal}("x^2", "x*y", "x^3*y^2"))
\]

M2 Ideal of ring $\mathbb{Q}[x,y,z]$ (grevlex) with generators:
\[
< x^2, x*y, x^3*y^2 >
\]

\[
> \text{radical}(I)
\]

M2 Ideal of ring $\mathbb{Q}[x,y,z]$ (grevlex) with generator:
\[
< x >
\]

\[
>
\]
Convenience functions

```r
> (I <- ideal("x^2", "x*y", "x^3*y^2"))
M2 Ideal of ring QQ[x,y,z] (grevlex) with generators:
< x^2, x y, x^3 y^2 >

> radical(I)
M2 Ideal of ring QQ[x,y,z] (grevlex) with generator:
< x >

> saturate(I,ideal("x^5"))
M2 Ideal of ring QQ[x,y,z] (grevlex) with generator:
< 1 >

>
```
Convenience functions

```r
> (I <- ideal("x^2", "x*y", "x^3*y^2"))
M2 Ideal of ring QQ[x,y,z] (grevlex) with generators:
< x^2, x y, x^3 y^2 >

> radical(I)
M2 Ideal of ring QQ[x,y,z] (grevlex) with generator:
< x >

> saturate(I, ideal("x^5"))
M2 Ideal of ring QQ[x,y,z] (grevlex) with generator:
< 1 >

> I+I
M2 Ideal of ring QQ[x,y,z] (grevlex) with generators:
< x^2, x y, x^3 y^2, x^2, x y, x^3 y^2 >

>
```
Convenience functions

> (I <- ideal("x^2", "x*y", "x^3*y^2"))
M2 Ideal of ring QQ[x,y,z] (grevlex) with generators :
< x^2, x y, x^3 y^2 >

> radical(I)
M2 Ideal of ring QQ[x,y,z] (grevlex) with generator :
< x >

> saturate(I,ideal("x^5"))
M2 Ideal of ring QQ[x,y,z] (grevlex) with generator :
< 1 >

> I+I
M2 Ideal of ring QQ[x,y,z] (grevlex) with generators :
< x^2, x y, x^3 y^2, x^2, x y, x^3 y^2 >

> gb(I+I)
x y
x^2
Convenience functions

> (I <- ideal("x^2", "x*y", "x^3*y^2"))
M2 Ideal of ring QQ[x,y,z] (grevlex) with generators :
< x^2, x y, x^3 y^2 >

>
Convenience functions

> (I <- ideal("x^2", "x*y", "x^3*y^2"))
M2 Ideal of ring QQ[x,y,z] (grevlex) with generators :
< x^2, x y, x^3 y^2 >

> primary_decomposition(I)
M2 List of ideals of QQ[x,y,z] (grevlex) :
< x >
< x^2, y >
Convenience functions

```r
> (I <- ideal("x^2", "x*y", "x^3*y^2"))
M2 Ideal of ring QQ[x,y,z] (grevlex) with generators:
< x^2, x y, x^3 y^2 >

> primary_decomposition(I)
M2 List of ideals of QQ[x,y,z] (grevlex):
< x 
< x^2, y >

> dimension(I)
[1] 2
```
Convenience functions

\[
> (I <- \text{ideal}("x^2", "x*y", "x^3*y^2"))
\]
M2 Ideal of ring \(\mathbb{Q}[x,y,z]\) (grevlex) with generators:
\[
< x^2, x \cdot y, x^3 \cdot y^2 >
\]

\[
> \text{primary_decomposition}(I)
\]
M2 List of ideals of \(\mathbb{Q}[x,y,z]\) (grevlex):
\[
< x >
\]
\[
< x^2, y >
\]

\[
> \text{dimension}(I)
\]
[1] 2

\[
> \text{ring}("x", "y", "z", \text{coefring} = "\mathbb{Q}"; \text{code} = \text{TRUE})
\]
m2rintring00000002 = \(\mathbb{Q}[x,y,z,\text{MonomialOrder}=>\{\text{GRevLex}=>3\}]\)

>
Convenience functions

```r
> (I <- ideal("x^2", "x*y", "x^3*y^2"))
M2 Ideal of ring QQ[x,y,z] (grevlex) with generators:
< x^2, x y, x^3 y^2 >

> primary_decomposition(I)
M2 List of ideals of QQ[x,y,z] (grevlex):
< x >
< x^2, y >

> dimension(I)
[1] 2

> ring("x", "y", "z", coefring = "QQ", code = TRUE)
m2rintring00000002 = QQ[x,y,z,MonomialOrder=>{GRevLex=>3}]

> dimension(I, code = TRUE)
dim(m2rintideal00000001)
```
The parser

\[
\begin{pmatrix}
1 & 2 \\
3 & 4
\end{pmatrix}
\]

M2 Matrix over ZZ

\[
\begin{pmatrix}
1 & 3 \\
2 & 4
\end{pmatrix}
\]

M2 Matrix over ZZ

Christopher O'Neill (UC Davis)
The parser

```r
> m2_matrix(matrix(c(1,2,3,4), nrow = 2, ncol = 2))
   [,1] [,2]
[1,] 1 3
[2,] 2 4
M2 Matrix over ZZ[]
>
```
> m2_matrix(matrix(c(1,2,3,4), nrow = 2, ncol = 2))
   [,1] [,2]
[1,]  1  3
[2,]  2  4
M2 Matrix over ZZ[]

> m2_matrix(matrix(c(1,2,3,4), nrow = 2, ncol = 2), code = TRUE)
m2rintmatrix00000002 = matrix {(1),(3)},{(2),(4)}

>
The parser

> m2_matrix(matrix(c(1,2,3,4), nrow = 2, ncol = 2))
  [,1] [,2]
[1,]  1  3
[2,]  2  4
M2 Matrix over ZZ[]

> m2_matrix(matrix(c(1,2,3,4), nrow = 2, ncol = 2), code = TRUE)
m2rintmatrix00000002 = matrix {{(1),(3)},{(2),(4)}}

> m2("m2rintmatrix00000002 = matrix {{(1),(3)},{(2),(4)}}")
[1] "map((ZZ)^2,(ZZ)^2,{{1, 3}, {2, 4}})"

>
The parser

> m2_matrix(matrix(c(1,2,3,4), nrow = 2, ncol = 2))

[,1] [,2]
[1,] 1 3
[2,] 2 4
M2 Matrix over ZZ[]

> m2_matrix(matrix(c(1,2,3,4), nrow = 2, ncol = 2), code = TRUE)
m2rintmatrix00000002 = matrix {{{1, 3},{2, 4}}}

> m2("m2rintmatrix00000002 = matrix {{{1, 3},{2, 4}}}")
[1] "map((ZZ)^2,(ZZ)^2,{{{1, 3},{2, 4}}})"

> m2_parse("map((ZZ)^2,(ZZ)^2,{{{1, 3},{2, 4}}})")

[,1] [,2]
[1,] 1 3
[2,] 2 4
M2 Matrix over ZZ[]
The parser

Parsing "map((ZZ)^2,(ZZ)^2,{{1, 3}, {2, 4}})"
The parser

Parsing "map((ZZ)^2,(ZZ)^2,{{1, 3}, {2, 4}})"

> m2_tokenize("map((ZZ)^2,(ZZ)^2,{{1, 3}, {2, 4}}})")

[1] "map" "(" "(" "ZZ" ")" "^" "2" "," "(" "ZZ"
[11] ")" "^" "2" "," "{" "{" "1" "," "3" "}"
[21] "," "{" "2" "," "4" "})" "}" "")"
The parser

Parsing "map((ZZ)^2,(ZZ)^2,{{1, 3}, {2, 4}})"

> m2_tokenize("map((ZZ)^2,(ZZ)^2,{{1, 3}, {2, 4}}))")
[1] "map" "(" "(" "ZZ" ")" "^" "2" "," "(" "ZZ"
[11] ")" "^" "2" "," "{" "{" "1" "," "3" "}"
[21] "," "{" "2" "," "4" "}" "}" ")"

map ( )

^ ^ {} {} "ZZ" "2" "ZZ" "2" {} {} "1" "3" "2" "4"
The parser

Parsing "map((ZZ)^2,(ZZ)^2,{1, 3}, {2, 4})"

```r
> m2_tokenize("map((ZZ)^2,(ZZ)^2,{1, 3}, {2, 4})")
[1] "map" "(" "(" "ZZ" ")" "^" "2" "," "(" "ZZ" ")" "^" "2" "," "{" "){" "1" "," "3" "}"
[21] "," "{" "2" "," "4" "}" "}" ")"
```
The parser

Parsing "map((ZZ)^2,(ZZ)^2,{{1, 3}, {2, 4}})"

> m2_tokenize("map((ZZ)^2,(ZZ)^2,{{1, 3}, {2, 4}})")
[1] "map" "(" "(" "ZZ" ")" "^" "2" ",," "(" "ZZ" [11] ")" "^" "2" ",," "{" "{" "1" ",," "3" "}" [21] ",," "{" "2" ",," "4" "}" "}" ")"

```
map()
```

```
^  ^  {}
ZZ  2  ZZ  2  list(1,3)  list(2,4)
```
The parser

Parsing "map((ZZ)^2,(ZZ)^2,{{1, 3}, {2, 4}})"

> m2_tokenize("map((ZZ)^2,(ZZ)^2,{{1, 3}, {2, 4}}))")
[1] "map" "(" "(" "ZZ" ")" "^" "2" "," "(" "ZZ" [11] ")" "^" "2" "," "{" "{" "1" "," "3" "}"
[21] "," "{" "2" "," "4" "}" "}" ")""

map ()

list(list(1,3),list(2,4))

"ZZ" "2" "ZZ" "2"
The parser

Parsing "map((ZZ)^2,(ZZ)^2,{{1, 3}, {2, 4}})"

> m2_tokenize("map((ZZ)^2,(ZZ)^2,{{1, 3}, {2, 4}})")

[1] "map" "(" "(" "ZZ" ")" "^" "2" ",", "(" "ZZ"
[11] ")" "^" "2" ",", "{" "{" "1" ",", "3" "}
[21] "," ",{" "2" ",", "4" ""} " })" ")" ")"
The parser

Parsing "map((ZZ)^2,(ZZ)^2,{{1, 3}, {2, 4}})"

> m2_tokenize("map((ZZ)^2,(ZZ)^2,{{1, 3}, {2, 4}}})")
[1] "map" "(" "(" "ZZ" ")" "^" "2" ",", "(" "ZZ" [11] ")" "^" "2" ",", 
[21] "," 
("{ "1" ",," "3" "}" ")" ")"

map ()

(Z-ring,2) (Z-ring,2) list(list(1,3),list(2,4))
The parser

Parsing "map((ZZ)^2,(ZZ)^2,{{1, 3}, {2, 4}})"

\[
\text{m2\_tokenize("map((ZZ)^2,(ZZ)^2,{{1, 3}, {2, 4}})})")}
\]

\[
[1] \ "map" \ "(" \ "(" \ "ZZ" \ ")")" \ "^" \ "2" \ "," \ "(" \ "ZZ"
[11] ")")" \ "^" \ "2" \ "," \ "{" \ "{" \ "1" \ "," \ "3" \ "}"\n[21] "," \ "{" \ "2" \ "," \ "4" \ "}"\n\]

\[
[\{1\} \ [,2]\]
[1,] 1 3
[2,] 2 4
\]

M2 Matrix over ZZ[]
The parser

>
The parser

```plaintext
> m2_parse(m2("x"))
M2 Symbol: x

> 
```
The parser

> m2_parse(m2("x"))
M2 Symbol: x

> m2_parse(m2("ZZ"))
M2 Ring: ZZ[], grevlex order

>
The parser

> m2_parse(m2("x"))
M2 Symbol: x

> m2_parse(m2("ZZ"))
M2 Ring: ZZ[], grevlex order

> m2("m2rintring00000002 = QQ[x,y,z,MonomialOrder=>{GRevLex=>3}]")
[1] "QQ(monoid[x..z, Degrees => {3:1}, Heft => {1}, MonomialOrder => VerticalList{MonomialSize => 32, GRevLex => {3:1}, Position => Up}, DegreeRank => 1])"

>
The parser

> m2_parse(m2("x"))
M2 Symbol: x

> m2_parse(m2("ZZ"))
M2 Ring: ZZ[], grevlex order

> m2("m2rintring00000002 = QQ[x,y,z,MonomialOrder=>{GRevLex=>3}]")
[1] "QQ(monoid[x..z, Degrees => {3:1}, Heft => {1}, MonomialOrder => VerticalList{MonomialSize => 32, GRevLex => {3:1}, Position => Up}, DegreeRank => 1])"

> m2_parse(m2("m2rintring00000002"))
M2 Ring: QQ[x,y,z], grevlex order

>
The parser

> m2_parse(m2("x"))
M2 Symbol: x

> m2_parse(m2("ZZ"))
M2 Ring: ZZ[], grevlex order

> m2("m2rintring00000002 = QQ[x,y,z,MonomialOrder=>{GRevLex=>3}]")
[1] "QQ(monoid[x..z, Degrees => {3:1}, Heft => {1}, MonomialOrder => VerticalList{MonomialSize => 32, GRevLex => {3:1}, Position => Up}, DegreeRank => 1])"

> m2_parse(m2("m2rintring00000002"))
M2 Ring: QQ[x,y,z], grevlex order

> m2("ideal({x^2+2*x,2*x+3})")
[1] "ideal map((m2rintring00000002)^1,(m2rintring00000002)^{{{-2}, {-1}},{{x^2+2*x, 2*x+3}}}"

Christopher O'Neill (UC Davis) August 3, 2017 15 / 20
Parser “extensibility”

> m2("ideal({x^2+2*x,2*x+3})")
[1] "ideal map((m2rintring00000002)^1,(m2rintring00000002)^{{-2},
{-1}},{x^2+2*x, 2*x+3})"
> m2("ideal({x^2+2*x,2*x+3})")
[1] "ideal map((m2rintring00000002)^1,(m2rintring00000002)^{\{-2\},
{-1}\},\{\{x^2+2*x, 2*x+3\}\})"

m2_parse_function.m2_map <- function(x) {
  R1 <- x[[1]]
  R2 <- x[[2]]
  
  m2_structure(
    mat,
    m2_name = "",
    m2_class = "m2_matrix",
    m2_meta = list(
      ring = R1
    ),
    base_class = "matrix"
  )
}

> m2("ideal({x^2+2*x,2*x+3})")
[1] "ideal map((m2r-intring00000002)^1,(m2r-intring00000002)^{-2},
{-1}),{{x^2+2*x, 2*x+3}})"
m2("ideal({x^2+2*x,2*x+3})")

> m2("ideal({x^2+2*x,2*x+3})")

[1] "ideal map((m2rintring00000002)^1,(m2rintring00000002)^{{-2},
{-1}},{x^2+2*x, 2*x+3})"

m2_parse_function.m2_ideal <- function(x) {
  m2_structure(
    m2_name = "",
    m2_class = "m2_ideal",
    m2_meta = list(
      ring = m2_meta(x[[1]], "ring"),
      gens = structure(x[[1]][1,,], class = "mpolyList")
    )
  )
}

Christopher O'Neill (UC Davis)
m2r is now in the cloud!

>
m2r is now in the cloud!

```r
> library("m2r")
Loading required package: mpoly
Loading required package: stringr
please cite mpoly if you use it; see citation("mpoly")
M2 not found; defaulting to cloud.
Use set_m2r_path("/path/to/m2") to run M2 locally.
```

>
m2r is now in the cloud!

```r
> library("m2r")
Loading required package: mpoly
Loading required package: stringr
please cite mpoly if you use it; see citation("mpoly")
M2 not found; defaulting to cloud.
Use set_m2r_path("/path/to/m2") to run M2 locally.

> start_m2()
Connecting to M2 in the cloud...
done.

>
```
m2r is now in the cloud!

> library("m2r")
Loading required package: mpoly
Loading required package: stringr
please cite mpoly if you use it; see citation("mpoly")
M2 not found; defaulting to cloud.
Use set_m2r_path("/path/to/m2") to run M2 locally.

> start_m2()
Connecting to M2 in the cloud...
done.

> m2("a = 5")
[1] "5"

>
m2r is now in the cloud!

```r
> library("m2r")
Loading required package: mpoly
Loading required package: stringr
please cite mpoly if you use it; see citation("mpoly")
M2 not found; defaulting to cloud.
Use set_m2r_path("/path/to/m2") to run M2 locally.

> start_m2()
Connecting to M2 in the cloud...
done.

> m2("a = 5")
[1] "5"

> m2("a")
[1] "5"
```
m2r is now in the cloud!

```r
> library("m2r")
Loading required package: mpoly
Loading required package: stringr
please cite mpoly if you use it; see citation("mpoly")
M2 not found; defaulting to cloud.
Use set_m2r_path("/path/to/m2") to run M2 locally.

> start_m2()
Connecting to M2 in the cloud...
done.

> m2("a = 5")
[1] "5"

> m2("a")
[1] "5"
```

![Diagram showing R Client connected to Python Server via Docker]
m2r is now in the cloud!

> library("m2r")
Loading required package: mpoly
Loading required package: stringr
please cite mpoly if you use it; see citation("mpoly")
M2 not found; defaulting to cloud.
Use set_m2r_path("/path/to/m2") to run M2 locally.

> start_m2()
Connecting to M2 in the cloud...
done.

> m2("a = 5")
[1] "5"

> m2("a")
[1] "5"
m2r is now in the cloud!

```r
> library("m2r")
Loading required package: mpoly
Loading required package: stringr
please cite mpoly if you use it; see citation("mpoly")
M2 not found; defaulting to cloud.
Use set_m2r_path("/path/to/m2") to run M2 locally.

> start_m2()
Connecting to M2 in the cloud...
done.

> m2("a = 5")
[1] "5"

> m2("a")
[1] "5"
```
m2r is now in the cloud!

```r
> library("m2r")
Loading required package: mpoly
Loading required package: stringr
please cite mpoly if you use it; see citation("mpoly")
M2 not found; defaulting to cloud.
Use set_m2r_path("/path/to/m2") to run M2 locally.

> start_m2()
Connecting to M2 in the cloud...
done.

> m2("a = 5")
[1] "5"

> m2("a")
[1] "5"
```
More fancy features out there: reference functions

```r
R <- ring("x", "y", "z", coefring = "QQ")
M2 Ring: QQ[\[x,y,z\]], grevlex order

(I <- ideal("x^2", "x*y", "x^3*y^2"))
M2 Ideal of ring QQ[\[x,y,z\]] (grevlex) with generators:
  < x^2, x y, x^3 y^2 >

> gb(I)
x y
x^2

(J <- ideal.("x^2", "x*y", "x^3*y^2"))
M2 Pointer Object

> gb(J)
x y
x^2

> m2_parse(J)
→ I
```
More fancy features out there: reference functions

```r
R <- ring("x", "y", "z", coefring = "QQ")
M2 Ring: QQ[x,y,z], grevlex order

> R <- ring("x", "y", "z", coefring = "QQ")
M2 Ring: QQ[x,y,z], grevlex order

> (I <- ideal("x^2", "x*y", "x^3*y^2"))
M2 Ideal of ring QQ[x,y,z] (grevlex) with generators: 
< x^2, x y, x^3 y^2 >

> gb(I)
x y
x^2

> (J <- ideal.("x^2", "x*y", "x^3*y^2"))
M2 Pointer Object

> gb(J)
x y
x^2

> m2_parse(J)
−→ I
```

Christopher O'Neill (UC Davis)
The m2r package
August 3, 2017
More fancy features out there: reference functions

```r
> R <- ring("x", "y", "z", coeftring = "QQ")
M2 Ring: QQ[x,y,z], grevlex order

> (I <- ideal("x^2", "x*y", "x^3*y^2"))
M2 Ideal of ring QQ[x,y,z] (grevlex) with generators:
< x^2, x y, x^3 y^2 >
```

Christopher O'Neill (UC Davis)
More fancy features out there: reference functions

\[
\text{R <- ring("x", "y", "z", coefring = "QQ")}
\]

M2 Ring: \( \mathbb{Q}[x,y,z] \), grevlex order

\[
\text{(I <- ideal("x^2", "x*y", "x^3*y^2"))}
\]

M2 Ideal of ring \( \mathbb{Q}[x,y,z] \) (grevlex) with generators:
< \( x^2 \), \( x*y \), \( x^3*y^2 \) >

\[
\text{gb(I)}
\]

\[
\text{x y}
\]

\[
\text{x^2}
\]

\[
\]
More fancy features out there: reference functions

\[
\begin{align*}
R & \leftarrow \text{ring("x", "y", "z", coefring = "QQ")} \\
\text{M2 Ring: } & \text{ QQ[x,y,z], grevlex order} \\
(I & \leftarrow \text{ideal("x^2", "x*y", "x^3*y^2")}) \\
\text{M2 Ideal of ring QQ[x,y,z] (grevlex) with generators : } \\
& < x^2, x y, x^3 y^2 > \\
gb(I) \\
x y \\
x^2 \\
(J & \leftarrow \text{ideal.("x^2", "x*y", "x^3*y^2")}) \\
\text{M2 Pointer Object} \\
\text{ExternalString : ideal map((m2rintring00000001)^1,(m2rin...} \\
\text{M2 Name : m2rintideal00000004} \\
\text{M2 Class : Ideal (Type)}
\end{align*}
\]
More fancy features out there: reference functions

\[ R \leftarrow \text{ring}("x", "y", "z", \text{coerfing} = "QQ") \]
M2 Ring: \( \mathbb{Q}[x,y,z] \), grevlex order

\[ (I \leftarrow \text{ideal}("x^2", "x*y", "x^3*y^2")) \]
M2 Ideal of ring \( \mathbb{Q}[x,y,z] \) (grevlex) with generators:
< \( x^2 \), \( x\ y \), \( x^3\ y^2 \) >

\[ \text{gb}(I) \]
\( x \ y \)
\( x^2 \)

\[ (J \leftarrow \text{ideal.}("x^2", "x*y", "x^3*y^2")) \]
M2 Pointer Object
   ExternalString : ideal map((m2rintring00000001)^1,(m2rint...)
       M2 Name : m2rintideal00000004
       M2 Class : Ideal (Type)

\[ \text{gb}(J) \]
\( x \ y \)
\( x^2 \)
More fancy features out there: reference functions

```r
R <- ring("x", "y", "z", coefring = "QQ")
M2 Ring: QQ[x,y,z], grevlex order

(I <- ideal("x^2", "x*y", "x^3*y^2"))
M2 Ideal of ring QQ[x,y,z] (grevlex) with generators:
< x^2, x y, x^3 y^2 >

> gb(I)
  x y
  x^2

(J <- ideal.("x^2", "x*y", "x^3*y^2"))
M2 Pointer Object
  ExternalString : ideal map((m2rintring00000001)^1,(m2rin...
  M2 Name : m2rintideal00000004
  M2 Class : Ideal (Type)

> gb(J)
  x y
  x^2

m2_parse(J) → I
```
Thank you MRC!
A computer algebra system for R: Macaulay2 and the m2r package

Macaulay2, a software system for research in algebraic geometry,
available at http://www.math.uiuc.edu/Macaulay2/.

D. Kahle (2013)
mpoly: Multivariate polynomials in R

R Core Team (2014)
R: A language and environment for statistical computing
R Foundation for Statistical Computing, Vienna, Austria
References

_A computer algebra system for R: Macaulay2 and the m2r package_

_Macaulay2, a software system for research in algebraic geometry,_
available at http://www.math.uiuc.edu/Macaulay2/.

D. Kahle (2013)
_mpoly: Multivariate polynomials in R_

R Core Team (2014)
_R: A language and environment for statistical computing_
R Foundation for Statistical Computing, Vienna, Austria

Thanks!
YOU should request lots of features!

https://github.com/coneill-math/m2r
YOU should request lots of features!

https://github.com/coneill-math/m2r
YOU should request lots of features!

https://github.com/coneill-math/m2r
YOU should request lots of features!

https://github.com/coneill-math/m2r