Elliptic Curves in Sage

John Cremona
University of Warwick, UK

Sage Days 10
11 October, 2008

Overview

- Elliptic Curves have been in Sage since (almost) the beginning.

Overview

- Elliptic Curves have been in Sage since (almost) the beginning.
- The source directory `sage/schemes/elliptic_curves` has 34 files and $21,628$ lines of code, and that does not count external packages such as my `eclib` (`mwrank` and friends), Runestein's `lcalc`, the `pari` library's elliptic curve functions, and Simon's gp scripts.

Overview

- Elliptic Curves have been in Sage since (almost) the beginning.
- The source directory `sage/schemes/elliptic_curves` has 34 files and $21,628$ lines of code, and that does not count external packages such as my `eclib` (`mwrank` and friends), Runestein's `lcalc`, the `pari` library's elliptic curve functions, and Simon's gp scripts.
- The Sage Tutorial and Constructions documents currently only mention a tiny part of the elliptic curve functionality in Sage. The reference manual, Chapter 39, has several sections on elliptic curves which contain *all* the docstrings of all the functions. Browsing these will give you a better idea of what is (and is not) there, but not necessarily in a coherent order.

The Elliptic Curve Classes

- The base class for elliptic curves in Sage is `EllipticCurve_generic` which builds on `ProjectiveCurve_generic` and lower level machinery for Curves and Schemes.

The Elliptic Curve Classes

- The base class for elliptic curves in Sage is `EllipticCurve_generic` which builds on `ProjectiveCurve_generic` and lower level machinery for Curves and Schemes.
- ```
  EllipticCurve_generic
      EllipticCurve_field
          EllipticCurve_finite_field
          EllipticCurve_number_field
              EllipticCurve_rational_field
  ```
  ```

The Elliptic Curve Classes

- The base class for elliptic curves in Sage is `EllipticCurve_generic` which builds on `ProjectiveCurve_generic` and lower level machinery for Curves and Schemes.
- `EllipticCurve_generic`
```
EllipticCurve_generic
    EllipticCurve_field
        EllipticCurve_finite_field
        EllipticCurve_number_field
            EllipticCurve_rational_field
```
- There is also the class `EllipticCurve_padic_field`.

Points on Elliptic Curves

- The "fancy" class `EllipticCurvePoint`, which derives from `SchemeMorphism_projective_coordinates_ring`, is not in fact used at all. It is there for when there is some functionality for elliptic curves defined over base schemes other than fields, which is not yet.

Points on Elliptic Curves

- The “fancy” class `EllipticCurvePoint`, which derives from `SchemeMorphism_projective_coordinates_ring`, is not in fact used at all. It is there for when there is some functionality for elliptic curves defined over base schemes other than fields, which is not yet.
- The class `EllipticCurvePoint_field` and its children does all the work.

Points on Elliptic Curves

- The "fancy" class `EllipticCurvePoint`, which derives from `SchemeMorphism_projective_coordinates_ring`, is not in fact used at all. It is there for when there is some functionality for elliptic curves defined over base schemes other than fields, which is not yet.
- The class `EllipticCurvePoint_field` and its children does all the work.
- `EllipticCurvePoint_field` (derived from `AdditiveGroupElement`)

Points on Elliptic Curves

- The “fancy” class `EllipticCurvePoint`, which derives from `SchemeMorphism_projective_coordinates_ring`, is not in fact used at all. It is there for when there is some functionality for elliptic curves defined over base schemes other than fields, which is not yet.
- The class `EllipticCurvePoint_field` and its children does all the work.
- `EllipticCurvePoint_field` (derived from `AdditiveGroupElement`)
  - `EllipticCurvePoint_number_field`

Points on Elliptic Curves

- The "fancy" class `EllipticCurvePoint`, which derives from `SchemeMorphism_projective_coordinates_ring`, is not in fact used at all. It is there for when there is some functionality for elliptic curves defined over base schemes other than fields, which is not yet.
- The class `EllipticCurvePoint_field` and its children does all the work.
- `EllipticCurvePoint_field` (derived from `AdditiveGroupElement`)
  - `EllipticCurvePoint_number_field`
  - `EllipticCurvePoint_finite_field`

Plan for the talk

- General elliptic curve functions

Plan for the talk

- General elliptic curve functions
- Elliptic curves over $\mathbb{Q}$

Plan for the talk

- General elliptic curve functions
- Elliptic curves over $\mathbb{Q}$
- Elliptic curves over number fields

Plan for the talk

- General elliptic curve functions
- Elliptic curves over $\mathbb{Q}$
- Elliptic curves over number fields
- Elliptic curves over finite fields.

Creating an elliptic curve

- The most common method is to give the usual 5 coefficients a_1, a_2, a_3, a_4, a_6 of a Weierstrass Equation:

```
sage: E1 = EllipticCurve([0,0,1,-7,6])
sage: E1
Elliptic Curve defined by y^2 + y = x^3 - 7*x + 6 over
 Rational Field
```

Creating an elliptic curve

- The most common method is to give the usual 5 coefficients a_1, a_2, a_3, a_4, a_6 of a Weierstrass Equation:

```
sage: E1 = EllipticCurve([0,0,1,-7,6])
sage: E1
Elliptic Curve defined by y^2 + y = x^3 - 7*x + 6 over
 Rational Field
```

Creating an elliptic curve

- The most common method is to give the usual 5 coefficients a_1, a_2, a_3, a_4, a_6 of a Weierstrass Equation:

```
sage: E1 = EllipticCurve([0,0,1,-7,6])
sage: E1
Elliptic Curve defined by y^2 + y = x^3 - 7*x + 6 over
 Rational Field
```

```
sage: E2 = EllipticCurve(GF(101),[0,0,1,-7,6])
sage: E2
Elliptic Curve defined by y^2 + y = x^3 + 94*x + 6 over
 Finite Field of size 101
```

Creating an elliptic curve

- The most common method is to give the usual 5 coefficients a_1, a_2, a_3, a_4, a_6 of a Weierstrass Equation:

```
sage: E1 = EllipticCurve([0,0,1,-7,6])
sage: E1
Elliptic Curve defined by y^2 + y = x^3 - 7*x + 6 over
 Rational Field

sage: E2 = EllipticCurve(GF(101),[0,0,1,-7,6])
sage: E2
Elliptic Curve defined by y^2 + y = x^3 + 94*x + 6 over
 Finite Field of size 101

sage: K = PolynomialRing(QQ,'T').fraction_field()
sage: T = K.gen()
sage: E3 = EllipticCurve([0,0,0,-T^2,0])
sage: E3
Elliptic Curve defined by y^2  = x^3 - T^2*x over Fraction Field of
 Univariate  Polynomial Ring in T over Rational Field
```

- Starting with a more general genus one curve and a base point is not yet implemented. For example,

```
sage: P2.<X, Y, Z> = ProjectiveSpace(QQ, 2)
sage: P2
Projective Space of dimension 2 over Rational Field
sage: C = Curve(X^3 + Y^2*Z - X*Y*Z - Z^3)
sage: C
Projective Curve over Rational Field defined by X^3 - X*Y*Z + Y^2*Z
sage: C.genus()
1
sage: pt = C([0, 1, 1]); pt
(0 : 1 : 1)
sage: EllipticCurve(C,pt)
...
TypeError: invalid input to EllipticCurve constructor
```

Changing models

Elliptic curves in Sage are currently always represented by (long) Weierstrass models. Standard transformations for changing models are available:

```
sage: E = EllipticCurve([1/2,3/4,5/6,7/8,9/10]); E
Elliptic Curve defined by y^2 + 1/2*x*y + 5/6*y = x^3 + 3/4*x^2 + 7/8*x
sage: Emin = E.minimal_model(); Emin
Elliptic Curve defined by y^2 + x*y  = x^3 - x^2 + 699258*x + 597561416
sage: t = Emin.isomorphism_to(E); t

Generic morphism:
  From: Abelian group of points on Elliptic Curve defined by y^2 + x*y
  To:   Abelian group of points on Elliptic Curve defined by y^2 + 1/2*
  Via:  (u,r,s,t) = (30, 244, 7, 11128)
```

Twists

Sage can construct quadratic (and higher) twists over any field, including fields of characteristic 2 or 3. It cannot (yet) detect when two curves are (quadratic or more general) twists except by comparing j-invariants:

```
sage: E = EllipticCurve([1,0,0,0,1])
sage: E3 = E.quadratic_twist(3)
sage: E;E3
Elliptic Curve defined by y^2 + x*y  = x^3 +1 over Rational Field
Elliptic Curve defined by y^2  = x^3 - 3*x + 1730 over Rational Field
sage: E.is_isomorphic(E3)
False
```

Twists

Sage can construct quadratic (and higher) twists over any field, including fields of characteristic 2 or 3. It cannot (yet) detect when two curves are (quadratic or more general) twists except by comparing j-invariants:

```
sage: E = EllipticCurve([1,0,0,0,1])
sage: E3 = E.quadratic_twist(3)
sage: E;E3
Elliptic Curve defined by y^2 + x*y  = x^3 +1 over Rational Field
Elliptic Curve defined by y^2  = x^3 - 3*x + 1730 over Rational Field
sage: E.is_isomorphic(E3)
False

sage: K = QuadraticField(3,'root3')
sage: phi = E.change_ring(K).isomorphism_to(E3.change_ring(K))
sage: E.gens()
[(-1 : 1 : 1), (0 : -1 : 1)]
sage: [phi(P) for P in E.gens()]
[(-11 : 12*root3 : 1), (1 : -24*root3 : 1)]
```

Twists (continued)

Quartic and sextic twists are only defined for curves with appropriate j-invariant:

```
sage: F = GF(101)
sage: E = EllipticCurve(F,[1,0,0,0,1])
sage: d = F.multiplicative_generator()
sage: E.quadratic_twist(d)
Elliptic Curve defined by y^2  = x^3 + 2*x^2 + 7 over Finite Field of s
sage: E = EllipticCurve(F,[0,0,0,0,1])
sage: E.j_invariant() == 0
True
sage: E.sextic_twist(d)
Elliptic Curve defined by y^2  = x^3 + 89 over Finite Field of size 101
sage: E = EllipticCurve(F,[0,0,0,1,0])
sage: E.j_invariant() == 1728
True
sage: E.quartic_twist(d)
Elliptic Curve defined by y^2  = x^3 + 67*x over Finite Field of size 1
```

Accessing basic invariants

Obviously one can access the coefficients and other basic invariants of a curve:

```
sage: E = EllipticCurve([0,0,1,-7,36])
sage: E.a_invariants()
[0, 0, 1, -7, 36]
sage: E.b_invariants()
(0, -14, 145, -49)
sage: E.c_invariants()
(336, -31320)
sage: E.discriminant()
-545723
sage: E.j_invariant()
-37933056/545723
```

The a_i, b_i and c_i are returned as lists so if you need to assign names to them you can do this:

```
sage: c4,c6 = E.c_invariants(); c4,c6
(336, -31320)
```

Note: elliptic curves in Sage are always defined over fields and so in the above examples the type of everything is `sage.rings.rational.Rational`, i.e. rational and not integer. This rarely causes problems:

```
sage: E.discriminant().factor()
-1 * 545723
sage: E.j_invariant().factor()
-1 * 2^12 * 3^3 * 7^3 * 545723^-1
sage: E.discriminant().is_prime()
...
AttributeError: 'sage.rings.rational.Rational' object has no attribute
sage: ZZ(E.discriminant()).is_prime()
False
sage: ZZ(E.discriminant()).is_irreducible()
True
```

Sets of points on a curve

Points on an elliptic curve E defined over a field F have as parent the object $E(K)$ (where K is any extension of the field F of definition of E). Over a finite field we can list its elements, or ask for a random point.

```
sage: F = GF(13)
sage: E = EllipticCurve(F,[0,-1,1,0,0])
sage: E(F)
Abelian group of points on Elliptic Curve defined by y^2 + y = x^3 + 12
 Finite Field of size 13
sage: E(GF(13^100,'a'))
Abelian group of points on Elliptic Curve defined by y^2 + y = x^3 + 12
 Finite Field in a of size 13^100
sage: E.points()
[(0 : 0 : 1), (0 : 1 : 0), (0 : 12 : 1), (1 : 0 : 1), (1 : 12 : 1),
 (2 : 5 : 1), (2 : 7 : 1), (8 : 2 : 1), (8 : 10 : 1), (10 : 6 : 1)]
sage: E.cardinality()
10
sage: E.abelian_group()
(Multiplicative Abelian Group isomorphic to C10, ((8 : 2 : 1),))
```

```
sage: E.change_ring(GF(13^10,'a')).abelian_group()
(Multiplicative Abelian Group isomorphic to C68929587450 x C2,
 ((9*a^9 + 12*a^8 + 3*a^7 + 7*a^6 + 2*a^5 + 6*a^4 + 9*a^3 + 12*a^2 + 4*
  5*a^9 + 11*a^8 + 11*a^7 + 4*a^6 + 5*a^5 + 12*a^4 + 12*a^3 + 4*a^2 + a
  (2*a^8 + 9*a^6 + 8*a^5 + 11*a^4 + 5*a^3 + 6*a^2 + 11*a + 4 : 6 : 1)))
sage: E.change_ring(GF(13^100,'a')).cardinality()
24793351109659725335110728847348651362387744678749411498748696227612229
 66104977552895203130235525308261778000000
```

```
sage: E.change_ring(GF(13^10,'a')).abelian_group()
(Multiplicative Abelian Group isomorphic to C68929587450 x C2,
 ((9*a^9 + 12*a^8 + 3*a^7 + 7*a^6 + 2*a^5 + 6*a^4 + 9*a^3 + 12*a^2 + 4*
  5*a^9 + 11*a^8 + 11*a^7 + 4*a^6 + 5*a^5 + 12*a^4 + 12*a^3 + 4*a^2 + a
  (2*a^8 + 9*a^6 + 8*a^5 + 11*a^4 + 5*a^3 + 6*a^2 + 11*a + 4 : 6 : 1)))
sage: E.change_ring(GF(13^100,'a')).cardinality()
24793351109659725335110728847348651362387744678749411498748696227612229
 66104977552895203130235525308261778000000

E = EllipticCurve(GF(next_prime(10^10)),[0,-1,1,0,0]); E
Elliptic Curve defined by y^2 + y = x^3 + 10000000018*x^2 over
 Finite Field of size 10000000019
E.cardinality()
9999910115
sage: E.random_point().order()
9999910115
```

Creation of points on a curve

All points are given in normalized projective coordinates (last nonzero coordinate $= 1$), so either $(0 : 1 : 0)$ or $(x : y : 1)$. To define a point, use $E(s)$ where s is a list $[x, y]$ or $[x, y, z]$; an error results if the equation is not satisfied. For the identity one can just use $E(0)$. Using `is_x_coord()` one can test whether an x value is the x-coordinate of a point, and use `lift_x()` to construct the point:

```
sage: E = EllipticCurve([0,0,1,-7,36])
sage: E(0)
(0 : 1 : 0)
sage: E(1,5)
(1 : 5 : 1)
sage: [a for a in srange(100) if E.is_x_coord(a)]
[1, 2, 3, 4, 6, 9, 15, 17, 32, 36, 40, 43]
sage: E.lift_x(6)
(6 : 14 : 1)
sage: E(6,14,1).order() # only over GF(q) or number fields
+Infinity
```

Point operations

To add and subtract points or multiply a point by an integer is easy:

```
sage: E = EllipticCurve('5077a1')
sage: P1,P2,P3 = E.gens()
sage: P1+P2
(4 : -7 : 1)
sage:  -P3
(1 : 0 : 1)
sage: 2*P1
(221/49 : -2967/343 : 1)
```

Point operations

To add and subtract points or multiply a point by an integer is easy:

```
sage: E = EllipticCurve('5077a1')
sage: P1,P2,P3 = E.gens()
sage: P1+P2
(4 : -7 : 1)
sage:  -P3
(1 : 0 : 1)
sage: 2*P1
(221/49 : -2967/343 : 1)
```

We can also (attempt to) divide points:

```
sage: P1.division_points(2)
[]
sage: Q=2*P1; Q
(-226/121 : -9374/1331 : 1)
sage: Q.division_points(2)
[(-2 : 3 : 1)]
```

The following shows that the three points are in fact independent:

```
sage: all([len(Q.division_points(2)) ==0 for Q in [P1,P2,P3,P1+P2,P1+P3
True
```

More simply (but only available over $\mathbb{Q}$ at present since it uses the canonical height pairing) the following shows that the three points are in fact independent:

```
sage: E.regulator([P1,P2,P3])
0.417143558758384
sage: E.regulator([P1,P2,P3],precision=280)
0.41714355875838396981711954462602952052616673147328427226778021551376
```

... and in fact

```
sage: E.rank()
3
```

... more on elliptic curves over number fields later ...

Division Polynomials

We can obtain the nth division polynomial of E (as a univariate polynomial):

```
sage: E = EllipticCurve([0,-1,1,0,0])
sage: f5 = E.division_polynomial(5); f5
5*x^12 - 20*x^11 + 16*x^10 + 95*x^9 - 285*x^8 + 360*x^7 - 255*x^6 + 94*
sage: f5.roots()
[(1, 1), (0, 1)]
sage: E(0).division_points(5)
[(0 : -1 : 1), (0 : 0 : 1), (0 : 1 : 0), (1 : -1 : 1), (1 : 0 : 1)]
```

`DivisionPolynomial()` has various options:

```
sage: E.division_polynomial(4,two_torsion_multiplicity=0)
2*x^6 - 4*x^5 + 10*x^3 - 10*x^2 + 4*x - 1
sage: E.division_polynomial(4,two_torsion_multiplicity=1)
4*x^6*y + 2*x^6 - 8*x^5*y - 4*x^5 + 20*x^3*y + 10*x^3 - 20*x^2*y - 10*x
sage: E.division_polynomial(4,two_torsion_multiplicity=2)
8*x^9 - 24*x^8 + 16*x^7 + 42*x^6 - 84*x^5 + 56*x^4 - 10*x^3 - 6*x^2 + 4
```

Morphisms and Isogenies

These are essentially not yet implemented, except for isomorphisms and automorphisms.

```
sage: E = EllipticCurve(GF(17),[0,0,0,1,0]); E
Elliptic Curve defined by y^2  = x^3 + x over Finite Field of size 17
sage: E.automorphisms()

[Generic endomorphism of Abelian group of points on Elliptic Curve defi
  Via:  (u,r,s,t) = (1, 0, 0, 0),
 Generic endomorphism of Abelian group of points on Elliptic Curve defi
  Via:  (u,r,s,t) = (4, 0, 0, 0),
 Generic endomorphism of Abelian group of points on Elliptic Curve defi
  Via:  (u,r,s,t) = (13, 0, 0, 0),
 Generic endomorphism of Abelian group of points on Elliptic Curve defi
  Via:  (u,r,s,t) = (16, 0, 0, 0)]
```

Elliptic Curves over Number Fields

We now describe some of the greater functionality provided in Sage dealing with elliptic curves over $\mathbb{Q}$ and other number fields. At present there are more functions available over $\mathbb{Q}$ than over general number fields, but the gap is closing. With more Sage developers, the gap would close faster!

What's available (over all number fields)

Extra functionality available both over $\mathbb{Q}$ and over general number fields:

- Conductor, local reduction data; global minimal models over class number one fields;

```
sage: K.<i> = NumberField(x^2+1)
sage: E = EllipticCurve([0,1,0,i,1+i])
sage: E.conductor()
Fractional ideal (-104*i - 472)
sage: E.conductor().factor()
(Fractional ideal (i + 1))^7 * (Fractional ideal (2*i + 1))^2 * (Fr
sage: E.local_data()

[Local data at Fractional ideal (i + 1) of Elliptic Curve defined b
Local minimal model: Elliptic Curve defined by y^2  = x^3 + x^2 + i
Minimal discriminant valuation: 8
Conductor exponent: 7
Kodaira Symbol: III
Tamagawa Number: 2,
 Local data at Fractional ideal (2*i + 1) of Elliptic Curve defined
Local minimal model: Elliptic Curve defined by y^2  = x^3 + x^2 + i
Minimal discriminant valuation: 2
```

- Rank and Mordell-Weil group via 2-descent (up to finite index in the case of number fields); torsion subgroup;

```
sage: E.simon_two_descent()
(2, 2, [(-i - 1 : 2 : 1), (-1 : 1 : 1)])
sage: E.rank() # Exercise: fix this bug today!
...
AttributeError: 'EllipticCurve_number_field' object has no attribut
sage: E.torsion_order()
1
```

- Rank and Mordell-Weil group via 2-descent (up to finite index in the case of number fields); torsion subgroup;

```
sage: E.simon_two_descent()
(2, 2, [(-i - 1 : 2 : 1), (-1 : 1 : 1)])
sage: E.rank() # Exercise: fix this bug today!
...
AttributeError: 'EllipticCurve_number_field' object has no attribut
sage: E.torsion_order()
1
```

- Rank and Mordell-Weil group via 2-descent (up to finite index in the case of number fields); torsion subgroup;

```
sage: E.simon_two_descent()
(2, 2, [(-i - 1 : 2 : 1), (-1 : 1 : 1)])
sage: E.rank() # Exercise: fix this bug today!
...
AttributeError: 'EllipticCurve_number_field' object has no attribut
sage: E.torsion_order()
1
```

```
sage: E = EllipticCurve('11a1')
sage: K.<a>=NumberField(x^4 + x^3 + 11*x^2 + 41*x + 101)
sage: EK=E.base_extend(K)
sage: tor = EK.torsion_subgroup()
sage: tor
Torsion Subgroup isomorphic to Multiplicative Abelian Group isomorp
 C5 x C5 associated to the Elliptic Curve defined by
 y^2 + y = x^3 + (-1)*x^2 + (-10)*x + (-20) over Number Field in a
 with defining polynomial x^4 + x^3 + 11*x^2 + 41*x + 101
sage: tor.gens()
((16 : 60 : 1), (a : 1/11*a^3 + 6/11*a^2 + 19/11*a + 48/11 : 1))
```

- Periods, elliptic logarithm and exponential ($z \mapsto (\wp(z), \wp'(z))$) (only for real embeddings so far);

```
sage: K.<a> = NumberField(x^2-2)
sage: E=EllipticCurve([0,0,0,a,2])
sage: embs=K.embeddings(RR); len(embs)
2
sage: # For each embedding we have a different period lattice:
sage: L = E.period_lattice(embs[0]); L
Period lattice associated to Elliptic Curve defined by y^2  = x^3 +
  From: Number Field in a with defining polynomial x^2 - 2
  To:   Real Field with 53 bits of precision
  Defn: a |--> -1.41421356237310
sage: L.basis()
(4.13107185270501681, -2.06553592635250840 + 0.98863042446910776*I
sage: P,Q = E.simon_two_descent()[2]
sage: P.elliptic_logarithm(embs[0])
1.5454184304794459018335238473
sage: P.elliptic_logarithm(embs[1])
0.62430211670806143068243968901
sage: sage: (2*P).elliptic_logarithm(embs[0]) / P.elliptic_logarit
2.0000000000000000000000000000
```

What's available (over $\mathbb{Q}$ only)

Extra functionality available over $\mathbb{Q}$ only:

- Point searching (up to a given bound on naive height).

```
sage: E = EllipticCurve('5077a1')
sage: E.point_search(10, verbose=False)
[(1 : -1 : 1), (-2 : 3 : 1), (-7/4 : 25/8 : 1)]
```

What's available (over $\mathbb{Q}$ only)

Extra functionality available over $\mathbb{Q}$ only:

- Point searching (up to a given bound on naive height).

```
sage: E = EllipticCurve('5077a1')
sage: E.point_search(10, verbose=False)
[(1 : -1 : 1), (-2 : 3 : 1), (-7/4 : 25/8 : 1)]
```

- Canonical heights and related functions (regulator, height pairing)

```
sage: [P.height() for P in E.gens()]
[1.36857250535393, 2.71735939281229, 0.668205165651928]
sage: E.height_pairing_matrix()
[ 1.36857250535393 -1.30957670708658 0.634867157837156]
[-1.30957670708658  2.71735939281229 -1.09981843056673]
[0.634867157837156 -1.09981843056673 0.668205165651928]
sage: E.regulator()
0.417143558758384
```

- Database of curves:

- Database of curves:
  - Default: Curves defined over $\mathbb{Q}$, conductor $< 10^4$ with equations and rank (but no generators);

- Database of curves:
  - Default: Curves defined over $\mathbb{Q}$, conductor $< 10^4$ with equations and rank (but no generators);
  - Optional (1.6MB extra): Curves defined over $\mathbb{Q}$, conductor $< 13 \cdot 10^4$ with equations, rank, generators and other data.

- Database of curves:
  - Default: Curves defined over $\mathbb{Q}$, conductor $< 10^4$ with equations and rank (but no generators);
  - Optional (1.6MB extra): Curves defined over $\mathbb{Q}$, conductor $< 13 \cdot 10^4$ with equations, rank, generators and other data.
  - Optional (2.6GB extra): Stein-Watkins database.

- Database of curves:
  - Default: Curves defined over $\mathbb{Q}$, conductor $< 10^4$ with equations and rank (but no generators);
  - Optional (1.6MB extra): Curves defined over $\mathbb{Q}$, conductor $< 13 \cdot 10^4$ with equations, rank, generators and other data.
  - Optional (2.6GB extra): Stein-Watkins database.
- ```
  sage: E.cremona_label()
  '5077a1'
  sage: for E in CremonaDatabase().iter(srange(1000,1002)):
      print E.label(), E.ainvs(), E.rank(), E.modular_degree()
  ....:
  1001a1 [0, -1, 1, -15881, 778423] 1 1680
  1001b1 [1, -1, 1, -16, -198] 0 152
  1001b2 [1, -1, 1, -621, -5764] 0 304
  1001b3 [1, -1, 1, -9916, -377564] 0 608
  1001b4 [1, -1, 1, -1006, 2552] 0 608
  1001c1 [0, 0, 1, -199, 1092] 2 1008
  ```
  ```

- Database of curves:
  - Default: Curves defined over $\mathbb{Q}$, conductor $< 10^4$ with equations and rank (but no generators);
  - Optional (1.6MB extra): Curves defined over $\mathbb{Q}$, conductor $< 13 \cdot 10^4$ with equations, rank, generators and other data.
  - Optional (2.6GB extra): Stein-Watkins database.
- ```
  sage: E.cremona_label()
  '5077a1'
  sage: for E in CremonaDatabase().iter(srange(1000,1002)):
      print E.label(), E.ainvs(), E.rank(), E.modular_degree()
  ....:
  1001a1 [0, -1, 1, -15881, 778423] 1 1680
  1001b1 [1, -1, 1, -16, -198] 0 152
  1001b2 [1, -1, 1, -621, -5764] 0 304
  1001b3 [1, -1, 1, -9916, -377564] 0 608
  1001b4 [1, -1, 1, -1006, 2552] 0 608
  1001c1 [0, 0, 1, -199, 1092] 2 1008
  ```
- Analytic rank;
  ```
  sage: E = EllipticCurve('5077a1')
  sage: E.analytic_rank()
  3
  ```
  ```

- Modular degree;

```
sage: E = EllipticCurve('5077a1')
sage: E.modular_degree()
1984
```

- Modular degree;

```
sage: E = EllipticCurve('5077a1')
sage: E.modular_degree()
1984
```

- integral points.

```
sage: E = EllipticCurve('5077a1')
sage: E.integral_points()
[(-3 : 0 : 1), (-2 : 3 : 1), (-1 : 3 : 1), (0 : 2 : 1), (1 : 0 : 1)
 (2 : 0 : 1), (3 : 3 : 1), (4 : 6 : 1), (8 : 21 : 1), (11 : 35 : 1)
 (14 : 51 : 1), (21 : 95 : 1), (37 : 224 : 1), (52 : 374 : 1), (93
 896 : 1), (342 : 6324 : 1), (406 : 8180 : 1), (816 : 23309 : 1)]
```

What's not yet available

Most of the following would be easy to implement; some would be more challenging!

- Canonical heights and related functions over general number fields

What's not yet available

Most of the following would be easy to implement; some would be more challenging!

- Canonical heights and related functions over general number fields
- Isogenies, and computation of isogeny classes (only available over $\mathbb{Q}$ so far);

What's not yet available

Most of the following would be easy to implement; some would be more challenging!

- Canonical heights and related functions over general number fields
- Isogenies, and computation of isogeny classes (only available over $\mathbb{Q}$ so far);
- higher descents;

What's not yet available

Most of the following would be easy to implement; some would be more challenging!

- Canonical heights and related functions over general number fields
- Isogenies, and computation of isogeny classes (only available over $\mathbb{Q}$ so far);
- higher descents;
- other models for curves of genus 1.

What's not yet available

Most of the following would be easy to implement; some would be more challenging!

- Canonical heights and related functions over general number fields
- Isogenies, and computation of isogeny classes (only available over $\mathbb{Q}$ so far);
- higher descents;
- other models for curves of genus 1.
- S-integral points.

Elliptic Curves over finite fields

Sage's elliptic curve functionality over function fields applies to curves defined over general fields $\mathbb{F}_q$, not just prime fields $\mathbb{F}_p$, for which q is of "reasonable size". Just one function has a more sophisticated implementation: the cardinality of elliptic curves defined over prime fields (or whose j-invariant lies in a prime field) is computed using an implementation of the SEA algorithm.

Elliptic Curves over finite fields

Sage's elliptic curve functionality over function fields applies to curves defined over general fields $\mathbb{F}_q$, not just prime fields $\mathbb{F}_p$, for which q is of "reasonable size". Just one function has a more sophisticated implementation: the cardinality of elliptic curves defined over prime fields (or whose j-invariant lies in a prime field) is computed using an implementation of the SEA algorithm.

```
sage: p=next_prime(10^100); F=GF(p)
sage: E=EllipticCurve(F,[F.random_element(),F.random_element()])
sage: time E.cardinality()
CPU times: user 0.00 s, sys: 0.00 s, total: 0.00 s
Wall time: 50.82 s
999999999999999999999999999999999999999999999924484601817481249\
 6664153533227107630096744347580
```

Elliptic Curves over finite fields

Sage's elliptic curve functionality over function fields applies to curves defined over general fields $\mathbb{F}_q$, not just prime fields $\mathbb{F}_p$, for which q is of "reasonable size". Just one function has a more sophisticated implementation: the cardinality of elliptic curves defined over prime fields (or whose j-invariant lies in a prime field) is computed using an implementation of the SEA algorithm.

```
sage: p=next_prime(10^100); F=GF(p)
sage: E=EllipticCurve(F,[F.random_element(),F.random_element()])
sage: time E.cardinality()
CPU times: user 0.00 s, sys: 0.00 s, total: 0.00 s
Wall time: 50.82 s
9999999999999999999999999999999999999999999999999924484601817481249\
 666415353322710763009674434747580

sage: time E.cardinality(extension_degree=2)
CPU times: user 0.03 s, sys: 0.00 s, total: 0.03 s
Wall time: 0.04 s

10000000000000000000000000000000000000000000000000000000000000000000\
 00000000000000000000000000000053542974246373356438188803510556168\
 45842378799234068724513647475437926957999836869874208002198708846480
```

```
sage: time E.cardinality(extension_degree=10)
CPU times: user 0.03 s, sys: 0.01 s, total: 0.04 s
Wall time: 0.03 s
10000000000000000000000000000000000000000000000000000000000000000000000
 0000000000000000000000267000000000000000000000000000000000000000000000
 0000000000000000000000000000000000000000000003208005000000000000000000
 0000000000000000000000000000000000000000000000000000000000000000000022
 9560000000000000000000000000000000000000000000000000000000000000000000
 0000000000000000010672455194100000000000000000000000000000000000000000
 0000000000000000000000000000000000000000034194546441896250990367930655
 8413992288628676723037333475277734238808648796181254290325896287730664
 6977098023406756762100771768307415676466144681323148754974851742336757
 2985220206385127443146007356771016247387298066163303608462469988742895
 4958816297084426889934185378760632059951175059642199636148487990929243
 2030474846677029455055134869488177457480108482235365008566720108764238
 4362300827190649821619217590522250066395576150367971604311391767416266
 55595237149578942394113993363715920400
```

As well as the cardinality of the group $E(\mathbb{F}_q)$ we can compute the abelian group structure and generators:

```
sage: K.<i> = QuadraticField(-1)
sage: OK = K.ring_of_integers()
sage: P=K.factor(10007)[0][0]
sage: OKmodP = OK.residue_field(P)
sage: E = EllipticCurve([0,0,0,i,i+3])
sage: Emod = E.change_ring(OKmodP); Emod
Elliptic Curve defined by y^2  = x^3 + ibar*x + (ibar+3) over Residue f
 in ibar of Fractional ideal (10007)
sage: Emod.abelian_group()
(Multiplicative Abelian Group isomorphic to C50067594 x C2,
 ((9538*ibar + 3564 : 9291*ibar + 8885 : 1), (2425*ibar + 4050 : 0 : 1)
```

We have elliptic logarithms, implemented using a generic algorithm based on Shanks' Baby-Step-Giant-step:

```
sage: P,Q = Emod.abelian_group()[1]
sage: P.order()
50067594
sage: n = randint(0,P.order())
sage: Q = n*P
sage: P.discrete_log(Q)
18055058
```

We have elliptic logarithms, implemented using a generic algorithm based on Shanks' Baby-Step-Giant-step:

```
sage: P,Q = Emod.abelian_group()[1]
sage: P.order()
50067594
sage: n = randint(0,P.order())
sage: Q = n*P
sage: P.discrete_log(Q)
18055058

sage: n
18055058
```

Authors

- Authors of Sage code: Nick Alexander, Jennifer Balakrishnan, Robert Bradshaw, John Cremona, David Harvey, David Kohel, Michael Mardaus, Tobias Nagel, William Stein, Chris Wuthrich, Liang Xiao.
- Authors of wrapped code, library code or other adapted code: Tim Dokchitser, the `pari/gp` team, Alice Silverberg, Denis Simon, Karl Rubin, Mike Rubinstein, Mark Watkins.

Apologies for any omissions!