In [1]:
```
import numpy as np
from scipy.optimize import fmin
import astropy.units as u
from astropy import constants

%matplotlib inline
import matplotlib.pyplot as plt
```

In [2]:
```
# change plotting defaults
plt.rc('axes', labelsize=14)
plt.rc('axes', labelweight='bold')
plt.rc('axes', titlesize=16)
plt.rc('axes', titleweight='bold')
plt.rc('font', family='sans-serif')
plt.rcParams['errorbar.capsize'] = 3
opts = {'mec':'k', 'mew': 0.5, 'lw': 1}
plt.rcParams['figure.figsize'] = (12, 8)
plt.rcParams['xtick.direction'] = 'out'
plt.rcParams['ytick.direction'] = 'out'
plt.rcParams['xtick.top'] = True
plt.rcParams['ytick.right'] = True
plt.rcParams['xtick.minor.visible'] = True
plt.rcParams['ytick.minor.visible'] = True
plt.rcParams['axes.grid'] = True
```

Dark Matter

Step 0. Background & Motivation

Over the course of this program, you've been introduced to the concept of dark matter, and its importance in galaxy formation. However, the discovery of dark matter was no mere feat. As its name implies, dark matter does not absorb nor reflect electromagnetic radiation (light), so we cannot rely on visual detection of this matter. Instead, our knowledge of the laws of gravity and motion led to the discovery of dark matter.

In this activity, you will be computing the *total mass* of the galaxy NGC 2742 using these same gravitational laws, and comparing/contrasting this against the mass one would estimate from the matter we can see, otherwise known as the *luminous mass*.

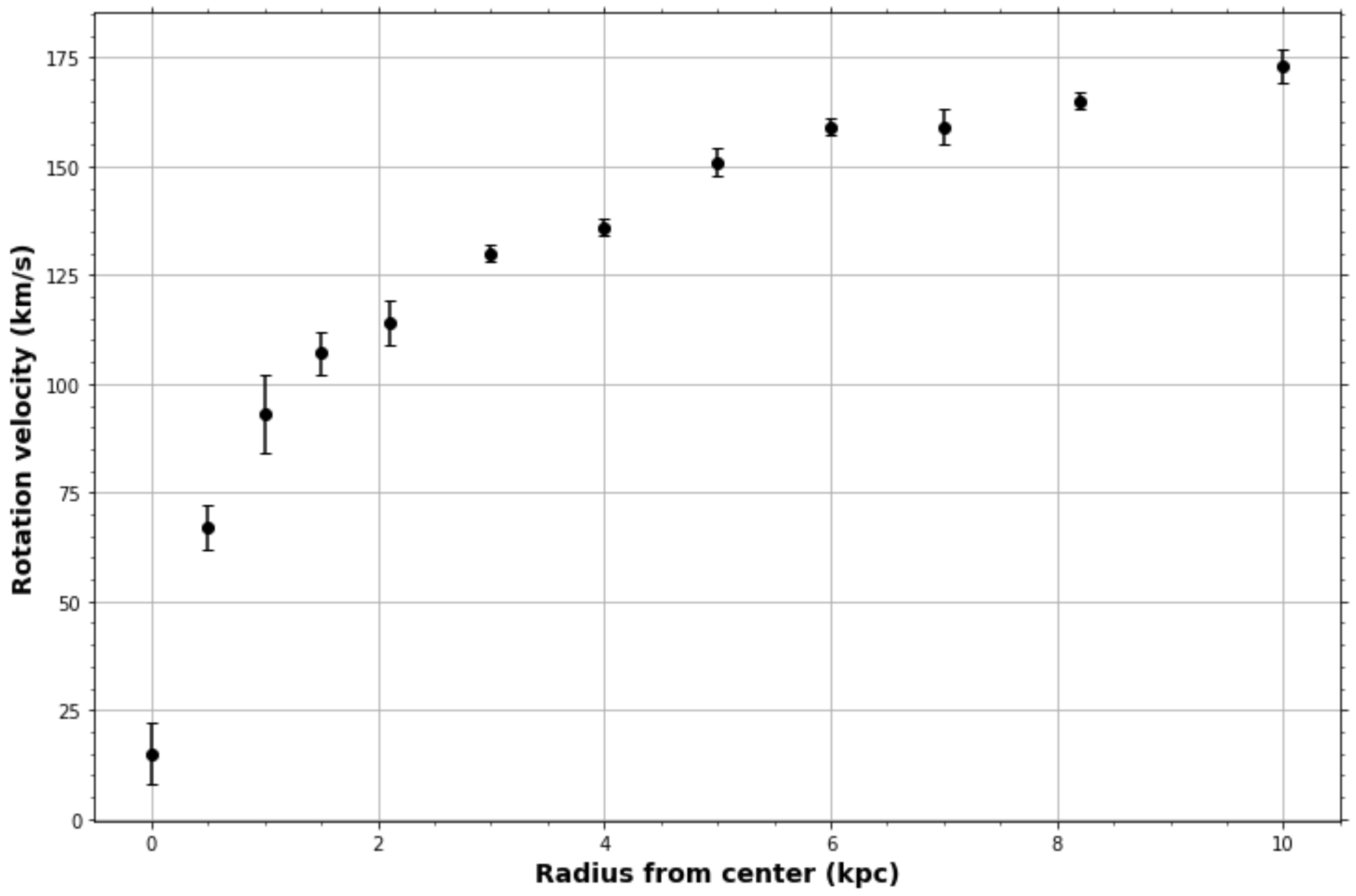
Since the overwhelming majority of this mass is represented by stars, this is also referred to as a galaxy's *stellar mass*. To do this, you will be using real rotation data collected by Vera Rubin et al. in 1985, and an analytical model of the galaxy's integrated light.

Optional Portion: For the first step of this activity, you may choose to use χ^2 minimization to better constrain your model parameters. For a tutorial on this technique, check out this video (https://www.youtube.com/watch?v=TSNV-4K3Fws) by Prof. Quimby.

Step 1. Compute a Rotation Curve

Provided in an ASCII file in this directory is data for the rotation curve of the galaxy NGC 2742 provided by Rubin et al. 1985 (https://ui.adsabs.harvard.edu/abs/1985ApJ...289...81R/abstract).

```
In [3]: data = np.genfromtxt('ngc2742rv.dat', comments='#', names='r, v, err')
        plt.errorbar(data['r'], data['v'], data['err'], marker='o', c='k', ls='none')
        plt.xlabel('Radius from center (kpc)')
        plt.ylabel('Rotation velocity (km/s)');
```



In the field of astronomy, we often like to find trends in data. This may be done by fitting a curve. Typically, we have to determine the nature of the function ourselves, but you can use the following relation for the Rubin et al. data:

$$V = aR^b$$

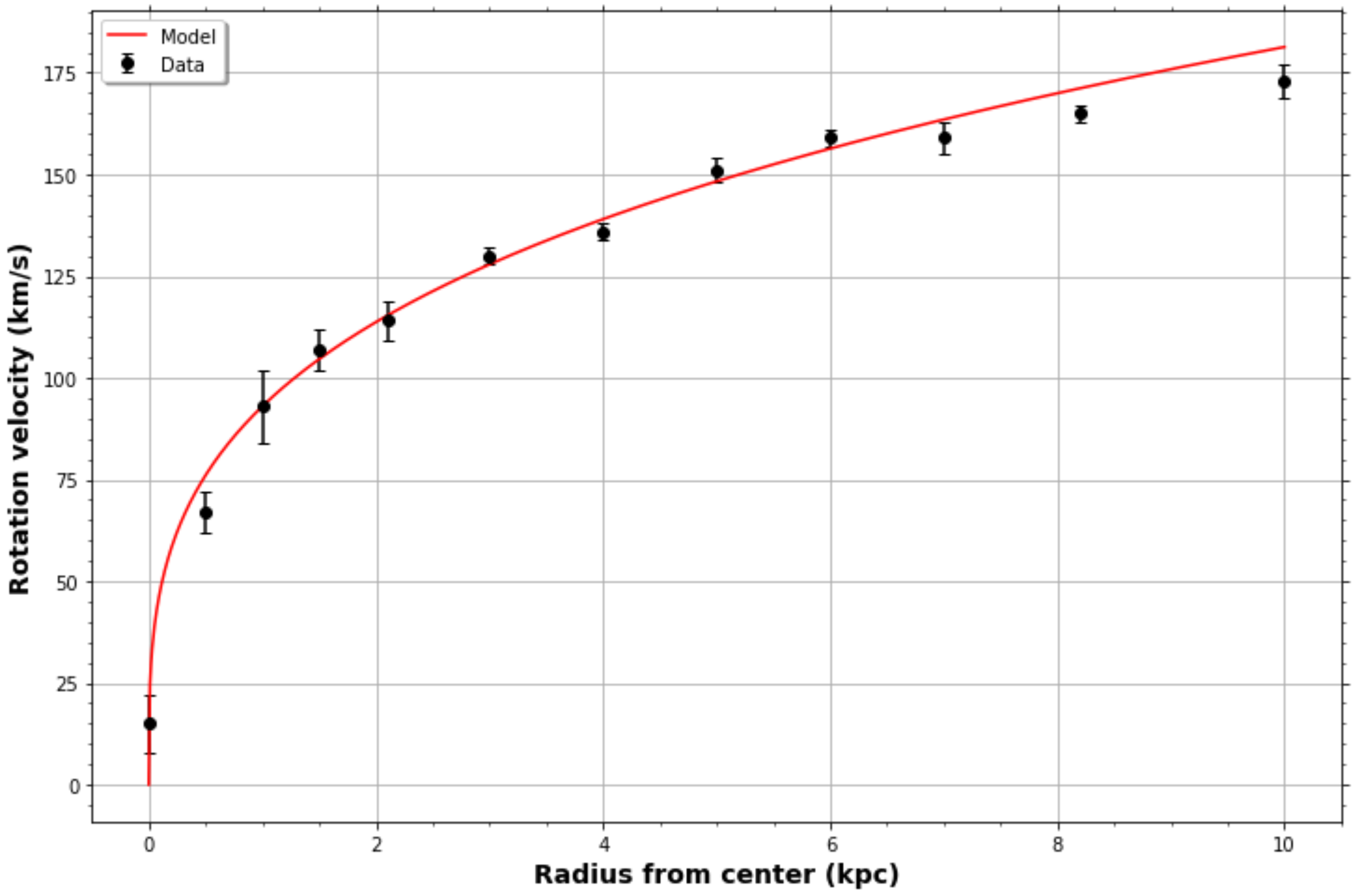
where V is the rotation velocity in km/s, R is the radius from center in kpc, and a and b are the fitting parameters. First, try to get a fit by eye utilizing `matplotlib` to visualize how the parameters a, b change the fit of the curve to the data.

In [4]:
```
# define your model
def rotation_model(params, rs):
    a, b = params
    vs = a*rs**b
    return vs
```

```
In [5]: # try a fit by-hand; plot your results to check as you go
        params = (93,0.29)

        rmod = np.linspace(0,10,1000)
        vmod = rotation_model(params, rmod)

        plt.errorbar(data['r'], data['v'], data['err'], marker='o', c='k', ls='none', label='Data')
        plt.plot(rmod, vmod, c='r', label='Model')
        plt.xlabel('Radius from center (kpc)')
        plt.ylabel('Rotation velocity (km/s)')
        plt.legend(shadow=True);
```



Optional Step 1.5: χ^2 Minimization

You may utilize χ^2 minimization to obtain best-fit parameters in a more numerically robust way. Remember:

$$\chi^2 = \sum_i \left(\frac{\Delta Y_i}{\sigma_i}\right)^2$$

In [6]:
```python
def get_chisq(params, data):
    model = rotation_model(params, data['r'])
    dRV = model - data['v']
    return np.sum(dRV**2/data['err']**2)
```

In [7]:
```python
# test your code
get_chisq(params, data)
```

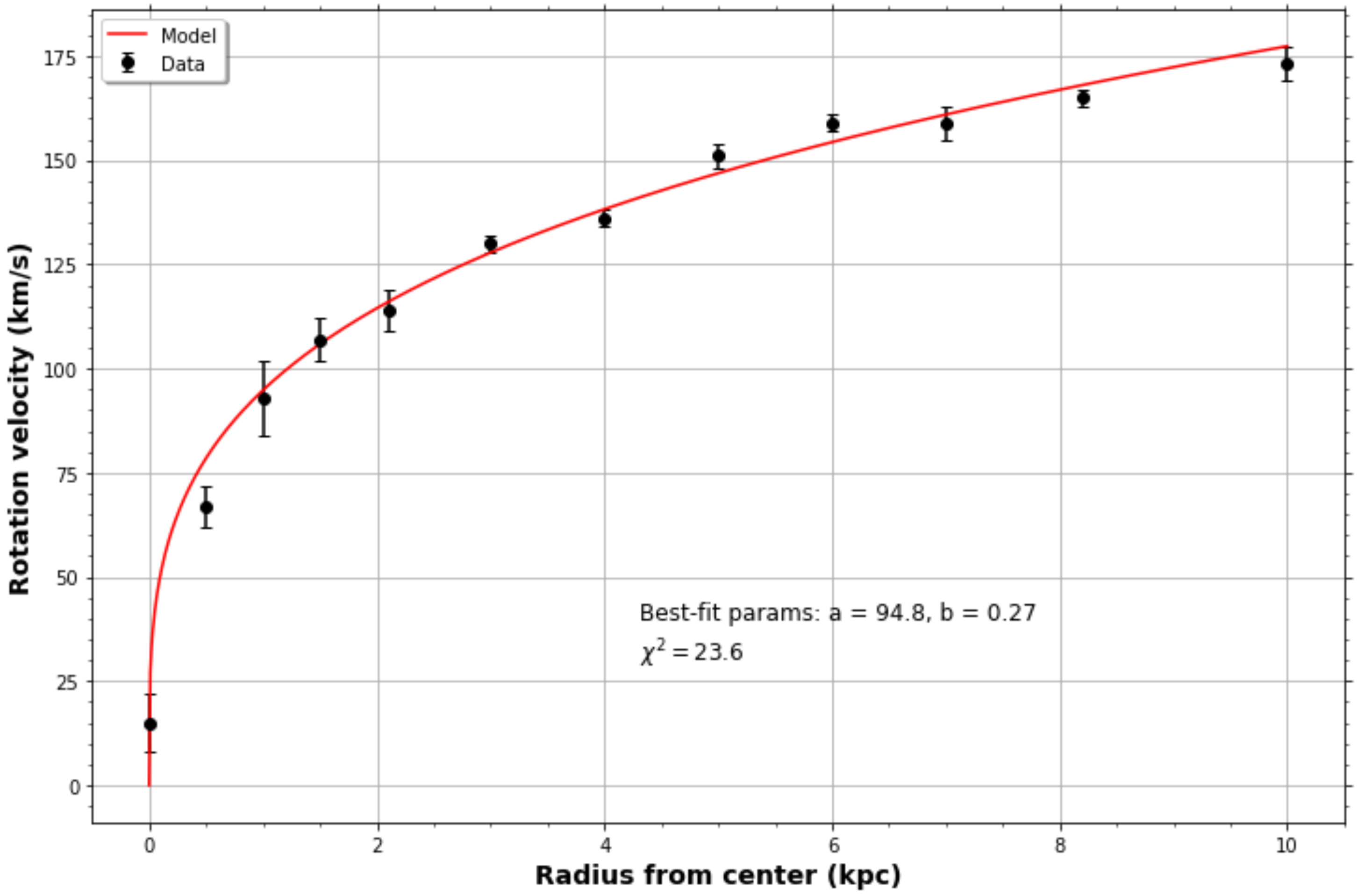
Out[7]:
```
29.31259808852238
```

In [8]:
```python
# get best-fit parameters with chi-square minimization
params = fmin(get_chisq, params, args=(data, ))
print(params)
```

```
Optimization terminated successfully.
         Current function value: 23.600363
         Iterations: 40
         Function evaluations: 77
[94.80592389  0.27202896]
```

In [9]:
```python
# evaluate a model
vmod = rotation_model(params, rmod)
chisq = get_chisq(params, data)

# plot your best-fit model vs. data
plt.errorbar(data['r'], data['v'], data['err'], marker='o', c='k', ls='none', label='Data')
plt.plot(rmod, vmod, c='r', label='Model')
plt.text(4.3,40, f"Best-fit params: a = {params[0]:.1f}, b = {params[1]:.2f}", size=12)
plt.text(4.3,30, f"$\\chi^2 = {chisq:.1f}$", size=12)
plt.xlabel('Radius from center (kpc)')
plt.ylabel('Rotation velocity (km/s)')
plt.legend(shadow=True);
```



Step 2. Compute a Cumulous Luminosity Distribution

In order to determine the luminous mass of a galaxy, we would need a surface brightness profile an a stellar mass-to-light ratio. Measuring these is hard work. To make your life easier, you can utilize this formula for the luminosity contained with the radius from center, or a cumulous luminosity distribution (CLD):

$$L_R = 2\pi h^2 \Sigma_0 \left[1 - e^{-R/h}\left(1 + \frac{R}{h}\right)\right]$$

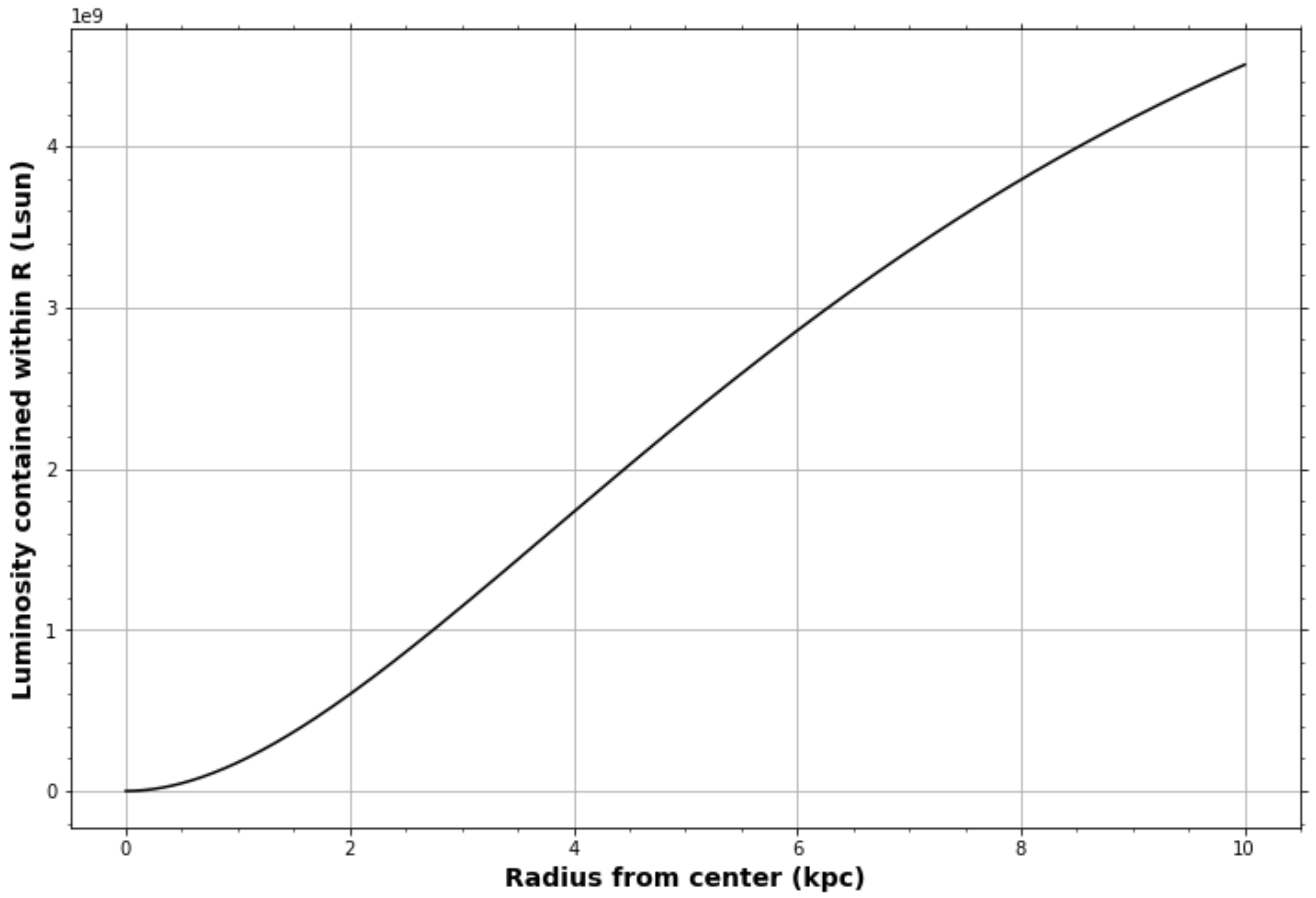
where L_R is the luminosity contained within R, $h = 3.8$ kpc is a scale height (the distance in R space across which L_R decreases by a factor of e), and $\Sigma_0 = 6.725 \times 10^7\ \mathrm{L_\odot/kpc^2}$ is the surface brightness density (the stellar light per unit area). You may want to write a function for this.

NOTE: These values of h and Σ_0 are measured specifically for NGC 2742

In [10]:
```
# compute the CLD as a function of radius
def get_clum(rs, h=3.8, Sig0=6.725e7):
    A = 2*np.pi*h**2*Sig0
    cls = A*(1 - np.exp(-rs/h)*(1 + rs/h)) # Lsol
    return cls

clmod = get_clum(rmod)
```

```
In [11]: # plot the CLD
         plt.plot(rmod, clmod, c='k')
         plt.xlabel('Radius from center (kpc)')
         plt.ylabel('Luminosity contained within R (Lsun)');
```



Step 3. Compute a Mass From Rotation

As discussed in the lectures by Prof. Sandquist on gravity, rotation velocity and mass contained within R (M_V) are related in the following way:

$$M_V = \frac{V^2 R}{G}$$

where G is Newton's gravitational constants. This value is available in `astropy.constants`. **Make sure you're using the correct units to compute a result in Solar masses.**

In [12]:
```
# compute the total mass
G = constants.G.to_value(u.kpc * u.km**2 / u.solMass / u.s**2) # ensure proper units
def get_mtot(vs, rs):
    global G
    mtot = vs**2*rs/G
    return mtot

mtot = get_mtot(vmod, rmod)
```

Step 4. Compute a Mass from Light

In order to compute the luminous (stellar) mass from this CLD, we will need a stellar mass-to-light ratio $\left(\Upsilon \equiv \frac{M_\star}{L}\right)$, which is the luminosity per unit mass produced by the stars in a galaxy. If given this quantity, the formula is simply stated as:

$$M_\star = \Upsilon L_R$$

For simplicity, assume a relatively typical value of $\Upsilon = 2.0\ \mathrm{M}_\odot/\mathrm{L}_\odot$.

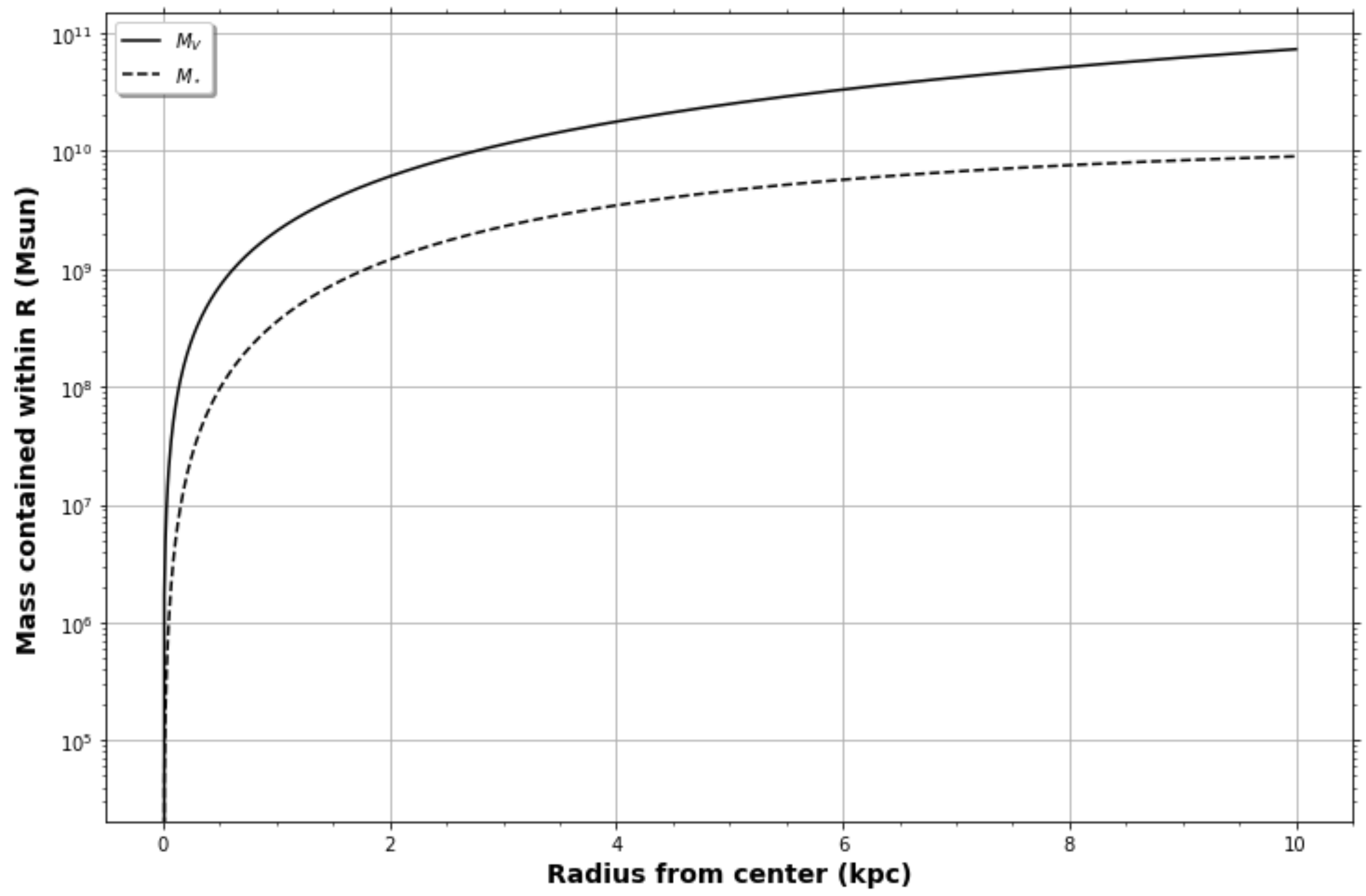
In [13]:
```
# compute the luminous mass
def get_mlum(clums, Ups=2.0):
    mlums = clums*Ups
    return mlums

mlum = get_mlum(clmod)
```

Step 5. Plot Your Results

On the same graph, plot the total mass computed from rotation and the luminous mass against the radius from center. You may want to make this a semi-log plot (i.e. make the y-axis scale logarithmic).

In [14]:
```
# make a plot of your total and luminous mass
plt.plot(rmod, mtot, ls='-', c='k', label='$M_V$')
plt.plot(rmod, mlum, ls='--', c='k', label='$M_\\star$')
plt.yscale('log')
plt.xlabel('Radius from center (kpc)')
plt.ylabel('Mass contained within R (Msun)')
plt.legend(shadow=True);
```



What do you notice about this curve? Is it surprising? Why or why not?

In [15]:
```
#### answer goes here
```