

Luminosity Distance - PHYS417 Project 2

Ryan Cox

2022-10-16

1 The Friedmann Equation

The Hubble parameter is given by

$$H(t)^2 = \frac{8\pi G}{3}\rho - \frac{k}{a^2} + \frac{\Lambda}{3}$$

We can define

$$\begin{aligned}\rho_c &= \frac{3H(t)^2}{8\pi G} \\ \Omega &= \frac{\rho}{\rho_c} = \frac{8\pi G}{3H(t)^2}\rho \\ \Omega_k &= -\frac{k}{H(t)^2 a^2} \\ \Omega_\Lambda &= \frac{\Lambda}{3H(t)^2}\end{aligned}$$

as seen in Lecture 2. We see we can rewrite our first equation as

$$\frac{H^2}{H_0^2} = \frac{8\pi G}{3H_0^2}\rho - \frac{k}{H_0^2 a^2} + \frac{\Lambda}{H_0^2}$$

where we have dropped the t in our notation for $H(t)$ and used a subscript zero to indicate quantities at the present time. If we assume

$$\rho = \frac{\rho_0}{a^3}$$

and use $a_0 = 1$ as holds for matter density, in a matter dominated universe or mixture, this transforms into

$$\frac{H^2}{H_0^2} = \frac{\Omega_0}{a^3} + \frac{\Omega_{k,0}}{a^2} + \Omega_{\Lambda,0}$$

Then it is just a matter of using

$$a = \frac{1}{1+z}$$

to find

$$\frac{H^2}{H_0^2} = \Omega_0(1+z)^3 + \Omega_{k,0}(1+z)^2 + \Omega_{\Lambda,0}$$

or

$$H = H_0 \sqrt{\Omega_0(1+z)^3 + \Omega_{k,0}(1+z)^2 + \Omega_{\Lambda,0}}$$

1.1 Luminosity Distance Behaviour

Luminosity distance is given by

$$d_L = d_p(1+z)$$

where the physical (comoving) distance is given by

$$d_p = \frac{c}{H_0} \int_0^z \frac{dz'}{E(z')}$$

In a universe dominated by dark energy

$$\begin{aligned}\Omega_{\Lambda,0} &= 1 \\ E(z) &= \sqrt{\Omega_{\Lambda,0}} = 1 \\ d_p &= \frac{cz}{H_0}\end{aligned}$$

Then luminosity distance is simply

$$d_L = \frac{cz}{H_0}(1+z)$$

For large redshifts we get

$$d_L \propto z^2$$

In a universe dominated by matter

$$\begin{aligned}\Omega_0 &= 1 \\ E(z) &= (1+z)^{3/2} \\ d_p &= \frac{2c}{H_0} \left(1 - \frac{1}{\sqrt{1+z}} \right)\end{aligned}$$

Then luminosity distance is

$$d_L = \frac{2c}{H_0} (1+z - \sqrt{1+z})$$

For large redshifts we get

$$d_L \propto z$$

1.2 Luminosity Distance Plots

In a universe with both matter and dark energy we need to find d_p by numerical integration of

$$d_p(z) = \frac{c}{H_0} \int_0^z \frac{dz'}{E(z')}$$

From here on out we'll assume a flat universe, unless stated otherwise.

```
using Unitful # Unit handling
using UnitfulAstro # Astronomical units
using PhysicalConstants.CODATA2018: c_0 # Speed of light from CODATA2018, with units
using QuadGK # Numerical integration
using Plots, LaTeXify, UnitfulLatexify, LaTeXStrings
using Measurements # Uncertainly handling

# Unitful doesn't export preferunits so we have to reference by package
Unitful.preferunits(u"Mpc", u"Msun")

# Define our cosmological parameters
Ω0::Real = 0.3
Ωk::Real = 0 # flat universe
ΩΛ::Real = 1 - Ω0
H0 = 70.0u"km/s/Mpc"

# This is a one line function definition
E(z::Real)::Number = sqrt(Ω0*(1+z)^3 + Ωk*(1+z)^2 + ΩΛ)

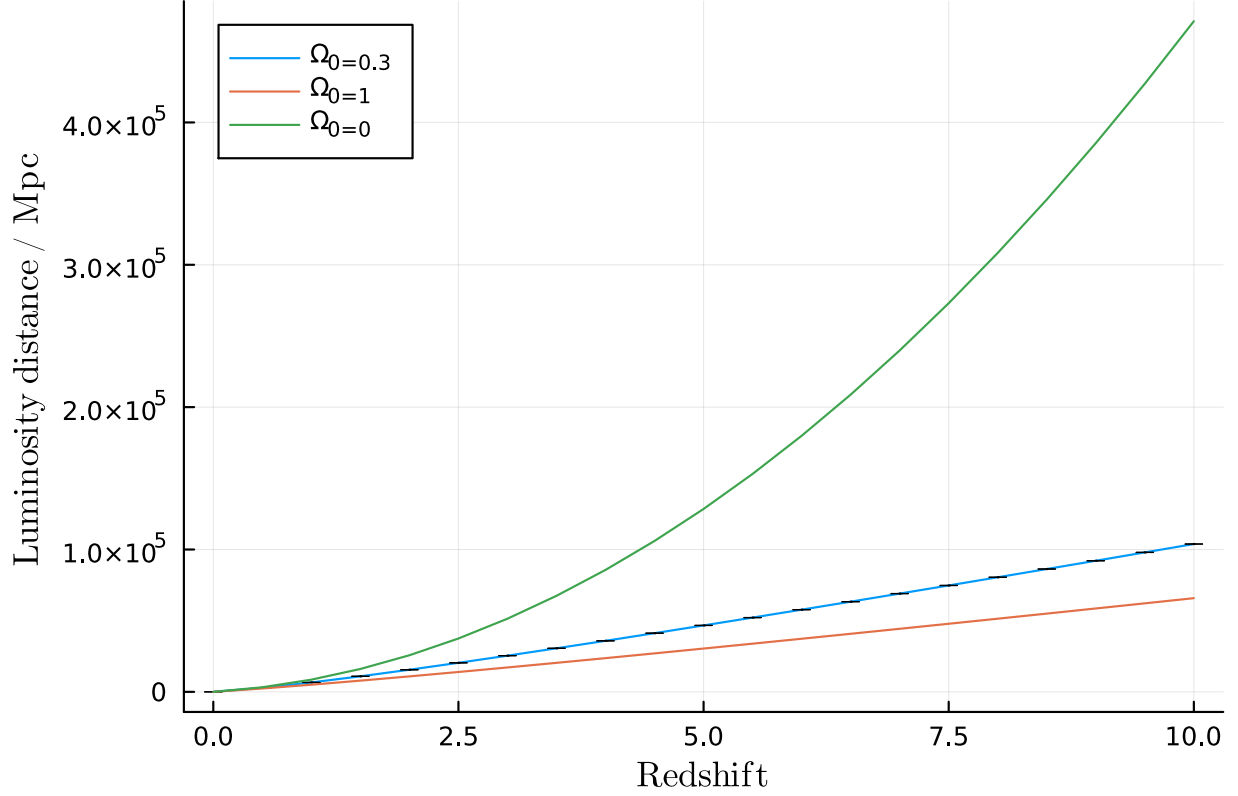
# Input type must be real and the output must be a length
# Unitful will determine and check the dimensions of the output
function dp(z::Real)::Unitful.Length
    """Calculate proper distance from redshift."""
    integral, err = quadgk(z->1/E(z), 0, z, rtol=1e-8)
    return c_0/H0 .* (integral ± err)
end

dl(z::Real) = dp(z) * (1+z)

z = 0:0.5:10
# dl.(z) vectorises dl so it acts elementwise on z
plot(z,
    upreferred.(dl.(z)),
    unitformat=latexify,
    label="\Omega_0=$(Ω0)",
    legend=:topleft,
    xlabel=L"\mathrm{Redshift}",
    ylabel="\mathrm{Luminosity distance}"
)

# Matter dominated -> no dark energy
Ω0::Real = 1
ΩΛ::Real = 1 - Ω0
plot!(z, 2 * c_0 / H0 * (1 + z - sqrt(1 + z)), label="\Omega_0=$(Ω0)") # plot!()
updates last plot
# Testing using the numerical solution agrees with the exact solution, but I've
# left it off the final plot since you can't see both lines at once.
# plot!(z, dl.(z), label="\Omega_0=$(Ω0)")
```

```
# Dark energy dominated -> no matter
Ω0::Real = 0
ΩΛ::Real = 1 - Ω0
plot!(z, c_0 / H0 * z .* (1 .+ z), label="\Omega_0=$(Ω0)")
# plot!(z, dl.(z), label="\Omega_0=$(Ω0)")
```



In our dark energy dominated universe the rate of expansion is approximately constant. In our matter dominated universe the rate of expansion is approximately proportional to $z^{3/2}$. This means the mass density was the primary driver of the expansion of the universe in the early days but as space expanded it fell and so the effect lessened. The cosmological constant remains constant regardless of redshift and so can dominate in the present if $\Omega_\Lambda > \Omega_0$.

2 Candles

We know that radial velocity is related to proper distance by

$$v_r = H_0 d_p$$

In the low redshift (and thus non-relativistic) limit

$$v_r \approx zc$$

so we can conclude

$$d_p \approx \frac{zc}{H_0}$$

Combining this with our luminosity distance equation and gives

$$d_l \approx \frac{z c}{H_0} (1 + z) \approx \frac{z c}{H_0} \text{ as } z \rightarrow 0$$

```
using CSV, DataFrames

dl_from_m(dm::Number)::Unitful.Length = 10u"pc" * 10^(dm/5)

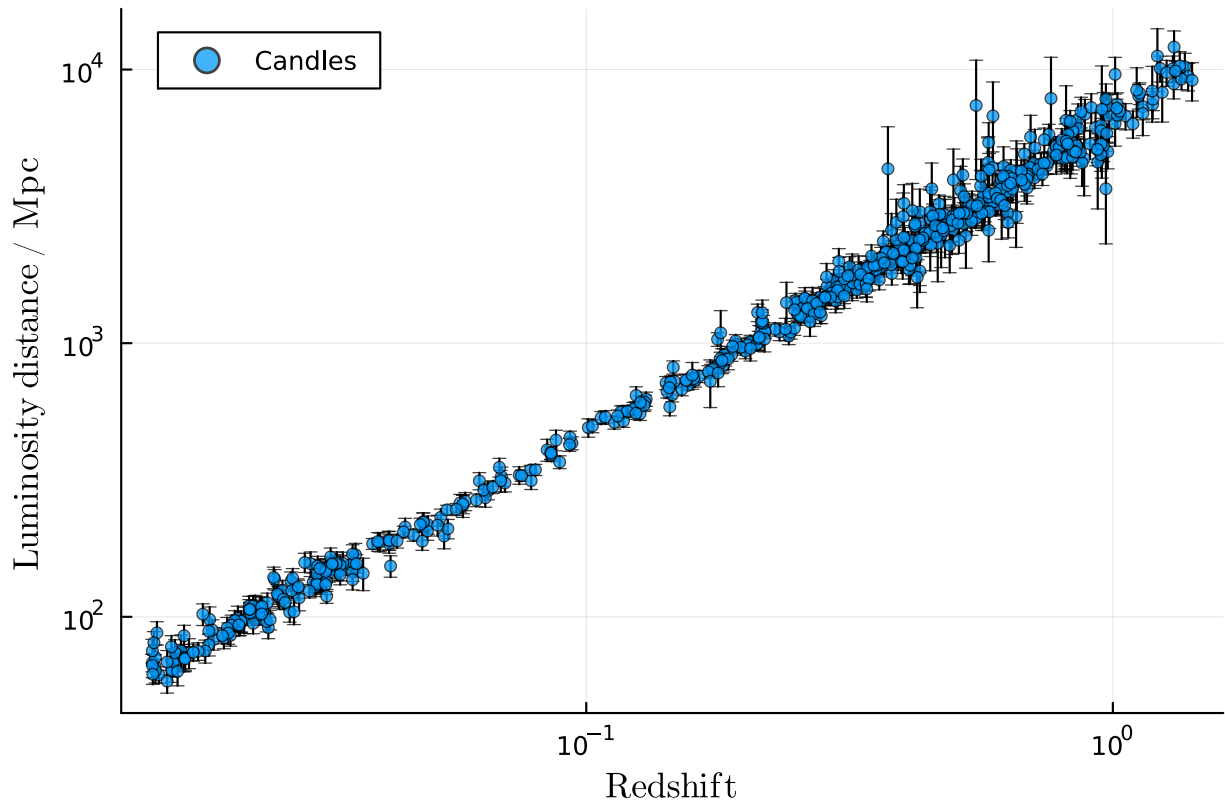
data = CSV.read("SCPUnion2.1_mu_vs_z.txt",
                DataFrame,
                header=["id", "z", "DMVal", "DMErr", "Prob"],
                skipto=6
            )

data.DM = data.DMVal .± data.DMErr

select!(data, :id, :z, :DM) # Get rid of the other columns

data.dl = upreferred.(dl_from_m.(data.DM))

scatter(data.z,
        data.dl,
        unitformat=latexify,
        label="Candles",
        legend=:topleft,
        xlabel=L"\mathrm{Redshift}",
        ylabel=L"\mathrm{Luminosity}\ distance",
        markersize=3,
        markeralpha=0.75,
        xscale=:log10,
        yscale=:log10
    )
```



2.1 Low Redshift

We can perform basic least squares in Julia with remarkable ease. Remarkably it propagates the uncertainties right through. However it doesn't like units so we'll have to strip them (carefully!). The line of best fit is underlaid by a light orange region showing variation from the propagated uncertainty in H_0 and a grey region showing the standard deviation of the data as estimated from the fitting. The first region is very narrow and thus hard to see.

```
using Statistics: mean

lowz_data = @view data[data.z .< 0.1, [:z, :dl]] # get z and DM for low redshift
kernel = lowz_data.z * ustrip(u"km/s", c_0) # construct kernel. dl = G*(1/H0)

reciprocal_H0 = kernel \ ustrip(u"Mpc", lowz_data.dl) # This performs least squares

lowz_H0 = 1u"km/s/Mpc" / reciprocal_H0
println("Estimated H0 = $(lowz_H0)")

dl_mean = mean(lowz_data.dl)
# sum of squares error
fit_err = sum(abs2, Measurements.value.(lowz_data.dl) -
lowz_data.z*c_0/Measurements.value(lowz_H0))
# sum of variance
var = sum(abs2, Measurements.value.(lowz_data.dl .- Measurements.value.(dl_mean)))
# Coefficient of determination
println("R^2 = $(1 - fit_err/var)")
# Std of dl from fit error
lowz_err = sqrt(1/(length(lowz_data.z)-1) * fit_err)

# plot data
scatter(lowz_data.z,
        lowz_data.dl,
        unitformat=latexify,
        label="Candles",
        legend=:topleft,
        xlabel=L"\mathrm{Redshift}",
        ylabel="\mathrm{Luminosity\\ distance}",
        markersize=3,
        markeralpha=0.75
)

z = 0:0.01:0.1
dlz = z*c_0/Measurements.value(lowz_H0)

# plot fitted line
plot!(z, dlz, linewidth=1, label="Fit")

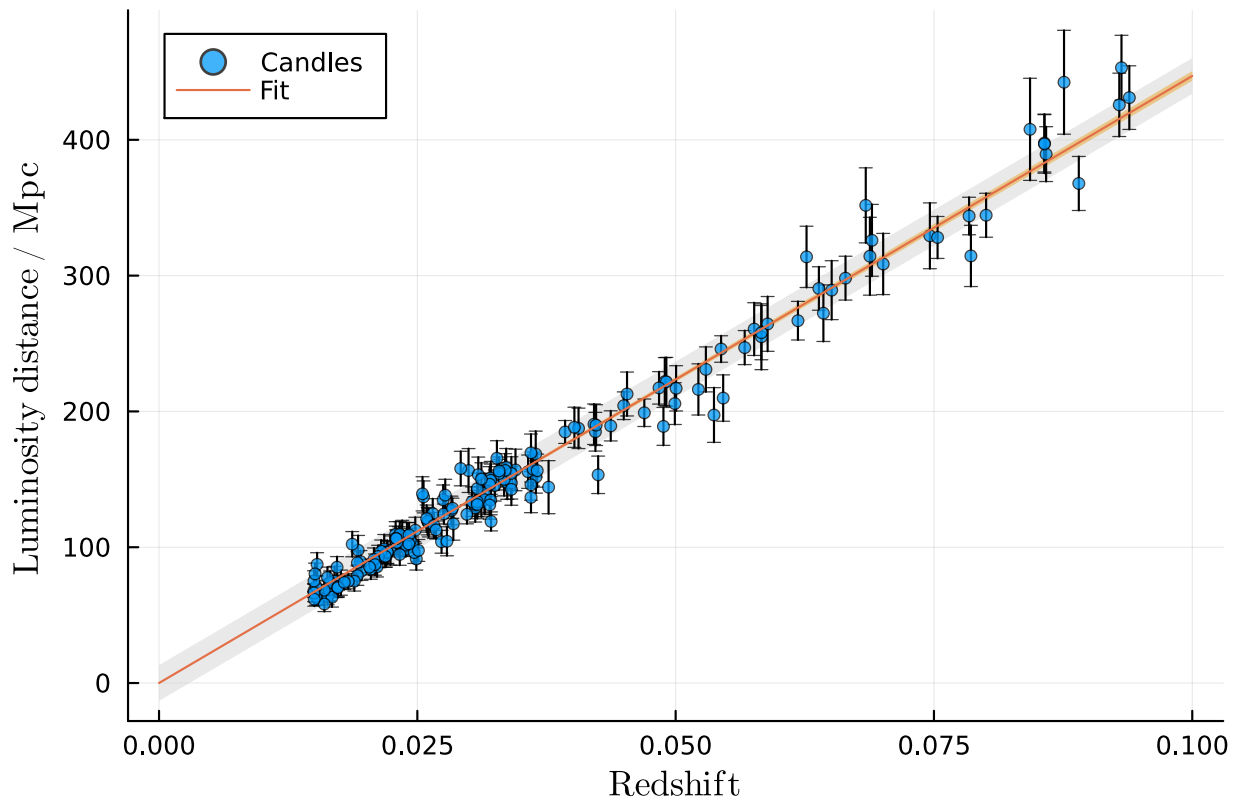
# plot area around fitted line equal to ± standard deviation
plot!(
    z,
    dlz .+ lowz_err,
    fillrange = dlz .- lowz_err,
    fillcolor = :lightgray,
    fillalpha = 0.5,
    linecolor = nothing,
    primary = false, # no legend entry
    z_order = :back
)
```

```

# plot area around fitted line equal to minimum and maximum from uncertainties
plot!(
    z,
    z*c_0/(Measurements.value(lowz_H0)-Measurements.uncertainty(lowz_H0)),
    fillrange = z*c_0/(Measurements.value(lowz_H0)+Measurements.uncertainty(lowz_H0)),
    fillcolor = :orange,
    fillalpha = 0.75,
    linecolor = nothing,
    primary = false, # no legend entry
    z_order = :back
)

```

Estimated $H_0 = 67.07 \pm 0.53 \text{ km Mpc}^{-1} \text{ s}^{-1}$
 $R^2 = 0.9791817338152677$



3 All Points

```

using LsqFit

# Implementing this curve fitting produced a succession of cryptic errors
# that I resolved by removing uncertainties and units, with some regret.

function dp(z, Ω, H)
    integral, err = quadgk(zz -> 1/sqrt(Ω*(1+zz)^3 + (1-Ω)), 0, z, rtol=1e-8)
    return ustrip(u"km/s", c_0)/H * (integral ± err)
end

function dl_model(z, p)
    r = dp.(z, p[1], p[2]) .* (1 .+ z)
    return Measurements.value.(r)
end

```

```

end

dl_in = ustrip(u"Mpc", Measurements.value.(data.dl))
weights = 1 ./ ustrip(u"Mpc", Measurements.uncertainty.(data.dl)).^2

fit = curve_fit(dl_model, data.z, dl_in, weights, [0.3, 70])

fit_err = stderror(fit)

# sum of squares error
fit_err2 = Measurements.value(sum(abs2, data.dl - dl_model(data.z, [ $\Omega_0$ ,
ustrip(H0)])*1u"Mpc"))
var = Measurements.value(sum(abs2, data.dl .- mean(data.dl)))
println("R^2 = $(1 - fit_err2/var)") # Coefficient of determination
dl_fit_err = sqrt(1/(length(data.z)-1) * fit_err2) # Std of dl from fit error

params = fit.param .± fit_err
println(" $\Omega_0$  = $(params[1])")
println("H0 = $(params[2])")
println("Sum of squares error: $(fit_err2)")

function E_integral(z::Real)::Number
    integral, err = quadgk(zz -> 1/E(zz), 0, z, rtol=1e-8)
    err = Measurements.value(Measurements.uncertainty(integral) +
Measurements.value(err) + Measurements.uncertainty(err))
    return Measurements.value(integral) ± err
end

z = 0:0.05:1.5

 $\Omega_0$  = params[1]
 $\Omega_\Lambda$  = 1 -  $\Omega_0$ 
H0 = params[2]*1u"km/s/Mpc"

dlz = c_0 / H0 * E_integral.(z) .* (1 .+ z)

scatter(data.z,
    data.dl,
    unitformat=latexify,
    label="Candles",
    legend=:topleft,
    xlabel=L"\mathrm{Redshift}",
    ylabel=L"\mathrm{Luminosity\\ distance}",
    markersize=3,
    markeralpha=0.75
)

plot!(z, Measurements.value.(dlz), linewidth=1, label="Fit")

plot!(
    z,
    Measurements.value.(dlz) .+ dl_fit_err,
    fillrange = Measurements.value.(dlz) .- dl_fit_err,
    fillcolor = :lightgray,
    fillalpha = 0.75,
    linecolor = nothing,
    primary = false, # no legend entry
    z_order = :back
)

```



```

plot!(
    z,
    Measurements.value.(dlz) + Measurements.uncertainty.(dlz),
    fillrange = Measurements.value.(dlz) - Measurements.uncertainty.(dlz),
    fillcolor = :orange,
    fillalpha = 0.75,
    linecolor = nothing,
    primary = false, # no legend entry
    z_order = :back
)

```

```
fe3 = fit_err2
```

```

R^2 = 0.8391220608855329
Ω0 = 0.0993 ± 0.0061
H0 = 71.02 ± 0.32
Sum of squares error: 5.002119360656467e8 Mpc^2
5.002119360656467e8 Mpc^2

```

This result is unconvincing. We'll also fit to the case with no dark energy, where $\Omega_0 = 1$.

```

using LsqFit
function dp(z, Ω, H)
    integral, err = quadgk(zz -> 1/sqrt(Ω*(1+z)^3 + (1-Ω)), 0, z, rtol=1e-8)
    return ustrip(u"km/s", c_0)/H * (integral ± err)
end

function dl_model(z, p)
    r = dp.(z, 1.0, p[1]) .* (1 .+ z)
    return Measurements.value.(r)
end

dl_in = ustrip(u"Mpc", Measurements.value.(data.dl))
weights = 1 ./ ustrip(u"Mpc", Measurements.uncertainty.(data.dl)).^2

fit = curve_fit(dl_model, data.z, dl_in, weights, [70.0])

fit_err = stderror(fit)
# sum of squares error
fit_err2 = Measurements.value(sum(abs2, data.dl - dl_model(data.z, fit.param)*1u"Mpc"))
var = Measurements.value(sum(abs2, data.dl .- mean(data.dl)))
println("R^2 = $(1 - fit_err2/var)") # Coefficient of determination
dl_fit_err = sqrt(1/(length(data.z)-1) * fit_err2) # Std of dl from fit error

params2 = fit.param .± fit_err
println("H0 = $(params2[1])")

println("Sum of squares error: $(fit_err2)")

function E_integral(z::Real)::Number
    integral, err = quadgk(zz -> 1/E(zz), 0, z, rtol=1e-8)
    err = Measurements.value(Measurements.uncertainty(integral) +
Measurements.value(err) + Measurements.uncertainty(err))
    return Measurements.value(integral) ± err
end

z = 0:0.05:1.5

Ω0 = 1

```

```

 $\Omega\Lambda = 1 - \Omega_0$ 
H0 = (params2[1])*1u"km/s/Mpc"

dlz = c_0 / H0 * E_integral.(z) .* (1 .+ z)

plot!(z, Measurements.value.(dlz), linewidth=1, label="\Omega_0 =  $\Omega_0$ ")

plot!(
    z,
    Measurements.value.(dlz) + Measurements.uncertainty.(dlz),
    fillrange = Measurements.value.(dlz) - Measurements.uncertainty.(dlz),
    fillcolor = :lightgreen,
    fillalpha = 0.75,
    linecolor = nothing,
    primary = false, # no legend entry
    z_order = :back
)

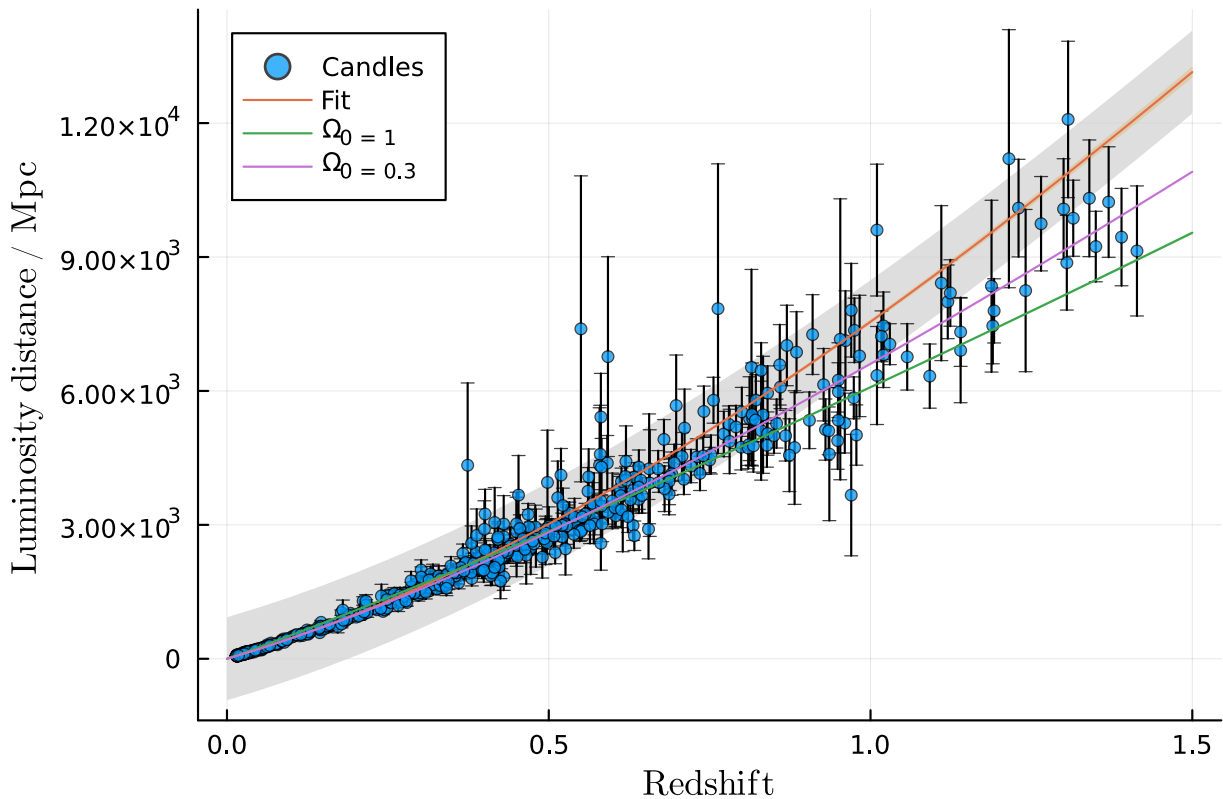
 $\Omega_0 = 0.3$ 
 $\Omega\Lambda = 1 - \Omega_0$ 
H0 = 70u"km/s/Mpc"

ss_err = sum(abs2, data.dl - dl.(data.z))
println("Sum of squares error ( $\Omega_0 = \Omega_0$ ):  $\Omega_0$ (ss_err)")

plot!(z, Measurements.value.(dl.(z)), linewidth=1, label="\Omega_0 =  $\Omega_0$ ")

R^2 = 0.6212502979126604
H0 = 57.73  $\pm$  0.19
Sum of squares error: 1.1776326997239494e9 Mpc^2
Sum of squares error ( $\Omega_0 = 0.3$ ): 1.54e8  $\pm$  4.8e7 Mpc^2

```



To the eye this does a good job of fitting the low redshift data but somewhat underestimates

the upper end. From the R^2 values I'd judge it a worse fit than the universe with dark energy, but I wouldn't say the dark energy fit is particularly good either. From comparing the sum of squares error we actually see that the $\Omega_0 = 0.3$ line, manually inserted by me, performs better than either fitted line, or at least it better satisfies the very condition least squares is optimising for. It would appear that the curve fitting is not very successful.

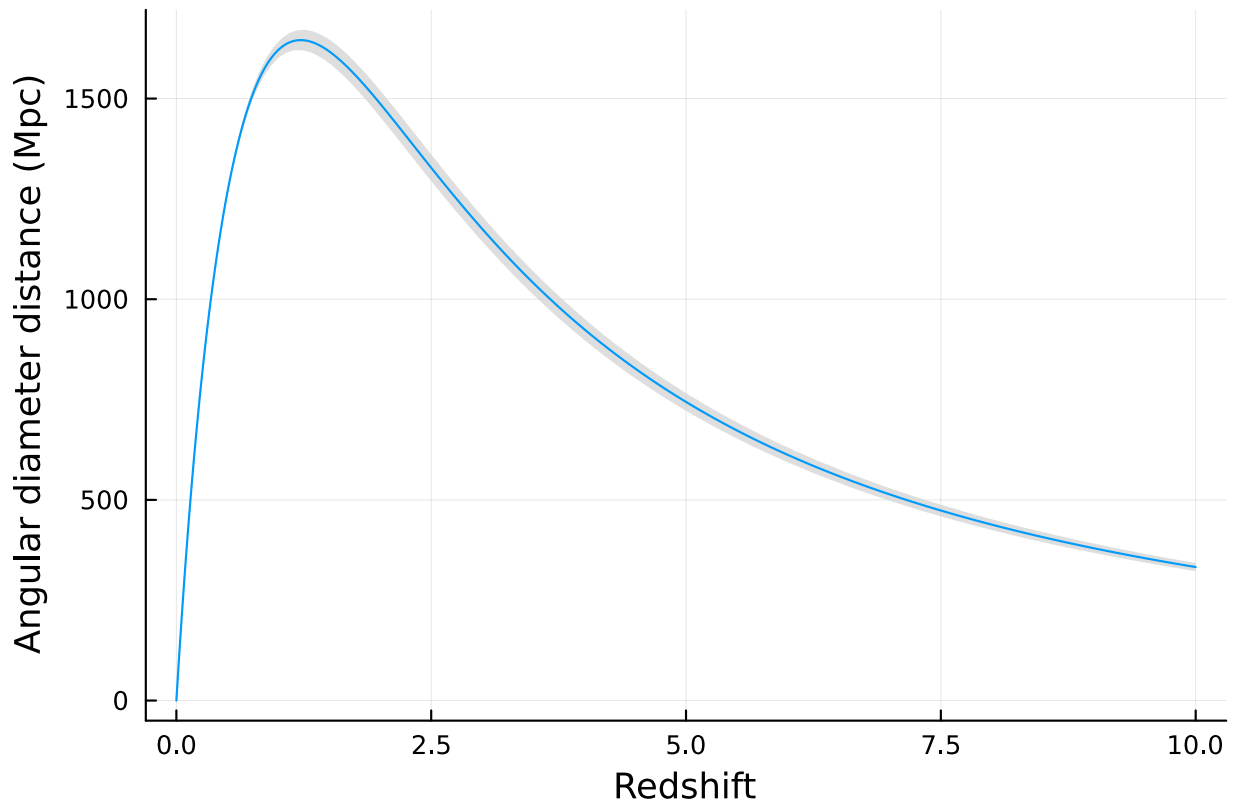
3.1 Angular Diameter Distance

```
function da(z, p)::Vector{<:Unitful.Length}
    r = dp.(z, p[1], p[2]) ./ (1 .+ z) * 1u"Mpc"
    return Measurements.value.(r)
end

z = 0:0.01:10
daz = da(z, params) # params is [ $\Omega_0$   $H_0$ ]

plot(z, Measurements.value.(daz), xlabel="Redshift", ylabel="Angular diameter distance",
legend=false)

plot!(
    z,
    Measurements.value.(daz) + Measurements.uncertainty.(daz),
    fillrange = Measurements.value.(daz) - Measurements.uncertainty.(daz),
    fillcolor = :lightgray,
    fillalpha = 0.75,
    linecolor = nothing,
    primary = false, # no legend entry
    z_order = :back
)
```



The smallest angular size in our dataset is

```
# If I use uconvert then the arcsecond symbol breaks the silly Julia -> latex converter  
I've  
# been trying in this report. This still converts, but then strips the units.  
ustrip(u"arcsecond", minimum(30u"kpc" ./ (data.dl ./ (1 .+ data.z).^2)) * 1u"rad")
```

2.01 ± 0.93

The theoretical minimum using our calculated parameters

```
# I could probably solve this analytically but this is faster and easier  
ustrip(u"arcsecond", minimum(30u"kpc" ./ daz) * 1u"rad")
```

3.76 ± 0.058

I believe the discrepancy is because there is a supernova at $z=0.55$ that is significantly above the and thus throwing our results off.

Actually observing the theoretically minimum is unlikely because in sufficiently high redshift observations galaxies are usually too young to have supernovae.