Luminosity Distance - PHYS417 Project 2

Ryan Cox

2022-10-09

1 The Friedmann Equation

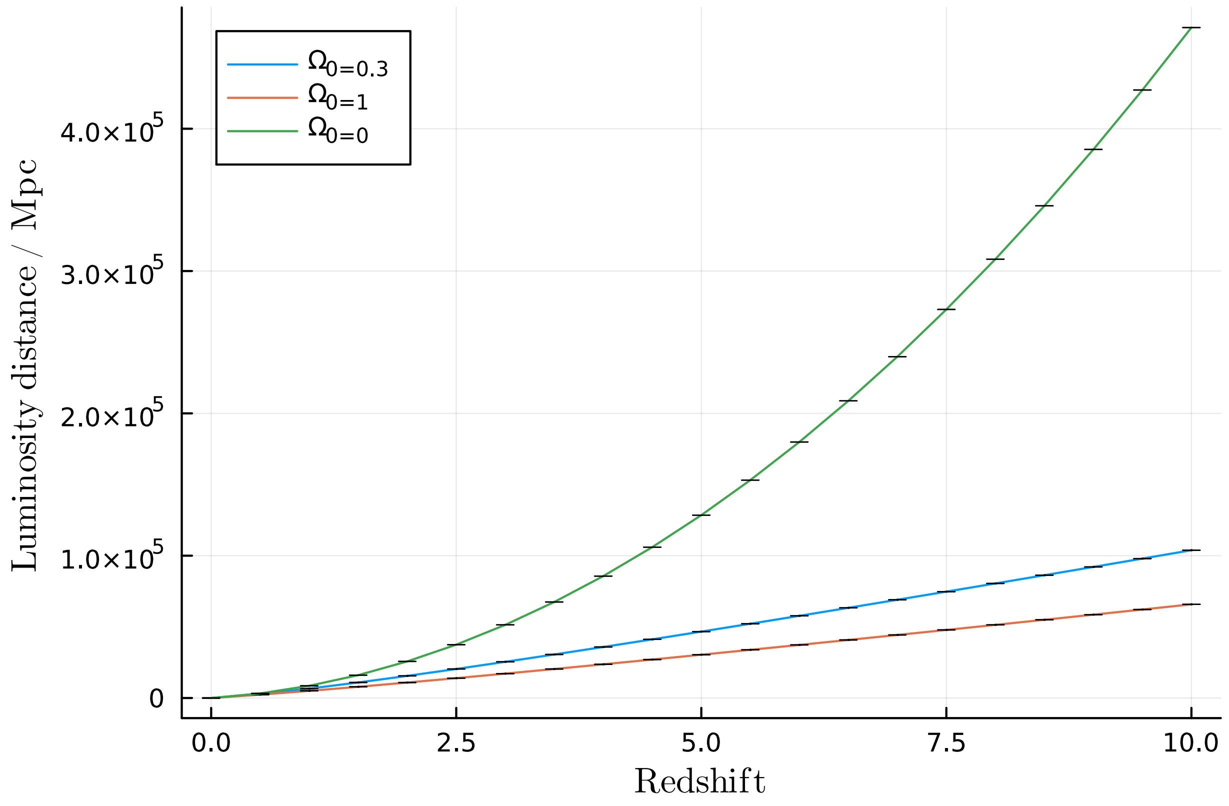
Show it can be written Predict behaviour (take limits)

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.

```julia
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(zz -> 1/E(zz), 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, dl.(z), label="\\Omega_0=$(Ω0)") # plot!() updates last plot

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



Candle time.

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

$$v_r \approx zc$$

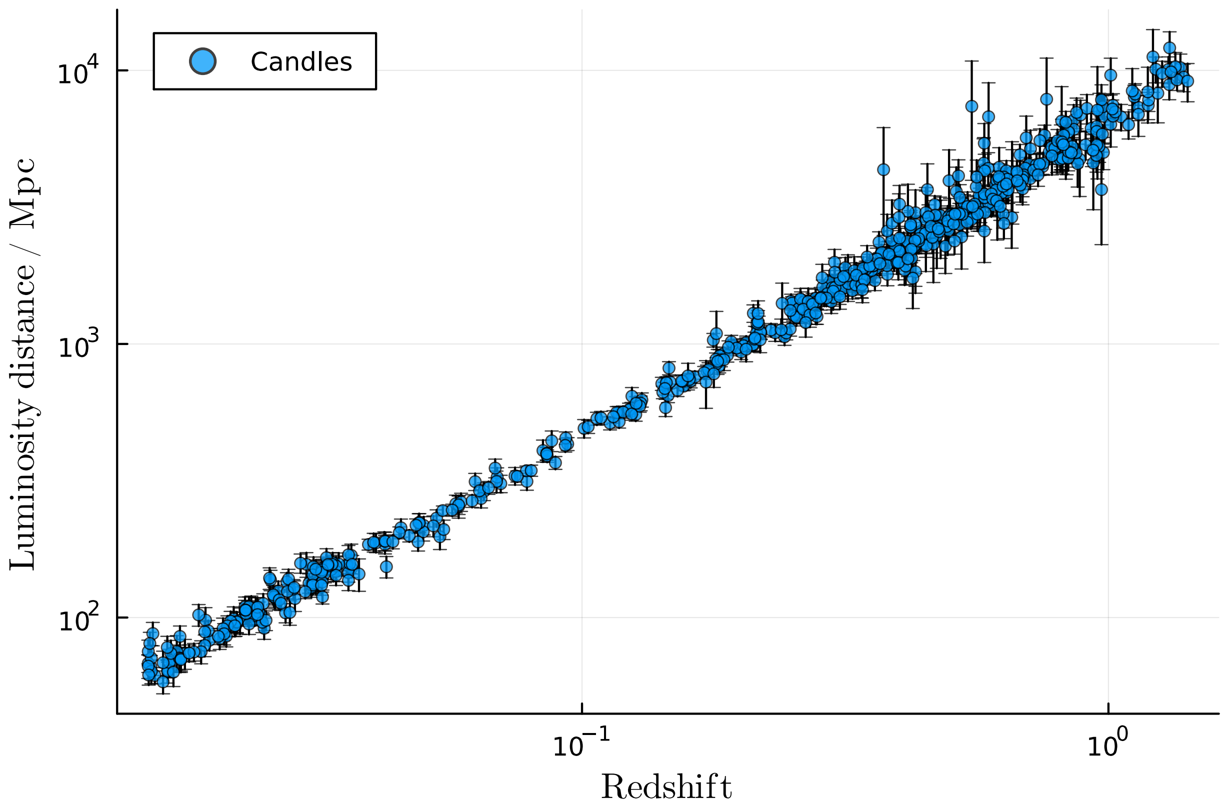
so we can conclude

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

Combining this with our luminosity distance equation and gives

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

```julia
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="\\mathrm{Luminosity\\ distance}",
        markersize=3,
        markeralpha=0.75,
        xscale=:log10,
        yscale=:log10
)
```

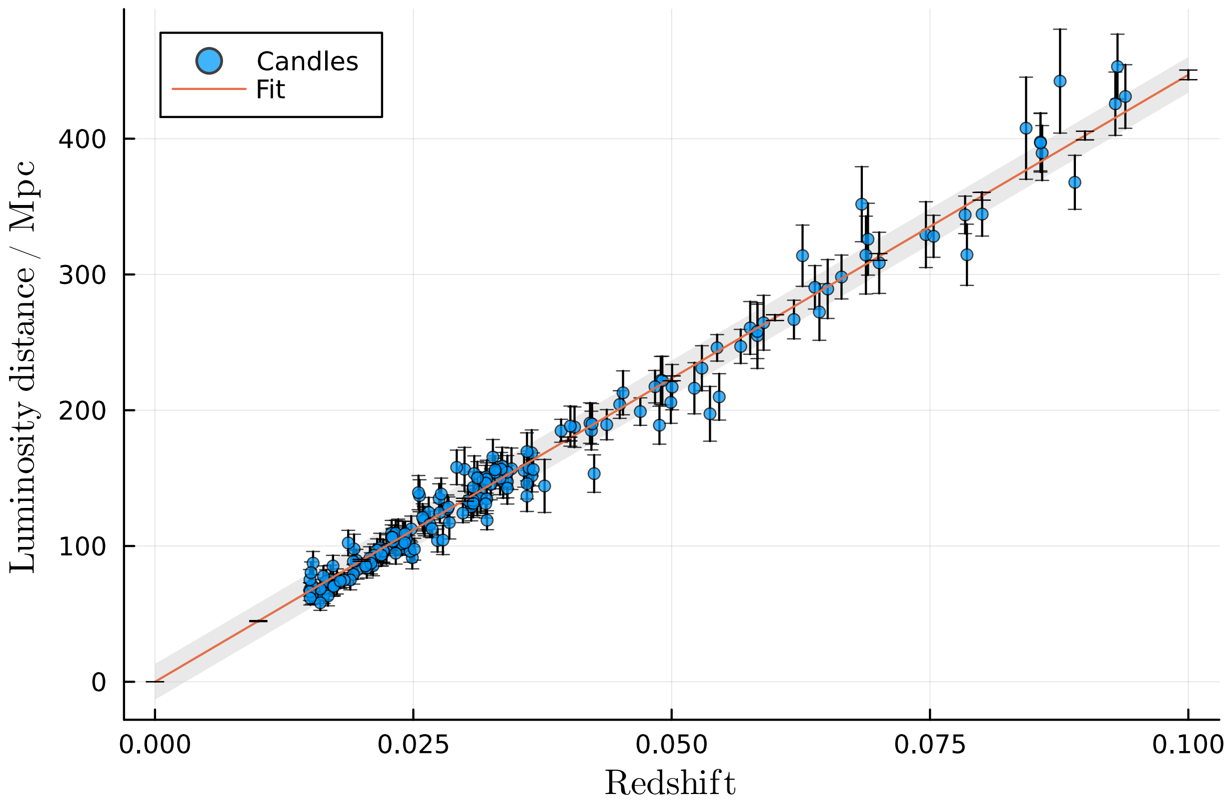


We can perform basic least squares in Julia with remarkable ease. Remarkably it handles uncertaintities. However it doesn't like units so we'll have to strip them (carefully!).

```
using Statistics: mean, cor

lowz_data = @view data[data.z .< 0.1, [:z, :dl]] # get z and DM for low redshift
G = lowz_data.z * ustrip(u"km/s", c_0) # construct kernel. dl = G*(1/H0)
reciprocal_H0 = G \ ustrip.(u"Mpc", lowz_data.dl) # This performs least squares
lowz_H0 = 1u"km/s/Mpc" / reciprocal_H0

println("Estimated H0 = $(lowz_H0)")

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

dl_fit_err = sqrt(1/(nrow(lowz_data)-1) * fit_err) # Std of dl from fit error

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
plot!(z, z*c_0/lowz_H0, linewidth=1, label="Fit")

dlz = z*c_0/Measurements.value(lowz_H0)
display(dl_fit_err)

plot!(
    z,
    dlz .+ dl_fit_err,
    fillrange = dlz .- dl_fit_err,
    fillcolor = :lightgray,
    fillalpha = 0.5,
    linecolor = nothing,
    primary = false, # no legend entry
    z_order = :back
)
```

```
Estimated H0 = 67.07 ± 0.53 km Mpc^-1 s^-1
R^2 = 0.9791817338152677
13.166821825313074 Mpc
```


● Candles
Fit
Luminosity distance / Mpc
Redshift