

- *# Import some helpful packages for loading and plotting data*
- using CSV , Dates , DataFrames , Gadfly , GLM , Statistics

	Time	Replicate	OD
1	13:23:00	1	0.038
2	13:23:00	2	0.031
3	13:23:00	3	0.042
4	13:23:00	4	0.04
5	13:43:00	1	0.044
6	13:43:00	2	0.034
7	13:43:00	3	0.05
8	13:43:00	4	0.042
9	14:03:00	1	0.065
10	14:03:00	2	0.048
more			
40	16:23:00	4	1.088

- begin
- *# Load CSV into a DataFrame*
- csvfile = "Growth Curve Data.csv"
- df = DataFrame(CSV.File(csvfile))
- end

	Time	Replicate	OD
1	0	1	0.038
2	0	2	0.031
3	0	3	0.042
4	0	4	0.04
5	20	1	0.044
6	20	2	0.034
7	20	3	0.05
8	20	4	0.042
9	40	1	0.065
10	40	2	0.048
	more		
40	180	4	1.088

```

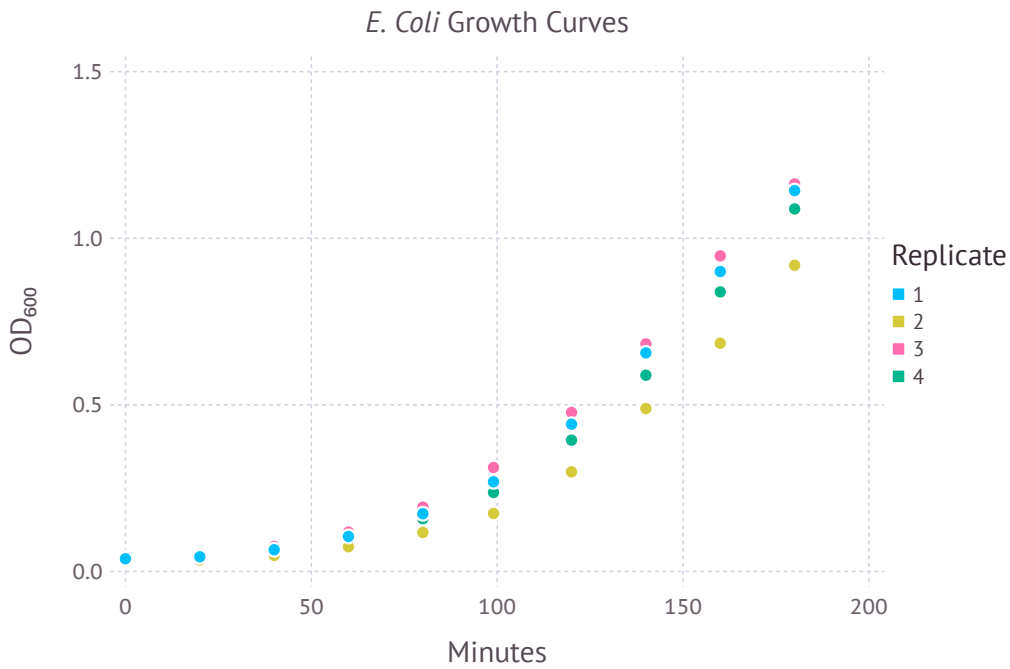
• begin
•   # Normalise times and convert to minutes
•   start = df[1, :Time]
•   pdf = transform(df, :Time => ByRow(t -> Dates.value(Minute(t - start))) => :Time)
• end

```

Growth Curve Data

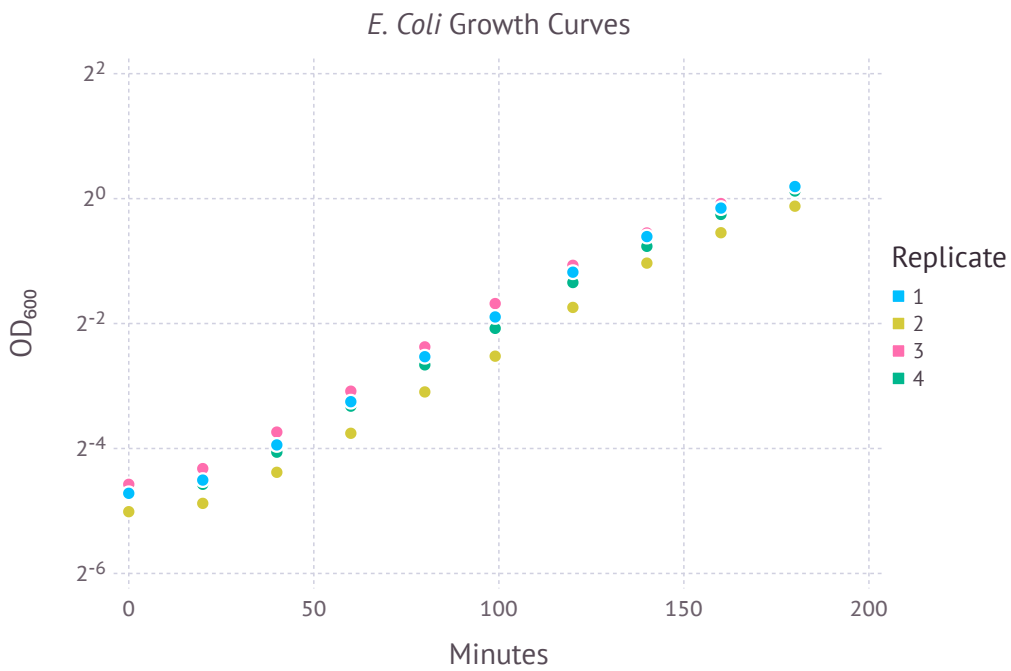
Given that OD is proportional to cell count (when properly diluted so that readings don't exceed 0.6), it can be used to track the growth of cells.

On a linear scale, this growth curve is an exponential, but be later made linear by applying a logarithmic transformation.



- *# Construct a line-scatter plot, grouping by biological replicate*
- `plot(pdf, x=:Time, y=:OD, color=:Replicate, Scale.color_discrete_hue,`
- `Guide.xlabel("Minutes"), Guide.ylabel("OD600"),`
- `Guide.title("<i>E. Coli</i> Growth Curves"))`

A log transformation reveals that the region between 60 and 120 minutes can be safely said to be linear



- *# Replot, but on a log-scale so that we can pick out the exponential growth region*
- `plot(pdf, x=:Time, y=:OD, color=:Replicate,`
- `Scale.color_discrete_hue, Scale.y_log2,`
- `Guide.xlabel("Minutes"), Guide.ylabel("OD600"),`
- `Guide.title("<i>E. Coli</i> Growth Curves"))`

logdf =

	Time	Replicate	OD
1	60	1	0.105
2	60	2	0.074
3	60	3	0.118
4	60	4	0.1
5	80	1	0.173
6	80	2	0.117
7	80	3	0.193
8	80	4	0.158
9	99	1	0.269
10	99	2	0.174
more			
16	120	4	0.394

- *# Trim the data to take a closer look at log-phase*
- `logdf = filter(:Time => t -> 60 <= t <= 120, pdf)`

	Time	Replicate	OD
1	60	1	-3.25154
2	60	2	-3.75633
3	60	3	-3.08314
4	60	4	-3.32193
5	80	1	-2.53116
6	80	2	-3.09542
7	80	3	-2.37333
8	80	4	-2.662
9	99	1	-1.89432
10	99	2	-2.52284
more			
16	120	4	-1.34373

- *# Log-transform the OD data*
- `transform!(logdf, :OD => ByRow(log2) => :OD)`

```
ols =
StatsModels.TableRegressionModel{LinearModel{GLM.LmResp{Vector{Float64}}}, GLM.DensePredCho
```

OD ~ 1 + Time

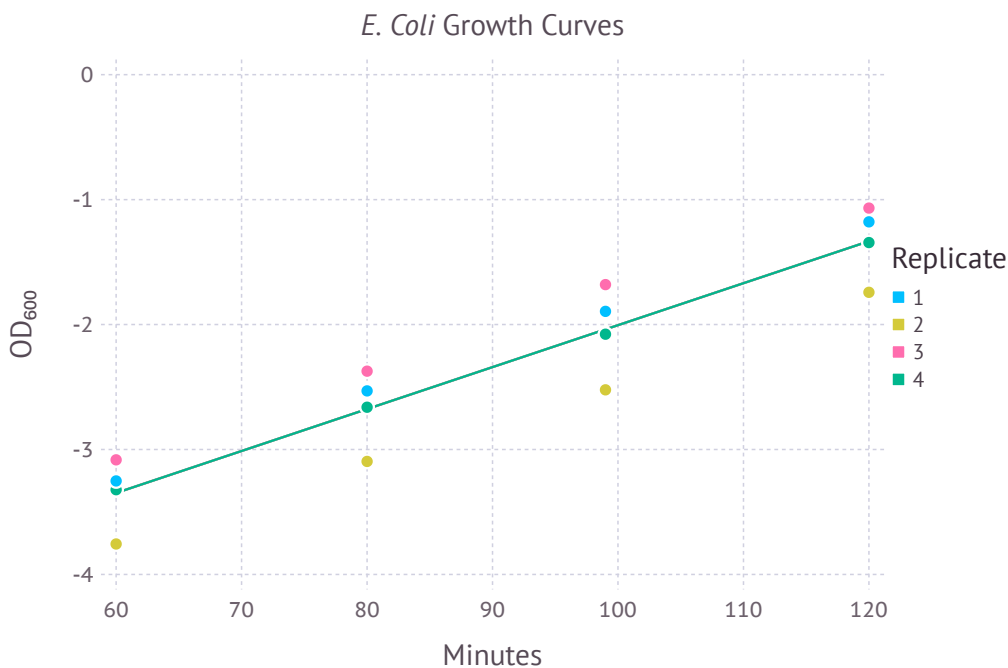
Coefficients:

	Coef.	Std. Error	t	Pr(> t)	Lower 95%	Upper 95%
(Intercept)	-5.36313	0.301881	-17.77	<1e-10	-6.0106	-4.71566
Time	0.0335859	0.00326472	10.29	<1e-07	0.0265837	0.040588

- *# Perform an ordinary least-squares regression for a linear model*
- `ols = lm(@formula(OD ~ Time), logdf)`

```
[-3.34798, -3.34798, -3.34798, -3.34798, -2.67626, -2.67626, -2.67626, -2.67626, -2.03813,
```

- *# Insert a new column into our dataframe representing the model predictions*
- `logdf[!,:Model] = predict(ols)`



- *# And then plot it on a log-scale*
- `plot(logdf, x=:Time, y=:OD, color=:Replicate,`
- `Scale.color_discrete_hue, Geom.point,`
- `Guide.xlabel("Minutes"), Guide.ylabel("OD600"),`
- `Guide.title("<i>E. Coli</i> Growth Curves"),`
- *# With the line-of-best-fit superimposed*
- `layer(x=:Time, y=:Model, Geom.line))`

Calculating Doubling-Time From Our Model

We can start with a fundamental equation that models the growth of microbes undergoing binary fission:

$$N = N_0 2^{\frac{t}{g}}$$

Where N is the current number of cells, N_0 is the initial number of cells, t is time, and g is generation or doubling-time. We want to rearrange this equation to fit the model `OD ~ Time` after calculating the \log_2 of all ODs.

Let's start by applying the \log_2 to both sides of the equation:

$$\log_2 N = \log_2 N_0 + \frac{t}{g}$$

Ignoring the intercept and separating terms, we get an expression that matches our model:

$$\log_2 N = \frac{1}{g}t$$

Therefore we can conclude that g is equal to the reciprocal of our regression gradient.

The doubling time was ~29.8 minutes

```
• begin
•   # Calculate doubling-time
•   g = 1/coef(ols)[2]
•   # Format it into a nice string
•   md"The doubling time was ~$(round(g, sigdigits=3)) minutes"
• end
```