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

	Time	Replicate	OD
1	11:17:00	1	0.025
2	11:17:00	2	0.025
3	11:17:00	3	0.026
4	11:37:00	1	0.024
5	11:37:00	2	0.021
6	11:37:00	3	0.024
7	11:57:00	1	0.021
8	11:57:00	2	0.018
9	11:57:00	3	0.029
10	12:17:00	1	0.017
more			
39	15:17:00	3	3.46

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

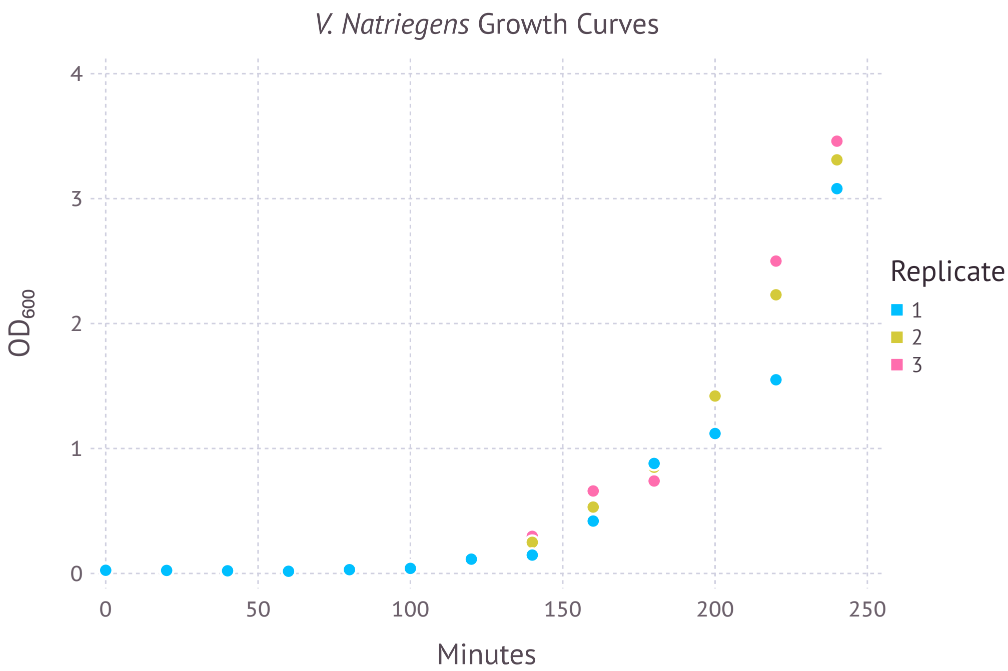
	Time	Replicate	OD
1	0	1	0.025
2	0	2	0.025
3	0	3	0.026
4	20	1	0.024
5	20	2	0.021
6	20	3	0.024
7	40	1	0.021
8	40	2	0.018
9	40	3	0.029
10	60	1	0.017
more			
39	240	3	3.46

```
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

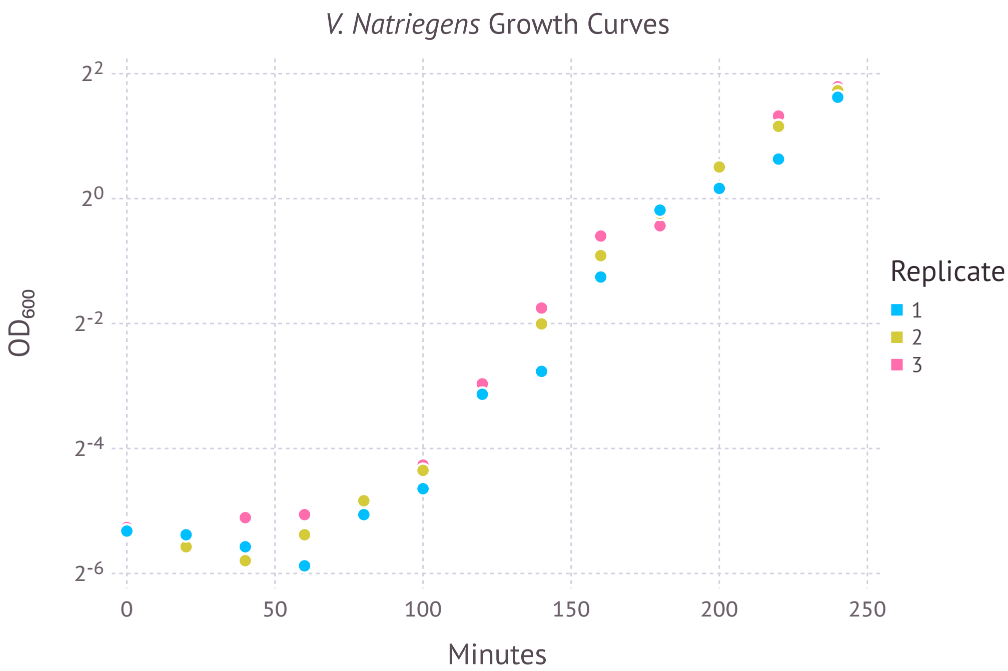
Given that OD is proportional to cell count (when properly diluted so that readings don't exceed 0.6), it can be used to track the growth of cells.

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>V. Natriegens</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>V. Natriegens</i> Growth Curves"))
```

logdf =

	Time	Replicate	OD
1	100	1	0.04
2	100	2	0.049
3	100	3	0.052
4	120	1	0.114
5	120	2	0.112
6	120	3	0.128
7	140	1	0.147
8	140	2	0.249
9	140	3	0.297
10	160	1	0.419
11	160	2	0.531
12	160	3	0.66

```
# Trim the data to take a closer look at log-phase
logdf = filter(:Time => t -> 100 <= t <= 160, pdf)
```

	Time	Replicate	OD
1	100	1	-4.64386
2	100	2	-4.35107
3	100	3	-4.26534
4	120	1	-3.13289
5	120	2	-3.15843
6	120	3	-2.96578
7	140	1	-2.76611
8	140	2	-2.00578
9	140	3	-1.75147
10	160	1	-1.25498
11	160	2	-0.913216
12	160	3	-0.599462

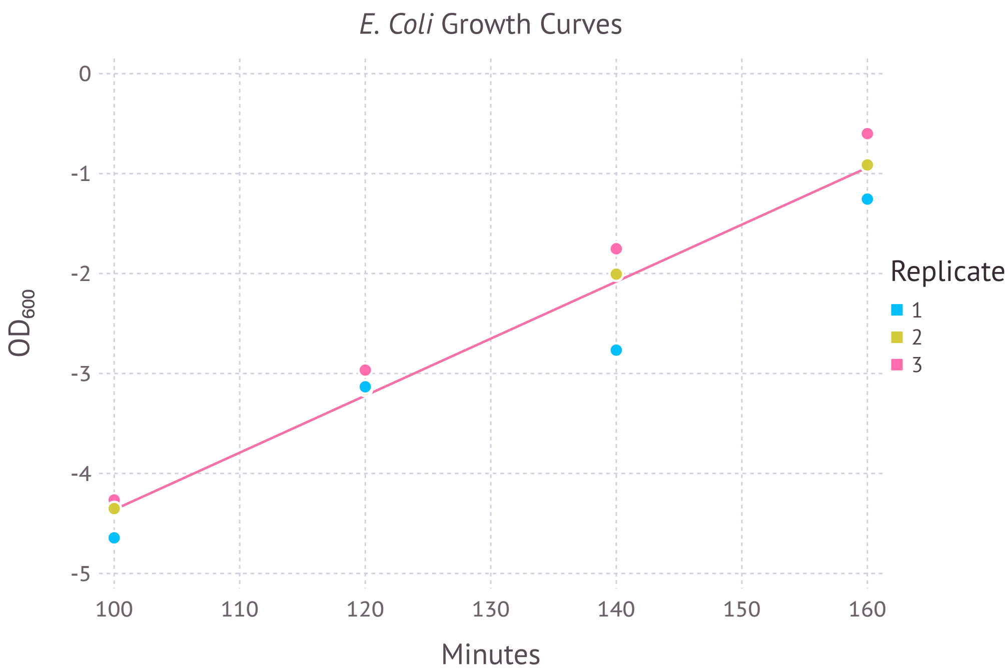
```
# Log-transform the OD data
transform!(logdf, :OD => ByRow(log2) => :OD)
```

```
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)  -10.0632    0.5292      -19.02    <1e-08  -11.2423    -8.88408
Time           0.0570193 0.00401185   14.21    <1e-07    0.0480804  0.0659583
──────────────────────────────────────────────────────────────────────────────
```

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

```
[-4.36128, -4.36128, -4.36128, -3.22089, -3.22089, -3.22089, -2.08051, -2.08051, -2.08051,
```

```
# 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 superimpoosed
     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 ~17.5 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
```