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

	Time	Replicate	OD
1	11:06:00	"H"	0.013
2	11:06:00	"S"	0.027
3	11:11:00	"1"	0.031
4	11:11:00	"2"	0.031
5	11:16:00	"3"	0.034
6	11:16:00	"4"	0.032
7	11:21:00	"5"	0.032
8	11:21:00	"6"	0.031
9	11:28:00	"H"	0.009
10	11:28:00	"S"	0.039
	more		
96	15:04:00	"6"	5.1

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

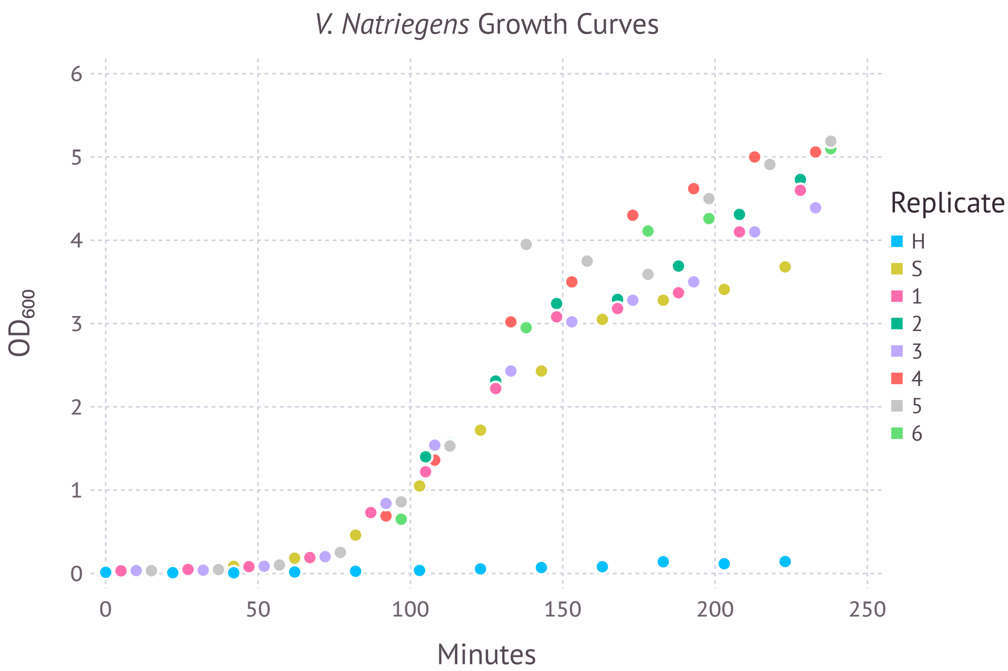
	Time	Replicate	OD
1	0	"H"	0.013
2	0	"S"	0.027
3	5	"1"	0.031
4	5	"2"	0.031
5	10	"3"	0.034
6	10	"4"	0.032
7	15	"5"	0.032
8	15	"6"	0.031
9	22	"H"	0.009
10	22	"S"	0.039
more			
96	238	"6"	5.1

```
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

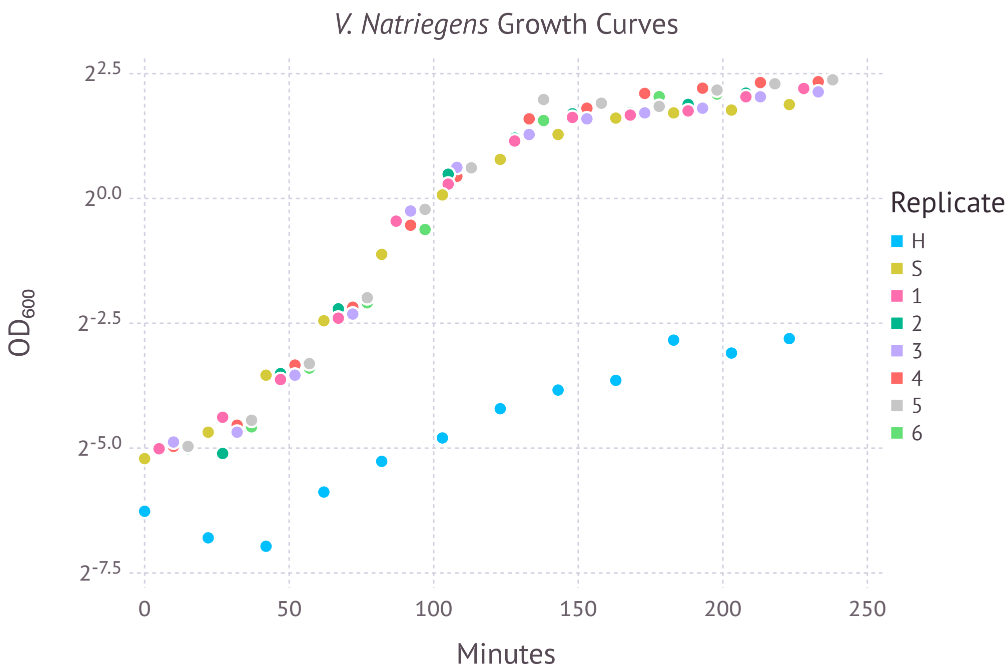
Given that OD is proportional to cell count (when properly diluted so that readings don't exceed 0.6), it can be used to track the growth of cells.

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

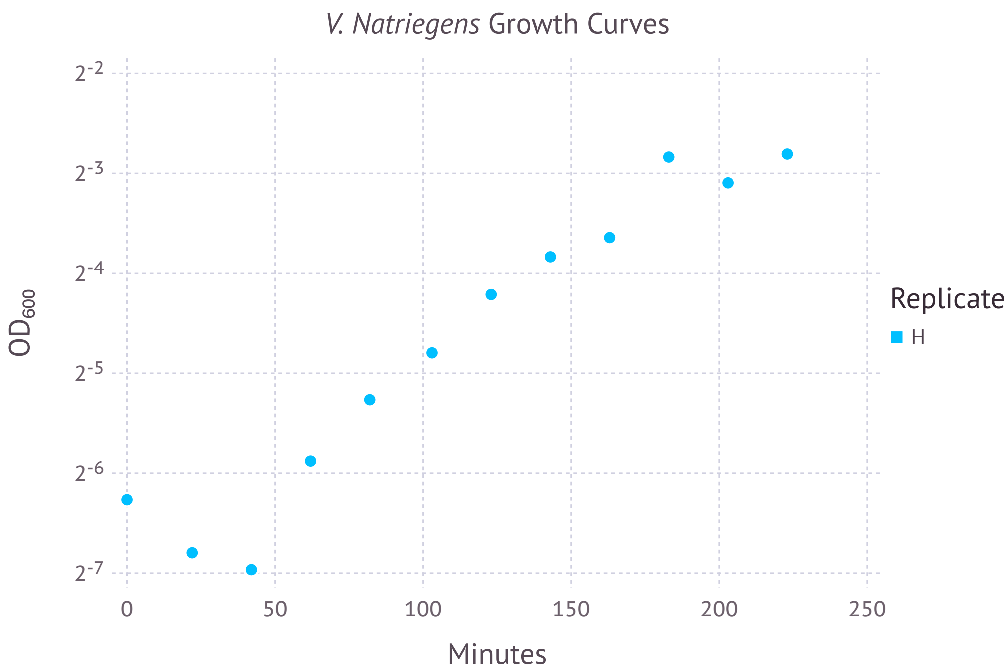



```
# Replot, but on a log-scale so that we can pick out the exponential growth region
plot(pdf, x=:Time, y=:OD, color=:Replicate,
     Scale.color_discrete_hue, Scale.y_log2,
     Guide.xlabel("Minutes"), Guide.ylabel("OD₆₀₀"),
     Guide.title("<i>V. Natriegens</i> Growth Curves"))
```

hlogdf =

	Time	Replicate	OD
1	0	"H"	0.013
2	22	"H"	0.009
3	42	"H"	0.008
4	62	"H"	0.017
5	82	"H"	0.026
6	103	"H"	0.036
7	123	"H"	0.054
8	143	"H"	0.07
9	163	"H"	0.08
10	183	"H"	0.14
11	203	"H"	0.117
12	223	"H"	0.143

```
# Isolate the H media
hlogdf = filter(:Replicate => r -> r == "H", pdf)
```




```
# Replot, but the isolated values so that we can pick out the exponential growth
region
plot(hlogdf, 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"))
```

thlogdf =

	Time	Replicate	OD
1	62	"H"	0.017
2	82	"H"	0.026
3	103	"H"	0.036
4	123	"H"	0.054

```
# Trim the data to take a closer look at log-phase
thlogdf = filter(:Time => t -> 60 <= t <= 123, hlogdf)
```

	Time	Replicate	OD
1	62	"H"	-5.87832
2	82	"H"	-5.26534
3	103	"H"	-4.79586
4	123	"H"	-4.2109

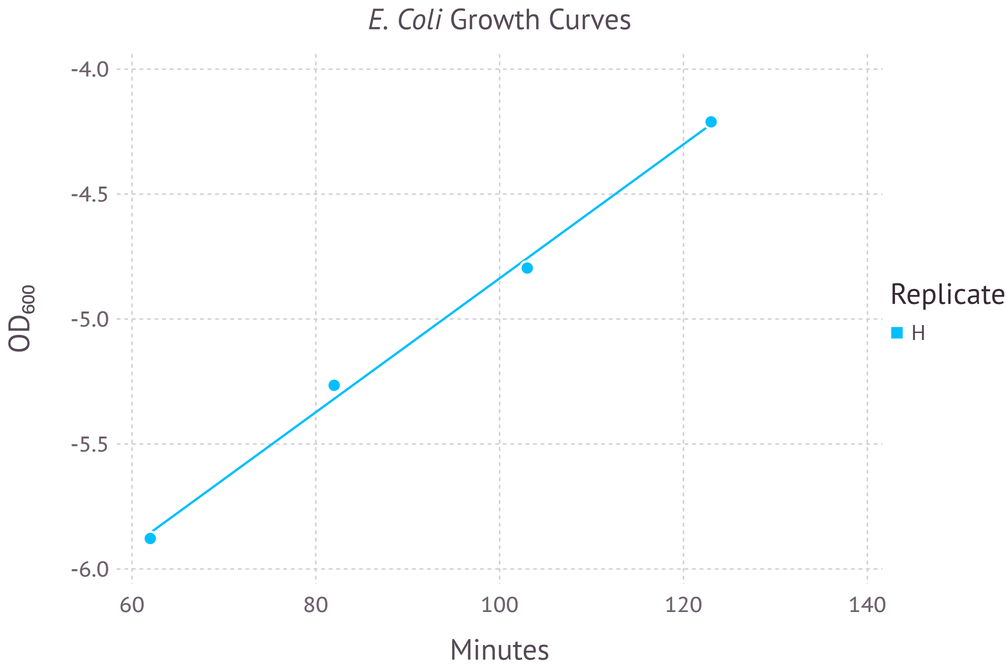
```julia
# Log-transform the OD data
transform!(thlogdf, :OD => ByRow(log2) => :OD)
```

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

OD ~ 1 + Time

Coefficients:
──────────────────────────────────────────────────────────────────────────────
                  Coef.  Std. Error       t  Pr(>|t|)  Lower 95%   Upper 95%
──────────────────────────────────────────────────────────────────────────────
(Intercept)  -7.51728    0.105276    -71.41    0.0002  -7.97025    -7.06432
Time          0.0268073  0.00110502   24.26    0.0017   0.0220528   0.0315618
──────────────────────────────────────────────────────────────────────────────
```

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

```
[-5.85523, -5.31908, -4.75613, -4.21998]
```

```julia
# Insert a new column into our dataframe representing the model predictions
thlogdf[!,:Model] = predict(hols)
```




```
# And then plot it on a log-scale
plot(thlogdf, 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.

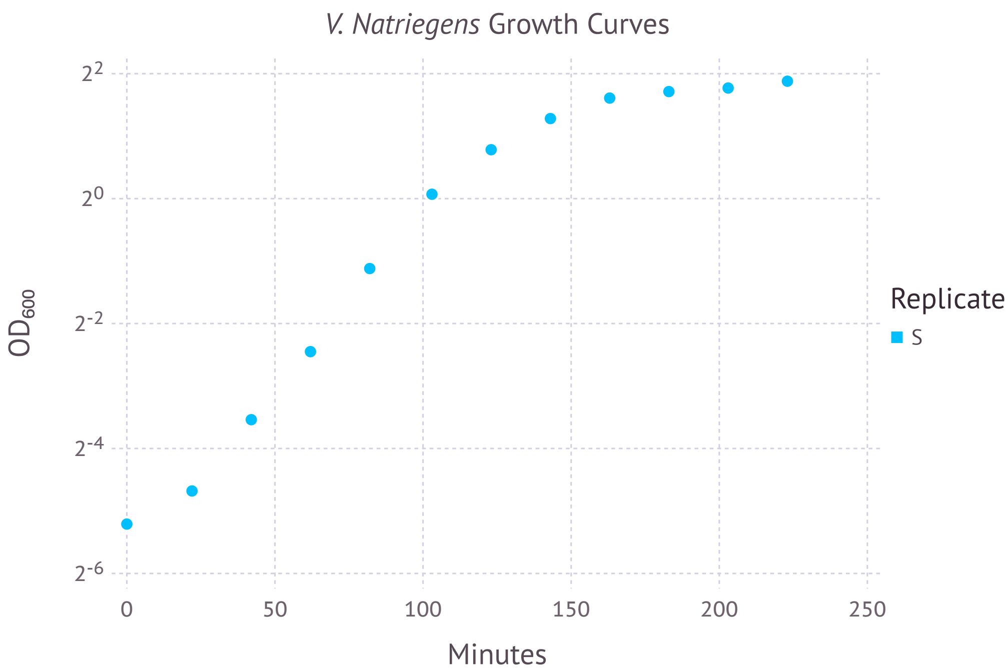
The doubling time for H was ~37.3 minutes, with a growth rate of 0.0268

```julia
begin
    # Calculate doubling-time
    g = 1/coef(hols)[2]
    # Format it into a nice string
    md"The doubling time for H was ~$(round(g, sigdigits=3)) minutes, with a growth rate of $(round(coef(hols)[2], sigdigits=3))"
end
```

slogdf =

	Time	Replicate	OD
1	0	"S"	0.027
2	22	"S"	0.039
3	42	"S"	0.086
4	62	"S"	0.183
5	82	"S"	0.46
6	103	"S"	1.05
7	123	"S"	1.72
8	143	"S"	2.43
9	163	"S"	3.05
10	183	"S"	3.28
11	203	"S"	3.41
12	223	"S"	3.68

```
# Isolate the S media
slogdf = filter(:Replicate => r -> r == "S", pdf)
```




```julia
# Replot, but the isolated values so that we can pick out the exponential growth region
plot(slogdf, 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"))
```

tslogdf =

	Time	Replicate	OD
1	22	"S"	0.039
2	42	"S"	0.086
3	62	"S"	0.183
4	82	"S"	0.46

```julia
# Trim the data to take a closer look at log-phase
tslogdf = filter(:Time => t -> 20 <= t <= 82, slogdf)
```

	Time	Replicate	OD
1	22	"S"	-4.68038
2	42	"S"	-3.53952
3	62	"S"	-2.45008
4	82	"S"	-1.12029

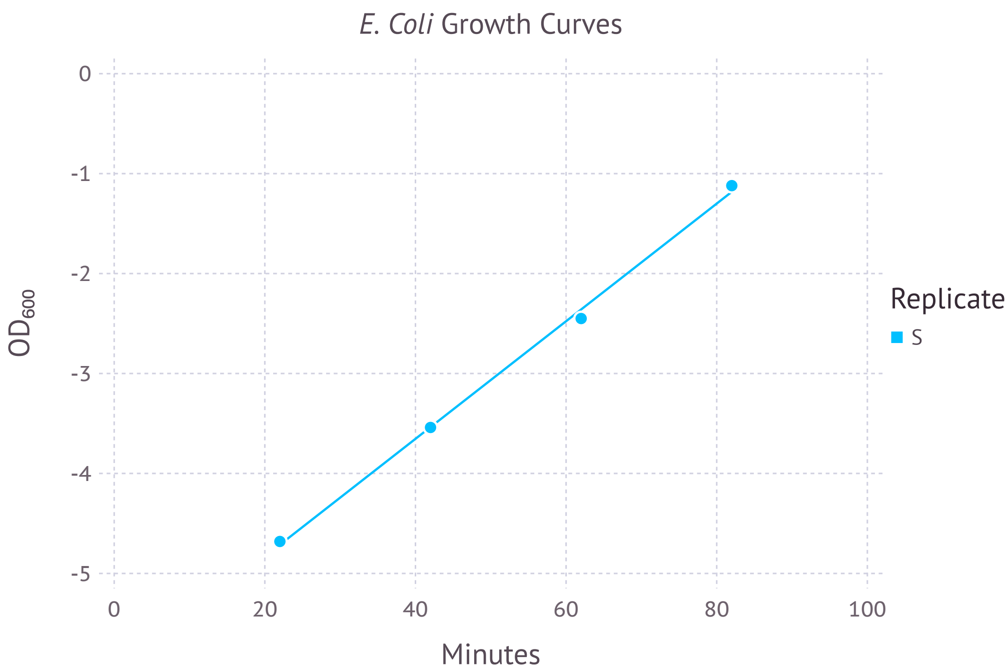
```
# Log-transform the OD data
transform!(tslogdf, :OD => ByRow(log2) => :OD)
```

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

OD ~ 1 + Time

Coefficients:
────────────────────────────────────────────────────────────────────────────
                  Coef.  Std. Error       t  Pr(>|t|)   Lower 95%  Upper 95%
────────────────────────────────────────────────────────────────────────────
(Intercept)  -6.00769    0.102749    -58.47    0.0003  -6.44979    -5.5656
Time          0.0588485  0.00181523   32.42    0.0010   0.0510382   0.0666588
────────────────────────────────────────────────────────────────────────────
```

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

```
[-4.71302, -3.53605, -2.35909, -1.18212]
```

```
# Insert a new column into our dataframe representing the model predictions
tslogdf[!,:Model] = predict(sols)
```




```
# And then plot it on a log-scale
plot(tslogdf, 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))
```

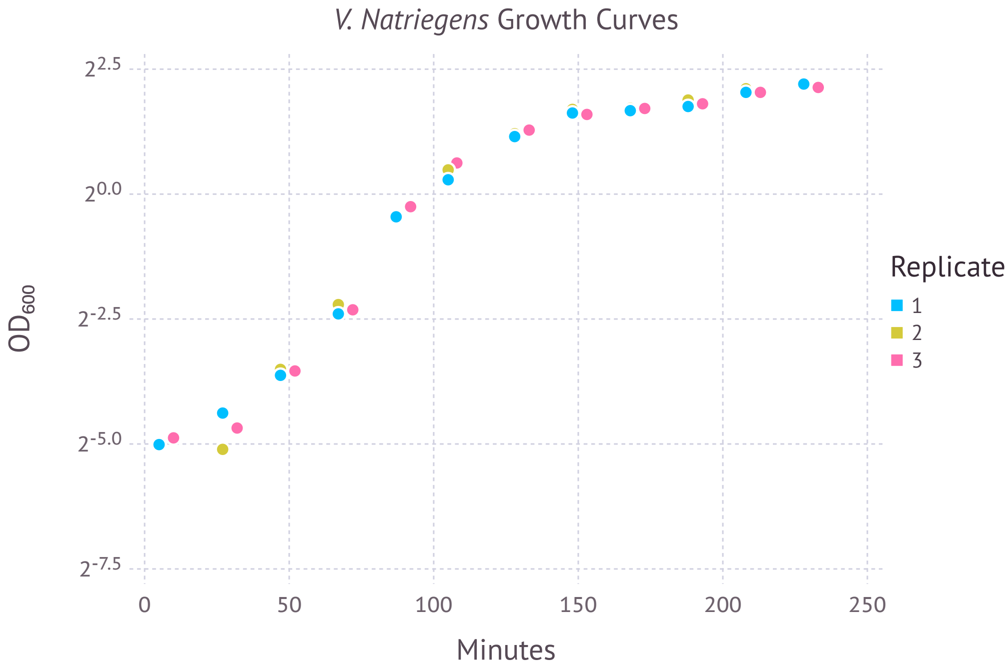
The doubling time for S was ~17.0 minutes, with a growth rate of 0.0588

```
begin
    # Calculate doubling-time
    sg = 1/coef(sols)[2]
    # Format it into a nice string
    md"The doubling time for S was ~$(round(sg, sigdigits=3)) minutes, with a growth rate of $(round(coef(sols)[2], sigdigits=3))"
end
```

nglogdf =

	Time	Replicate	OD
1	5	"1"	0.031
2	5	"2"	0.031
3	10	"3"	0.034
4	27	"1"	0.048
5	27	"2"	0.029
6	32	"3"	0.039
7	47	"1"	0.081
8	47	"2"	0.088
9	52	"3"	0.086
10	67	"1"	0.19
	more		
36	233	"3"	4.39

```
# Isolate the no glucose media
nglogdf = filter(:Replicate => r -> r in ["1", "2", "3"], pdf)
```




```julia
# Replot, but the isolated values so that we can pick out the exponential growth region
plot(nglogdf, 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"))
```

tnglogdf =

	Time	Replicate	OD
1	47	"1"	0.081
2	47	"2"	0.088
3	52	"3"	0.086
4	67	"1"	0.19
5	67	"2"	0.216
6	72	"3"	0.201
7	87	"1"	0.73
8	87	"2"	0.71
9	92	"3"	0.84

```julia
# Trim the data to take a closer look at log-phase
tnglogdf = filter(:Time => t -> 47 <= t <= 92, nglogdf)
```

	Time	Replicate	OD
1	47	"1"	-3.62593
2	47	"2"	-3.50635
3	52	"3"	-3.53952
4	67	"1"	-2.39593
5	67	"2"	-2.2109
6	72	"3"	-2.31473
7	87	"1"	-0.454032
8	87	"2"	-0.494109
9	92	"3"	-0.251539

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

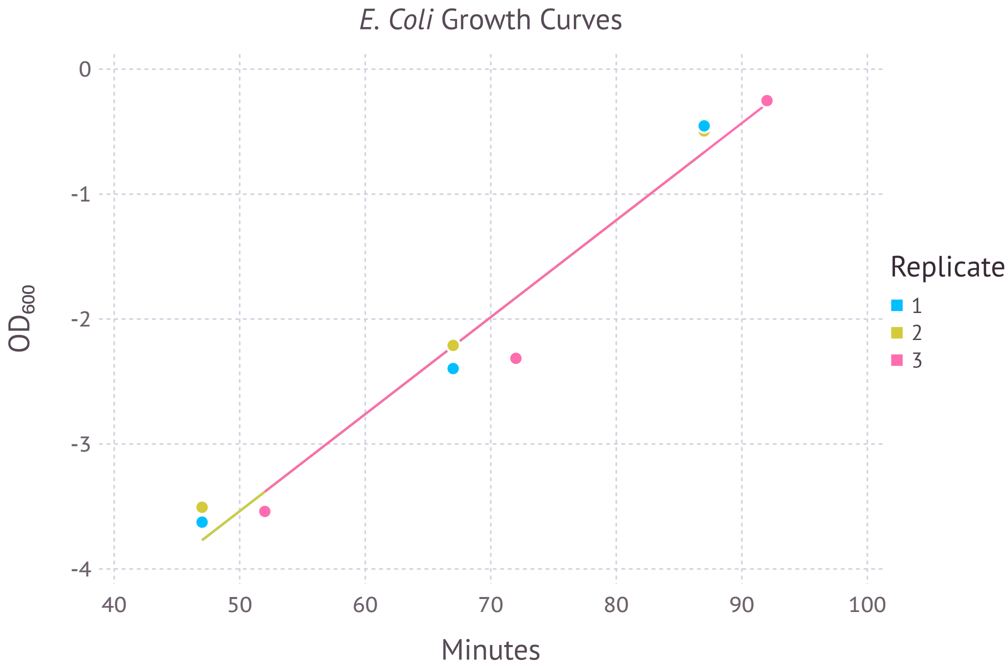
```
ngols =
StatsModels.TableRegressionModel{LinearModel{GLM.LmResp{Vector{Float64}}, GLM.DensePredCho

OD ~ 1 + Time

Coefficients:
```

	Coef.	Std. Error	t	Pr(>|t|)	Lower 95%	Upper 95%
(Intercept)	-7.41988	0.364521	-20.36	<1e-06	-8.28184	-6.55793
Time	0.0776471	0.00516165	15.04	<1e-05	0.0654417	0.0898524

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

```
[-3.77047, -3.77047, -3.38223, -2.21753, -2.21753, -1.82929, -0.664586, -0.664586, -0.2763
```

```
# Insert a new column into our dataframe representing the model predictions
tnglogdf[!,:Model] = predict(ngols)
```




```
# And then plot it on a log-scale
plot(tnglogdf, 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))
```

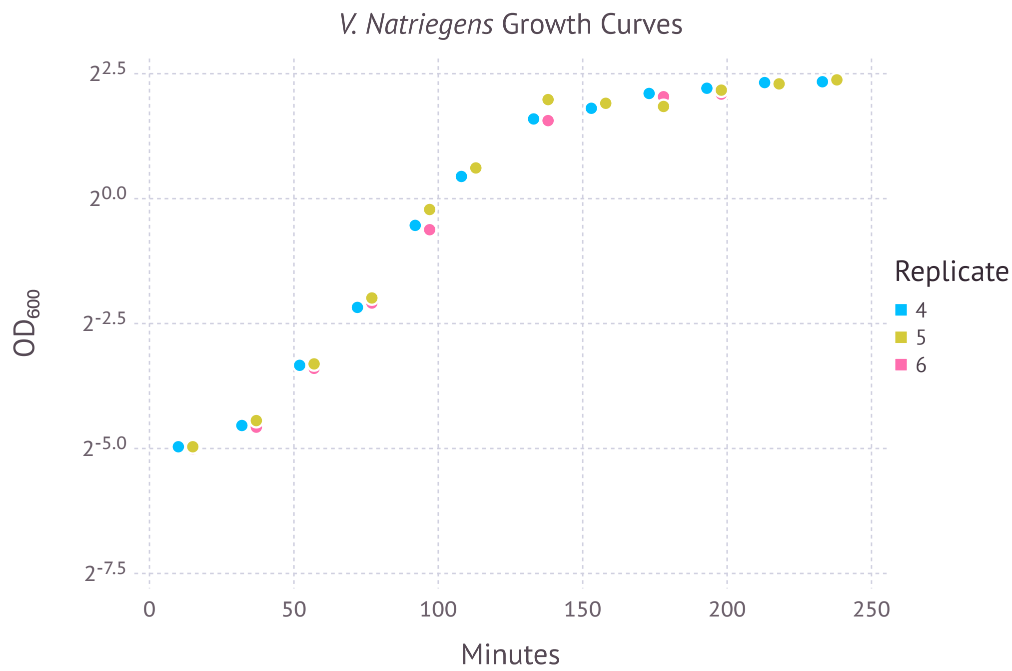
The doubling time for no glucose was ~12.9 minutes, with a growth rate of 0.0776

```
begin
    # Calculate doubling-time
    ngg = 1/coef(ngols)[2]
    # Format it into a nice string
    md"The doubling time for no glucose was ~$(round(ngg, sigdigits=3)) minutes, with
    a growth rate of $(round(coef(ngols)[2], sigdigits=3))"
end
```

glogdf =

	Time	Replicate	OD
1	10	"4"	0.032
2	15	"5"	0.032
3	15	"6"	0.031
4	32	"4"	0.043
5	37	"5"	0.046
6	37	"6"	0.042
7	52	"4"	0.099
8	57	"5"	0.101
9	57	"6"	0.095
10	72	"4"	0.221
more			
36	238	"6"	5.1

```
# Isolate the glucose media
glogdf = filter(:Replicate => r -> r in ["4", "5", "6"], pdf)
```




```
# Replot, but the isolated values so that we can pick out the exponential growth region
plot(glogdf, 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"))
```

tglogdf =

	Time	Replicate	OD
1	52	"4"	0.099
2	57	"5"	0.101
3	57	"6"	0.095
4	72	"4"	0.221
5	77	"5"	0.252
6	77	"6"	0.235
7	92	"4"	0.69
8	97	"5"	0.86
9	97	"6"	0.65

```
# Trim the data to take a closer look at log-phase
tglogdf = filter(:Time => t -> 52 <= t <= 97, glogdf)
```

	Time	Replicate	OD
1	52	"4"	-3.33643
2	57	"5"	-3.30757
3	57	"6"	-3.39593
4	72	"4"	-2.17788
5	77	"5"	-1.9885
6	77	"6"	-2.08927
7	92	"4"	-0.535332
8	97	"5"	-0.217591
9	97	"6"	-0.621488

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

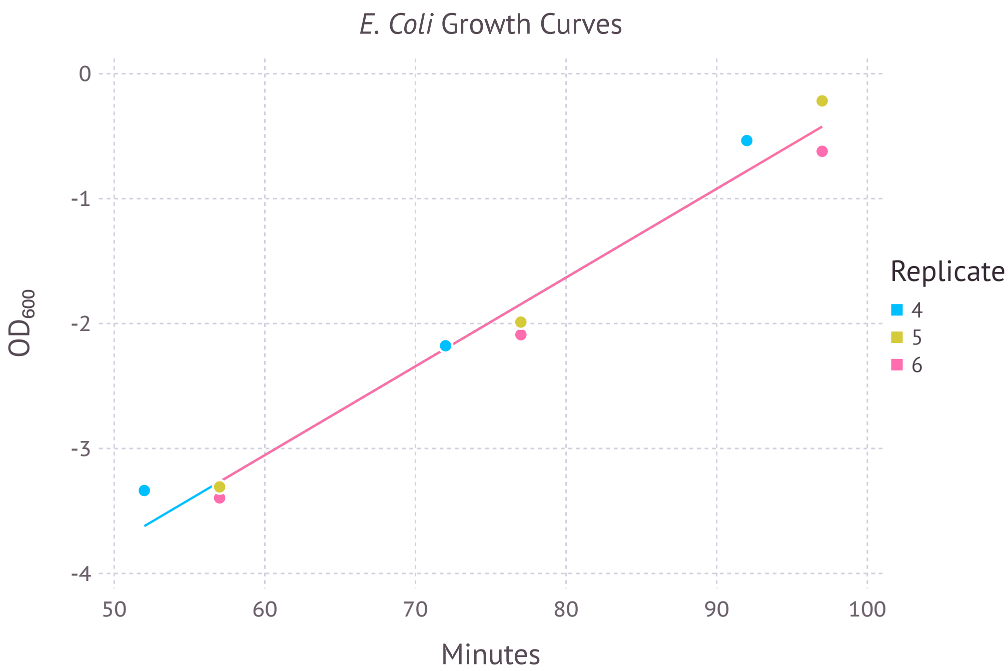
```
gols =
StatsModels.TableRegressionModel{LinearModel{GLM.LmResp{Vector{Float64}}, GLM.DensePredCho

OD ~ 1 + Time

Coefficients:
```

	Coef.	Std. Error	t	Pr(>|t|)	Lower 95%	Upper 95%
(Intercept)	-7.31688	0.333952	-21.91	<1e-06	-8.10655	-6.5272
Time	0.0710647	0.00433035	16.41	<1e-06	0.0608251	0.0813044

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

```
[-3.62151, -3.26619, -3.26619, -2.20022, -1.84489, -1.84489, -0.778921, -0.423597, -0.4235
```

```
# Insert a new column into our dataframe representing the model predictions
tglogdf[!,:Model] = predict(gols)
```




```
# And then plot it on a log-scale
plot(tglogdf, 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))
```

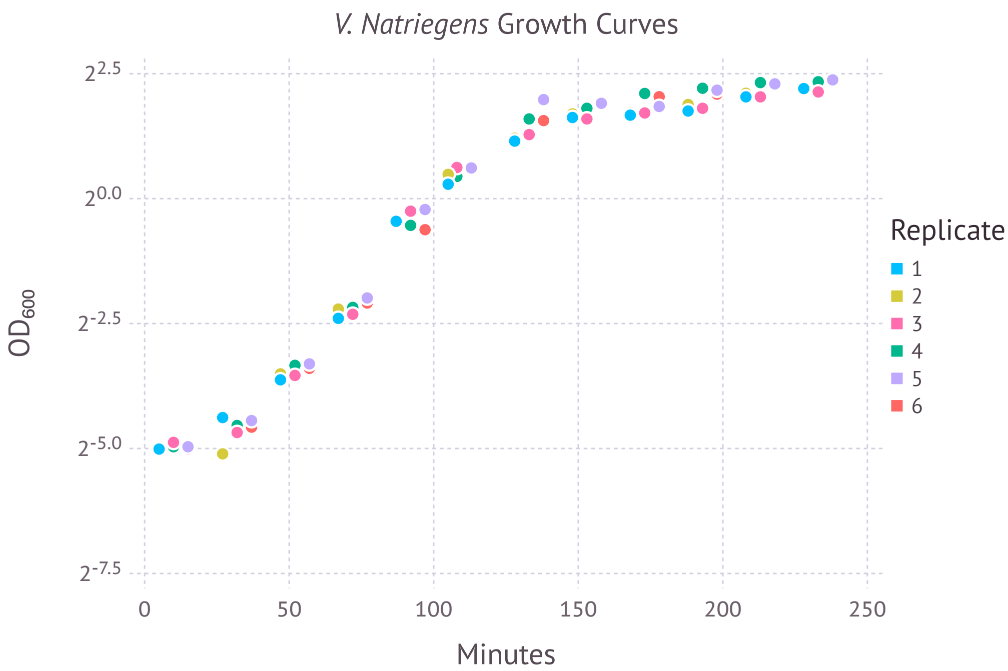
The doubling time for no glucose was ~14.1 minutes, with a growth rate of 0.0711

```
begin
    # Calculate doubling-time
    gg = 1/coef(gols)[2]
    # Format it into a nice string
    md"The doubling time for no glucose was ~$(round(gg, sigdigits=3)) minutes, with
    a growth rate of $(round(coef(gols)[2], sigdigits=3))"
end
```

comparedf =

	Time	Replicate	OD
1	5	"1"	0.031
2	5	"2"	0.031
3	10	"3"	0.034
4	10	"4"	0.032
5	15	"5"	0.032
6	15	"6"	0.031
7	27	"1"	0.048
8	27	"2"	0.029
9	32	"3"	0.039
10	32	"4"	0.043
	more		
72	238	"6"	5.1

```
# Isolate the with / without glucose media
comparedf = filter(:Replicate => r -> isdigit(r[1]) , pdf)
```




```
# Replot, but the isolated values so that we can pick out the exponential growth
region
plot(comparedf, 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"))
```