

# Survival Analysis – Handout

Marc Los Huertos

November 19, 2025

## 1 Goal

We have absorbance (optical density, OD) measurements of algal cultures over time for three treatments. This R Markdown walks through an analysis that **uses a survival-analysis approach**: we define the event as *time to reach a event* (i.e. reach end of experiment milestone). This allows handling of cultures that never reach the threshold within the experiment (right-censored observations) and comparing groups using KaplanMeier curves, log-rank tests, and Cox proportional hazards models.

## 2 Justification!

Welcome to survival analysis for short-term experiments! I know 72 hours seems quick for “survival” analysis, but this technique is perfect for understanding *when* growth rates change or fail, not just *if* they change.

In your experiment, you’re tracking algae growth under pollution stress. We’ll use survival analysis to answer:

- When do algae colonies first show growth impairment?
- How quickly does pollution affect growth rates?
- What proportion maintains normal growth throughout the 72 hours?

## Contents

<b>1</b>	<b>Goal</b>	<b>1</b>
<b>2</b>	<b>Justification!</b>	<b>1</b>
<b>3</b>	<b>Why use a survival analysis?</b>	<b>2</b>
3.1	1. Our event is the end of the experiment . . . . .	3
3.2	2. All cultures are censored by design . . . . .	3
3.3	3. It lets us compare treatments without assuming perfect growth curves . . . . .	3

3.4	4. It gives us simple comparisons between treatments . . . . .	3
3.5	5. We do not lose data by only looking at the final OD measurement	3
3.6	6. It fits the story of the experiment . . . . .	4
3.7	Summary . . . . .	4
<b>4</b>	<b>1. Packages &amp; session info</b>	<b>4</b>
<b>5</b>	<b>2. Data: two options</b>	<b>4</b>
5.1	Option A use your own CSV . . . . .	5
5.2	Option B simulated example data . . . . .	5
5.3	End Of experiment . . . . .	6
5.4	3. Visualize raw growth curves . . . . .	9
5.5	Define event: time to reach a threshold . . . . .	10
5.6	5. KaplanMeier curves by treatment . . . . .	12
5.7	6. Log-rank test (overall) . . . . .	14
5.8	8. Cox proportional hazards model . . . . .	15
5.9	9. Check proportional hazards assumption . . . . .	16
5.10	10. Visual diagnostics and alternative visualizations . . . . .	16
5.11	11. Reporting recommended outputs . . . . .	18
<b>6</b>	<b>Rationale Notes for Marc</b>	<b>19</b>
6.1	Rationale for Using Survival Analysis . . . . .	19
6.2	Focus on Time-to-Threshold Outcomes . . . . .	19
6.3	Natural Handling of Right-Censoring . . . . .	19
6.4	Minimal Assumptions About Growth Curves . . . . .	19
6.5	Interpretable Effect Sizes via Cox Models . . . . .	19
6.6	Avoiding Information Loss . . . . .	20
6.7	Flexibility for Complex Experiments . . . . .	20
6.8	Summary . . . . .	20

### 3 Why use a survival analysis?

Alright Korey and Maria, here is an explanation of why I think you should and CAN use this approach.

We are not looking for a specific growth threshold. The issue is that the experiment has a hard stop. Some algae cultures get close to something interesting, and others do not, but everything ends at the same final time point because the experiment has to end. In other words, we censor the data. This is called right censored data.

So in this circumstance, the real question becomes: “How long does each culture continue growing before the experiment forces us to stop?”

This is exactly the kind of situation survival analysis is made for. Here is why it fits what we are doing.

### **3.1 1. Our event is the end of the experiment**

Instead of waiting for the algae to reach some specific level, the “event” in our case is that the experiment ends. Some cultures are still growing. Some appear to have slowed down. Others might be about to separate from the pack but never get the chance because the experiment stops.

Survival analysis lets us model this without pretending the algae hit a threshold that they did not actually reach – in other words, we are removing a confounding factor: data that is not there because the experiment ended.

### **3.2 2. All cultures are censored by design**

Normally, censoring happens when something fails to happen in time. But here, censoring is simply the structure of the experiment. Every culture that does not finish whatever process we care about by the final measurement is censored at the end-of-experiment time.

This is not a problem for survival analysis. In fact, survival analysis is built to handle exactly this type of “we stopped watching before the event happened” data.

### **3.3 3. It lets us compare treatments without assuming perfect growth curves**

The algae do not follow perfect smooth logistic curves. They wiggle, plateau, slow down, and then take off again. If we tried to fit nonlinear models, we would be forcing the data into shapes it does not actually follow.

Survival analysis does not care about the exact shape of the growth curve. It only cares about time until the event or censoring. That makes it realistic and robust for the type of noisy, real-world growth dynamics we have.

### **3.4 4. It gives us simple comparisons between treatments**

Kaplan Meier curves show how much “growth potential” is still left in each group at each time point. Even though the event is the experiment ending, the KM curves still tell us who is ahead or behind as time goes on.

Cox models give hazard ratios. In our case, the hazard ratio tells us whether one treatment is “progressing toward the cutoff” faster or slower than another. This becomes a practical way to compare treatments even when none of them reach a traditional endpoint.

### **3.5 5. We do not lose data by only looking at the final OD measurement**

If we just looked at the final absorbance for each treatment, we would miss all the differences in how fast each culture was growing over time. Survival analysis uses all the data:

- every time point,
- every culture,
- all the differences in growth speed,
- all the censored cases.

Nothing gets thrown away. Are you starting to see why I think this is a good method? :-)

### 3.6 6. It fits the story of the experiment

The goal is not “Did it reach some number?” Instead, we might ask “How long did it keep progressing until we had to stop the experiment?”

Survival analysis is exactly the framework that treats our forced stop correctly and lets us compare treatments in a statistically meaningful way.

### 3.7 Summary

We are using survival analysis because the experiment does not run long enough for the cultures to reach a common endpoint. Instead, everyone gets cut off at the end of the experiment, and survival analysis is designed to handle that kind of censoring. It uses all of our data across time, does not require perfect growth curves, and gives us straightforward comparisons between treatments.

In short:

Survival analysis matches the way the experiment actually works, and it gives us the cleanest and most meaningful way to compare how the treatments influence growth over time.

## 4 1. Packages & session info

To run the analysis, I am relying on a few packages, e.g. `tidyverse` for data wrangling and plotting, `survival` for survival analysis, and `survminer` for nice plotting functions.

```
library(tidyverse)
library(survival)
library(survminer)
```

## 5 2. Data: two options

For Maria and Korey, I have created simulated example data below (Option B). You can also use your own data (Option A) if you have it in the suggested format. Hopefully, this handout will give you a sense of what this approach looks like and how to implement it!

## 5.1 Option A use your own CSV

a suggested format is long: one row per sample per timepoint.

Columns:

- ‘sample\_id’ (unique for each biological replicate)
- ‘treatment’ (factor with 3 levels, e.g. ‘A’, ‘B’, ‘C’)
- ‘time’ (time in hours or days; numeric)
- ‘absorbance’ (OD or fluorescence reading; numeric)

Read it with:

```
# df_raw <- read.csv("your_algae_data.csv")
# glimpse(df_raw)
```

## 5.2 Option B simulated example data

used in this report so the analysis runs end-to-end). I think this simulation creates realistic growth curves with noise, i.e. variation needed for a statistical analysis!

More over, I created 10 replicates – where some replicates that never hit the threshold.

```
set.seed(2025)
# parameters
n_rep <- 10 # replicates per treatment
times <- seq(0, 72, by = 8) # sample every 8 hours for 72 hours

make_growth <- function(n, treatment_label, mu_time_to_mid = 30, sd_time = 6, maxOD = 1.2, p_no_reach = 0.1) {
  tibble(sample = paste0(treatment_label, "_", seq_len(n))) %>%
    rowwise() %>%
    mutate(
      # latent time to reach midpoint (control over growth speed)
      t_mid = rnorm(1, mu_time_to_mid, sd_time) |> pmax(6),
      slope = (maxOD) / (t_mid + 0.1),
      # for a small proportion, set very slow growth so they won't reach threshold
      never = runif(1) < prop_no_reach
    ) %>%
    ungroup() %>%
    select(-t_mid, -slope, -never) -> samples

  expand_grid(sample = samples$sample, time = times) %>%
    left_join(samples, by = "sample") %>%
    rowwise() %>%
```

```

mutate(
  # generate an underlying logistic-like growth curve with noise
  mu = plogis((time - rnorm(1, mu_time_to_mid, sd_time))/7) * maxOD,
  absorbance = mu + rnorm(1, 0, 0.03)
) %>%
ungroup() %>%
mutate(treatment = treatment_label)
}

# create three treatments with different typical growth rates
df_A <- make_growth(n_rep, "A", mu_time_to_mid = 28, sd_time = 6, maxOD = 1.1, prop_no_reach = 0.1)
df_B <- make_growth(n_rep, "B", mu_time_to_mid = 34, sd_time = 7, maxOD = 1.15, prop_no_reach = 0.1)
df_C <- make_growth(n_rep, "C", mu_time_to_mid = 22, sd_time = 5, maxOD = 1.0, prop_no_reach = 0.1)

df_raw <- bind_rows(df_A, df_B, df_C) %>%
  mutate(treatment = factor(treatment))

# inspect a handful
df_raw %>% group_by(treatment) %>% slice_head(n = 6)

## # A tibble: 18 x 5
## # Groups:   treatment [3]
##   sample time      mu absorbance treatment
##   <chr> <dbl>    <dbl>      <dbl> <fct>
## 1 A_1      0 0.0233      0.0581 A
## 2 A_1      8 0.0623      0.0175 A
## 3 A_1     16 0.123      0.122  A
## 4 A_1     24 0.699      0.692  A
## 5 A_1     32 0.199      0.189  A
## 6 A_1     40 0.960      0.941  A
## 7 B_1      0 0.00495    -0.0645 B
## 8 B_1      8 0.213      0.199  B
## 9 B_1     16 0.0448      0.0604 B
## 10 B_1    24 0.365      0.342  B
## 11 B_1    32 0.793      0.842  B
## 12 B_1    40 0.960      0.968  B
## 13 C_1      0 0.0587      0.0488 C
## 14 C_1      8 0.104      0.0963 C
## 15 C_1     16 0.233      0.217  C
## 16 C_1     24 0.432      0.420  C
## 17 C_1     32 0.482      0.503  C
## 18 C_1     40 0.959      1.02   C

```

### 5.3 End Of experiment

```

# df_raw contains: sample, treatment, time, absorbance
# Step 1: find the end-of-experiment time
end_time <- max(df_raw$time)

# Step 2: create a time-to-event dataset where the event is "experiment ended"
# All samples have the same event time (the end) and all are censored (status = 0)

time_to_event <- df_raw %>%
  group_by(sample, treatment) %>%
  summarise(
    time = end_time, # everyone ends at the final observed time
    status = 0L,      # 0 = censored, because the event never actually happened
    .groups = "drop"
  )

time_to_event

## # A tibble: 30 x 4
##   sample treatment    time status
##   <chr>   <fct>      <dbl>  <int>
## 1 A_1     A           72      0
## 2 A_10    A           72      0
## 3 A_2     A           72      0
## 4 A_3     A           72      0
## 5 A_4     A           72      0
## 6 A_5     A           72      0
## 7 A_6     A           72      0
## 8 A_7     A           72      0
## 9 A_8     A           72      0
## 10 A_9    A           72      0
## # i 20 more rows

```

Some notes on the data:

1. Why status = 0 for everyone?

Because in your design:

- The experiment ends before anything biologically meaningful happens, so the event never occurs, and the only thing that happens is that measurement stops.
- This is exactly what right-censoring represents.

2. If you want to define a different event, like growth slowed or plateau started
3. You can define an event inside the experiment, for example: slope falls below a threshold, OD stops increasing, time to max observed absorbance, etc.

4. Then some samples would have status = 1 and some status = 0.
5. If you want that, tell me the biological definition and Ill write the code for it.

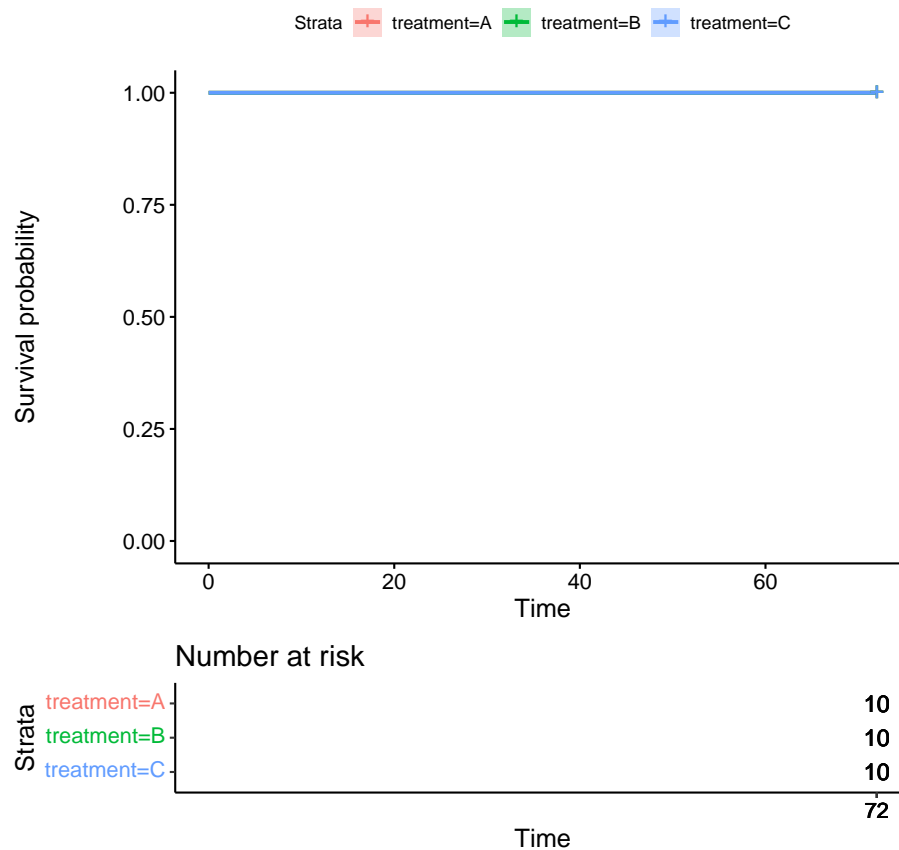
If you want to run KaplanMeier with end-of-experiment censoring

```
library(survival)
library(survminer)

km_fit <- survfit(Surv(time, status) ~ treatment, data = time_to_event)

ggsurvplot(km_fit, data = time_to_event,
            risk.table = TRUE,
            conf.int = TRUE,
            xlab = "Time",
            title = "Survival curves where experiment-ending is censoring")
```

Survival curves where experiment-ending is censoring





This will produce survival curves showing how much time under observation each treatment had all curves will drop only at the end (because everyone is censored at the same time), but this sets the correct structure for modeling.

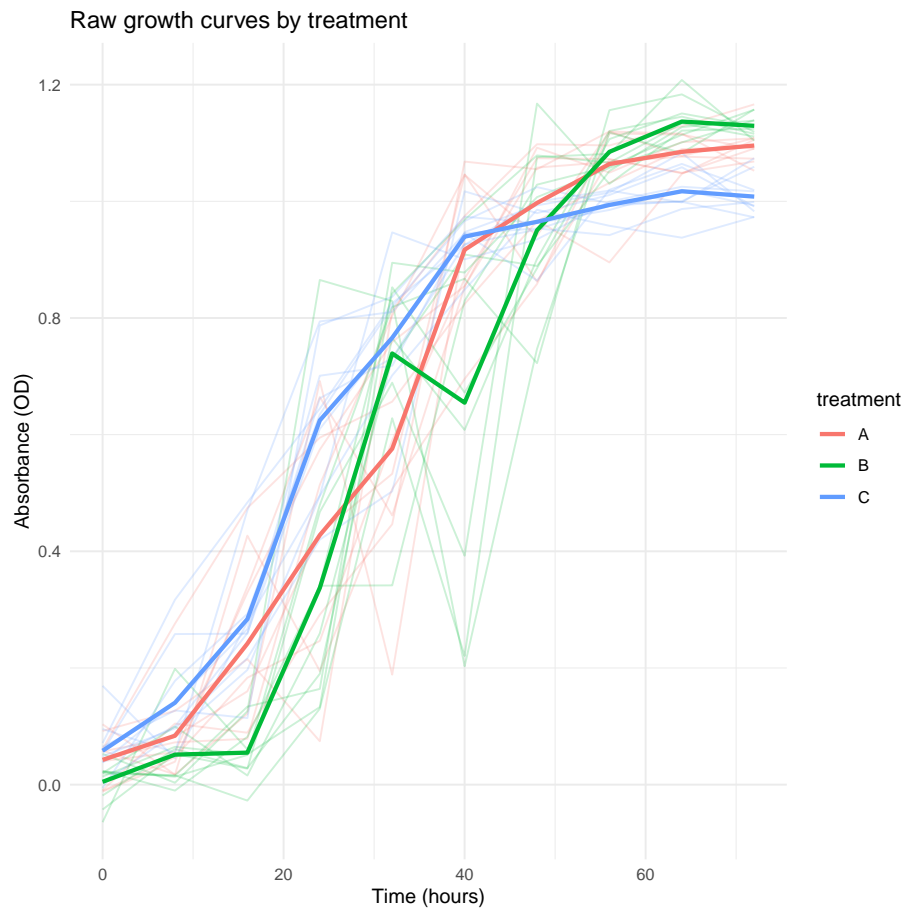
### 5.4 3. Visualize raw growth curves

Plotting raw absorbance over time for each replicate is important to see shapes, noise, and whether a single threshold makes sense.

```
# spaghetti plot: individual curves (light) + treatment mean (bold)
df_means <- df_raw %>%
  group_by(treatment, time) %>%
  summarise(mean_abs = mean(absorbance), .groups = "drop")

p_raw <- ggplot() +
  geom_line(data = df_raw,
            aes(time, absorbance, group = sample, color = treatment),
            alpha = 0.2, show.legend = FALSE) +
  geom_line(data = df_means,
            aes(time, mean_abs, color = treatment),
            size = 1.1) +
  labs(x = "Time (hours)",
       y = "Absorbance (OD)",
       title = "Raw growth curves by treatment") +
  theme_minimal()

p_raw
```



## 5.5 Define event: time to reach a threshold

Choose a threshold that represents a biologically meaningful growth milestone. For example, ‘threshold = 0.6’ (you can change this). We compute, per sample, the **first time** the absorbance is  $\geq$  threshold. If a sample never reaches the threshold during the recorded times, we treat it as **right-censored** at the last observed timepoint.

```
threshold <- 0.6

time_to_event <- df_raw %>%
  group_by(sample, treatment) %>%
  arrange(time) %>%
  summarise(
    event_time = {
```

```

    hit_rows <- which(absorbance >= threshold)
    if(length(hit_rows) == 0) NA_real_ else time[min(hit_rows)]
  },
  last_time = max(time),
  .groups = "drop"
) %>%
mutate(
  status = if_else(is.na(event_time), 0L, 1L),      # 1 = event observed, 0 = censored
  time = if_else(is.na(event_time), last_time, event_time)
)

# show table
time_to_event %>% count(treatment, status)

## # A tibble: 3 x 3
##   treatment status      n
##   <fct>      <int> <int>
## 1 A           1     10
## 2 B           1     10
## 3 C           1     10

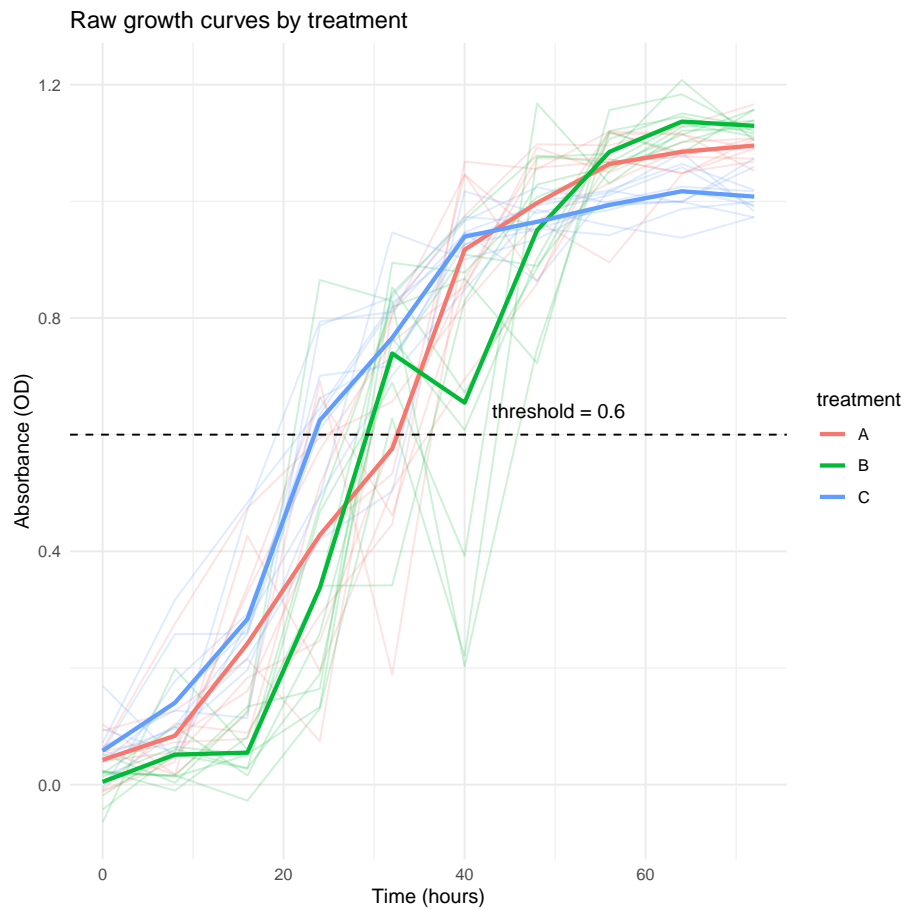
```

Plot the threshold on the raw curves so readers see what the event means.

```

p_raw + geom_hline(yintercept = threshold, linetype = "dashed") +
  annotate("text", x = max(df_raw$time)*0.7, y = threshold + 0.04, label = paste0("threshold"))

```



## 5.6 5. KaplanMeier curves by treatment

Create a 'Surv' object and fit KaplanMeier curves. We will visualize them and report median time-to-event where available.

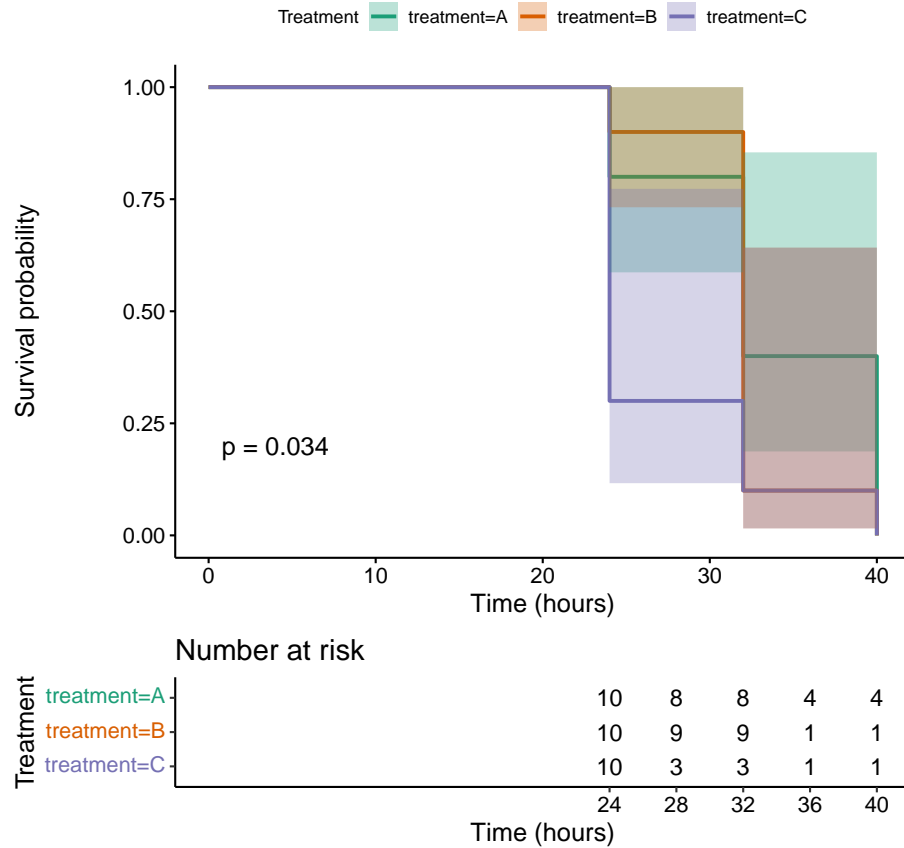
```
km_fit <- survfit(Surv(time, status) ~ treatment, data = time_to_event)
summary(km_fit)
```

```
## Call: survfit(formula = Surv(time, status) ~ treatment, data = time_to_event)
##
##               treatment=A
##   time n.risk n.event survival std.err lower 95% CI upper 95% CI
##   24      10      2      0.8  0.126    0.587      1.000
##   32       8      4      0.4  0.155    0.187      0.855
##   40       4      4      0.0   NaN      NA         NA
```

```
##
##           treatment=B
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##    24     10      1     0.9  0.0949    0.7320    1.000
##    32      9      8     0.1  0.0949    0.0156    0.642
##    40      1      1     0.0    NaN        NA        NA
##
##           treatment=C
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##    24     10      7     0.3  0.1449    0.1164    0.773
##    32      3      2     0.1  0.0949    0.0156    0.642
##    40      1      1     0.0    NaN        NA        NA

ggsurvplot(km_fit, data = time_to_event, risk.table = TRUE, pval = TRUE,
            conf.int = TRUE, palette = "Dark2",
            title = "KaplanMeier: time to reach absorbance threshold by treatment",
            xlab = "Time (hours)", legend.title = "Treatment")
```

### Kaplan...Meier: time to reach absorbance threshold by tre



**\*\*Interpretation\*\*:** The survival curve here shows the probability that a culture has *not yet* reached the threshold at each timepoint. Lower curves correspond to faster attainment of the threshold.

## 5.7 6. Log-rank test (overall)

The log-rank test (implemented as 'survdif') checks for differences in survival functions across groups.

```
logrank <- survdiff(Surv(time, status) ~ treatment, data = time_to_event)
logrank

## Call:
## survdiff(formula = Surv(time, status) ~ treatment, data = time_to_event)
##
##           N Observed Expected (O-E)^2/E (O-E)^2/V
```

```
## treatment=A 10      10      12.93      0.6653      3.318
## treatment=B 10      10      10.63      0.0377      0.153
## treatment=C 10      10      6.43      1.9774      6.069
##
##  Chisq= 6.7  on 2 degrees of freedom, p= 0.03

# compute approximate p-value
p_val <- 1 - pchisq(logrank$chisq, df = length(logrank$n) - 1)
cat("Omnibus log-rank p-value:", signif(p_val, 3), "\n")

## Omnibus log-rank p-value: 0.0345
```

subsection7. Pairwise comparisons (adjusted)

If the omnibus test is significant, we may want pairwise comparisons. We'll compute pairwise log-rank tests and adjust p-values using the Benjamini-Hochberg method.

```
# pairwise_survdif is available via surminer
pairwise_res <- pairwise_survdif(Surv(time, status) ~ treatment, data = time_to_event)
pairwise_res

##
##  Pairwise comparisons using Log-Rank test
##
##  data:  time_to_event and treatment
##
##    A      B
## B 0.365 -
## C 0.073 0.073
##
## P value adjustment method: BH

# extract p-values and adjust
pw <- pairwise_res$p.value
pw_vec <- na.omit(as.vector(pw))
adj <- p.adjust(pw_vec, method = "BH")
adj

## [1] 0.3652911 0.1088803 0.1088803
```

## 5.8 8. Cox proportional hazards model

A Cox model estimates hazard ratios between treatments, which can be interpreted as relative rates of reaching the threshold.

```

# encode treatment as factor; choose reference
time_to_event <- time_to_event %>% mutate(treatment = relevel(treatment, ref = "A"))
cox1 <- coxph(Surv(time, status) ~ treatment, data = time_to_event)
summary(cox1)

## Call:
## coxph(formula = Surv(time, status) ~ treatment, data = time_to_event)
##
##      n= 30, number of events= 30
##
##              coef exp(coef) se(coef)      z Pr(>|z|)
## treatmentB 0.3785      1.4601  0.4586 0.825  0.4092
## treatmentC 0.9301      2.5347  0.4554 2.042  0.0411 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##              exp(coef) exp(-coef) lower .95 upper .95
## treatmentB      1.460      0.6849    0.5943    3.587
## treatmentC      2.535      0.3945    1.0382    6.189
##
## Concordance= 0.722 (se = 0.085 )
## Likelihood ratio test= 4.04 on 2 df,  p=0.1
## Wald test              = 4.23 on 2 df,  p=0.1
## Score (logrank) test = 4.45 on 2 df,  p=0.1

```

Notes: the hazard ratio (HR) > 1 means faster attainment of the threshold relative to the reference.

## 5.9 9. Check proportional hazards assumption

```

ph_test <- cox.zph(cox1)
ph_test

##              chisq df    p
## treatment    3.61  2 0.16
## GLOBAL       3.61  2 0.16

plot(ph_test)

```

If the proportional hazards assumption fails for treatment, consider alternatives: stratified Cox, time-varying covariates, or non-parametric comparisons.

## 5.10 10. Visual diagnostics and alternative visualizations

Plot the cumulative hazard or complementary log-log plots if helpful. We can also overlay median times and forest plots of HRs.



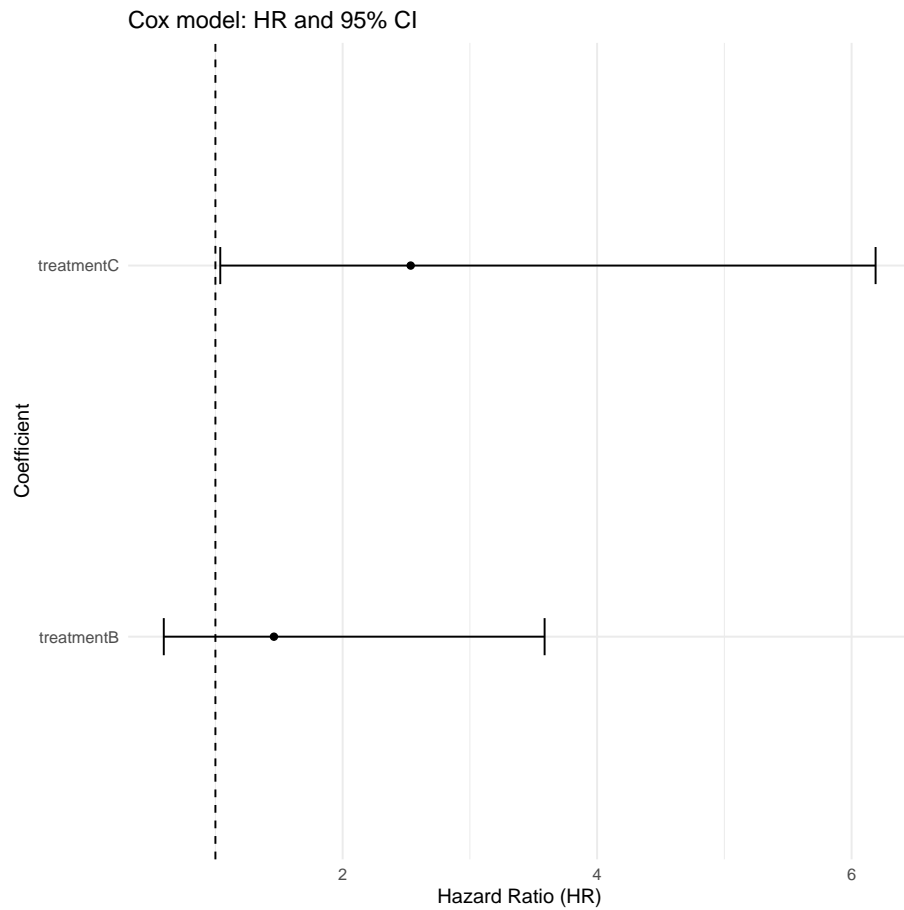
```

# simple forest plot for Cox HRs
hr <- broom::tidy(cox1, exponentiate = TRUE, conf.int = TRUE)
hr

## # A tibble: 2 x 7
##   term          estimate std.error statistic p.value conf.low conf.high
##   <chr>         <dbl>     <dbl>     <dbl>   <dbl>   <dbl>   <dbl>
## 1 treatmentB     1.46      0.459      0.825  0.409    0.594    3.59
## 2 treatmentC     2.53      0.455      2.04   0.0411    1.04    6.19

ggplot(hr, aes(x = term, y = estimate)) +
  geom_point() +
  geom_errorbar(aes(ymin = conf.low, ymax = conf.high), width = 0.1) +
  geom_hline(yintercept = 1, linetype = "dashed") +
  coord_flip() +
  labs(y = "Hazard Ratio (HR)", x = "Coefficient", title = "Cox model: HR and 95% CI") +
  theme_minimal()

```



## 5.11 11. Reporting recommended outputs

A minimal reproducible report should include:

- Raw growth curve figure with threshold annotated (Step 3 & 4).
- Table of ‘time-to-event’ with ‘sample’, ‘treatment’, ‘time’, ‘status’ (observed/censored).
- Kaplan–Meier plot with risk table and p-value (Step 5).
- Omnibus log-rank test results and pairwise p-values with multiple-testing corrections (Steps 6–7).
- Cox model coefficients, hazard ratios and 95% CI (Step 8) and asses

## 6 Rationale Notes for Marc

### 6.1 Rationale for Using Survival Analysis

Survival analysis is a powerful and appropriate statistical framework for analyzing algae growth when the primary question concerns *how long* it takes cultures to reach a biologically meaningful milestone, such as exceeding a specified absorbance threshold. This section explains why survival methods are preferred in such situations.

### 6.2 Focus on Time-to-Threshold Outcomes

Many algal growth studies aim to compare how long cultures under different treatments require to reach: - a specified absorbance (OD) level, - the onset of exponential growth, or - a density indicating establishment or competitive success.

Survival analysis directly models *time-to-event* outcomes, aligning the statistical method with the biological question.

### 6.3 Natural Handling of Right-Censoring

Not every culture reaches the absorbance threshold during the experimental window. Traditional approaches (e.g., ANOVA on final OD or curve-fitting) often discard these observations or treat them incorrectly. Survival analysis handles such cases as *right-censored* data: - the event did not occur during the observation period, - but the culture still provides valid information up to its last measurement time.

This prevents bias and maximizes data use.

### 6.4 Minimal Assumptions About Growth Curves

KaplanMeier curves and log-rank tests compare groups *without requiring a specific growth model*. This is helpful because real algal populations may deviate from ideal logistic or Gompertz growth curves due to: - environmental fluctuations, - physiological acclimation, - measurement noise, and - treatment-specific growth dynamics.

Survival methods remain robust under these conditions.

### 6.5 Interpretable Effect Sizes via Cox Models

The Cox proportional hazards model provides hazard ratios that quantify how treatments affect the *rate at which cultures reach the absorbance threshold*: - hazard ratio (HR)  $> 1$  indicates faster growth, - HR  $< 1$  indicates delayed growth.

These effect sizes are intuitive and directly comparable across treatments.

## 6.6 Avoiding Information Loss

Analyzing only end-point absorbance values ignores valuable temporal data. Survival analysis incorporates: - timing of all measurements up to the event, - differences in growth trajectories, - censored observations.

This increases statistical power and provides a more complete view of growth dynamics.

## 6.7 Flexibility for Complex Experiments

Survival frameworks easily extend to: - multiple covariates, - time-varying predictors, - stratified models, - non-proportional hazards scenarios, - mixed-effects survival models.

This flexibility allows the approach to scale with more sophisticated experimental designs.

## 6.8 Summary

Survival analysis is an ideal approach for algae growth experiments where the goal is to compare the \*\*time required to reach a growth milestone\*\*, especially when:

- treatments create different growth rates,
- some cultures never reach the threshold,
- the full growth curves are noisy or irregular, or
- time-based comparisons provide the most biological insight.

It allows full use of the data, handles censoring