Survival Models

Fitting Models to Data, Not Data to Models
Model Fitting Series - With Applications in R

Jesse Ghashti
Source code by Jesse Ghashti

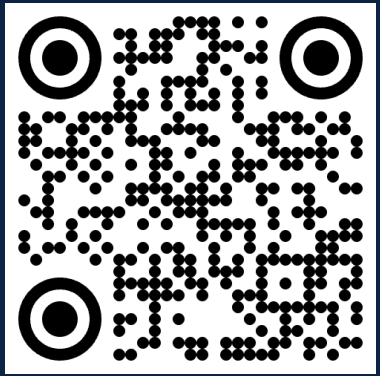
November 25, 2025

Centre for Scholarly Communication
The University of British Columbia | Okanagan Campus | Syilx Okanagan Nation Territory

Workshop Series Overview

Session	Topic	Date/Time
1	Simple Linear Regression	Oct 7, 9:00 AM
2	Fitting Linear Models in R	Oct 8, 10:30 AM
3	Multiple Linear Regression in R	Oct 16, 4:00 PM
4	Interaction Terms & Hierarchical Linear Models	Oct 21, 11:00 AM
5	Generalized Linear Models	Oct 23, 4:00 PM
6	Generalized Additive Models (GAMs)	Oct 28, 11:00 AM
7	Interpreting & Predicting from GAMs	Oct 29, 10:30 AM
8	Hierarchical GAMs	Nov 4, 12:00 PM
9	Penalized Models	Nov 18, 11:00 AM
10	Survival Models	Nov 25, 11:00 AM
11	Nonparametric Models	Dec 2, 11:00 AM

New Here?

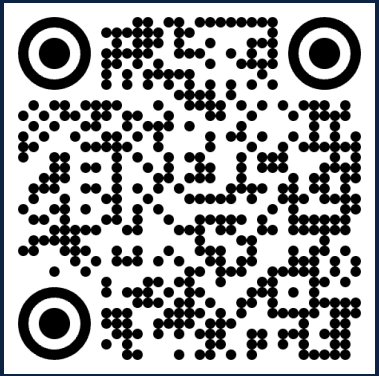


<———————-

New to R? Check out the Fundamentals of R series!

GitHub code and slides for today's workshop (and previous workshops)

———————————>



Alternatively, code/slides available at the bottom of `https://csc-ubc-okanagan.github.io/workshops/`

Today: Survival Regression and Prediction

Today we will...

- Build a realistic **simulated reliability dataset** (pumps in plants)
- Fit and interpret **Cox PH models** with and without plant-level frailty
- Allow for **time-varying covariate effects** (non-proportional hazards)
- Compare **parametric AFT models** (Weibull, lognormal) with Cox
- Fit a **piecewise exponential model** via Poisson regression
- Evaluate **predictive accuracy**: AIC, C-index, prediction error curves

We will use these packages...

```
# install.packages(c("survival", "survminer", "dplyr", "ggplot2", "broom", "pec"))
library(survival)
library(survminer)
library(dplyr)
library(ggplot2)
library(broom)
library(pec)
```

Data Generating Story: Pumps in Three Plants

Reliability Setup

We simulate time-to-failure for industrial pumps installed in **3 plants**:

- Each pump belongs to a plant $\in \{\texttt{P1}, \texttt{P2}, \texttt{P3}\}$
- **Continuous covariates:**
  - Temperature (`temp`): degrees C above baseline
  - Load (`load`): relative design load
- **Categorical covariates:**
  - Material (`material` = A/B)
  - Maintenance program (`maint` = 0/1)
- **Right censoring** between 40 and 120 months
- True model has a **plant-level frailty** (unobserved plant stressors)

Simulating Pump Reliability Data

Simulation code

```r
## Simulated Data
## Features:
## 1. 3 plants with different baseline reliability (cluster effect)
## 2. Continuous covariates: temperature, load
## 3. Categorical: material (A/B), maintenance program (0/1)
## Right censoring, true model has a plant-level random effect (frailty)
numberOFplants <- 3
plant    <- factor(sample(paste0("P", 1:numberOFplants), size = 500, replace = TRUE))
temp     <- rnorm(500, mean = 25, sd = 4)          # degrees C above baseline
load     <- runif(500, 0.7, 1.3)                   # design load
material <- factor(sample(c("A","B"), 500, TRUE)) # material it was made from
maint    <- rbinom(500, 1, 0.5)                    # maintenance program (think of regular or sparse)

## plant frailty: log-normal random effect on hazard. this can be though of as unobserved plant-level stressors
plantsEFFECT <- rnorm(numberOFplants, mean = 0, sd = 0.5)
names(plantsEFFECT) <- paste0("P", 1:numberOFplants)
## baseline hazards and their respective coefficients
lambda0 <- 0.005          # baseline rate
baselineTEMP  <- 0.04     # per °C
baselineLOAD  <- 1.2
baselineMATERIALB  <- -0.5
baselineMAINTENANCE <- -0.7
```

Simulating Pump Reliability Data

Simulation code

```r
## predictor for the hazard occurring
linpred <- baselineTEMP*(temp-mean(temp)) +
  baselineLOAD*(load-1) +
  baselineMATERIALB*(material=="B") +
  baselineMAINTENANCE*maint +
  plantsEFFECT[plant]

lambdaI <- lambda0*exp(linpred)

## we will model with exponential survival times
U <- runif(500)
trueTIME <- -log(U)/lambdaI
## Random censoring between 40 and 120 months
censorTIME <- runif(500, 40, 120)
time   <- pmin(trueTIME, censorTIME)
status <- as.integer(trueTIME <= censorTIME)  # 1=failure, 0=censored

dat <- data.frame(pump_id  = 1:500, plant,
  time, status, temp, load, material, maint = factor(maint))
head(dat)
```

	pump_id	plant	time	status	temp	load	material	maint
1	1	P1	38.34740	1	28.34623	1.2570109	A	0
2	2	P1	35.70169	1	26.16657	1.2670218	A	1
3	3	P1	86.44448	0	24.45903	0.9121598	B	0
4	4	P1	60.23994	1	22.21728	1.1337975	A	0
5	5	P1	40.82726	0	24.54467	0.9440624	B	1

Kaplan–Meier Curves by Plant

Nonparametric survival

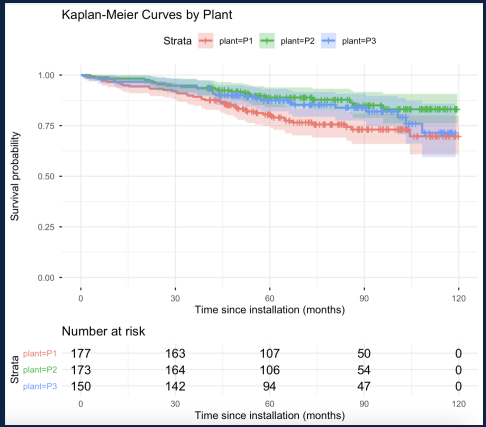
We first explore survival differences **across plants**:

- Construct $S(t)$ using **Kaplan–Meier** for each plant
- Check:
  - Which plant has higher survival (better reliability)?
  - Are differences mostly early or late in follow-up?
- This is a model-free, descriptive step before regression

KM Curves by Plant in R

Kaplan–Meier

```r
## KM by plant
survivalOBJECT <- Surv(dat$time, dat$status)
kmPLANT <- survfit(survivalOBJECT ~ plant,
                   data = dat)
ggsurvplot(
  kmPLANT,
  conf.int   = TRUE,
  risk.table = TRUE,
  ggtheme    = theme_minimal(),
  xlab       = "Time since installation (months)",
  ylab       = "Survival probability",
  title      = "Kaplan-Meier Curves by Plant"
)
```



Cox Proportional Hazards Model

Cox PH

Model...

$$\lambda(t \mid X) = \lambda_0(t)\, \exp(\beta^\top X)$$

- $\lambda_0(t)$: unspecified baseline hazard
- β: log hazard ratios for covariates
- Proportional hazards assumption: hazard ratios do not change over time

To interpret...

- $\exp(\beta_j)$ is the hazard ratio for a `1`-unit increase in covariate X_j
- > 1: higher failure rate (shorter survival); < 1: lower failure rate (longer survival)

Cox PH Model for Pump Data

Fit Cox model

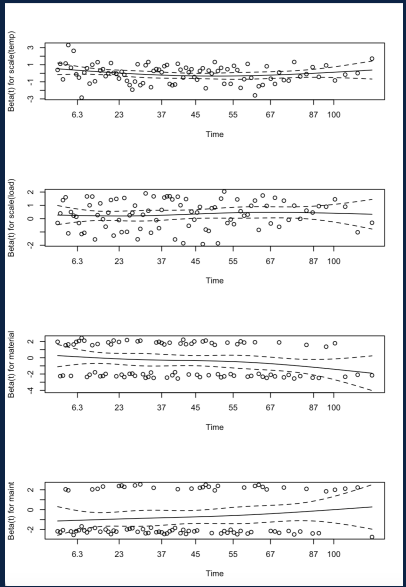
```
## Cox PH model
coxBASIC <- coxph(
  Surv(time, status) ~ scale(temp)+
                scale(load)+
                material+maint,
  data = dat,
  x = TRUE,
  y = TRUE
)
summary(coxBASIC)
broom::tidy(coxBASIC, exponentiate = TRUE,
            conf.int = TRUE)
```

```
Call:
coxph(formula = Surv(time, status) ~ scale(temp) + scale(load) +
    material + maint, data = dat, x = TRUE, y = TRUE)

  n= 500, number of events= 89

                coef exp(coef) se(coef)      z Pr(>|z|)
scale(temp) -0.05947   0.94226  0.10591 -0.562  0.57442
scale(load)  0.32119   1.37877  0.11196  2.869  0.00412 **
materialB   -0.38906   0.67769  0.21403 -1.818  0.06910 .
maint1      -0.71371   0.48983  0.22512 -3.170  0.00152 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

            exp(coef) exp(-coef) lower .95 upper .95
scale(temp)    0.9423     1.0613    0.7656    1.1596
scale(load)    1.3788     0.7253    1.1071    1.7171
materialB      0.6777     1.4756    0.4455    1.0309
maint1         0.4898     2.0415    0.3151    0.7615

Concordance= 0.638  (se = 0.03 )
Likelihood ratio test= 22.47  on 4 df,   p=2e-04
Wald test            = 21.13  on 4 df,   p=3e-04
Score (logrank) test = 21.8  on 4 df,   p=2e-04

> broom::tidy(coxBASIC, exponentiate = TRUE, conf.int = TRUE)
# A tibble: 4 × 7
  term         estimate std.error statistic p.value conf.low conf.high
  <chr>           <dbl>     <dbl>     <dbl>   <dbl>    <dbl>     <dbl>
1 scale(temp)     0.942     0.106    -0.562 0.574       0.766     1.16
2 scale(load)     1.38      0.112     2.87  0.00412     1.11      1.72
3 materialB       0.678     0.214    -1.82  0.0691      0.446     1.03
4 maint1          0.490     0.225    -3.17  0.00152     0.315     0.761
```

Cox PH Model for Pump Data

Check PH

```
## check the proportional hazards assumptions
coxBASIC_zph <- cox.zph(coxBASIC)
coxBASIC_zph
plot(coxBASIC_zph)
```



Shared Frailty: Plant-Level Random Effects

Unobserved heterogeneity

Pumps within the same plant may share unmeasured risk factors (frailty):

$$\lambda_{ij}(t) = \lambda_0(t)\, \exp(\beta^\top X_{ij} + u_{k(i)})$$

- i: pump, $k(i)$: plant
- u_k: plant-level random effect (**frailty**)
- Captures unobserved plant stressors (e.g., operating conditions, management)

Why use frailty models?

- Account for within-plant clustering
- Get more realistic uncertainty and plant-specific hazard multipliers

Cox Model with Plant Frailty

Fitting frailty model

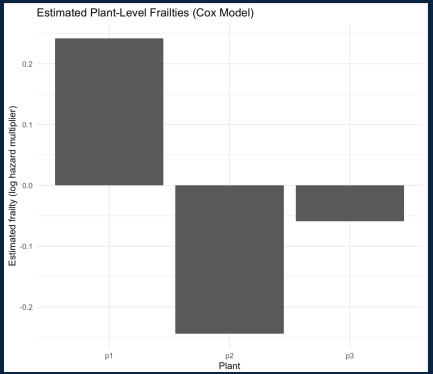
```
## Cox model with plant-level frailty
coxFRAILTY <- coxph(
  Surv(time, status) ~ scale(temp)+scale(load)
            +material+maint + frailty(plant),
  data = dat,
  x = TRUE,
  y = TRUE
)
summary(coxFRAILTY)
## compare log-likelihood/AIC with and without frailty
AIC(coxBASIC, coxFRAILTY)
```

```
coxph(formula = Surv(time, status) ~ scale(temp) + scale(load) +
    material + maint + frailty(plant), data = dat, x = TRUE,
    y = TRUE)

  n= 500, number of events= 89

                coef     se(coef) se2    Chisq DF   p
scale(temp)    -0.06023 0.1051   0.1051  0.33 1.00 0.5700
scale(load)     0.32854 0.1120   0.1120  8.61 1.00 0.0033
materialB      -0.39137 0.2141   0.2141  3.34 1.00 0.0680
maint1         -0.74047 0.2257   0.2255 10.77 1.00 0.0010
frailty(plant)                           5.72 1.31 0.0260

              exp(coef) exp(-coef) lower .95 upper .95
scale(temp)      0.9415     1.0621    0.7663    1.1569
scale(load)      1.3889     0.7200    1.1152    1.7298
materialB        0.6761     1.4790    0.4444    1.0288
maint1           0.4769     2.0969    0.3064    0.7422
gamma:P1         1.2737     0.7851    0.9010    1.8005
gamma:P2         0.7836     1.2762    0.5261    1.1671
gamma:P3         0.9427     1.0608    0.6443    1.3793

Iterations: 10 outer, 29 Newton-Raphson
     Variance of random effect= 0.06355876   I-likelihood = -511.7
Degrees of freedom for terms= 1.0 1.0 1.0 1.0 1.3
Concordance= 0.658  (se = 0.03 )
Likelihood ratio test= 30.06  on 5.3 df,   p=2e-05

> ## compare log-likelihood/AIC with and without frailty
> AIC(coxBASIC, coxFRAILTY)
                df      AIC
coxBASIC   4.000000 1033.967
coxFRAILTY 5.304768 1028.984
```

Cox Model with Plant Frailty

Frailty model coefficient visualization

```r
## random effect estimates (frailties)
frailtyVALS <- data.frame(
  plant = c("p1", "p2", "p3"),
  frailty = coxFRAILTY$coefficients[c("gamma:P1",
            "gamma:P2", "gamma:P3")]
)

ggplot(frailtyVALS, aes(x = plant, y = frailty)) +
  geom_col() +
  theme_minimal() +
  labs(
    x = "Plant",
    y = "Estimated frailty (log hazard multiplier)",
    title = "Estimated Plant-Level Frailties (Cox Model)"
  )
```



Non-Proportional Hazards and Time-Varying Effects

When PH fails

If diagnostics suggest violation of PH for a covariate (e.g. load):

- Effect of that covariate may change over time
- A simple approach: allow a time-varying coefficient

$$\lambda(t \mid X) = \lambda_0(t)\, \exp\{\beta^\top X + \gamma \cdot \texttt{load} \cdot f(t)\}$$

- Here we choose $f(t) = \log(t+1)$
- Interpretation: the importance of `load` increases/decreases over $\log(\text{time})$

Time-Varying Effect of Load

Cox with `tt()`

```
## non-proportional hazards: time-varying effect
## suppose diagnostic suggested load has
## time-varying effect. We will now allow
## an effect of load that changes ~ log(time).
## tt() gets covariate*function of time;
coxTV <- coxph(
  Surv(time, status) ~ scale(temp)+material
                       +maint+tt(load),
  data = dat,
  tt = function(x, t, ...) x*log(t+1)
)
summary(coxTV)
## the coefficient of tt(load) describes
## how effect of load increases/decreases
## over log(time). In reality, this is
## interpreted as load matters more/less
## as pumps age.
```

```
Call:
coxph(formula = Surv(time, status) ~ scale(temp) + material +
    maint + tt(load), data = dat, tt = function(x, t, ...) x *
    log(t + 1))

  n= 500, number of events= 89

                coef exp(coef) se(coef)      z Pr(>|z|)
scale(temp) -0.05822   0.94344  0.10589 -0.550  0.58244
materialB   -0.39300   0.67503  0.21416 -1.835  0.06649 .
maint1      -0.71585   0.48878  0.22517 -3.179  0.00148 **
tt(load)     0.52372   1.68829  0.18177  2.881  0.00396 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

            exp(coef) exp(-coef) lower .95 upper .95
scale(temp)    0.9434     1.0600    0.7666    1.1610
materialB      0.6750     1.4814    0.4436    1.0271
maint1         0.4888     2.0459    0.3144    0.7599
tt(load)       1.6883     0.5923    1.1823    2.4109

Concordance= 0.639  (se = 0.031 )
Likelihood ratio test= 22.56  on 4 df,   p=2e-04
Wald test            = 21.15  on 4 df,   p=3e-04
Score (logrank) test = 21.84  on 4 df,   p=2e-04
```

Parametric Survival Models (AFT View)

Weibull and lognormal AFT

Accelerated Failure Time (AFT) models:

$$\log T = \mu + \beta^\top X + \sigma\varepsilon$$

- Weibull and lognormal are two common parametric choices
- Coefficients exponentiated $\Rightarrow$ time ratios, not hazard ratios
- Time ratio > 1: longer survival (stretching the time scale)
- Time ratio < 1: shorter survival (compressing the time scale)

We fit

- Fit Weibull and lognormal AFT models
- Compare their AIC and interpret covariate time ratios

Weibull and Lognormal AFT Models

Fit parametric AFT models

```r
## Parametric models: Weibull and Lognormal
weibullAFT <- survreg(
  Surv(time, status) ~ temp+load+
                       material+maint,
  data = dat,
  dist = "weibull"
)
lognormalAFT <- survreg(
  Surv(time, status) ~ temp+load+
                       material+maint,
  data = dat,
  dist = "lognormal"
)
summary(weibullAFT)
summary(lognormalAFT)
AIC(weibullAFT, lognormalAFT)
```

```
Call:
survreg(formula = Surv(time, status) ~ temp + load + material +
    maint, data = dat, dist = "weibull")
              Value Std. Error     z       p
(Intercept)  6.8823     0.9072  7.59 3.3e-14
temp         0.0142     0.0252  0.56  0.5738
load        -1.7517     0.6287 -2.79  0.0053
materialB    0.3730     0.2032  1.84  0.0663
maint1       0.6750     0.2190  3.08  0.0021
Log(scale)  -0.0645     0.0989 -0.65  0.5143

Scale= 0.938

Weibull distribution
Loglik(model)= -611.6   Loglik(intercept only)= -623
        Chisq= 22.8 on 4 degrees of freedom, p= 0.00014
Number of Newton-Raphson Iterations: 8
n= 500

> summary(lognormalAFT)

Call:
survreg(formula = Surv(time, status) ~ temp + load + material +
    maint, data = dat, dist = "lognormal")
              Value Std. Error     z       p
(Intercept)  7.5431     1.0421  7.24 4.5e-13
temp        -0.0018     0.0298 -0.06 0.95181
load        -1.8864     0.7292 -2.59 0.00968
materialB    0.2843     0.2421  1.17 0.24031
maint1       0.8388     0.2514  3.34 0.00085
Log(scale)   0.6139     0.0867  7.08 1.4e-12

Scale= 1.85

Log Normal distribution
Loglik(model)= -614.5   Loglik(intercept only)= -624.7
        Chisq= 20.39 on 4 degrees of freedom, p= 0.00042
Number of Newton-Raphson Iterations: 4
n= 500

> ## compare AIC
> AIC(weibullAFT, lognormalAFT)
              df      AIC
weibullAFT     6 1235.151
lognormalAFT   6 1240.910
```

Weibull and Lognormal AFT Models

Fit parametric AFT models

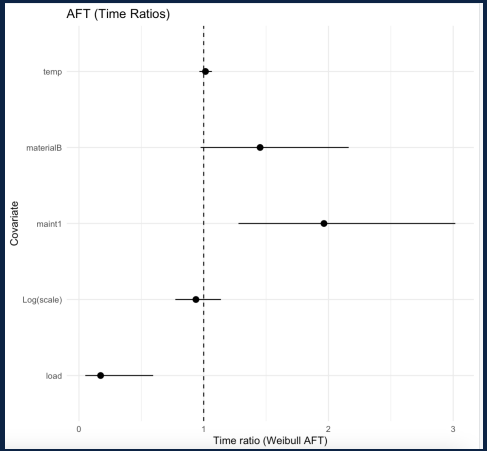
```
## convert AFT coefficients to time
## ratios (exp on log-time scale)
weibullAFT_timeRatio <- broom::tidy(weibullAFT) %>%
  filter(term != "(Intercept)") %>%
  mutate(
    timeRatio = exp(estimate),
    lower     = exp(estimate-1.96*std.error),
    upper     = exp(estimate+1.96*std.error)
  )
weibullAFT_timeRatio
```

	term	estimate	std.error	statistic	p.value	timeRatio	lower	upper
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	temp	0.0142	0.0252	0.562	0.574	1.01	0.965	1.07
2	load	-1.75	0.629	-2.79	0.00533	0.173	0.0506	0.595
3	materialB	0.373	0.203	1.84	0.0663	1.45	0.975	2.16
4	maint1	0.675	0.219	3.08	0.00205	1.96	1.28	3.02
5	Log(scale)	-0.0645	0.0989	-0.652	0.514	0.938	0.772	1.14

Time Ratios from Weibull AFT

Visualizing time ratios

```
## plot time ratios with CI
ggplot(weibullAFT_timeRatio,
       aes(x = term, y = timeRatio,
       ymin = lower, ymax = upper)) +
  geom_pointrange() +
  coord_flip() +
  theme_minimal() +
  geom_hline(yintercept = 1,
                linetype = "dashed") +
  labs(
    x = "Covariate",
    y = "Time ratio (Weibull AFT)",
    title = "AFT (Time Ratios)"
  )
```



KM vs Weibull: Model vs Nonparametric

Checking parametric fit

Compare...

- Kaplan–Meier curve for a subset of pumps
- Weibull AFT fitted survival curve for the same profile

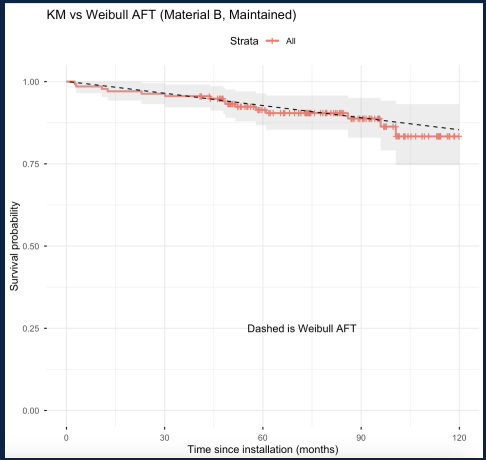
Questions...

- Does the Weibull curve track the KM reasonably well?
- Where does it deviate (early / late follow-up)?

KM vs Weibull AFT for a Profile

Profile and curves

```
## Weibull and KM visual comparison
## we will look at the profile associated with
## Plant P1, material B, maint=1, avg temp/load
profile <- data.frame(temp = mean(dat$temp),
  load     = mean(dat$load),
  material = factor("B", levels = levels(dat$material)),
  maint    = factor(1, levels = levels(dat$maint)))
## Predicted median survival time (months)
predict(weibullAFT, newdata = profile, type = "quantile", p = 0.5)
## Weibull survival curve from survreg params
t_grid <- seq(1, max(dat$time), length.out = 200)
xb     <- predict(weibullAFT, newdata = profile, type = "lp")
sigma <- weibullAFT$scale
survivalWeibull <- exp(-(t_grid*exp(-xb))^(1/sigma))
weibullDF <- data.frame(time = t_grid, S = survivalWeibull)
kmSUBSET <- survfit(Surv(time, status) ~ 1,
  data = dat %>% filter(material == "B", maint == 1))
gKM <- ggsurvplot(kmSUBSET, conf.int   = TRUE,
  risk.table = FALSE, ggtheme = theme_minimal(),
  xlab = "Time since installation (months)",
  ylab = "Survival probability",
  title = "KM vs Weibull AFT (Material B, Maintained)")
gKM$plot + geom_line(data = weibullDF,
        aes(x = time, y = S), linetype = "dashed") +
        annotate("text", x = max(dat$time)*0.6,
           y = 0.25, label = "Dashed is Weibull AFT")
```



Piecewise Exponential Model

Approximate changing hazards

Idea: model the hazard as constant within time intervals, but allow it to change between intervals.

- Split follow-up into intervals, e.g. $[0, 40), [40, 80), [80, \infty)$ months
- Within each interval: exponential model with its own rate
- Implement as a Poisson regression with log(time) offset

This is a flexible semi-parametric approach that still yields interpretable rate ratios by interval.

Piecewise Exponential via Poisson Regression

Splitting time and fitting model

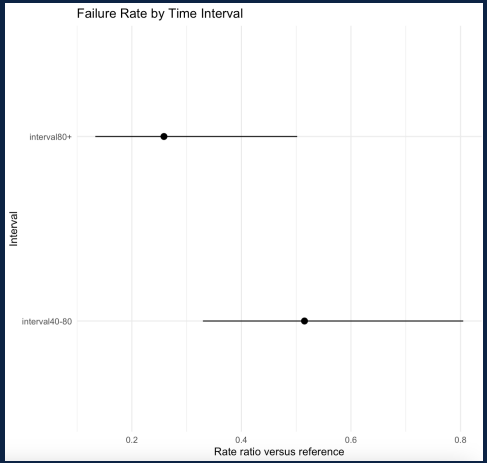
```
## Piecewise Exponential Model
cuts <- c(0, 40, 80, Inf)
datSPLIT <- survSplit(Surv(time, status) ~., 
  data = dat, cut = cuts[-c(1, length(cuts))],
  episode = "interval")
## interval factor labels
datSPLIT$interval <- factor(datSPLIT$interval,
                  labels = c("0-40", "40-80", "80+"))
## poisson regression on event counts
## with log(time) offset
poissonWeightedGLM <- glm(status ~ interval+temp+
           load+material+maint, family = poisson(),
           offset = log(time), data = datSPLIT)
summary(poissonWeightedGLM)
## coefficients conversion
pweRR <- broom::tidy(poissonWeightedGLM) %>%
  mutate(rate_ratio = exp(estimate),
    lower = exp(estimate-1.96*std.error),
    upper = exp(estimate+1.96*std.error))
pweRR
```

```
Call:
glm(formula = status ~ interval + temp + load + material + maint, 
    family = poisson(), data = datSPLIT, offset = log(time))

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)    -7.14409    0.94012  -7.599 2.98e-14 ***
interval40-80 -0.66334    0.22773  -2.913  0.00358 **
interval80+   -1.35254    0.33826  -3.999 6.37e-05 ***
temp          -0.01633    0.02688  -0.608  0.54343
load           1.85783    0.64821   2.866  0.00416 **
materialB     -0.38419    0.21403  -1.795  0.07264 .
maint1        -0.69620    0.22498  -3.094  0.00197 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 580.97  on 1158  degrees of freedom
Residual deviance: 536.12  on 1152  degrees of freedom
AIC: 728.12

Number of Fisher Scoring iterations: 6

> pweRR
# A tibble: 7 × 8
  term          estimate std.error statistic  p.value rate_ratio    lower    upper
  <chr>            <dbl>     <dbl>     <dbl>    <dbl>      <dbl>    <dbl>    <dbl>
1 (Intercept)    -7.14      0.940     -7.60  2.98e-14  0.000790 0.000125  0.00498
2 interval40-80  -0.663     0.228     -2.91  3.58e- 3  0.515    0.330     0.805
3 interval80+    -1.35      0.338     -4.00  6.37e- 5  0.259    0.133     0.502
4 temp           -0.0163    0.0269    -0.608 5.43e- 1  0.984    0.933     1.04
5 load            1.86      0.648      2.87  4.16e- 3  6.41     1.80     22.8
6 materialB      -0.384     0.214     -1.80  7.26e- 2  0.681    0.448     1.04
7 maint1         -0.696     0.225     -3.09  1.97e- 3  0.498    0.321     0.775
```

Piecewise Exponential via Poisson Regression

Splitting time and fitting model

```
## interval-specific baseline rates
intervalEFFECTS <- pweRR %>%
  filter(grepl("^interval", term))

ggplot(intervalEFFECTS,
       aes(x = term, y = rate_ratio,
       ymin = lower, ymax = upper)) +
  geom_pointrange() + coord_flip() +
  theme_minimal() +
  labs(
    x = "Interval",
    y = "Rate ratio versus reference",
    title = "Failure Rate by Time Interval"
  )
```



Comparing Models: Fit and Prediction

Which model is best?

Compare...

- AIC: penalized likelihood; lower is better (but scale differs across model classes)
- Harrell's C-index: rank-based discrimination; 0.5 = random, 1.0 = perfect
- Prediction error / Brier score: absolute prediction error over time

Notes...

- Compare Cox (with/without frailty) to Weibull and lognormal AFT
- Use `pec` to obtain prediction error curves for Cox-type models

AIC, C-index, and Prediction Error

Model comparison code

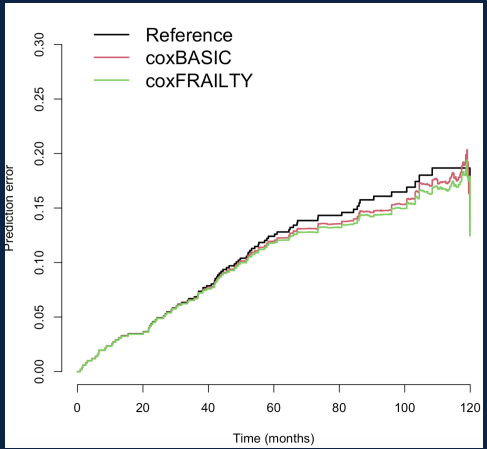
```
## Which model predicts best?
AIC(coxBASIC, coxFRAILTY, weibullAFT, lognormalAFT)

## Predictive accuracy with Harrell's C-index
## C-index measures rank-order predictive accuracy
cINDEX <- function(fit, data) {
  s <- Surv(data$time, data$status)
  lp <- predict(fit, newdata = data, type = "lp")
  conc <- concordance(s ~ lp)$concordance
  return(conc)
}
c_basic   <- cINDEX(coxBASIC,   dat)
c_frail   <- cINDEX(coxFRAILTY, dat)
c_weibull <- cINDEX(weibullAFT,    dat)
c_logn    <- cINDEX(lognormalAFT,    dat)

data.frame(
  model = c("Cox (no frailty)",
  "Cox (frailty)", "Weibull AFT", "Lognormal AFT"),
  cINDEX = c(c_basic, c_frail, c_weibull, c_logn)
)
```

```
                    df      AIC
coxBASIC      4.000000 1033.967
coxFRAILTY    5.304768 1028.984
weibullAFT    6.000000 1235.151
lognormalAFT 6.000000 1240.910
> data.frame(
+   model = c("Cox (no frailty)
+   cINDEX = c(c_basic, c_frail
+ )
              model    cINDEX
1 Cox (no frailty) 0.3621266
2    Cox (frailty) 0.3423481
3      Weibull AFT 0.6376427
4    Lognormal AFT 0.6407565
```

AIC, C-index, and Prediction Error

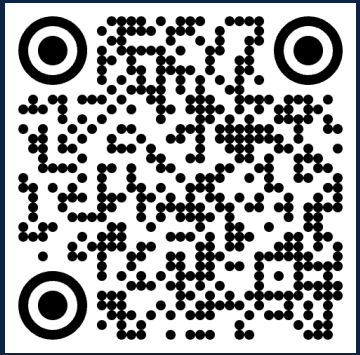
Model comparison code

```r
## Brier score absolute prediction error at time t.
## still figuring out how to do this
## for the other two models
models <- list(
  coxBASIC   = coxBASIC,
  coxFRAILTY = coxFRAILTY
)
## prediction error curves
pecFIT <- pec(
  object  = models,
  formula = Surv(time, status) ~ 1,
  data    = dat,
  times   = quantile(dat$time,
        probs = seq(0.1, 0.9, by = 0.1)))

plot(pecFIT,
     xlab = "Time (months)",
     ylab = "Prediction error",
     legend = TRUE,
     main = "Prediction Error Curves")
```



What's Next?

Additional Questions?
Book an Appointment!



Next Workshop:

Nonparametric Models
–> December 2, 11:00 AM

Thank You!

Questions?

Workshop Materials:
`https://github.com/csc-ubc-okanagan/ubco-csc-modeling-workshop`

Contact:
jesse.ghashti@ubc.ca