PH 716 Applied Survival Analysis

Part III: Comparing survival curves

Zhiyang Zhou (zhou67@uwm.edu, zhiyanggeezhou.github.io)

2024/Feb/07 11:08:59

Recall Ex. 2.2

```
data.ex22 = survival::pbc[complete.cases(survival::pbc[,1:4]), 1:4]
data.ex22$status = 1*(data.ex22$status %in% c(1,2)) # merging status 1 and 2
survminer::ggsurvplot(
  survival::survfit(survival::Surv(time, status)~trt, data=data.ex22, conf.type="log-log"),
  xlab="Time",
  conf.int = T,
  conf.int.style="step",
  censor=F,
  risk.table = F,
  cumevents = F,
  tables.height = 0.15
)
```

Recall the hypothesis testing (from the perspective of binary classification)

- Make a decision between the null hypothesis H_0 and the alternative one H_1
- Potential outcomes
  - True positive (TP) = H_0 correctly rejected
  - False positive (FP, i.e., type I error) = H_0 incorrectly rejected
  - True negative (TN) = H_0 is correctly accepted
  - False negative (FN, i.e., type II error) = H_0 incorrectly accepted
  - E.g., H_0 : healthy vs H_1 : sick
    - TP: sick people identified as sick
    - FP: healthy people identified as sick
    - TN: healthy people identified as healthy
    - FN: sick people identified as healthy

	Accept H_0	Reject H_0
H_0 is true	True negative (TN)	False positive (FP, i.e., type I error)
H_0 is false	False negative (FN, i.e., type II error)	True positive (TP)

- Evaluating the error rate
  - Misclassification rate = $\Pr(\text{FP}) + \Pr(\text{FN})$
  - False discovery rate (FDR) = $\Pr(\text{FP})/\{\Pr(\text{FP}) + \Pr(\text{TP})\}$
    - controlling for sequential/simultaneous testing

  - True positive rate (TPR, i.e., sensitivity) = $\Pr(\text{TP})/\{\Pr(\text{TP}) + \Pr(\text{FN})\}$
  - False positive rate (FPR) = $\Pr(\text{FP})/\{\Pr(\text{FP}) + \Pr(\text{FN})\}$
  - Receiver operating characteristic curve (ROC curve): plot of TPR vs FPR
    - Area under the ROC curve (AUC)
  - True negative rate (TNR, i.e., specificity) = $\Pr(\text{TN})/\{\Pr(\text{TN}) + \Pr(\text{FP})\}$
- The (optimal) hypothesis testing is a strategy minimizing Pr(FN) subject to capped Pr(FP), i.e.,

$$\text{minimize}\, \Pr(\text{type II error}) \qquad \text{subject to}\ \Pr(\text{type I error}) \leq \alpha$$

  - α is the significance level

Assumptions

- The censoring is noninformative
- All the subjects are independent from each other
- Subjects in group k share the identical hazard rate $\lambda_k(t)$

Hypotheses to be tested

- Null hypothesis $H_0 : \lambda_1(t) = \lambda_2(t) = \lambda(t)$ for all t
- Alternative hypothesis H_1 could be:
  - One-sided $H_1 : \lambda_1(t) \geq \lambda_2(t)$ for all t and $\lambda_1(t) > \lambda_2(t)$ for some t
  - One-sided $H_1 : \lambda_1(t) \leq \lambda_2(t)$ for all t and $\lambda_1(t) < \lambda_2(t)$ for some t
  - Two-sided $H_1 : \lambda_1(t) \neq \lambda_2(t)$ for some t

Two-sample log-rank test

- Distinct observed event times across the POOLED sample are $t_1 < \cdots < t_{n_D}$
  - At time t_j , there are d_{kj} events in group k, $k = 1, 2$, and $d_j = d_{1j} + d_{2j}$
  - Just prior to t_j, there are r_{kj} at risk in group k and $r_j = r_{1j} + r_{2j}$
- Test statistic
  - $U_k/\sqrt{V} \approx N(0, 1)$ under H_0, $k = 1, 2$
    - $U_k = \sum_{j=1}^{n_D} r_{kj}(d_{kj}/r_{kj} - d_j/r_j) = r_{kj}\{\hat{\lambda}_1(t_j) - \hat{\lambda}(t_j)\}$
      - $\hat{\lambda}_1(t_j)$: estimated hazard rate at t_j for group k
      - $\hat{\lambda}(t_j)$: estimated hazard rate at t_j for pooled population
      - $d_{kj} = r_{kj}\hat{\lambda}_1(t_j)$: observed number of events from sample k at time t_j
      - $r_{kj}\hat{\lambda}(t_j)$: expected number of events from sample k at time t_j under H_0
    - $V = \text{var}(U_k) = \sum_{j=1}^{n_D} \frac{d_j r_{1j} r_{2j}(r_j - d_j)}{r_j^2(r_j - 1)}$
    - $U_1 = U_2$
  - The log-rank test is rank-based; one could construct the test statistic using only the order of observed event times alone.
- Rejection region
  - 2-sided: $|U_k/\sqrt{V}| > z_{1-\alpha/2}$ or equiv. $U_k^2/V > \chi^2_{1,1-\alpha}$
    - $z_{1-\alpha/2}$ is the $1 - \alpha/2$ quantile of $N(0, 1)$
    - $\chi^2_{1,1-\alpha}$ is the $1 - \alpha$ quantile of $\chi^2(1)$
  - 1-sided ($H_1 : \lambda_1(t) \geq \lambda_2(t)$ for all t and $\lambda_1(t) > \lambda_2(t)$ for some t): $U_1/\sqrt{V} > z_{1-\alpha}$
  - 1-sided ($H_1 : \lambda_1(t) \leq \lambda_2(t)$ for all t and $\lambda_1(t) < \lambda_2(t)$ for some t): $-U_1/\sqrt{V} > z_{1-\alpha}$
- p-value
  - 2-sided: $p = 2\{1 - \Phi(|U_k/\sqrt{V}|)\}$
    - $\Phi(\cdot)$ is the cdf of $N(0, 1)$
  - 1-sided ($H_1 : \lambda_1(t) \geq \lambda_2(t)$ for all t and $\lambda_1(t) > \lambda_2(t)$ for some t): $p = \{1 - \Phi(U_1/\sqrt{V})\}$
  - 1-sided ($H_1 : \lambda_1(t) \leq \lambda_2(t)$ for all t and $\lambda_1(t) < \lambda_2(t)$ for some t): $p = \{1 - \Phi(-U_1/\sqrt{V})\}$

Ex. 3.1. Revisit the PBC data

```
data.ex22 = survival::pbc[complete.cases(survival::pbc[,1:4]), 1:4]
data.ex22$status = 1*(data.ex22$status %in% c(1,2)) # merging status 1 and 2
# For 2-sided only
survival::survdiff(
  formula = survival::Surv(time, status)~trt, data=data.ex22
)
survminer::surv_pvalue(
  fit = survival::survfit(formula = survival::Surv(time, status)~trt, data=data.ex22),
  method = 'log-rank'
)
# For 2-sided or 1-sided
nph::logrank.test(
  time = data.ex22$time,
  event = data.ex22$status,
  group = data.ex22$trt,
  alternative = 'two.sided' # 'two.sided','less','greater'
)$test
```

- Demo report of testing results (covering necessary components: hypotheses, test name, p-value/rejection region, significance level, and conclusion):
  - "Testing hypotheses H_0 : ___ vs. H_1 : ___ , we carried on the ___ test."
    - "The p-value is ___ . So, at the ___ level, there was/wasn't a strong statistical evidence against H_0, i.e., we believed that ___ ."
    - OR "The value of test statistic is $T =$ ___ . Given the level ___ rejection region $T >$ ___ , there was/wasn't a strong statistical evidence against H_0, i.e., we believed that ___ ."

Testing multiple (>2) survival curves

- Hypotheses to be tested
  - Null hypothesis $H_0 : \lambda_1(t) = \cdots = \lambda_K(t) = \lambda(t)$ for all t
  - Alternative hypothesis $H_1 : \lambda_{k_1}(t) \neq \lambda_{k_2}(t)$ for certain t and certain 2-tuple (k_1, k_2)
- Ex. 3.2. (Bladder Cancer Recurrences) A dataset on recurrences of bladder cancer. It contains three treatment arms for 118 subjects.

```
data.ex32 = survival::bladder1[
  complete.cases(survival::bladder1[,c('id', 'treatment', 'start', 'stop', 'status')]),
  c('id', 'treatment', 'start', 'stop', 'status')
]
data.ex32$status = 1*(data.ex32$status %in% c(1,2,3)) # merging status 1, 2,3
data.ex32$time = data.ex32$stop - data.ex32$start
# For 2-sided only
survival::survdiff(
  formula = survival::Surv(time, status)~treatment, data=data.ex32
)
survminer::surv_pvalue(
  fit = survival::survfit(formula = survival::Surv(time, status)~treatment, data=data.ex32),
  method = 'log-rank'
)

survminer::surv_pvalue(
  fit = survival::survfit(formula = survival::Surv(time, status)~treatment, data=data.ex32),
  method = 'log-rank',
  test.for.trend = T
```

```
)
```

Testing for trend

- Hypotheses to be tested
  - Null hypothesis $H_0 : \lambda_1(t) = \cdots = \lambda_K(t) = \lambda(t)$ for all t, $K > 2$
  - Alternative hypothesis $H_1 : \lambda_1(t) \geq \cdots \geq \lambda_K(t)$ or $\lambda_1(t) \leq \cdots \leq \lambda_K(t)$, with at least one strict inequality
- Ex. 3.3. Revisit the data of bladder cancer recurrences

```
data.ex33 = survival::bladder1[
  complete.cases(survival::bladder1[,c('id', 'treatment', 'start', 'stop', 'status')]),
  c('id', 'treatment', 'start', 'stop', 'status')
]
data.ex33$status = 1*(data.ex33$status %in% c(1,2,3)) # merging status 1, 2,3
data.ex33$time = data.ex33$stop - data.ex33$start
data.ex33$treatment = factor(data.ex33$treatment, levels = c("placebo","pyridoxine","thiotepa"))
survminer::surv_pvalue(
  fit = survival::survfit(formula = survival::Surv(time, status)~treatment, data=data.ex33),
  method = 'log-rank',
  test.for.trend = T
)
# Change the order of treatments
data.ex33$treatment = factor(data.ex33$treatment, levels = c("placebo","thiotepa","pyridoxine"))
survminer::surv_pvalue(
  fit = survival::survfit(survival::Surv(time, status)~treatment, data=data.ex33),
  method = 'log-rank',
  test.for.trend = T
)
```