Messina Experiment 1: Classical Survival Approaches

March 8, 2015

1 Preparation

```
library(messina)

## Loading required package:  survival
## Loading required package:  splines
## Loading required package:  methods

library(plyr)
library(reshape2)
library(ggplot2)

library(doMC)

## Loading required package:  foreach
## Loading required package:  iterators
## Loading required package:  parallel

registerDoMC(32)
```

2 Accessory Functions

```
calcCensorWeibullRate = function(surv.shape, surv.scale, cens.shape, cens.scale, plot = FALSE)
{
        if (plot)
        {
                x = seq(0, 10*min(surv.scale, cens.scale), length.out = 1e4)
                y = dweibull(x, shape = cens.shape, scale = cens.scale) * (1 - pweibull(x, shape = surv.
                plot(y ~ x, type = "l")
        }
        integrate(function(t) dweibull(t, shape = cens.shape, scale = cens.scale) * (1 - pweibull(t, sha
}
# calcCensorWeibullRate(1/0.7253, exp(6.9), 1/0.3216, exp(7.1))
# temp.ws = rweibull(1e6, 1/0.7253, exp(6.9))
# temp.wc = rweibull(1e6, 1/0.3216, exp(7.1))
# mean(temp.wc < temp.ws)
# AW FUK YEH
# calcCensorWeibullRate(1.379,601.8,3.109,10188,plot = TRUE)

# Only approximate due to instability at endpoints, but who's interested in 95% censoring anyway?
```

```
calcCensorWeibullScale = function(surv.shape, surv.scale, cens.shape, cens.target_rate)
{
        exp(optimize(function(log.cens.scale) abs(calcCensorWeibullRate(surv.shape, surv.scale, cens.sha
}


calcCensorWeibullScaleTwoGroup = function(surv.shape, surv.scale.0, surv.scale.1, group.fraction.0, cens
{
        exp(optimize(function(log.cens.scale)
                abs(
                        group.fraction.0*calcCensorWeibullRate(surv.shape, surv.scale.0, cens.shape, exp
                        (1 - group.fraction.0)*calcCensorWeibullRate(surv.shape, surv.scale.1, cens.shap
                        cens.target_rate), interval = log(c(1/50, 50)*range(c(surv.scale.0, surv.scale.1
}


fastCoxCoef = function(x, y)
{
        require(messina)
        sort_perm = order(y[,1])
        time = y[sort_perm,1]
        event = y[sort_perm,2]
        x = x[sort_perm]
        zstat = messina:::messinaSurvLRT(as.logical(x), time, event)
        zstat * sqrt(4/sum(event))
}


buildDetResult = function(x, y, detresult)
{
        require(messina)
        xc = x > detresult[2,] # = cutoffs
        coefs = apply(xc, 1, function(xc1) fastCoxCoef(xc1, y))
        cbind(det = detresult[1,], coefs = coefs)
}
```

3 Data generator

```
generator1 = function(delta_x, noise_sd, surv_dists, censor_dist, balance, n)
{
        n1 = round(balance * n)
        n2 = n - n1
        true_class = rep(c(0, 1), c(n1, n2))

        x = true_class*delta_x + rnorm(n, sd = noise_sd)

        time_event = rep(NA, n)
        time_event[true_class == 0] = surv_dists[["0"]](n1)
        time_event[true_class == 1] = surv_dists[["1"]](n2)
        time_cens = censor_dist(n)
```

```
        time_observed = pmin(time_event, time_cens)
        event_observed = time_event <= time_cens

        y = Surv(time_observed, event_observed)

        x = matrix(x, nrow = 1, ncol = length(x))
        result = list(x = x, y = y, c = true_class)
}
```

4 Detectors

```
# For ncuts = 1, this equates to median cut.
detector_multicut = function(x, y, p = 0.05, ncuts = 10, correct = "none")
{
        if (ncuts == 1) { correct = "none" }
        buildDetResult(x, y, apply(x, 1, function(x1) {
                cutpoints = quantile(x1, probs = (1:ncuts)/(ncuts + 1))
                pvals = sapply(cutpoints, function(c) {
                        x1c = x1 > c
                        test = survdiff(y ~ x1c)
                        pval = pchisq(test$chisq, df = 1, lower.tail = FALSE)
                        pval
                })
                pvals = p.adjust(pvals, correct)
                pvals[is.na(pvals)] = 1
                c(min(pvals) < p, cutpoints[which.min(pvals)])
        }))
}


# A 'best-approach' to all-cutoff testing
detector_maxstat = function(x, y, p = 0.05, pmethod = "HL")
{
        require(maxstat)

        buildDetResult(x, y, apply(x, 1, function(x1) {
                temp.data = data.frame(x1 = x1, time = y[,1], event = y[,2])
                test = try(maxstat.test(Surv(time, event) ~ x1, data = temp.data, smethod = "LogRank", p
                if (class(test) == "try-error")
                {
                        return(c(NA, NA))
                }
                c(test$p.value < p, test$estimate[[1]])
        }))
}
```

5 Experiment function

```
exp1.singlerun = function(ed)
{
        message(ed[["RunIndex"]], "/", ed[["RunTotal"]])
        require(plyr)
        surv_scale_0 = exp(6.9)/sqrt(exp(ed[["log.hazard.ratio"]]))
        surv_scale_1 = exp(6.9)*sqrt(exp(ed[["log.hazard.ratio"]]))
        surv_shape = 1/0.7253
        cens_shape = 1/0.3216
        censor_scale = calcCensorWeibullScaleTwoGroup(surv_shape, surv_scale_0, surv_scale_1, ed[["class

        data = lapply(1:ed[["replicates"]], function(i) generator1(
                delta_x = ed[["delta.x"]],
                noise_sd = ed[["noise.sd"]],
                surv_dists = list(
                        "0" = function(n) rweibull(n, shape = surv_shape, scale = surv_scale_0),
                        "1" = function(n) rweibull(n, shape = surv_shape, scale = surv_scale_1)),
                censor_dist = function(n) rweibull(n, shape = cens_shape, scale = censor_scale),
                balance = ed[["class.1.fraction"]],
                n = ed[["cohort.size"]])))
        detector_results = laply(data, function(data1) exp1.detectors[[ed[["detector"]]]](data1$x, data1

        detected = detector_results[,1]
        effectsizes = detector_results[,2]

        failed = is.na(detected)
        effectsizes[failed] = NA
        detected[is.na(detected)] = FALSE

        c("detrate" = mean(detected), "failrate" = mean(failed), "effectsize" = median(effectsizes, na.r
}
```

6 The Experiment

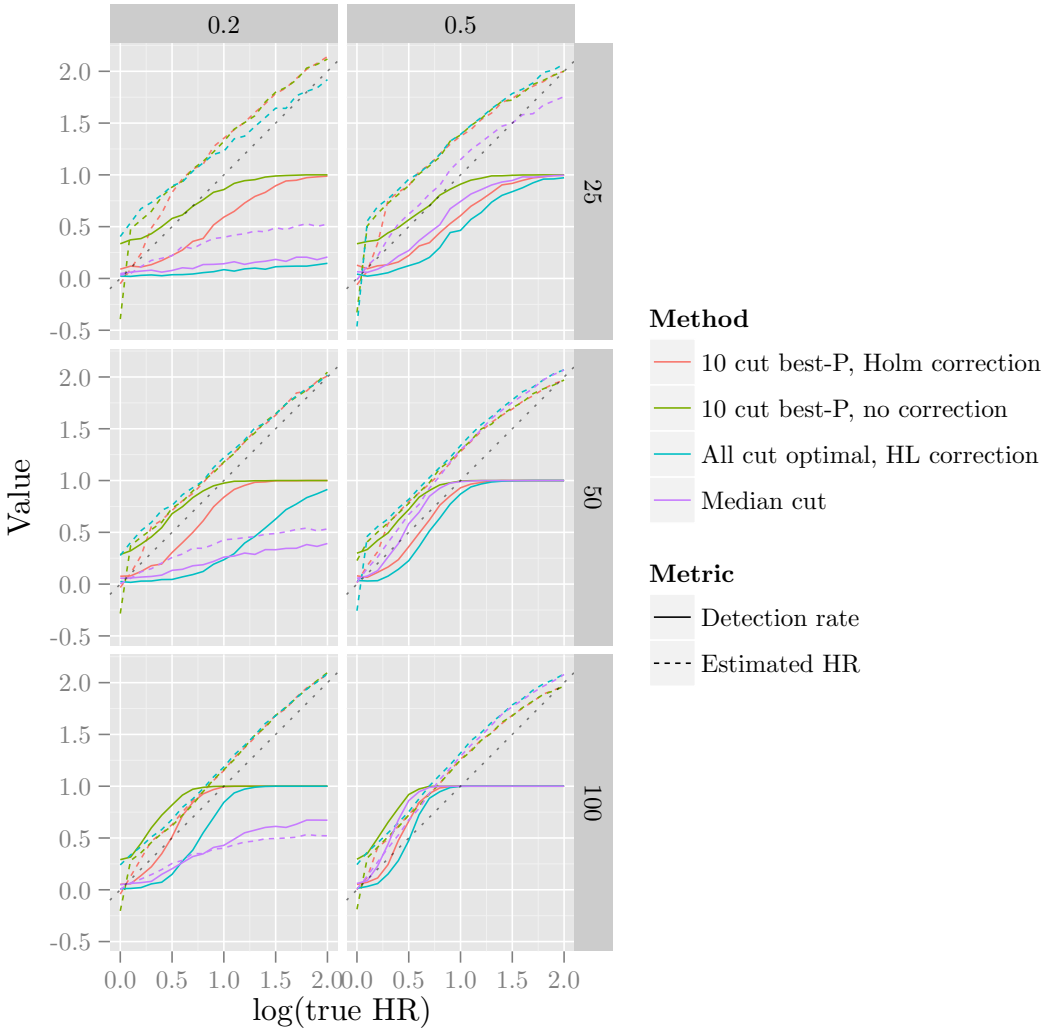
```
exp1.detectors = list(
        "1-cut" = function(x, y) detector_multicut(x, y, p = 0.05, ncuts = 1, correct = "none"),
        "10-cutNone" = function(x, y) detector_multicut(x, y, p = 0.05, ncuts = 10, correct = "none"),
        "10-cutHolm" = function(x, y) detector_multicut(x, y, p = 0.05, ncuts = 10, correct = "holm"),
        "all-cutHL" = function(x, y) detector_maxstat(x, y, p = 0.05, pmethod = "HL"))

exp1.design = expand.grid(
        delta.x = 5,
        noise.sd = 1,
        # class.1.fraction = c(0.2, 0.5),
        class.1.fraction = c(0.2, 0.5, 0.8),
        detector = c("1-cut", "10-cutNone", "10-cutHolm", "all-cutHL"),
        # cohort.size = c(25, 50, 100, 200),
        cohort.size = c(25, 50, 100),
        log.hazard.ratio = seq(0, 2, 0.1),
        censoring.rate = c(0.2, 0.5, 0.8),
        replicates = 1000)
```

```
set.seed(20150306)
exp1.measurements = laply(1:nrow(exp1.design), function(i) exp1.singlerun(exp1.design[i,,drop=FALSE]),
```

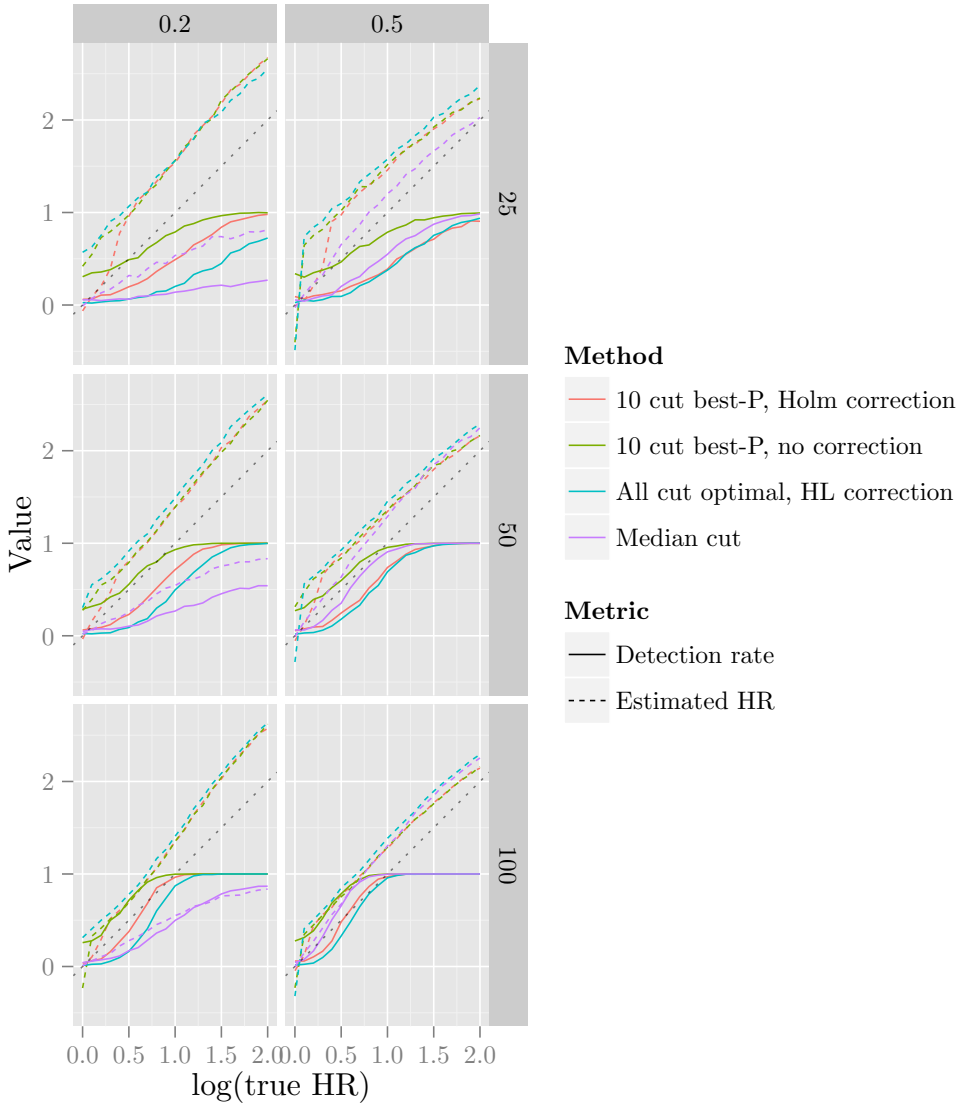
```
exp1.result = cbind(exp1.design, exp1.measurements)
```

```
save.image("02_surv_exp1.rda")
```

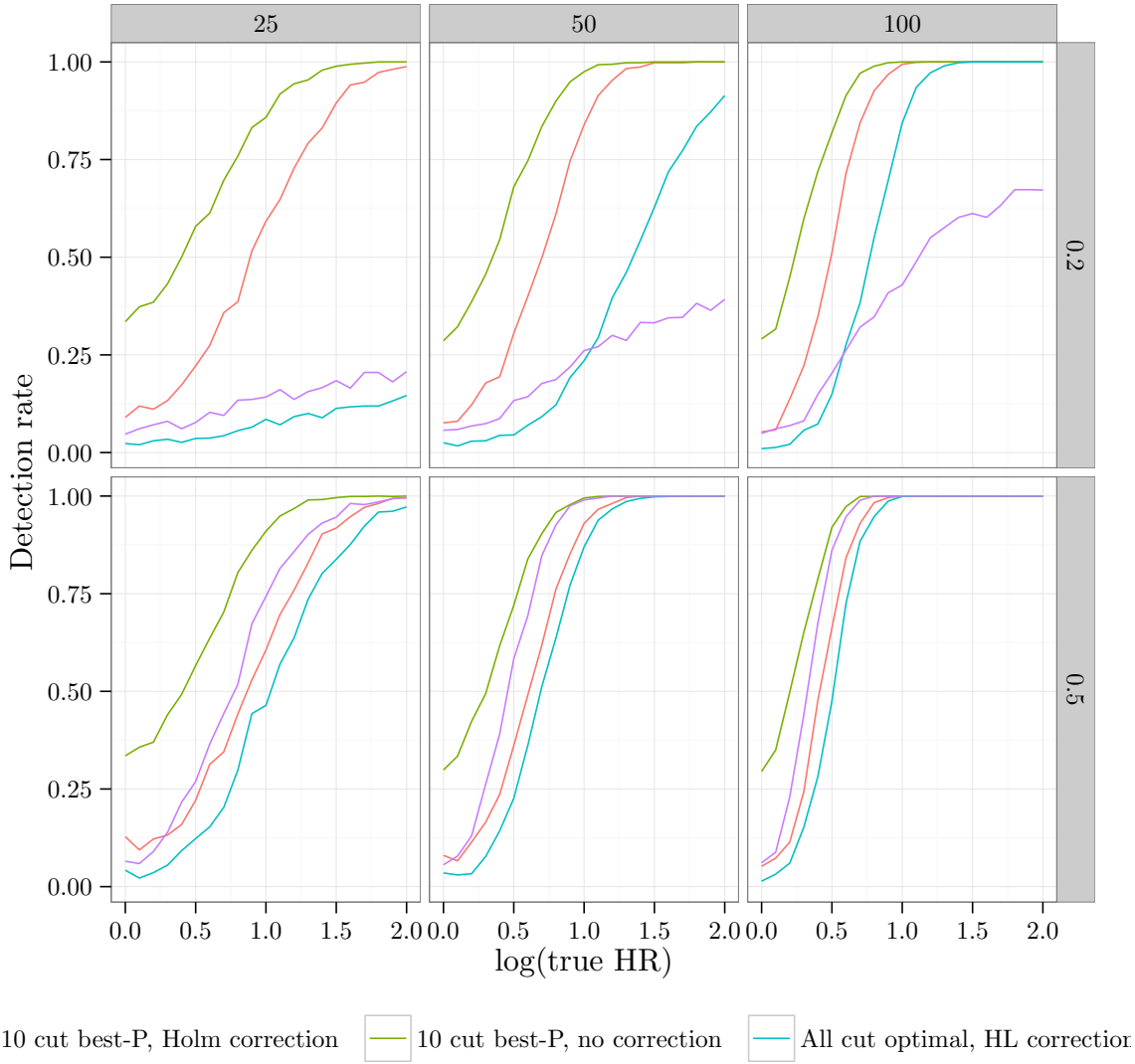
```
library(reshape)
exp1.result.melted = melt(exp1.result, measure.vars = c("detrate", "failrate", "effectsize", "censrate")

exp1.result.melted = exp1.result.melted[exp1.result.melted$class.1.fraction %in% c(0.2, 0.5),]
exp1.result.melted = exp1.result.melted[exp1.result.melted$censoring.rate %in% c(0.2, 0.5),]
exp1.result.melted$detector = c("1-cut" = "Median cut", "10-cutNone" = "10 cut best-P, no correction", "
exp1.result.melted$variable = c("detrate" = "Detection rate", "effectsize" = "Estimated HR", "failrate"
exp1.result.melted$detector = as.factor(exp1.result.melted$detector)
exp1.result.melted$variable = as.factor(exp1.result.melted$variable)
colnames(exp1.result.melted)[colnames(exp1.result.melted) == "detector"] = "Method"
colnames(exp1.result.melted)[colnames(exp1.result.melted) == "variable"] = "Metric"

ggplot(exp1.result.melted[exp1.result.melted$censoring.rate == 0.2 & !(exp1.result.melted$Metric %in% c
        geom_line() +
        facet_grid(cohort.size ~ class.1.fraction) +
        geom_abline(intercept = 0, slope = 1, alpha = 0.5, linetype = "dotted") +
        coord_fixed() +
        labs(x = "log(true HR)", y = "Value")
```



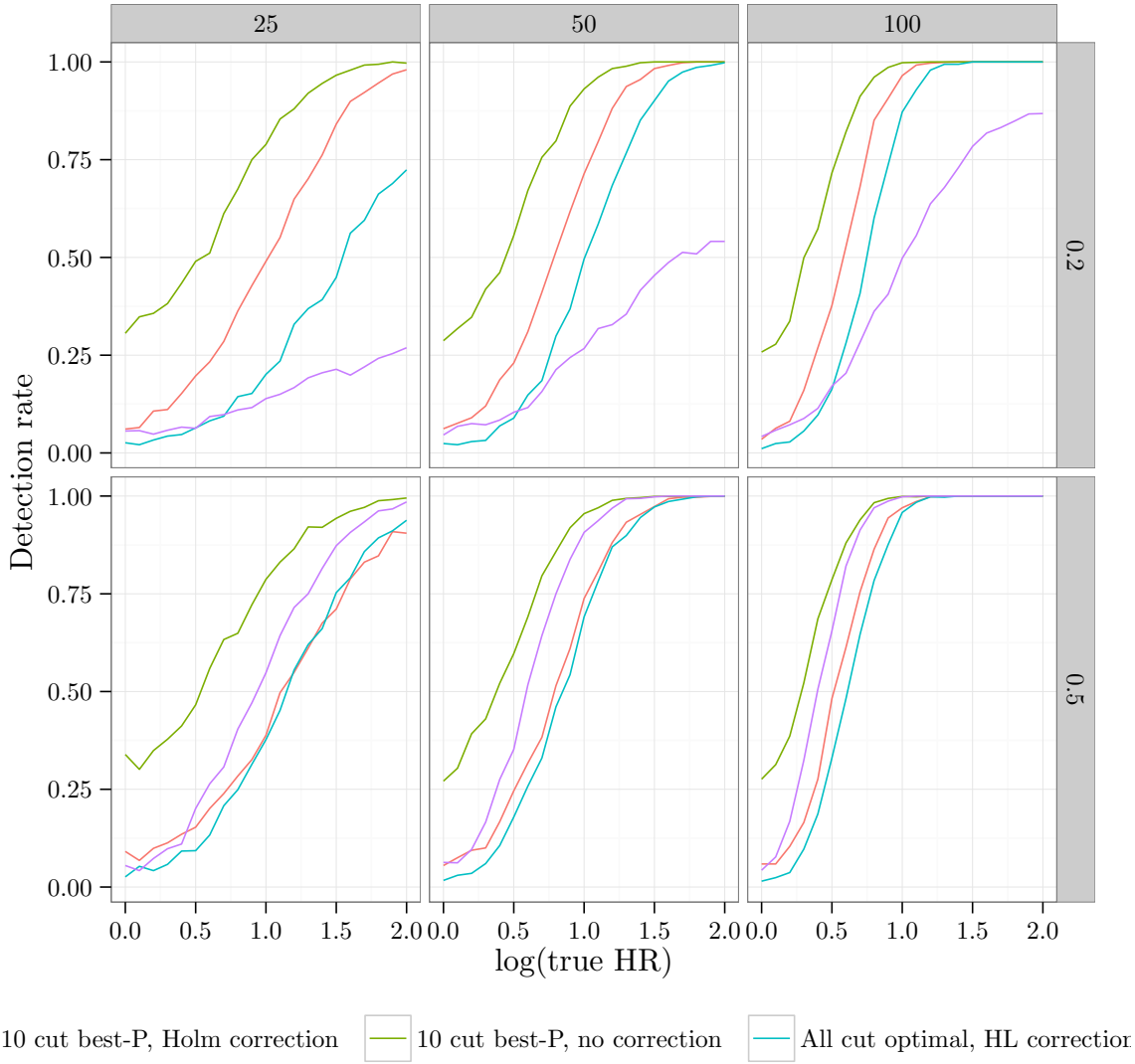

```
ggplot(exp1.result.melted[exp1.result.melted$censoring.rate == 0.5 & !(exp1.result.melted$Metric %in% c(
        geom_line() +
        facet_grid(cohort.size ~ class.1.fraction) +
        geom_abline(intercept = 0, slope = 1, alpha = 0.5, linetype = "dotted") +
        coord_fixed() +
        labs(x = "log(true HR)", y = "Value")
```



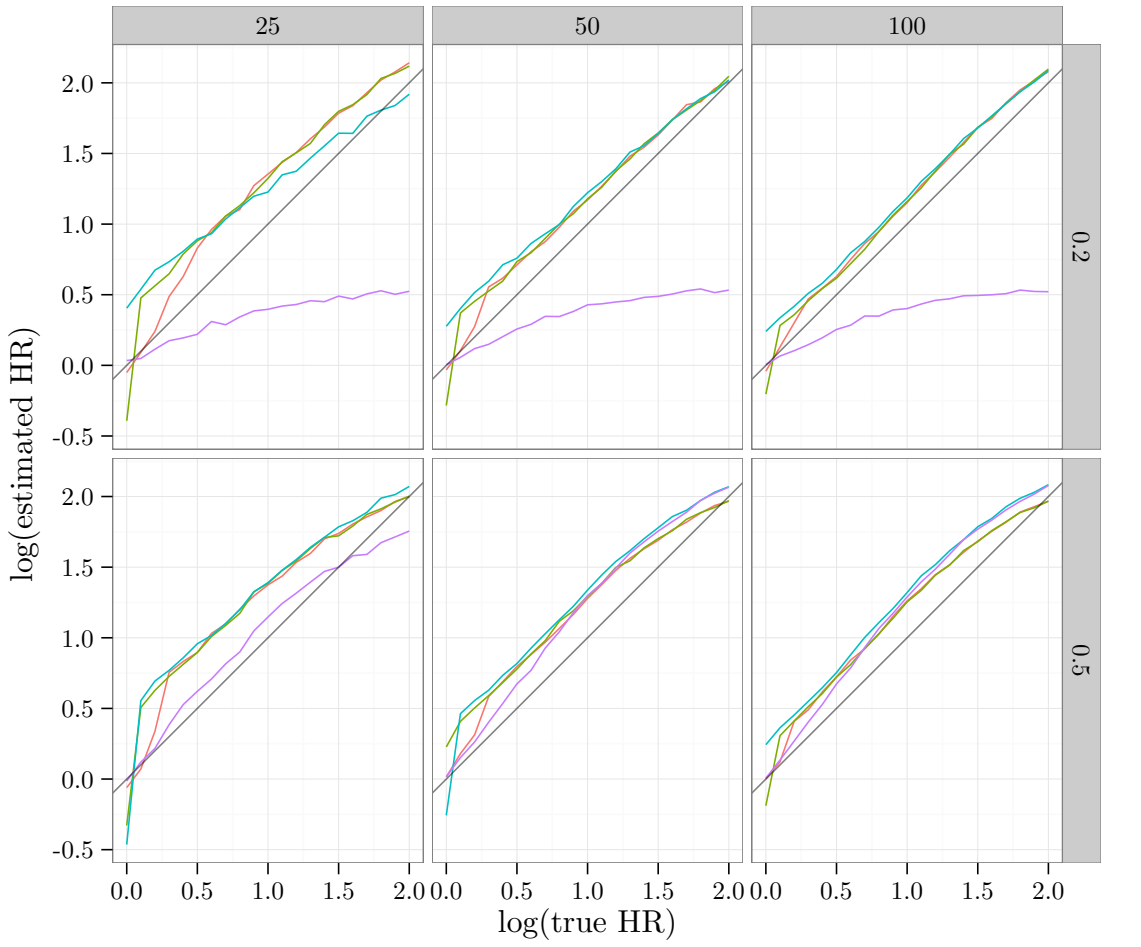

```
ggplot(exp1.result.melted[exp1.result.melted$censoring.rate == 0.2 & exp1.result.melted$Metric == "Detec
        geom_line() +
        facet_grid(class.1.fraction ~ cohort.size) +
        labs(x = "log(true HR)", y = "Detection rate") +
        theme_bw() + theme(legend.position = "bottom")
```



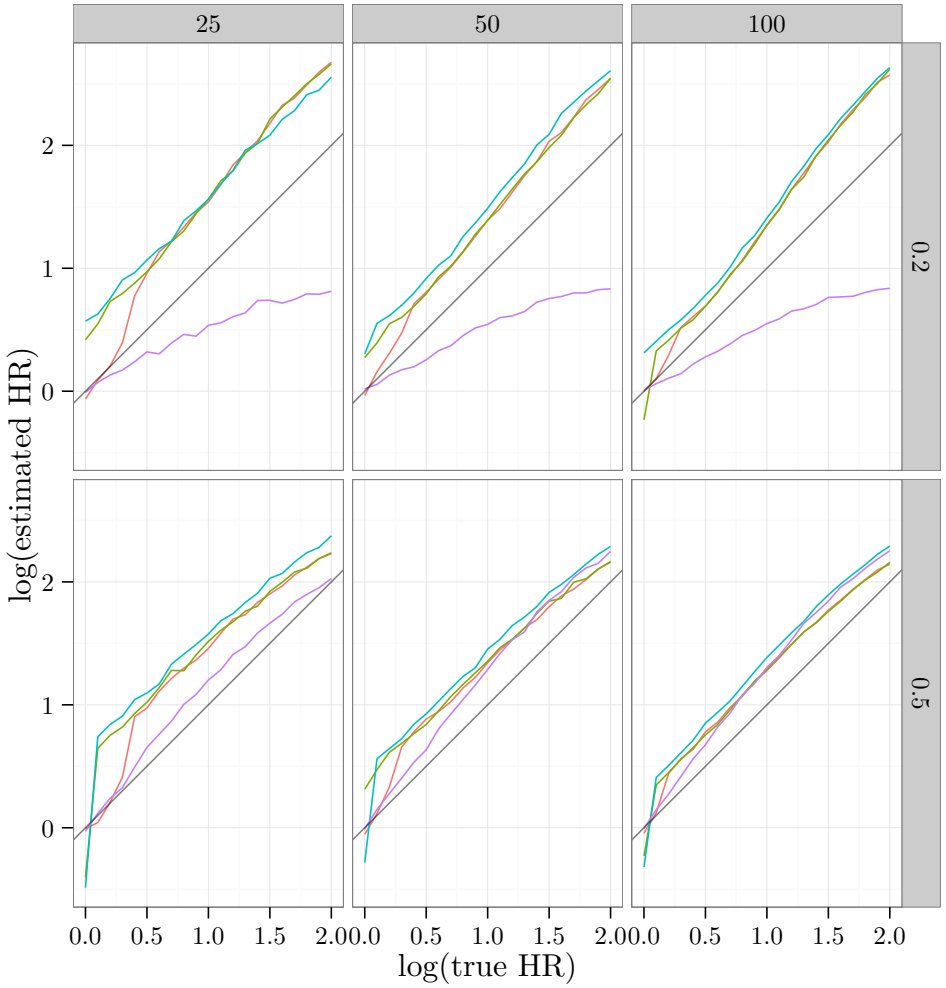

```
ggplot(exp1.result.melted[exp1.result.melted$censoring.rate == 0.5 & exp1.result.melted$Metric == "Detec
        geom_line() +
        facet_grid(class.1.fraction ~ cohort.size) +
        labs(x = "log(true HR)", y = "Detection rate") +
        theme_bw() + theme(legend.position = "bottom")
```




```
ggplot(exp1.result.melted[exp1.result.melted$censoring.rate == 0.2 & exp1.result.melted$Metric == "Estim
        geom_line() +
        facet_grid(class.1.fraction ~ cohort.size) +
        geom_abline(intercept = 0, slope = 1, alpha = 0.5) +
        coord_fixed() +
        labs(x = "log(true HR)", y = "log(estimated HR)") +
        theme_bw() + theme(legend.position = "bottom")
```




```
ggplot(exp1.result.melted[exp1.result.melted$censoring.rate == 0.5 & exp1.result.melted$Metric == "Estim
        geom_line() +
        facet_grid(class.1.fraction ~ cohort.size) +
        geom_abline(intercept = 0, slope = 1, alpha = 0.5) +
        coord_fixed() +
        labs(x = "log(true HR)", y = "log(estimated HR)") +
        theme_bw() + theme(legend.position = "bottom")
```



10 cut best-P, Holm correction | 10 cut best-P, no correction | All cut optimal, HL correction