Technical report for "Conformal prediction for exponential families and generalized linear models"

Daniel J. Eck

September 4, 2019

Abstract

In this supplementary materials document we provide all of the code to reproduce the analyses which appear in the paper Eck et al. [2019] and the R package Eck [2018]. Several additional analyses are provided which illustrate the advantages of conformal prediction regions. In particular, we provide evidence that the parametric conformal prediction region, developed in Eck et al. [2019], compares favorably to other prediction regions even when the parametric model is misspecified.

Contents

The following R packages are required in order to replicate the calculations within this document.

```r
library(parallel)
library(MASS)
library(statmod)
library(HDInterval)
library(conformal.glm) ## https://github.com/DEck13/conformal.glm
library(conformalInference) ## https://github.com/ryantibs/conformal
library(faraway)
library(geometry)
library(xtable)
library(rgl)
```

We set the error tolerance to be $\alpha = 0.10$ for all prediction regions unless otherwise noted.

1 Introduction/Summary of simulation results

This manuscript (and corresponding .Rnw file) provides all of the code to reproduce the analyses which appear in the paper Eck et al. [2019], the README file in the R package Eck [2018], and this document. Several additional analyses to those presented in Eck et al. [2019] are provided in this document. Of particular intherest, we investigate the performance of parametric conformal prediction regions under model misspecification. Specifically, we focus on settings where the underlying data is generated via a Gamma distribution and parametric prediction regions are obtained using a cubic regression model assuming homoscedastic normal errors. The cubic fit is chosen because it is intuitive and it fits the Gamma data better than a simple linear regression model or a quadratic model.

Our goal in this manuscript is to demonstrate the advantages and disadvantages of our binned and transformation based parametric conformal prediction regions [Eck et al., 2019] compared with the nonparametric conformal prediction region [Lei and Wasserman, 2014], the least squares (LS) conformal prediction region [Lei et al., 2018] obtained from conformalized residual scores, the least squares locally weighted (LSLW) conformal prediction region [Lei et al., 2018, Section 5.2] obtained from conformalized locally weighted residual scores, and the highest density (HD) region. In analyses with model misspecification, the parametric, LS, and LSLW conformal prediction regions and HD prediction region are constructed under the misspecified model. The binning used to construct the parametric and nonparametric conformal prediction regions follows the bin width asymptotics of Lei and Wasserman [2014].

We find that the binned parametric conformal prediction region performs well even when the model is misspecified. By construction, this region, along with the nonparametric conformal prediction region, maintains finite-sample local validity with respect to binning. These conformal prediction regions therefore achieves finite-sample marginal validity. The guarantee of finite-sample marginal and local validity are noted benefits of these conformal prediction regions [Lei and Wasserman, 2014, Eck et al., 2019]. However, the binned parametric and nonparametric conformal prediction regions are visually very different, as seen in Section 8, and give different prediction errors at small to moderate sample sizes. We see that the binned parametric conformal prediction region adapts naturally to the data when the model is correctly specified or modest deviations from the specified model are present. On the other hand, the nonparametric conformal prediction region does not adapt well to data obtained from a Gamma regression model or data obtained from a linear regression model with a steep mean function where steepness is relative to the variability about the mean function.

We find that the transformation based parametric conformal prediction region performs similarly to the HD prediction region under the assumed model when the assumed model is either the data generating model or only modest departures exist between the assumed model and the data generating model. This parametric conformal prediction region can be prohibitively large and inapproriate when the assumed model is in strong disagreement with the data generating model. However, this parametric conformal prediction region maintains finite sample marginal coverage in such settings.

The LS conformal prediction region obtains marginal validity [Lei et al., 2018] but performs poorly when deviations about the estimated mean function are either not symmetric, not constant, or both. When heterogeneity is present, the LS conformal prediction region exhibits undercoverage in regions where variability about the mean function is large and overcoverage in regions where variability about the mean function is small. This conformal prediction region is very sensitive to model misspecification. The LSLW conformal prediction region also obtains marginal validity [Lei et al., 2018, Section 5.2] and it is far less sensitive to model misspecification than the LS conformal prediction region. However, the LSLW conformal prediction region is not appropriate when deviations about an estimated mean function are obviously not symmetric, as evidenced in Section 8.1.

2 Illustrative example from the README file

We provide a gamma regression example with perfect model specification to illustrate the performance of conformal predictions when the model is known and the model does not have additive symmetric errors. We also compare conformal prediction regions to the oracle highest density region under the correct model. This example is included in the corresponding paper Eck et al. [2019] and the README file of the corresponding R package `conformal.glm` [Eck, 2018].

```
alpha <- 0.10
n <- 500
shape <- 2
beta <- c(1/4, 2)

set.seed(13)
x <- matrix(runif(n), ncol = 1)
rate <- cbind(1, x) %*% beta * shape
y <- rgamma(n = n, shape = shape, rate = rate)
data.readme <- data.frame(y = y, x = x)
colnames(data.readme)[2] <- c("x1")

fit.readme = glm(y ~ x1, family = Gamma, data = data.readme)
summary(fit.readme)

##
## Call:
## glm(formula = y ~ x1, family = Gamma, data = data.readme)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.0141   -0.7055   -0.1658    0.3386    1.7546
##
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.24070    0.02671   9.012   <2e-16 ***
## x1           2.04663    0.10584  19.336   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Gamma family taken to be 0.4786753)
##
##     Null deviance: 469.94  on 499  degrees of freedom
## Residual deviance: 257.53  on 498  degrees of freedom
## AIC: 817.06
##
## Number of Fisher Scoring iterations: 6
```

We now compute the binned and transformation based parametric conformal prediction regions [Eck et al., 2019] and the nonparametric conformal prediction region [Lei and Wasserman, 2014] using the `conformal.glm` function in the `conformal.glm` package. For illustration, we set the number of bins to be equal to 3. Since $X_i \sim U(0,1)$, we expect for there to be about 167 cases per bin with this choice of 3 bins. This choice for the number of bins is consistent with the bin width asymptotics in Lei and Wasserman [2014]. The number of cores is set to 7, if your computer has fewer than 7 cores at its disposal then you have to reduce the number of cores. This will not effect the final output but it will take longer to run.

```
bins <- 3
system.time(cpred <- conformal.glm(fit.readme,
  nonparametric = TRUE, bins = bins, cores = 7,
  method = "both"))

##    user  system elapsed
## 654.599   1.305 107.699

parabinCI <- cpred$paraconfbin
nonparabinCI <- cpred$nonparaconfbin
transformCI <- cpred$transformconf
```

We now compute the least squares with local weighting (LSLW) conformal prediction region [Lei et al., 2018, Section 5.2] using the `conformal.pred` function in the `conformalInference` package [Tibshirani, 2016]. In this procedure, the residuals from a regression fit are weighted by an estimate of the conditional mean absolute deviation (MAD) of $(Y - \mu(X))|X = x$, as a function of x. For augmented data including a proposed y, we let $\hat{\rho}_y(x)$ denote the estimate of the MAD. The conformal prediction procedure now uses

$$R_{y,i} = \frac{|Y_i - \hat{\mu}_y(X_i)|}{\hat{\rho}_y(X_i)}, \qquad (i = 1, ..., n+1)$$

as a anti-conformity measure. See Section 5.2 in Lei et al. [2018] for more details. The code below implements this method computing residuals from a cubic regression model, the cubic model fits the Gamma data better than a simple linear or quadratic model. We estimate $\hat{\rho}_y$ using a smoothing spline whose parameters are chosen by cross-validation.

```
funs <- lm.funs(intercept = TRUE)
train.fun <- funs$train.fun
predict.fun <- funs$predict.fun

cubic.model <- lm(y ~ x + I(x^2) + I(x^3))
abs.resid <- abs(cubic.model$resid)
smooth.call <- smooth.spline(x, abs.resid,
  nknots = 10)
lambda <- smooth.call$lambda
df <- smooth.call$df
mad.train.fun <- function(x, y, out = NULL){
  smooth.spline(x[, 1], y, lambda = lambda,
    df = df, nknots = 10)
}
mad.predict.fun <- function(out, newx){
  predict(out, newx[, 1])$y
}
system.time(p1.tibs <- conformal.pred(x = cbind(x,x^2,x^3),
  y = y, x0 = cbind(x,x^2,x^3),
  train.fun = train.fun, predict.fun = predict.fun,
  mad.train.fun = mad.train.fun,
  mad.predict.fun = mad.predict.fun,
  alpha = alpha))

##    user  system elapsed
## 309.964   0.177 310.344

LSLW = cbind(p1.tibs$lo, p1.tibs$up)
```

We now compute the highest density region using the `hdi` function in the `HDInterval` package [Meredith and Kruschke, 2018].

```
betaMLE <- coefficients(fit.readme)
shapeMLE <- as.numeric(gamma.shape(fit.readme)[1])
rateMLE <- cbind(1, x) %*% betaMLE * shapeMLE
HDCI <- do.call(rbind,
  lapply(1:nrow(x), function(j){
    hdi(qgamma, 0.90, shape = shapeMLE, rate = rateMLE[j, 1])
  }))
```

The four prediction regions for this data are depicted below. The top row depicts the parametric conformal prediction region (left panel) and the nonparametric conformal prediction region (right panel). The bin width was specified as 1/3 for these conformal prediction regions. The bottom row depicts the least squares conformal prediction region (left panel) and the highest density region (right panel). We see that the parametric conformal prediction region is a close discretization of the highest density region, the nonparametric conformal prediction region is quite jagged and unnatural, and the least squares conformal prediction region exhibits undercoverage for small x, exhibits overcoverage for large x, and includes negative response values of large magnitude.

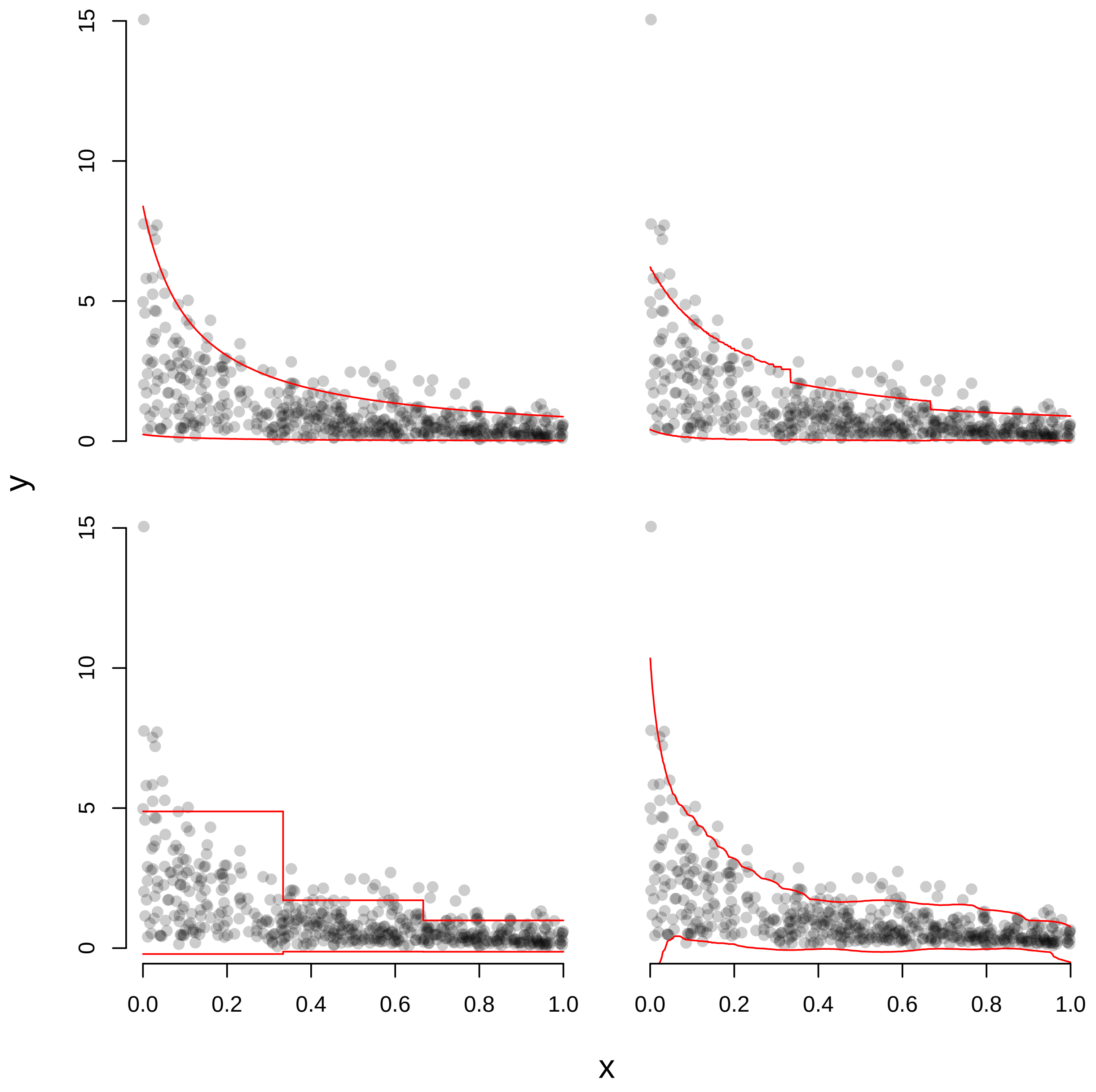
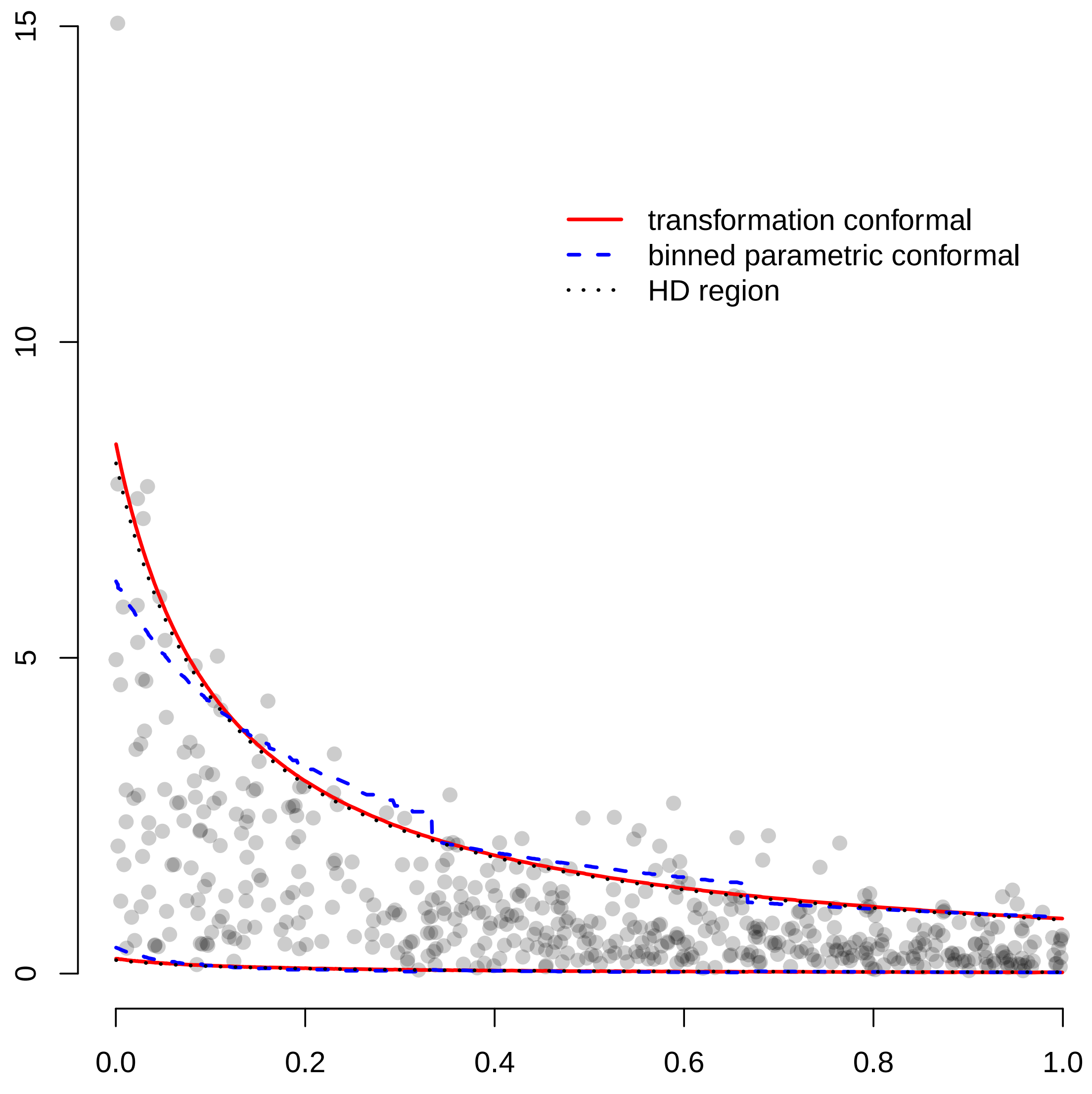


Figure 1: Sim setting: $n = 500$, shape = 2, bins = 3. The regions are depicted as follows: transformation based parametric conformal prediction (top-left panel), binned parametric conformal prediction region (top-right panel), nonparametric conformal prediction region (bottom-left panel), and LSLW conformal prediction region (bottom-right panel).

In Figure 2 we see that the transformation parametric conformal prediction region closely resembles the HD prediction region, and that the binned parametric conformal prediction region is a close descretization of the HD prediction region.

All of the presented prediction regions exhibit close to finite-sample marginal validity and local validity with respect to binning. However, the transformation conformal prediction region, LSLW conformal prediction region, and the HD prediction region do not exhibit finite-sample local validity in the second bin



and the HD prediction region does not quite possess finite-sample marginal validity. The binned parametric conformal prediction region is smallest in size with an estimated area of 2.19. The HD prediction region is a close second with an estimated area of 2.21. The tranformation conformal prediction region is a respectable third with an estimated area of 2.26. LSLW conformal prediction region has an estimated area of 2.56 and The nonparametric conformal prediction region has an estimated area of 2.68. Under correct model specification, the parametric conformal prediction regions are similar in performance to that of the highest density prediction region.

```
##########################################
## area and coverage
```

```r
# nonparametric conformal prediction region
# estimated area
area.nonparametric <- function(region){
  if(class(region) != "list"){
    stop("Only appropriate for nonparametric conformal prediction region")
  }
  bins <- length(region); wn <- 1 / bins
  area <- 0
  for(i in 1:bins){
    nonpar.region <- region[[i]]
    area <- area + wn * as.numeric(rep(c(-1,1),
      length(nonpar.region)/2) %*% nonpar.region)
  }
  area
}
# estimated area of prediction regions
area <- c(
  mean(apply(transformCI, 1, diff)),
  mean(apply(parabinCI, 1, diff)),
  area.nonparametric(nonparabinCI),
  mean(apply(LSLW, 1, diff)),
  mean(apply(HDCI, 1, diff))
)

# marginal local coverage of prediction regions
p <- 1
marginalcov <- c(
  local.coverage(region = transformCI, data = data.readme, d = p,
    bins = 1, at.data = "TRUE"),
  local.coverage(region = parabinCI, data = data.readme, d = p,
    bins = 1, at.data = "TRUE"),
  local.coverage(region = nonparabinCI, data = data.readme, d = p,
    nonparametric = "TRUE", bins = 1, at.data = "TRUE"),
  local.coverage(region = LSLW, data = data.readme, d = p,
    bins = 1, at.data = "TRUE"),
  local.coverage(region = HDCI, data = data.readme, d = p,
    bins = 1, at.data = "TRUE")
)

# local coverage of prediction regions
localcov <- cbind(
  local.coverage(region = transformCI, data = data.readme, d = p,
    bins = bins, at.data = "TRUE"),
  local.coverage(region = parabinCI, data = data.readme, d = p,
    bins = bins, at.data = "TRUE"),
  local.coverage(region = nonparabinCI, data = data.readme, d = p,
    nonparametric = "TRUE", bins = bins, at.data = "TRUE"),
  local.coverage(region = LSLW, data = data.readme, d = p,
    bins = bins, at.data = "TRUE"),
  local.coverage(region = HDCI, data = data.readme, d = p,
    bins = bins, at.data = "TRUE")
)

# diagnostics
diagnostics <- rbind(area, marginalcov, localcov)
colnames(diagnostics) <- c("transformCI", "binparaCI",
  "binnonparaCI", "LSLW", "HDCI")
rownames(diagnostics) <- c("area", "marginal coverage", "bin 1 coverage",
  "bin 2 coverage", "bin 3 coverage")
```

```
xtable(diagnostics, digits = c(0, 2, 2, 2, 2, 2))
```

	transformCI	binparaCI	binnonparaCI	LSLW	HDCI
area	2.26	2.19	2.68	2.56	2.21
marginal coverage	0.91	0.91	0.91	0.92	0.90
bin 1 coverage	0.91	0.91	0.93	0.92	0.90
bin 2 coverage	0.89	0.90	0.90	0.89	0.88
bin 3 coverage	0.93	0.91	0.90	0.95	0.91

3 Extension of the illustrative example from the README file

In this extension we investigate the local coverage properties of six prediction regions, the five from the previous section and a least squares (LS) conformal prediction region without local weighting. Local coverage is assessed via a Monte Carlo simulation of size $B = 50$. At each iteration of this simulation a new dataset is generated under the same specifications as those in the README file. We then compute the local coverage probabilities with respect to each bin and also approximate conditional coverage across x. We do not assess coverage properties directly at x for all $x \in (0, 1]$. This corresponds to the notion of conditional validity which cannot be achieved in tandem with an oracle estimation in finite sample settings when distributions are continuous [Lei and Wasserman, 2014, Section 2.2]. What we do instead is we assess local coverage with a binning in x that is much finer than the binning that was used to create the misspecified parametric and nonparametric conformal prediction regions. We form 25 bins of length 0.04.

3.1 Diagnostic measures

Several diagnostic measures are used to compare conformal prediction regions. These diagnostic measures compare prediction regions by their prediction error, volume, and coverage properties. Our prediction error diagnostic metric will be an average of the squared distances of observations outside of the prediction region to the closest boundary of the prediction region, averaged over all observations. An observation that falls within the prediction region has an error of 0. More formally this prediction error metric is

$$\text{prediction error} = n^{-1} \sum_{i=1}^{n} \mathbb{1}\left\{Y_i \notin C^{(\alpha)}(X_i)\right\} \left(\min_{j=1,\ldots,m_i} \{\min\{|Y_i - a_{i,j}|, |Y_i - b_{i,j}|\}\}\right)^2,$$

where $a_{i,j}$ and $b_{i,j}$ are, respectively, the lower and upper boundaries of possible $j = 1, \ldots, m_i$ disjoint intervals forming the prediction region.

The volume of each prediction region will be estimated by the average of the upper boundary minus the lower boundary across observed $\mathcal{X}$, written as

$$\text{area} = n^{-1} \sum_{i=1}^{n} \sum_{j=1}^{m_i} (b_{i,j} - a_{i,j}).$$

To assess finite-sample marginal validity we calculate the proportion of responses that fall within the prediction region. To assess finite-sample local validity with respect to binning we first bin the predictor data and then, for each bin, we calculate the proportion of responses that fall within the prediciton region. The same procedure is used to assess finite-sample conditional validity, but we use a much finer binning regime than what was used to assess finite-sample local validity.

The following function computes our estimate of prediction error at the observed data when the prediction region is not the nonparametric conformal prediction region.

```
absolute.error <- function(y, region, squared = TRUE){
  lwr <- region[, 1]
  upr <- region[, 2]
  index <- which(!(lwr <= y & y <= upr))
  out <- sum(unlist(lapply(index, function(j){
    error <- NULL
    if(squared == FALSE) error <-
      min(abs(y[j] - lwr[j]), abs(y[j] - upr[j]))
    if(squared == TRUE) error <-
      (min(abs(y[j] - lwr[j]), abs(y[j] - upr[j])))^2
    error
  }))) / length(y)
  out
}
```

The following function computes our estimate of prediction error at the observed data when the prediction region is the nonparametric conformal prediction region.

```
absolute.error.nonparametric <- function(data, region,
  squared = TRUE){
  n <- nrow(data)
  n.bins.region <- length(region)
  d <- ncol(data) - 1
  x <- as.matrix(data[,1:d + 1], col = d)
  index.bins.region <- find.index(x, wn = 1/n.bins.region, d = d)
  y <- data[, 1]
  area <- 0
  for(j in 1:n){
    index.j <- which(index.bins.region == j)
    error <- c(y[j] - region[[index.bins.region[j]]])
    index <- 0
    if(any(error > 0)) index <- max(which(error > 0))
    if(index %% 2 == 0){
      if(squared == FALSE) area <-
        area + min(abs(error)) / n
      if(squared == TRUE) area <-
        area + min(abs(error))^2 / n
    }
  }
  area
}
```

Our Monte Carlo simulator follows. This function computes all of the dignostics and local coverage probabilities for each prediction region for every generated dataset.

Revised up to here

```
funs <- lm.funs(intercept = TRUE)
train.fun <- funs$train.fun
predict.fun <- funs$predict.fun
mad.predict.fun <- function(out, newx){
  newx <- as.numeric(newx)
  predict(out, newx[, 1])$y
}

gamma_simulator <- function(beta, n = 500, alpha = 0.10, bins = 3,
  family = "Gamma", link = "inverse", shape = 2,
  parametric = TRUE, nonparametric = TRUE, method = "both",
  LS = TRUE, LSLW = TRUE, HD = TRUE, cores = 7){

  ## in this univariate problem, p and d are the same
  p <- d <- length(beta) - 1
  x <- matrix(runif(n*p), ncol = p)
  y <- rep(0,n)
  dat <- NULL

  ## set up partition
  if(class(bins) == "NULL"){
    wn <- min(1/ floor(1 / (log(n)/n)^(1/(d+3))), 1/2)
    bins <- 1 / wn
  }

  ## generate the data (has functionality for different
  ## families and link functions)
  if(family == "Gamma"){
    if(link == "identity"){
      rate <- (1 / (cbind(1, x) %*% beta)) * shape
      y <- rgamma(n = n, shape = shape, rate = rate)
      y <- y / sd(y)
      dat <- data.frame(y = y, x = x)
      colnames(dat)[2:(p+1)] <- paste("x", 1:p, sep = "")
    }
    if(link == "inverse"){
      rate <- (cbind(1, x) %*% beta) * shape
      y <- rgamma(n = n, shape = shape, rate = rate)
      y <- y / sd(y)
      dat <- data.frame(y = y, x = x)
      colnames(dat)[2:(p+1)] <- paste("x", 1:p, sep = "")
    }
    if(link == "log"){
      rate <- (1 / exp(cbind(1, x) %*% beta)) * shape
      y <- rgamma(n = n, shape = shape, rate = rate)
      y <- y / sd(y)
      dat <- data.frame(y = y, x = x)
      colnames(dat)[2:(p+1)] <- paste("x", 1:p, sep = "")
    }
  }
  if(family == "gaussian"){
    mu <- cbind(1, x) %*% beta
    y <- rnorm(n = n, mean = mu, sd = sd)
    dat <- data.frame(y = y, x = x)
    colnames(dat)[2:(p+1)] <- paste("x", 1:p, sep = "")
  }
  if(family == "inverse.gaussian"){
    mu = 1 / sqrt(cbind(1, x) %*% beta)
    y <- rinvgauss(n = n, mean = mu)
    y <- y / sd(y)
    dat <- data.frame(y = y, x = x)
    colnames(dat)[2:(p+1)] <- paste("x", 1:p, sep = "")
  }

  ## fit the Gamma regression model
  fit <- glm(y ~ x1, family = "Gamma", data = dat)
  assign("dat", dat, .GlobalEnv)
  parabinCI <- transformCI <- nonparabinCI <- LSCI <-
    LSLWCI <- HDCI <- NULL
  formula <- fit$formula
  newdata <- dat
  respname <- all.vars(formula)[1]
  newdata <- newdata[, !(colnames(dat) %in% respname)]
  newdata <- as.matrix(newdata)

  ## obtain the prediction regions
  output.parabin <- output.transform <- output.nonparabin <-
    output.LS <- output.LSLW <- output.HD <- rep(NA, bins + 1)
```

```r
  if(parametric){
    cpred <- conformal.glm(fit, parametric = TRUE,
      nonparametric = FALSE, alpha = alpha, method = "both",
      bins = bins, cores = cores, newdata = newdata,
      precision = 0.02)

    # bin parametric conformal
    parabinCI <- cpred$paraconfbin
    marginal.parabin <- local.coverage(region = parabinCI,
      data = dat, d = p, bins = 1, at.data = "TRUE")
    local.parabin <- local.coverage(region = parabinCI,
      data = dat, d = p, bins = bins, at.data = "TRUE")
    local.inx.parabin <- local.coverage(region = parabinCI,
      data = dat, d = p, bins = 25, at.data = "TRUE")
    output.parabin <- c(marginal.parabin, local.parabin,
      local.inx.parabin,
      mean(apply(parabinCI, 1, diff)),
      absolute.error(y = y, region = parabinCI))

    # transform parametric conformal
    transformCI <- cpred$transformconf
    marginal.transform <- local.coverage(region = transformCI,
      data = dat, d = p, bins = 1, at.data = "TRUE")
    local.transform <- local.coverage(region = transformCI,
      data = dat, d = p, bins = bins, at.data = "TRUE")
    local.inx.transform <- local.coverage(region = transformCI,
      data = dat, d = p, bins = 25, at.data = "TRUE")
    output.transform <- c(marginal.transform, local.transform,
      local.inx.transform,
      mean(apply(transformCI, 1, diff)),
      absolute.error(y = y, region = transformCI))
  }
  if(nonparametric){
    cpred <- conformal.glm(fit, parametric = FALSE,
      nonparametric = TRUE, alpha = alpha, method = "both",
      bins = bins, cores = cores, precision = 0.02)
    nonparabinCI <- cpred$nonparaconfbin

    marginal.nonparabin <- local.coverage(region = nonparabinCI,
      nonparametric = "TRUE", data = dat, d = p, bins = 1,
      at.data = "TRUE")
    local.nonparabin <- local.coverage(region = nonparabinCI,
      nonparametric = "TRUE", data = dat, d = p, bins = bins,
      at.data = "TRUE")
    local.inx.nonparabin <- local.coverage(region = nonparabinCI,
      nonparametric = "TRUE", data = dat, d = p, bins = 25,
      at.data = "TRUE")
    output.nonparabin <-
      c(marginal.nonparabin, local.nonparabin,
        local.inx.nonparabin,
        area.nonparametric(nonparabinCI),
        absolute.error.nonparametric(data = dat,
          region = nonparabinCI))
  }
  if(LS){
    p1.tibs <- conformal.pred(x = cbind(x,x^2,x^3), y = y,
      x0 = cbind(x,x^2,x^3),
      train.fun = train.fun, predict.fun = predict.fun,
      alpha = alpha)
    LSCI <- cbind(p1.tibs$lo, p1.tibs$up)

    marginal.LS <- local.coverage(region = LSCI,
      data = dat, d = p, bins = 1, at.data = "TRUE")
    local.LS <- local.coverage(region = LSCI,
      data = dat, d = p, bins = bins, at.data = "TRUE")
    local.inx.LS <- local.coverage(region = LSCI,
      data = dat, d = p, bins = 25, at.data = "TRUE")
    output.LS <- c(marginal.LS, local.LS, local.inx.LS,
      mean(apply(LSCI, 1, diff)),
      absolute.error(y = y, region = LSCI))
  }
  if(LSLW){
    cubic.model <- lm(y ~ x + I(x^2) + I(x^3))
    abs.resid <- abs(cubic.model$resid)
    smooth.call <- smooth.spline(x, abs.resid,
      nknots = 10)
    lambda <- smooth.call$lambda
    df <- smooth.call$df
    mad.train.fun <- function(x, y, out = NULL){
      smooth.spline(x[, 1], y, lambda = lambda,
      df = df, nknots = 10)
    }
    p2.tibs <- conformal.pred(x = cbind(x,x^2,x^3), y = y,
      x0 = cbind(x,x^2,x^3),
      train.fun = train.fun, predict.fun = predict.fun,
      mad.train.fun = mad.train.fun,
      mad.predict.fun = mad.predict.fun,
      alpha = alpha)
    LSLWCI <- cbind(p2.tibs$lo, p2.tibs$up)

    marginal.LSLW <- local.coverage(region = LSLWCI,
      data = dat, d = p, bins = 1, at.data = "TRUE")
    local.LSLW <- local.coverage(region = LSLWCI,
      data = dat, d = p, bins = bins, at.data = "TRUE")
```

```r
    local.inx.LSLW <- local.coverage(region = LSLWCI,
      data = dat, d = p, bins = 25, at.data = "TRUE")
    output.LSLW <- c(marginal.LSLW, local.LSLW, local.inx.LSLW,
      mean(apply(LSLWCI, 1, diff)),
      absolute.error(y = y, region = LSLWCI))
  }
  if(HD){
    betaMLE <- coefficients(fit)
    shapeMLE <- as.numeric(gamma.shape(fit)[1])
    rateMLE <- cbind(1, newdata) %*% betaMLE * shapeMLE
    HDCI <- do.call(rbind, lapply(1:nrow(newdata), function(j){
      hdi(qgamma, 1 - alpha, shape = shapeMLE, rate = rateMLE[j, 1])
    }))

    marginal.HD <- local.coverage(region = HDCI,
      data = dat, d = p, bins = 1, at.data = "TRUE")
    local.HD <- local.coverage(region = HDCI,
      data = dat, d = p, bins = bins, at.data = "TRUE")
    local.inx.HD <- local.coverage(region = HDCI,
      data = dat, d = p, bins = 25, at.data = "TRUE")
    output.HD <- c(marginal.HD, local.HD, local.inx.HD,
      mean(apply(HDCI, 1, diff)),
      absolute.error(y = y, region = HDCI))
  }

  output <- list(output.parabin = output.parabin,
    output.transform = output.transform,
    output.nonparabin = output.nonparabin,
    output.LS = output.LS,
    output.LSLW = output.LSLW,
    output.HD = output.HD)
  output
}
```

The following performs our Monte Carlo simulation with $B = 50$ iterations.

```r
set.seed(13)
B <- 50
system.time(local.500.3.2 <- do.call(cbind, lapply(1:B,
  FUN = function(j){
    unlist(gamma_simulator(beta = beta))
})))

##      user    system   elapsed
## 20480.974    88.025  6143.265
```

	parametric conformal	nonparametric conformal	LS conformal	LSLW conformal	HD region
marginal coverage	0.909 (0.0005)	0.912 (0.0006)	0.901 (0.0005)	0.904 (0.0004)	0.915 (0.0007)
local coverage when $0 < x < 1/3$	0.909 (0.001)	0.911 (0.0023)	0.903 (0.0011)	0.75 (0.0024)	0.913 (0.0015)
local coverage when $1/3 \leq x < 2/3$	0.908 (0.0006)	0.913 (0.0026)	0.9 (0.0006)	0.967 (0.0015)	0.907 (0.0024)
local coverage when $2/3 \leq x < 1$	0.909 (0.0008)	0.912 (0.0028)	0.898 (0.0007)	0.994 (0.0008)	0.925 (0.0022)
area	1.832 (0.0203)	1.853 (0.0186)	2.035 (0.0231)	2.341 (0.0175)	1.963 (0.0152)
prediction error	0.084 (0.0054)	0.065 (0.0039)	0.137 (0.0077)	0.13 (0.0057)	0.054 (0.0034)

Table 1: Diagnostics of prediction regions. This table gives the area, prediction error, marginal coverage, and local coverages with respect to our binning scheme for the parametric, nonparametric, LS, and LSLW conformal prediction regions and the HD prediction region.

```r
local.gamma.500.3.2 <- cbind(
  rowMeans(local.500.3.2, na.rm = TRUE),
  apply(local.500.3.2, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))

options(scipen = 999)
local.gamma.500.3.2[, 1] <- round(local.gamma.500.3.2[, 1], 3)
local.gamma.500.3.2[, 2] <- round(local.gamma.500.3.2[, 2], 4)
```

Diagnostics for each of the siz prediction regions are given in Table 1 and Figure 2. We see that the parametric conformal prediction region performs as advertised. When the model is correctly specified, the parametric conformal prediction region is similar to the minimum length HD prediction region in area and prediction error (and in appearance as seen in Figure 1). The parametric conformal prediction region exhibits some conservative overcoverage marginally and with respect to binning, and some undercoverage in x for values close to the break points of the bins. The LSLW conformal prediction region has lower prediction error than the parametric conformal prediction error but such a benefit comes with the cost of lack of precision (increase in size) and dramatic overcoverage. The low prediction error of the LSLW conformal prediction region appears to stem from its ability to be far wider than the other prediction intervals at the values of x where the response data is most variable, as seen in Figure 1. This feature combined with its symmetric errors construction is what leads to its increase in size. The LSLW conformal prediction region is seen to provide conservative finite-sample local coverage in x. The nonparametric and LS conformal prediction regions are larger and have larger prediction error than the parametric conformal prediction region. The LS conformal prediction exhibits large undercoverage when the predictor is small and large overcoverage when the predictor is large. This is best evidenced by Figure 2.

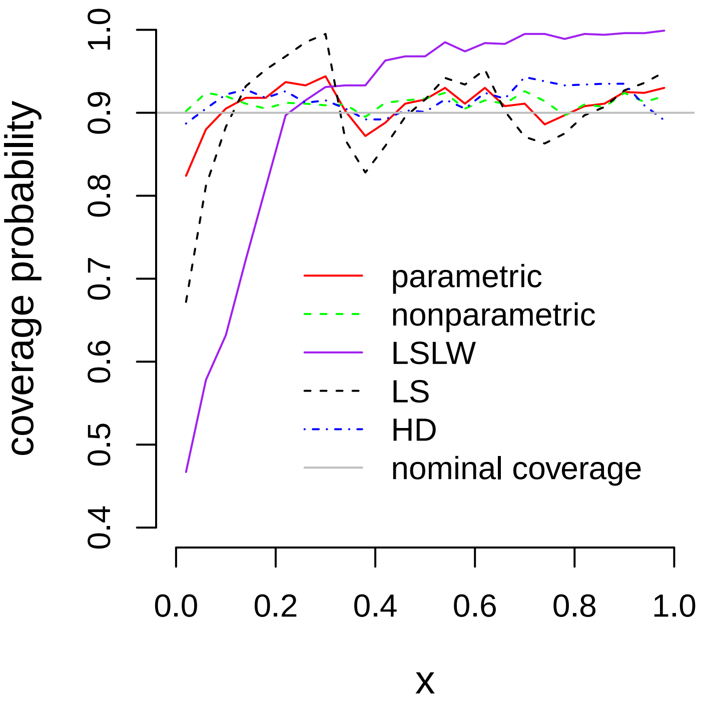


Figure 2: Plot of the estimated coverage probabilities of prediction regions across x.

4 Predicting the risk of diabetes

In this Section we reproduce the data analysis that appears in Section 5 of Eck et al. [2019]. We examine the influence of several variables on blood sugar, or glycosylated hemoglobin percentage (also known as HbA1c), an important risk factor for diabetes. A glycosylated hemoglobin value of 6.5% can be used as a cutoff for positive diagnosis of diabetes [World Health Organization, 2011].

We predict an individual's glycosylated hemoglobin from their height, weight, age, and gender, all of which are easy to measure, inexpensive, and do not require any laboratory testing. The data in this analysis come from a population-based sample of 403 rural African-Americans in Virginia [Willems et al., 1997], taken from the `faraway` R package [Faraway, 2016]. We considered a gamma regression model that only includes linear terms for each covariate, a linear regression model with homoskedastic normal errors and the same linear terms for each covariate, and a linear regression model with homoskedastic normal errors that also included quadratic terms for each covariate. Of these considered the models, the gamma regression model fit the data best on the basis that it had the lowest AIC value and it gave the best predictive predictive performance on the basis that it had the lowest sum of squares prediction error. That being said, we do not know the data generating process.

Based on these covariates, conformal prediction regions provide finite sample valid prediction regions for glycosylated hemoglobin that may be useful for diagnosing diabetes in this study population. The data set is loaded from the R package Faraway [2016].

```r
data(diabetes)
```

We throw away data points that contain missing values of any variable of interest.

```
dat <- diabetes[, c("glyhb", "height", "weight", "age", "gender")]
dat <- dat[complete.cases(dat), ]
```

The Gamma regression model with with log link function is selected among several candidate models as the model used for inference in this analysis. This model has the lowest AIC value among candidates.

```
m.gamma.log <- glm(glyhb ~ height + weight + age + gender,
  data = dat, family = Gamma(link = log))
m.gamma <- glm(glyhb ~ height + weight + age + gender,
  data = dat, family = "Gamma")
m.gamma.identity <- glm(glyhb ~ height + weight + age + gender,
  data = dat, family = Gamma(link = identity))
m.gaussian <- glm(glyhb ~ height + weight + age + gender,
  data = dat, family = "gaussian", x = TRUE)
X <- m.gaussian$x
m.gaussian.2 <- glm(glyhb ~ height + weight + age + I(height^2)
  + I(weight^2) + I(age^2) + gender,
  data = dat, family = "gaussian", x = TRUE)

AIC(m.gamma.log, m.gamma, m.gamma.identity, m.gaussian, m.gaussian.2)

##                  df      AIC
## m.gamma.log       6 1452.012
## m.gamma           6 1452.328
## m.gamma.identity  6 1453.972
## m.gaussian        6 1644.621
## m.gaussian.2      9 1648.256
```

This model also provides satisfactory prediction as seen in Figure 3.

Four conformal prediction regions are considered for predicting glycosylated hemoglobin percentage. These conformal prediction regions are the parametric conformal prediction region with a Gamma model fit, the parametric conformal prediction region with a Gaussian model fit, the least squares conformal prediction region, and the least square conformal prediction region with local weighting. All conformal prediction regions correspond to models that only include linear terms for each of the covariates. The Gamma and Gaussian parametric conformal prediction regions were computed with binning across the binary gender factor variable, the predictor space is partitioned across genders. However, no additional binning structure within the levels of gender was employed.

These conformal prediction regions are now constructed.

```
## Gamma conformal
system.time(conformal.gamma <-
  conformal.glm(m.gamma.log, cores = 6, bins = 1))

##    user  system elapsed
##  85.879   0.303  19.841

conformal.gamma <- conformal.gamma$paraconformal
head(conformal.gamma)
```

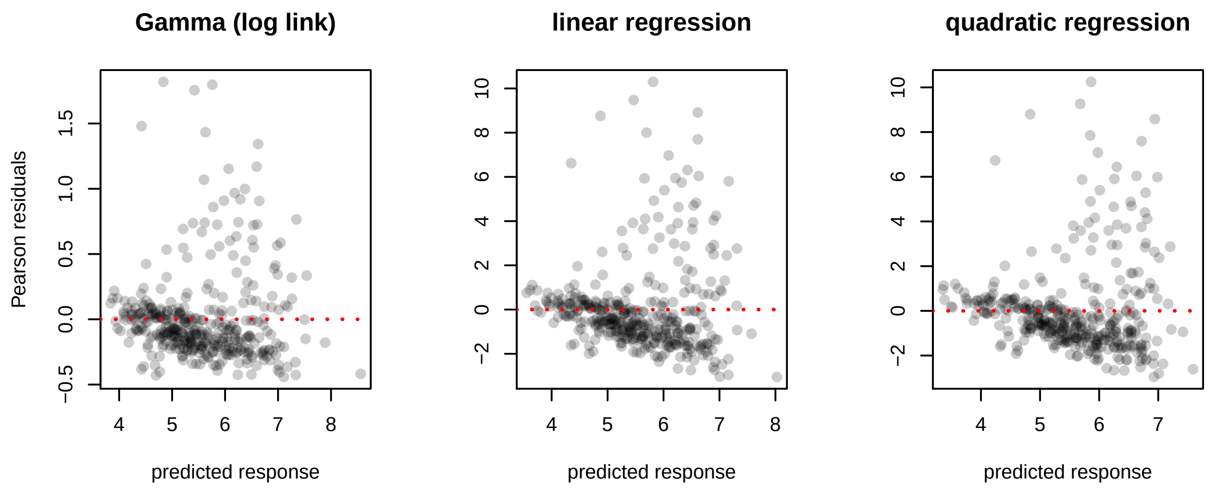


Figure 3: Predictive performance of candidate models.

```
##         lwr      upr
## [1,] 2.485 7.352500
## [2,] 2.545 7.495000
## [3,] 3.590 9.620031
## [4,] 2.925 8.752500
## [5,] 3.270 9.400000
## [6,] 2.500 7.777500

## Gaussian conformal
system.time(conformal.gaussian <-
  conformal.glm(m.gaussian, cores = 6, bins = 1))

##    user  system elapsed
##  58.904   0.392   9.269

conformal.gaussian <- conformal.gaussian$paraconformal
head(conformal.gaussian)

##         lwr      upr
## [1,] 2.015 7.862500
## [2,] 2.130 8.012500
## [3,] 3.670 9.625027
## [4,] 2.925 8.880000
## [5,] 3.445 9.390000
## [6,] 2.190 8.092500

## LS conformal
funs <- lm.funs(intercept = TRUE)
train.fun <- funs$train.fun
```

```
predict.fun <- funs$predict.fun
alpha <- 0.10
y <- dat$glyhb
system.time(p1.tibs <- conformal.pred(x = X[, -1],
  y = y, x0 = X[, -1],
  train.fun = train.fun, predict.fun = predict.fun,
  alpha = alpha))

##    user  system elapsed
##   2.481   0.011   2.493

LS = cbind(p1.tibs$lo, p1.tibs$up)
colnames(LS) <- c("lwr", "upr")
head(LS)

##             lwr      upr
## [1,] 2.034091 7.792133
## [2,] 2.253147 7.917308
## [3,] 3.751768 9.545698
## [4,] 3.022078 8.730944
## [5,] 3.513430 9.319835
## [6,] 2.298523 8.045960

## LSLW conformal
mad.train.fun <- function(x, y, out = NULL){
  lm(y ~ x[, -1], x = TRUE, y = TRUE)
}
mad.predict.fun <- function(out, newx){
  predict(out, as.data.frame(newx))
}
system.time(p2.tibs <- conformal.pred(x = X[, -1],
  y = y, x0 = X[, -1],
  train.fun = train.fun, predict.fun = predict.fun,
  mad.train.fun = mad.train.fun,
  mad.predict.fun = mad.predict.fun,
  alpha = alpha))

##    user  system elapsed
##  47.584   0.008  47.591

LSLW = cbind(p2.tibs$lo, p2.tibs$up)
colnames(LSLW) <- c("lwr", "upr")
head(LSLW)

##             lwr       upr
## [1,] 2.802525  7.058296
## [2,] 2.974858  7.246449
## [3,] 2.169697 11.255303
## [4,] 2.359546  9.509375
## [5,] 2.196818 10.577273
## [6,] 2.905844  7.413132
```

4.1 Results in Eck et al. [2019]

We now compute the diagnostics of these four conformal prediction regions and reproduce Table 2 in Eck et al. [2019]. We start with the computation of marginal and approximate average conditional coverage using functionality in the `conformal.glm` software.

```
## marginal coverages
marginal.gamma <- local.coverage(region = conformal.gamma, data = dat,
  bins = 1, d = 2)
marginal.gaussian <- local.coverage(region = conformal.gaussian,
  data = dat, bins = 1, d = 2)
marginal.LS <- local.coverage(region = LS, data = dat, bins = 1, d = 2)
marginal.LSLW <- local.coverage(region = LSLW, data = dat, bins = 1, d = 2)
table.marginal <- matrix(c(marginal.gamma, marginal.gaussian,
  marginal.LSLW, marginal.LS), ncol = 4)
table.marginal <- round(table.marginal, 3)
colnames(table.marginal) <- c("gamma", "gaussian", "LSLW", "LS")
rownames(table.marginal) <- "marginal coverage"

## conditional coverages
d <- ncol(dat) - 2
conditional.gamma <- local.coverage(region = conformal.gamma, data = dat,
  bins = 3, d = d)
conditional.gaussian <- local.coverage(region = conformal.gaussian,
  data = dat, bins = 3, d = d)
conditional.LS <- local.coverage(region = LS, data = dat, bins = 3, d = d)
conditional.LSLW <- local.coverage(region = LSLW, data = dat, bins = 3, d = d)
table.conditional <- matrix(cbind(conditional.gamma, conditional.gaussian,
  conditional.LSLW, conditional.LS), ncol = 4)
table.conditional <- round(table.conditional, 3)
colnames(table.conditional) <- c("gamma", "gaussian", "LSLW", "LS")
avg.cond.coverage <- colMeans(table.conditional,na.rm = TRUE)
```

We now compute the prediction error of each conformal prediction region.

```
pred.error.gamma <- absolute.error(y = dat$glyhb, region = conformal.gamma)
pred.error.gaussian <- absolute.error(y = y, region = conformal.gaussian)
pred.error.LSLW <- absolute.error(y = y, region = LSLW)
pred.error.LS <- absolute.error(y = y, region = LS)
table.pred.error <- matrix(c(pred.error.gamma, pred.error.gaussian,
  pred.error.LSLW, pred.error.LS), ncol = 4)
colnames(table.pred.error) <- c("gamma", "gaussian", "LSLW", "LS")
rownames(table.pred.error) <- "pred error"
```

We now compute the volume of each conformal prediction region.

```
## when no factor variables are present
volume.regions <- function(region, preds){
  if(class(region) == "list"){
    stop("Not appropriate for nonparametric conformal prediction regions")
  }
  n <- nrow(region)
  mat <- as.matrix(cbind(preds, region[, 1]))
  mat2 <- as.matrix(cbind(preds, region[, 2]))
  mat3 <- rbind(mat, mat2)
  vol <- as.numeric(convhulln(mat3, option = "FA")$vol)
  vol
}

preds <- dat[, 2:4]
volume.gamma <- volume.regions(region = conformal.gamma, preds = preds)
volume.gaussian <- volume.regions(region = conformal.gaussian, preds = preds)
volume.LSLW <- volume.regions(region = LSLW, preds = preds)
volume.LS <- volume.regions(region = LS, preds = preds)
table.volume <- matrix(c(volume.gamma, volume.gaussian,
  volume.LSLW, volume.LS), ncol = 4)
colnames(table.volume) <- c("gamma", "gaussian", "LSLW", "LS")
rownames(table.volume) <- "volume"

volume.factors <- function(region, bins = 1, data, object){
  if(class(region) == "list"){
    stop("Not appropriate for nonparametric conformal prediction regions")
  }
  n <- nrow(region)

  call <- object$call
  formula <- call$formula
  respname <- all.vars(formula)[1]
  index.factor.variables <- which(attr(terms(object), "dataClasses") == "factor")
  index.numeric.variables <- which(attr(terms(object), "dataClasses") == "numeric")
  data <- data[, respname != colnames(data)]
  index.data.factor.variables <- which(colnames(data) %in% names(index.factor.varia
  index.data.numeric.variables <- which(colnames(data) %in% names(index.numeric.var
  factors <- lapply(index.data.factor.variables, function(j) as.numeric(as.factor(c
  data.by.factors <- split(data, factors, drop = FALSE)
  region.by.factors <- split(region, factors, drop = FALSE)
  matrix.region.by.factors <- lapply(region.by.factors, function(x){
    matrix(x, ncol = 2)
  })
  numeric.by.factors <- lapply(data.by.factors, function(x){
    mat <- x[, index.data.numeric.variables]
    colnames(mat) <- colnames(data)[index.data.numeric.variables]
    mat
  })
  bin.index.by.factors <- lapply(data.by.factors, function(x){
    if(nrow(x) == 0) return(0)
    mat <- as.matrix(x[, index.data.numeric.variables[-1]],
      ncol = length(index.data.numeric.variables[-1]))
    find.index(mat = mat, wn = 1/bins, d = ncol(mat))
  })

  volume.factor <- sapply(1:length(region.by.factors), function(i){
    out <- sum(sapply(1:length(unique(bin.index.by.factors[[i]])), function(j){
      X <- rbind(
        cbind(numeric.by.factors[[i]][which(bin.index.by.factors[[i]] == j), ],
          matrix.region.by.factors[[i]][which(bin.index.by.factors[[i]] == j), 1]),
        cbind(numeric.by.factors[[i]][which(bin.index.by.factors[[i]] == j), ],
          matrix.region.by.factors[[i]][which(bin.index.by.factors[[i]] == j), 2])
      )
      X <- X[, -1]
      vol <- as.numeric(convhulln(X, option = "FA")$vol) * nrow(X) / (2*n)
      vol
    }))
    return(out)
  })

  sum(volume.factor)
}

volume.gamma <- volume.factors(region = conformal.gamma,
  data = dat, object = m.gamma.log)
volume.gaussian <- volume.factors(conformal.gaussian,
  data = dat, object = m.gaussian)
volume.LSLW <- volume.factors(LSLW,
  data = dat, object = m.gaussian)
volume.LS <- volume.factors(LS,
  data = dat, object = m.gaussian)
table.volume <- matrix(c(volume.gamma, volume.gaussian,
  volume.LSLW, volume.LS), ncol = 4)
colnames(table.volume) <- c("gamma", "gaussian", "LSLW", "LS")
rownames(table.volume) <- "volume"
table.volume <- table.volume / 1e4
```

Table 2 and Figure 4 in Eck et al. [2019] are shown below.

```
diagnostics <- rbind(table.marginal, table.volume,
  table.pred.error, avg.cond.coverage)
#diagnostics <- t(diagnostics)
xtable(diagnostics, digits = c(0, 3, 3, 3, 3))
```

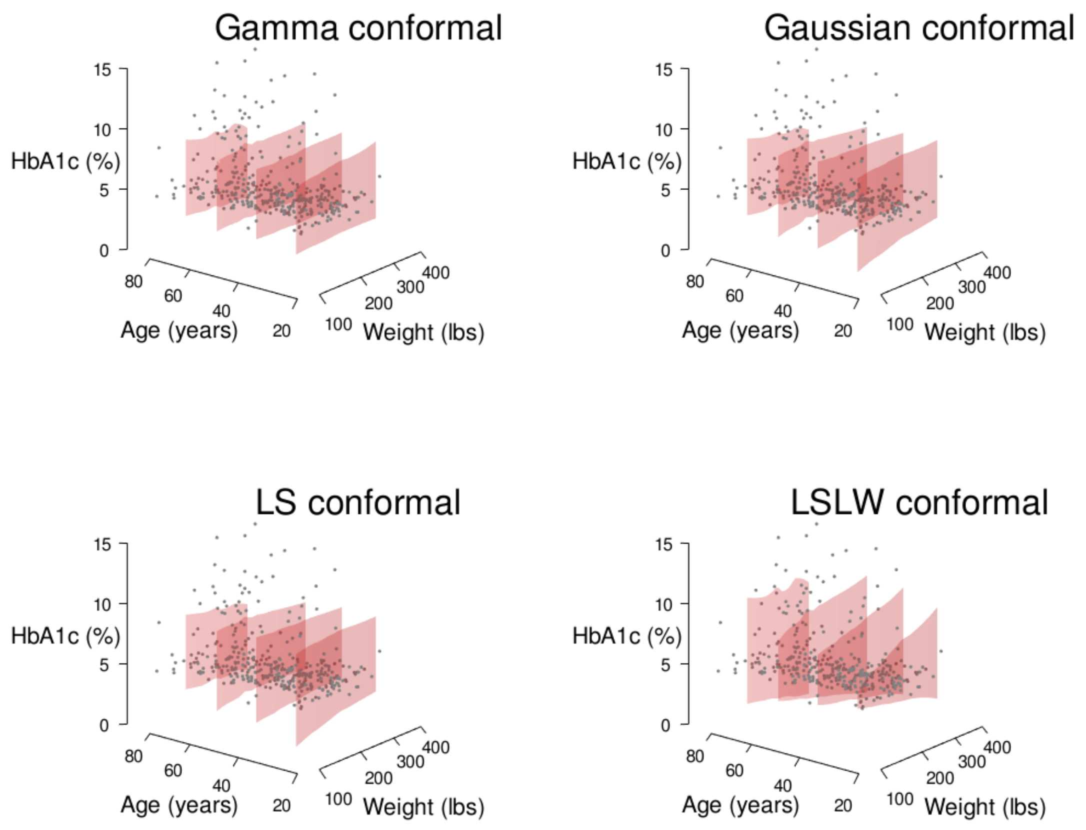


Figure 4: Figure 4 in Eck et al. [2019]. Conformal prediction regions for glycosylated hemoglobin projected onto the age and weight predictor axes. Upper and lower bounds of the conformal prediction region are loess smoothed for visual appearance. The code that constructed this figure is displayed in the accompanying .Rnw file.

	gamma	gaussian	LSLW	LS
marginal coverage	0.909	0.906	0.906	0.909
volume	7.349	7.730	8.983	7.601
pred error	0.931	0.849	0.744	0.883
avg.cond.coverage	0.874	0.849	0.889	0.860

5 Additional Gamma simulations

For the simulations in this section we generate responses using a gamma regression model with one variable. We specify the inverse link function (the defualt in the `glm` function) and we set $\beta = (1.25, -1)^T$. This value of β is chosen so that the generated Gamma data is increasing, on average, in x and exhibits increasing variability in x. We investigate the performance of the five prediction regions across different sample size and shape parameter combinations. We consider sample sizes of $n \in \{150, 250, 500\}$ and shape parameter values of $\{0.5, 0.75, 1, 1.5, 2, 5, 7, 10, 12, 15, 20, 25, 30, 40, 50, 60, 70, 80, 90, 100\}$. The LS and LSLW conformal prediction regions are fit using a cubic regression model. This model is simple and it fits this type of Gamma data better than a simple linear or quadratic regression model. Note that as the shape parameter increases the cubic regression model fits the data better. When $n = 150$ we build the parametric and nonparametric conformal prediction regions using 2 bins. When $n = 250, 500$ we build the parametric and nonparametric conformal prediction regions using 3 bins. These number of bin choices correspond to the bin width asymptotics of Lei and Wasserman [2014].

The next subsection contains all the code necessary to reproduce our results in Section 5.2.

5.1 Simulations

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 0.5.

```r
set.seed(13)
beta <- c(1.25, -1)
bins <- 2
n <- 150
B <- 250
system.time(out.gamma.150.2.0.5 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma_simulator(beta = beta, n = n,
      bins = bins, shape = 0.5))
})))

## Error in r[, 1]/mad.predict.fun(out.mad, xx):  non-numeric argument to
binary operator
## Timing stopped at:  47.47 1.415 9.022
```

```r
gamma.150.2.0.5 <- cbind(
  rowMeans(out.gamma.150.2.0.5, na.rm = TRUE),
  apply(out.gamma.150.2.0.5, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))

## Error in is.data.frame(x):  object 'out.gamma.150.2.0.5' not found
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 0.75.

```
system.time(out.gamma.150.2.0.75 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma_simulator(beta = beta, n = n,
      bins = bins, shape = 0.75))
})))

## Error in r[, 1]/mad.predict.fun(out.mad, xx):  non-numeric argument to
binary operator
## Timing stopped at:  54.82 0.8 9.444
```

```
gamma.150.2.0.75 <- cbind(
  rowMeans(out.gamma.150.2.0.75, na.rm = TRUE),
  apply(out.gamma.150.2.0.75, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))

## Error in is.data.frame(x):  object 'out.gamma.150.2.0.75' not found
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 1.

```
system.time(out.gamma.150.2.1 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma_simulator(beta = beta, n = n,
      bins = bins, shape = 1))
})))

## Error in r[, 1]/mad.predict.fun(out.mad, xx):  non-numeric argument to
binary operator
## Timing stopped at:  54.56 0.745 9.501
```

```
gamma.150.2.1 <- cbind(
  rowMeans(out.gamma.150.2.1, na.rm = TRUE),
  apply(out.gamma.150.2.1, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))

## Error in is.data.frame(x):  object 'out.gamma.150.2.1' not found
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 1.5.

```
system.time(out.gamma.150.2.1.5 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma_simulator(beta = beta, n = n,
      bins = bins, shape = 1.5))
})))

## Error in r[, 1]/mad.predict.fun(out.mad, xx):  non-numeric argument to
binary operator
## Timing stopped at:  40.15 1.424 7.519
```

```
gamma.150.2.1.5 <- cbind(
  rowMeans(out.gamma.150.2.1.5, na.rm = TRUE),
  apply(out.gamma.150.2.1.5, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))

## Error in is.data.frame(x):  object 'out.gamma.150.2.1.5' not found
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 2.

```
system.time(out.gamma.150.2.2 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma_simulator(beta = beta, n = n,
      bins = bins, shape = 2))
})))

## Error in r[, 1]/mad.predict.fun(out.mad, xx):  non-numeric argument to
binary operator
## Timing stopped at:  42.46 0.648 7.737
```

```
gamma.150.2.2 <- cbind(
  rowMeans(out.gamma.150.2.2, na.rm = TRUE),
  apply(out.gamma.150.2.2, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))

## Error in is.data.frame(x):  object 'out.gamma.150.2.2' not found
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 5.

```
system.time(out.gamma.150.2.5 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma_simulator(beta = beta, n = n,
      bins = bins, shape = 5))
})))

## Error in r[, 1]/mad.predict.fun(out.mad, xx):  non-numeric argument to
binary operator
## Timing stopped at:  38.53 0.663 7.254
```

```
gamma.150.2.5 <- cbind(
  rowMeans(out.gamma.150.2.5, na.rm = TRUE),
  apply(out.gamma.150.2.5, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))

## Error in is.data.frame(x):  object 'out.gamma.150.2.5' not found
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 7.

```
system.time(out.gamma.150.2.7 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma_simulator(beta = beta, n = n,
      bins = bins, shape = 7))
})))

## Error in r[, 1]/mad.predict.fun(out.mad, xx):  non-numeric argument to
binary operator
## Timing stopped at:  47.88 0.738 8.801
```

```
gamma.150.2.7 <- cbind(
  rowMeans(out.gamma.150.2.7, na.rm = TRUE),
  apply(out.gamma.150.2.7, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))

## Error in is.data.frame(x):  object 'out.gamma.150.2.7' not found
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 10.

```
system.time(out.gamma.150.2.10 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma_simulator(beta = beta, n = n,
      bins = bins, shape = 10))
})))

## Error in r[, 1]/mad.predict.fun(out.mad, xx):  non-numeric argument to
binary operator
## Timing stopped at:  47.18 1.557 8.531
```

```
gamma.150.2.10 <- cbind(
  rowMeans(out.gamma.150.2.10, na.rm = TRUE),
  apply(out.gamma.150.2.10, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))

## Error in is.data.frame(x):  object 'out.gamma.150.2.10' not found
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 12.

```
system.time(out.gamma.150.2.12 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma_simulator(beta = beta, n = n,
      bins = bins, shape = 12))
})))

## Error in r[, 1]/mad.predict.fun(out.mad, xx):  non-numeric argument to
binary operator
## Timing stopped at:  49.33 0.623 8.555
```

```
gamma.150.2.12 <- cbind(
  rowMeans(out.gamma.150.2.12, na.rm = TRUE),
  apply(out.gamma.150.2.12, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))

## Error in is.data.frame(x):  object 'out.gamma.150.2.12' not found
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 15.

```
system.time(out.gamma.150.2.15 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma_simulator(beta = beta, n = n,
      bins = bins, shape = 15))
})))

## Error in r[, 1]/mad.predict.fun(out.mad, xx):  non-numeric argument to
binary operator
## Timing stopped at:  38.89 0.791 7.055
```

```
gamma.150.2.15 <- cbind(
  rowMeans(out.gamma.150.2.15, na.rm = TRUE),
  apply(out.gamma.150.2.15, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))

## Error in is.data.frame(x):  object 'out.gamma.150.2.15' not found
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 20.

```
system.time(out.gamma.150.2.20 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma_simulator(beta = beta, n = n,
      bins = bins, shape = 20))
})))

## Error in r[, 1]/mad.predict.fun(out.mad, xx):  non-numeric argument to
binary operator
## Timing stopped at:  34.47 0.768 6.439
```

```r
gamma.150.2.20 <- cbind(
  rowMeans(out.gamma.150.2.20, na.rm = TRUE),
  apply(out.gamma.150.2.20, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))

## Error in is.data.frame(x):  object 'out.gamma.150.2.20' not found
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 25.

```r
system.time(out.gamma.150.2.25 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma_simulator(beta = beta, n = n,
      bins = bins, shape = 25))
})))

## Error in r[, 1]/mad.predict.fun(out.mad, xx):  non-numeric argument to
binary operator
## Timing stopped at:  31.57 0.787 5.861
```

```r
gamma.150.2.25 <- cbind(
  rowMeans(out.gamma.150.2.25, na.rm = TRUE),
  apply(out.gamma.150.2.25, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))

## Error in is.data.frame(x):  object 'out.gamma.150.2.25' not found
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 30.

```r
system.time(out.gamma.150.2.30 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma_simulator(beta = beta, n = n,
      bins = bins, shape = 30))
})))

## Error in r[, 1]/mad.predict.fun(out.mad, xx):  non-numeric argument to
binary operator
## Timing stopped at:  38.46 0.766 6.823
```

```
gamma.150.2.30 <- cbind(
  rowMeans(out.gamma.150.2.30, na.rm = TRUE),
  apply(out.gamma.150.2.30, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))

## Error in is.data.frame(x):  object 'out.gamma.150.2.30' not found
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 40.

```
system.time(out.gamma.150.2.40 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma_simulator(beta = beta, n = n,
      bins = bins, shape = 40))
})))

## Error in r[, 1]/mad.predict.fun(out.mad, xx):  non-numeric argument to
binary operator
## Timing stopped at:  35.17 0.76 6.279
```

```
gamma.150.2.40 <- cbind(
  rowMeans(out.gamma.150.2.40, na.rm = TRUE),
  apply(out.gamma.150.2.40, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))

## Error in is.data.frame(x):  object 'out.gamma.150.2.40' not found
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 50.

```
system.time(out.gamma.150.2.50 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma_simulator(beta = beta, n = n,
      bins = bins, shape = 50))
})))

## Error in r[, 1]/mad.predict.fun(out.mad, xx):  non-numeric argument to
binary operator
## Timing stopped at:  39.38 1.183 7.213
```

```
gamma.150.2.50 <- cbind(
  rowMeans(out.gamma.150.2.50, na.rm = TRUE),
  apply(out.gamma.150.2.50, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))

## Error in is.data.frame(x):  object 'out.gamma.150.2.50' not found
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 60.

```
system.time(out.gamma.150.2.60 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma_simulator(beta = beta, n = n,
      bins = bins, shape = 60))
})))

## Error in r[, 1]/mad.predict.fun(out.mad, xx):  non-numeric argument to
binary operator
## Timing stopped at:  45.26 0.76 7.967
```

```
gamma.150.2.60 <- cbind(
  rowMeans(out.gamma.150.2.60, na.rm = TRUE),
  apply(out.gamma.150.2.60, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))

## Error in is.data.frame(x):  object 'out.gamma.150.2.60' not found
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 70.

```
system.time(out.gamma.150.2.70 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma_simulator(beta = beta, n = n,
      bins = bins, shape = 70))
})))

## Error in r[, 1]/mad.predict.fun(out.mad, xx):  non-numeric argument to
binary operator
## Timing stopped at:  37.39 0.786 6.828
```

```
gamma.150.2.70 <- cbind(
  rowMeans(out.gamma.150.2.70, na.rm = TRUE),
  apply(out.gamma.150.2.70, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))

## Error in is.data.frame(x):  object 'out.gamma.150.2.70' not found
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 80.

```
system.time(out.gamma.150.2.80 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma_simulator(beta = beta, n = n,
      bins = bins, shape = 80))
})))

## Error in r[, 1]/mad.predict.fun(out.mad, xx):  non-numeric argument to
binary operator
## Timing stopped at:  34.85 0.883 6.449
```

```
gamma.150.2.80 <- cbind(
  rowMeans(out.gamma.150.2.80, na.rm = TRUE),
  apply(out.gamma.150.2.80, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))

## Error in is.data.frame(x):  object 'out.gamma.150.2.80' not found
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 90.

```
system.time(out.gamma.150.2.90 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma_simulator(beta = beta, n = n,
      bins = bins, shape = 90))
})))

## Error in r[, 1]/mad.predict.fun(out.mad, xx):  non-numeric argument to
binary operator
## Timing stopped at:  33.05 1.506 6.207
```

```
gamma.150.2.90 <- cbind(
  rowMeans(out.gamma.150.2.90, na.rm = TRUE),
  apply(out.gamma.150.2.90, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))

## Error in is.data.frame(x):  object 'out.gamma.150.2.90' not found
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 100.

```
system.time(out.gamma.150.2.100 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma_simulator(beta = beta, n = n,
      bins = bins, shape = 100))
})))

## Error in r[, 1]/mad.predict.fun(out.mad, xx):  non-numeric argument to
binary operator
## Timing stopped at:  43.15 0.733 7.725
```

```
gamma.150.2.100 <- cbind(
  rowMeans(out.gamma.150.2.100, na.rm = TRUE),
  apply(out.gamma.150.2.100, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))

## Error in is.data.frame(x):  object 'out.gamma.150.2.100' not found
```

Reorganize the output.

```
para.area.gamma.150 <- nonpara.area.gamma.150 <-
  LS.area.gamma.150 <- LSLW.area.gamma.150 <-
  HD.area.gamma.150 <- NULL
para.error.gamma.150 <- nonpara.error.gamma.150 <-
  LS.error.gamma.150 <- LSLW.error.gamma.150 <-
  HD.error.gamma.150 <- NULL
para.marginal.gamma.150 <- nonpara.marginal.gamma.150 <-
  LS.marginal.gamma.150 <- LSLW.marginal.gamma.150 <-
  HD.marginal.gamma.150 <- NULL
para.local.gamma.150 <- nonpara.local.gamma.150 <-
```

```
  LS.local.gamma.150 <- LSLW.local.gamma.150 <-
  HD.local.gamma.150 <- NULL
para.inx.gamma.150 <- nonpara.inx.gamma.150 <-
  LS.inx.gamma.150 <- LSLW.inx.gamma.150 <-
  HD.inx.gamma.150 <- NULL
shapes <- c(0.5, 0.75, 1, 1.5, 2, 5, 7, 10, 12, 15, 20, 25,
  30, 40, 50, 60, 70, 80, 90, 100)
for(j in shapes ){
  internal.output <-
    eval(parse(text=paste("gamma.150.2", j, sep = ".")))
  k <- which(shapes == j)
  para.marginal.gamma.150[k] <- as.numeric(internal.output[1, 1])
  para.local.gamma.150[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[2:3, 1])
  para.inx.gamma.150[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[4:28, 1])
  para.error.gamma.150[k] <- as.numeric(internal.output[29, 1])
  para.area.gamma.150[k] <- as.numeric(internal.output[30, 1])
  nonpara.marginal.gamma.150[k] <- as.numeric(internal.output[31, 1])
  nonpara.local.gamma.150[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[32:33, 1])
  nonpara.inx.gamma.150[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[34:58, 1])
  nonpara.error.gamma.150[k] <- as.numeric(internal.output[59, 1])
  nonpara.area.gamma.150[k] <- as.numeric(internal.output[60, 1])
  LS.marginal.gamma.150[k] <- as.numeric(internal.output[61, 1])
  LS.local.gamma.150[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[62:63, 1])
  LS.inx.gamma.150[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[64:88, 1])
  LS.error.gamma.150[k] <- as.numeric(internal.output[89, 1])
  LS.area.gamma.150[k] <- as.numeric(internal.output[90, 1])
  LSLW.marginal.gamma.150[k] <- as.numeric(internal.output[91, 1])
  LSLW.local.gamma.150[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[92:93, 1])
  LSLW.inx.gamma.150[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[94:118, 1])
  LSLW.error.gamma.150[k] <- as.numeric(internal.output[119, 1])
  LSLW.area.gamma.150[k] <- as.numeric(internal.output[120, 1])
  HD.marginal.gamma.150[k] <- as.numeric(internal.output[121, 1])
  HD.local.gamma.150[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[122:123, 1])
  HD.inx.gamma.150[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[124:148, 1])
  HD.error.gamma.150[k] <- as.numeric(internal.output[149, 1])
  HD.area.gamma.150[k] <- as.numeric(internal.output[150, 1])
}
```

```
## Error in eval(parse(text = paste("gamma.150.2", j, sep = "."))):  object
'gamma.150.2.0.5' not found
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 250$ and shape = 0.5.

```
set.seed(13)
beta <- c(1.25, -1)
bins <- 3
n <- 250
B <- 50
system.time(out.gamma.250.3.0.5 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma.simulator(beta = beta, n = n,
      bins = bins, shape = 0.5))
})))

##     user   system  elapsed
## 2511.339   16.860 1496.168
```

```
gamma.250.3.0.5 <- cbind(
  rowMeans(out.gamma.250.3.0.5, na.rm = TRUE),
  apply(out.gamma.250.3.0.5, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 250$ and shape = 0.75.

```
system.time(out.gamma.250.3.0.75 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma.simulator(beta = beta, n = n,
      bins = bins, shape = 0.75))
})))

##     user   system  elapsed
## 2577.735   16.795 1500.989
```

```
gamma.250.3.0.75 <- cbind(
  rowMeans(out.gamma.250.3.0.75, na.rm = TRUE),
  apply(out.gamma.250.3.0.75, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 250$ and shape = 1.

```
system.time(out.gamma.250.3.1 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma.simulator(beta = beta, n = n,
      bins = bins, shape = 1))
})))

##     user   system  elapsed
## 2694.797   16.807 1523.346
```

```
gamma.250.3.1 <- cbind(
  rowMeans(out.gamma.250.3.1, na.rm = TRUE),
  apply(out.gamma.250.3.1, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 250$ and shape = 1.5.

```
system.time(out.gamma.250.3.1.5 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma.simulator(beta = beta, n = n,
      bins = bins, shape = 1.5))
})))

##     user   system  elapsed
## 2820.080   17.131 1561.781
```

```
gamma.250.3.1.5 <- cbind(
  rowMeans(out.gamma.250.3.1.5, na.rm = TRUE),
  apply(out.gamma.250.3.1.5, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 250$ and shape = 2.

```r
system.time(out.gamma.250.3.2 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma.simulator(beta = beta, n = n,
      bins = bins, shape = 2))
})))

##     user   system  elapsed
## 3105.011   16.995 1603.930
```

```r
gamma.250.3.2 <- cbind(
  rowMeans(out.gamma.250.3.2, na.rm = TRUE),
  apply(out.gamma.250.3.2, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 250$ and shape = 5.

```r
system.time(out.gamma.250.3.5 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma.simulator(beta = beta, n = n,
      bins = bins, shape = 5))
})))

##     user   system  elapsed
## 2969.224   17.372 1572.685
```

```r
gamma.250.3.5 <- cbind(
  rowMeans(out.gamma.250.3.5, na.rm = TRUE),
  apply(out.gamma.250.3.5, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 250$ and shape = 7.

```r
system.time(out.gamma.250.3.7 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma.simulator(beta = beta, n = n,
      bins = bins, shape = 7))
})))

##     user   system  elapsed
## 2935.660   17.456 1564.401
```

```r
gamma.250.3.7 <- cbind(
  rowMeans(out.gamma.250.3.7, na.rm = TRUE),
  apply(out.gamma.250.3.7, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 250$ and shape = 10.

```r
system.time(out.gamma.250.3.10 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma.simulator(beta = beta, n = n,
      bins = bins, shape = 10))
})))

##     user   system  elapsed
## 2842.656   17.363 1539.173
```

```r
gamma.250.3.10 <- cbind(
  rowMeans(out.gamma.250.3.10, na.rm = TRUE),
  apply(out.gamma.250.3.10, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 250$ and shape = 12.

```r
system.time(out.gamma.250.3.12 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma.simulator(beta = beta, n = n,
      bins = bins, shape = 12))
})))

##     user   system  elapsed
## 2760.976   17.350 1526.356
```

```r
gamma.250.3.12 <- cbind(
  rowMeans(out.gamma.250.3.12, na.rm = TRUE),
  apply(out.gamma.250.3.12, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 250$ and shape = 15.

```
system.time(out.gamma.250.3.15 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma.simulator(beta = beta, n = n,
      bins = bins, shape = 15))
})))

##     user   system  elapsed
## 2729.205   17.260 1519.564
```

```
gamma.250.3.15 <- cbind(
  rowMeans(out.gamma.250.3.15, na.rm = TRUE),
  apply(out.gamma.250.3.15, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 250$ and shape = 20.

```
system.time(out.gamma.250.3.20 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma.simulator(beta = beta, n = n,
      bins = bins, shape = 20))
})))

##     user   system  elapsed
## 2701.140   17.335 1504.800
```

```
gamma.250.3.20 <- cbind(
  rowMeans(out.gamma.250.3.20, na.rm = TRUE),
  apply(out.gamma.250.3.20, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 250$ and shape = 25.

```
system.time(out.gamma.250.3.25 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma.simulator(beta = beta, n = n,
      bins = bins, shape = 25))
})))

##     user   system  elapsed
## 2881.666   17.454 1544.467
```

```
gamma.250.3.25 <- cbind(
  rowMeans(out.gamma.250.3.25, na.rm = TRUE),
  apply(out.gamma.250.3.25, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

Reorganize the output.

```
para.area.gamma.250 <- nonpara.area.gamma.250 <-
  LS.area.gamma.250 <- LSLW.area.gamma.250 <-
  HD.area.gamma.250 <- NULL
para.error.gamma.250 <- nonpara.error.gamma.250 <-
  LS.error.gamma.250 <- LSLW.error.gamma.250 <-
  HD.error.gamma.250 <- NULL
para.marginal.gamma.250 <- nonpara.marginal.gamma.250 <-
  LS.marginal.gamma.250 <- LSLW.marginal.gamma.250 <-
  HD.marginal.gamma.250 <- NULL
para.local.gamma.250 <- nonpara.local.gamma.250 <-
  LS.local.gamma.250 <- LSLW.local.gamma.250 <-
  HD.local.gamma.250 <- NULL
para.inx.gamma.250 <- nonpara.inx.gamma.250 <-
  LS.inx.gamma.250 <- LSLW.inx.gamma.250 <-
  HD.inx.gamma.250 <- NULL
shapes <- c(0.5, 0.75, 1, 1.5, 2, 5, 7, 10, 12, 15, 20, 25)
for(j in shapes ){
  internal.output <- eval(parse(text=paste("gamma.250.3", j, sep = ".")))
  k <- which(shapes == j)
  para.marginal.gamma.250[k] <- as.numeric(internal.output[1, 1])
  para.local.gamma.250[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[2:4, 1])
  para.inx.gamma.250[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[5:29, 1])
  para.error.gamma.250[k] <- as.numeric(internal.output[30, 1])
  para.area.gamma.250[k] <- as.numeric(internal.output[31, 1])
  nonpara.marginal.gamma.250[k] <- as.numeric(internal.output[32, 1])
  nonpara.local.gamma.250[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[33:35, 1])
  nonpara.inx.gamma.250[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[36:60, 1])
  nonpara.error.gamma.250[k] <- as.numeric(internal.output[61, 1])
  nonpara.area.gamma.250[k] <- as.numeric(internal.output[62, 1])
  LS.marginal.gamma.250[k] <- as.numeric(internal.output[63, 1])
  LS.local.gamma.250[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[64:66, 1])
  LS.inx.gamma.250[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[67:91, 1])
  LS.error.gamma.250[k] <- as.numeric(internal.output[92, 1])
  LS.area.gamma.250[k] <- as.numeric(internal.output[93, 1])
  LSLW.marginal.gamma.250[k] <- as.numeric(internal.output[94, 1])
  LSLW.local.gamma.250[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[95:97, 1])
  LSLW.inx.gamma.250[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[98:122, 1])
  LSLW.error.gamma.250[k] <- as.numeric(internal.output[123, 1])
  LSLW.area.gamma.250[k] <- as.numeric(internal.output[124, 1])
  HD.marginal.gamma.250[k] <- as.numeric(internal.output[125, 1])
  HD.local.gamma.250[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[126:128, 1])
  HD.inx.gamma.250[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[129:153, 1])
  HD.error.gamma.250[k] <- as.numeric(internal.output[154, 1])
  HD.area.gamma.250[k] <- as.numeric(internal.output[155, 1])
}
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 500$ and shape = 0.5.

```
set.seed(13)
beta <- c(1.25, -1)
bins <- 3
n <- 500
B <- 50
system.time(out.gamma.500.3.0.5 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma.simulator(beta = beta, n = n,
      bins = bins, shape = 0.5))
})))

##     user   system  elapsed
## 9094.705   23.855 5496.216
```

```r
gamma.500.3.0.5 <- cbind(
  rowMeans(out.gamma.500.3.0.5, na.rm = TRUE),
  apply(out.gamma.500.3.0.5, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 500$ and shape = 0.75.

```r
system.time(out.gamma.500.3.0.75 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma.simulator(beta = beta, n = n,
      bins = bins, shape = 0.75))
})))

##     user   system  elapsed
## 8411.859   21.506 4823.067
```

```r
gamma.500.3.0.75 <- cbind(
  rowMeans(out.gamma.500.3.0.75, na.rm = TRUE),
  apply(out.gamma.500.3.0.75, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 500$ and shape = 1.

```r
system.time(out.gamma.500.3.1 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma.simulator(beta = beta, n = n,
      bins = bins, shape = 1))
})))

##     user   system  elapsed
## 9375.044   23.025 5350.013
```

```r
gamma.500.3.1 <- cbind(
  rowMeans(out.gamma.500.3.1, na.rm = TRUE),
  apply(out.gamma.500.3.1, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 500$ and shape = 1.5.

```
system.time(out.gamma.500.3.1.5 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma.simulator(beta = beta, n = n,
      bins = bins, shape = 1.5))
})))

##     user   system  elapsed
## 9709.719   19.786 5023.810
```

```
gamma.500.3.1.5 <- cbind(
  rowMeans(out.gamma.500.3.1.5, na.rm = TRUE),
  apply(out.gamma.500.3.1.5, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 500$ and shape = 2.

```
system.time(out.gamma.500.3.2 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma.simulator(beta = beta, n = n,
      bins = bins, shape = 2))
})))

##      user    system   elapsed
## 11685.253    20.810  5403.206
```

```
gamma.500.3.2 <- cbind(
  rowMeans(out.gamma.500.3.2, na.rm = TRUE),
  apply(out.gamma.500.3.2, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 500$ and shape = 5.

```
system.time(out.gamma.500.3.5 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma.simulator(beta = beta, n = n,
      bins = bins, shape = 5))
})))

##      user    system   elapsed
## 10175.065    20.472  5060.978
```

```
gamma.500.3.5 <- cbind(
  rowMeans(out.gamma.500.3.5, na.rm = TRUE),
  apply(out.gamma.500.3.5, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 500$ and shape = 7.

```
system.time(out.gamma.500.3.7 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma.simulator(beta = beta, n = n,
      bins = bins, shape = 7))
})))

##      user    system   elapsed
## 10187.202    20.175 12619.584
```

```
gamma.500.3.7 <- cbind(
  rowMeans(out.gamma.500.3.7, na.rm = TRUE),
  apply(out.gamma.500.3.7, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 500$ and shape = 10.

```
system.time(out.gamma.500.3.10 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma.simulator(beta = beta, n = n,
      bins = bins, shape = 10))
})))
```

```
##       user    system   elapsed
## 10417.609    20.990  5142.265
```

```
gamma.500.3.10 <- cbind(
  rowMeans(out.gamma.500.3.10, na.rm = TRUE),
  apply(out.gamma.500.3.10, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 500$ and shape = 12.

```
system.time(out.gamma.500.3.12 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma.simulator(beta = beta, n = n,
      bins = bins, shape = 12))
})))

##      user   system  elapsed
## 9750.862   19.680 4878.616
```

```
gamma.500.3.12 <- cbind(
  rowMeans(out.gamma.500.3.12, na.rm = TRUE),
  apply(out.gamma.500.3.12, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 500$ and shape = 15.

```
system.time(out.gamma.500.3.15 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma.simulator(beta = beta, n = n,
      bins = bins, shape = 15))
})))

##      user   system  elapsed
## 9450.264   19.315 4806.449
```

```
gamma.500.3.15 <- cbind(
  rowMeans(out.gamma.500.3.15, na.rm = TRUE),
  apply(out.gamma.500.3.15, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 500$ and shape = 20.

```
system.time(out.gamma.500.3.20 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma.simulator(beta = beta, n = n,
      bins = bins, shape = 20))
})))

##     user   system  elapsed
## 9359.494   18.956 4786.326
```

```
gamma.500.3.20 <- cbind(
  rowMeans(out.gamma.500.3.20, na.rm = TRUE),
  apply(out.gamma.500.3.20, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 500$ and shape = 25.

```
system.time(out.gamma.500.3.25 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(gamma.simulator(beta = beta, n = n,
      bins = bins, shape = 25))
})))

##     user   system  elapsed
## 9465.436   19.651 4784.069
```

```
gamma.500.3.25 <- cbind(
  rowMeans(out.gamma.500.3.25, na.rm = TRUE),
  apply(out.gamma.500.3.25, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

Reorganize the output.

```
para.area.gamma.500 <- nonpara.area.gamma.500 <-
  LS.area.gamma.500 <- LSLW.area.gamma.500 <-
  HD.area.gamma.500 <- NULL
para.error.gamma.500 <- nonpara.error.gamma.500 <-
  LS.error.gamma.500 <- LSLW.error.gamma.500 <-
  HD.error.gamma.500 <- NULL
para.marginal.gamma.500 <- nonpara.marginal.gamma.500 <-
  LS.marginal.gamma.500 <- LSLW.marginal.gamma.500 <-
  HD.marginal.gamma.500 <- NULL
para.local.gamma.500 <- nonpara.local.gamma.500 <-
  LS.local.gamma.500 <- LSLW.local.gamma.500 <-
  HD.local.gamma.500 <- NULL
para.inx.gamma.500 <- nonpara.inx.gamma.500 <-
  LS.inx.gamma.500 <- LSLW.inx.gamma.500 <-
  HD.inx.gamma.500 <- NULL
shapes <- c(0.5, 0.75, 1, 1.5, 2, 5, 7, 10, 12, 15, 20, 25)
for(j in shapes ){
  internal.output <- eval(parse(text=paste("gamma.500.3", j, sep = ".")))
  k <- which(shapes == j)
  para.marginal.gamma.500[k] <- as.numeric(internal.output[1, 1])
  para.local.gamma.500[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[2:4, 1])
  para.inx.gamma.500[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[5:29, 1])
  para.error.gamma.500[k] <- as.numeric(internal.output[30, 1])
  para.area.gamma.500[k] <- as.numeric(internal.output[31, 1])
  nonpara.marginal.gamma.500[k] <- as.numeric(internal.output[32, 1])
  nonpara.local.gamma.500[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[33:35, 1])
  nonpara.inx.gamma.500[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[36:60, 1])
  nonpara.error.gamma.500[k] <- as.numeric(internal.output[61, 1])
  nonpara.area.gamma.500[k] <- as.numeric(internal.output[62, 1])
  LS.marginal.gamma.500[k] <- as.numeric(internal.output[63, 1])
  LS.local.gamma.500[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[64:66, 1])
  LS.inx.gamma.500[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[67:91, 1])
  LS.error.gamma.500[k] <- as.numeric(internal.output[92, 1])
  LS.area.gamma.500[k] <- as.numeric(internal.output[93, 1])
  LSLW.marginal.gamma.500[k] <- as.numeric(internal.output[94, 1])
  LSLW.local.gamma.500[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[95:97, 1])
  LSLW.inx.gamma.500[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[98:122, 1])
  LSLW.error.gamma.500[k] <- as.numeric(internal.output[123, 1])
  LSLW.area.gamma.500[k] <- as.numeric(internal.output[124, 1])
  HD.marginal.gamma.500[k] <- as.numeric(internal.output[125, 1])
  HD.local.gamma.500[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[126:128, 1])
  HD.inx.gamma.500[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[129:153, 1])
  HD.error.gamma.500[k] <- as.numeric(internal.output[154, 1])
  HD.area.gamma.500[k] <- as.numeric(internal.output[155, 1])
}
```

5.2 Results

Results form our simulations are depicted in Figures 5-10. For all five prediction regions we depict the estimatated area, prediction error, and local coverage probabilities with respect to binning in Figures 5, 7, and 9 for $n = 150$, 250, and 500 respectively. For all five prediction regions we depict the local coverage probabilities across x in Figures 6, 8, and 10 for $n = 150$, 250, and 500 respectively.

From these simulations we see that the parametric conformal prediction region is similar to the HD prediciton region in area, prediction error, and appearance (as seen in Section 8.1). Moreover, the parametric conformal prediction region possess finite-sample, albeit slightly conservative, local validity with respect to binning and possess nominal finite-sample conditional coverage over most of the support. The LSLW conformal prediction region is smaller than than both the parametric conformal prediction region and the HD prediction region. However, it has a higher prediciton error than these regions and it is more conservative than the parametric conformal prediction region in most of our simulation settings. It also does not fit the data well when the deviations about the estimated mean function are clearly not symmetric as evidenced by in Section 8.1. These figures correspond to data generated from a Gamma distribution with small shape parameter values. The nonparametric conformal prediction region gives closer to nominal coverage and it possesses finite-sample local validity with respect to binning. However, this prediction region is larger and gives higher prediciton error than the parametric and LSLW conformal prediction regions and HD prediction region. The LS conformal prediciton region provides extreme overcoverage at small values of x and extreme undercoverage at large values of x. This prediction region is also larger and gives higher prediciton error than the parametric and LSLW conformal prediction regions and HD prediction region.

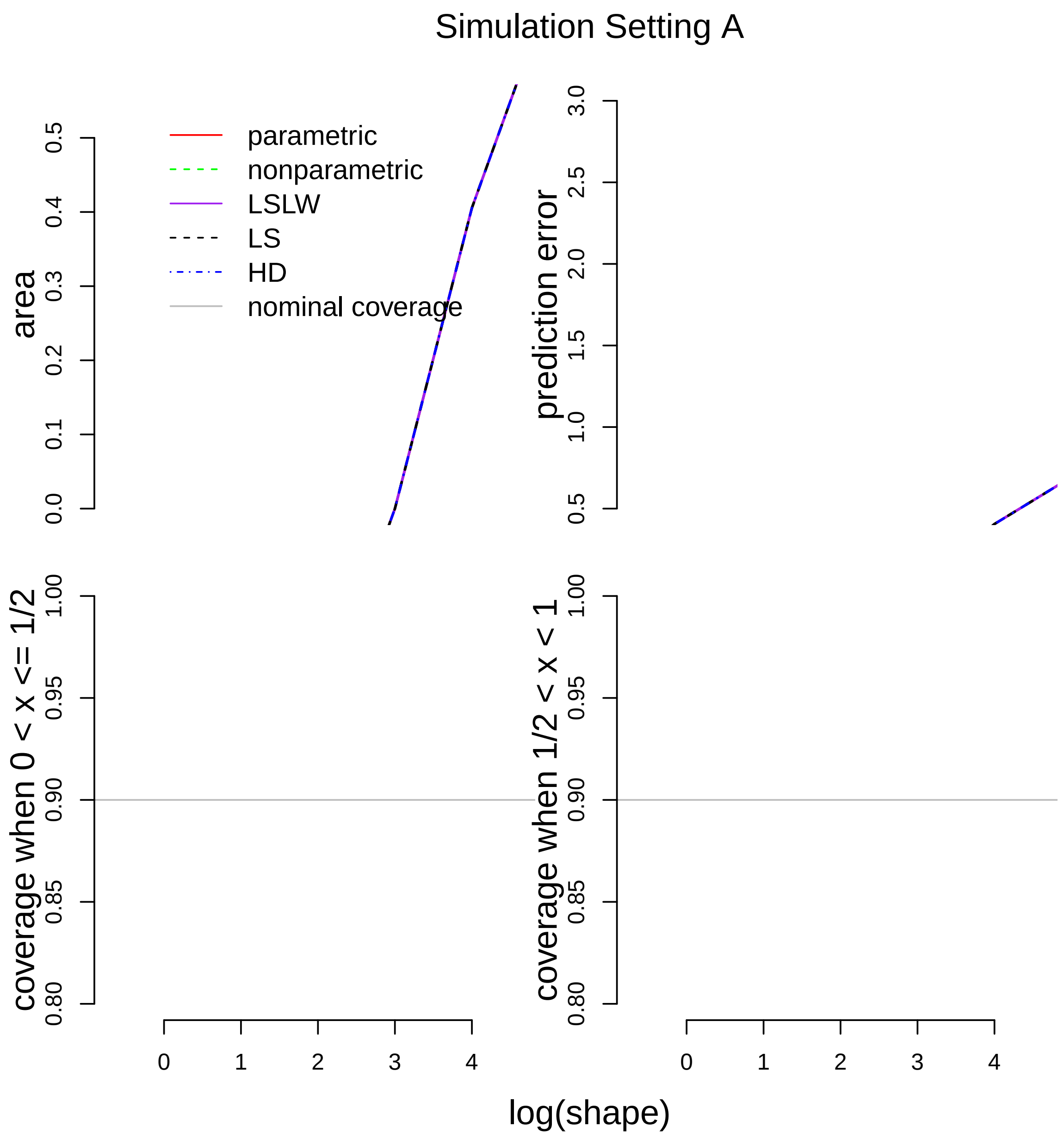


Figure 5: This figure compares the performance of the parametric, nonparametric, least squares, and least squares locally weighted conformal prediciton region and the highest density predcition region when $n = 150$ and the number of bins equals 2. The specific diagnostics used to compare these prediciton regions is the area (top-left panel), prediction error (top-right panel), and the coverage probability with respect to binning (bottom row) across shape parameter values. The average of 250 Monte Carlo samples at each shape parameter value in these simulation settings form the lines that are depicted in this figure.

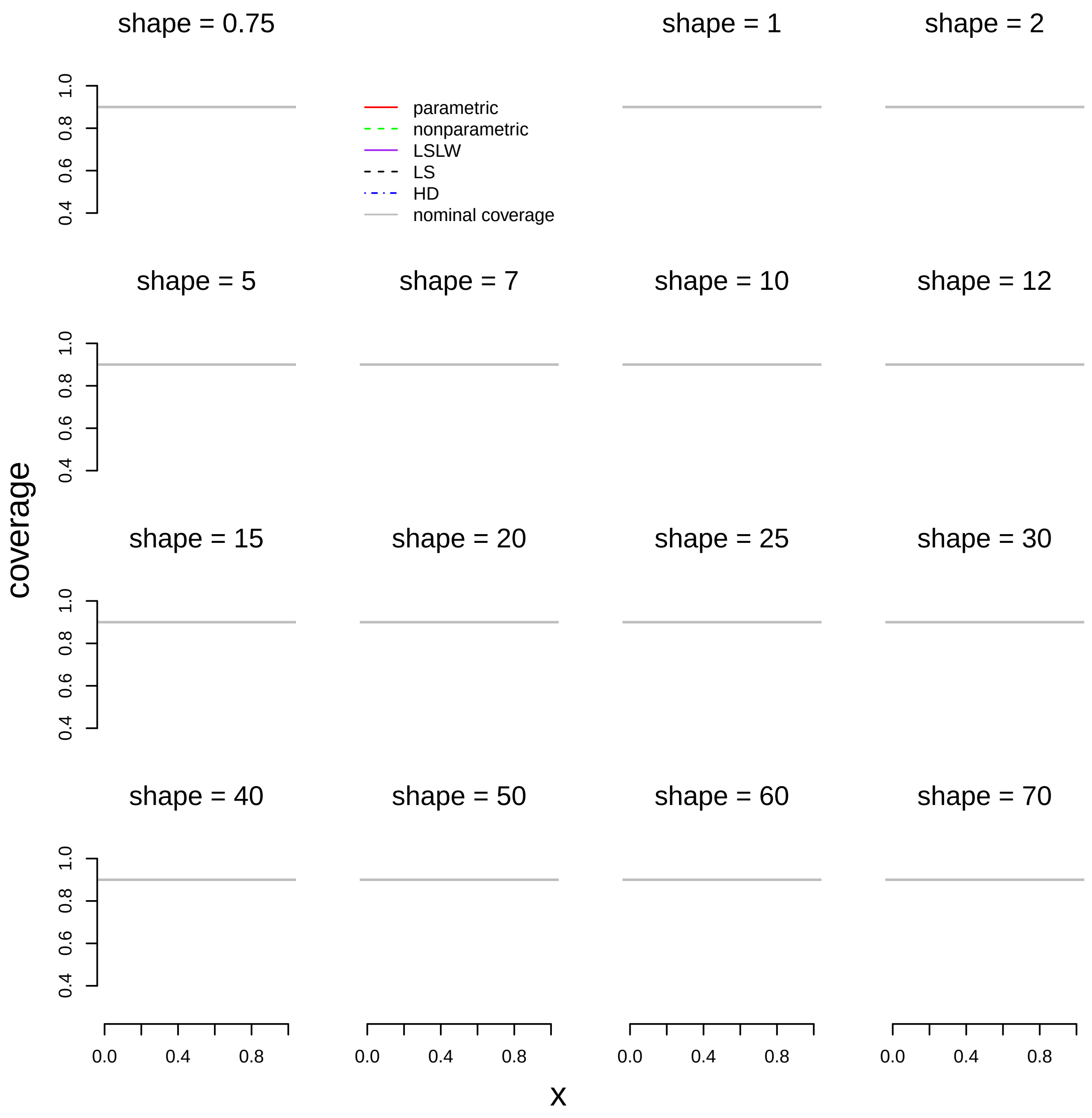


Figure 6: Plot of the estimated coverage probabilities of prediction regions across x and shape parameter values when the model is correctly specified, $n = 150$, and the number of bins is equal to 2.

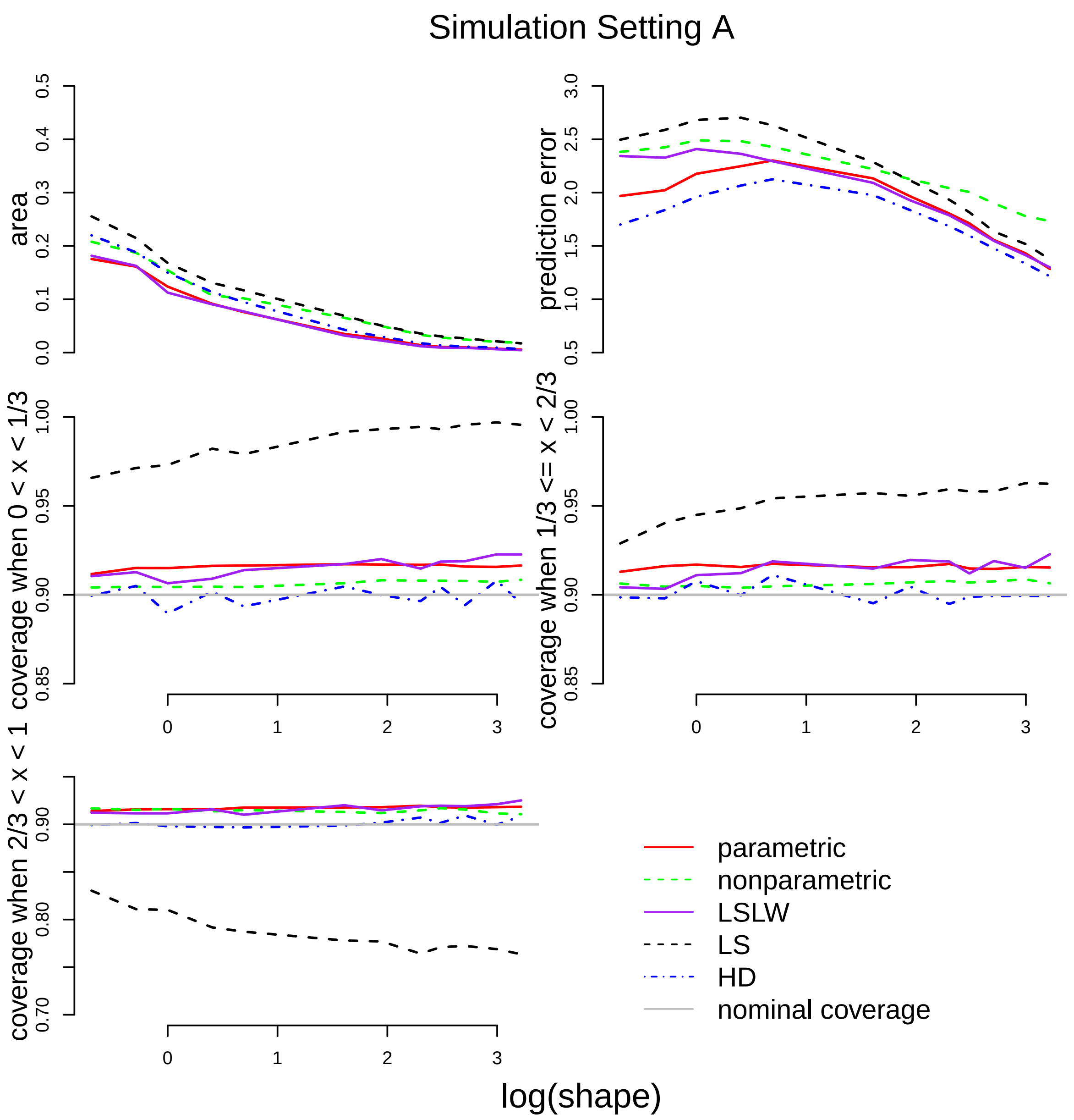


Figure 7: This figure compares the performance of the parametric, nonparametric, least squares, and least squares locally weighted conformal prediciton region and the highest density prediction region when $n = 250$ and the number of bins equals 3. The specific diagnostics used to compare these prediciton regions is the area (top-left panel), prediction error (top-right panel), and the coverage probability with respect to binning (bottom row) across shape parameter values. The average of 50 Monte Carlo samples at each shape parameter value in these simulation settings form the lines that are depicted in this figure.

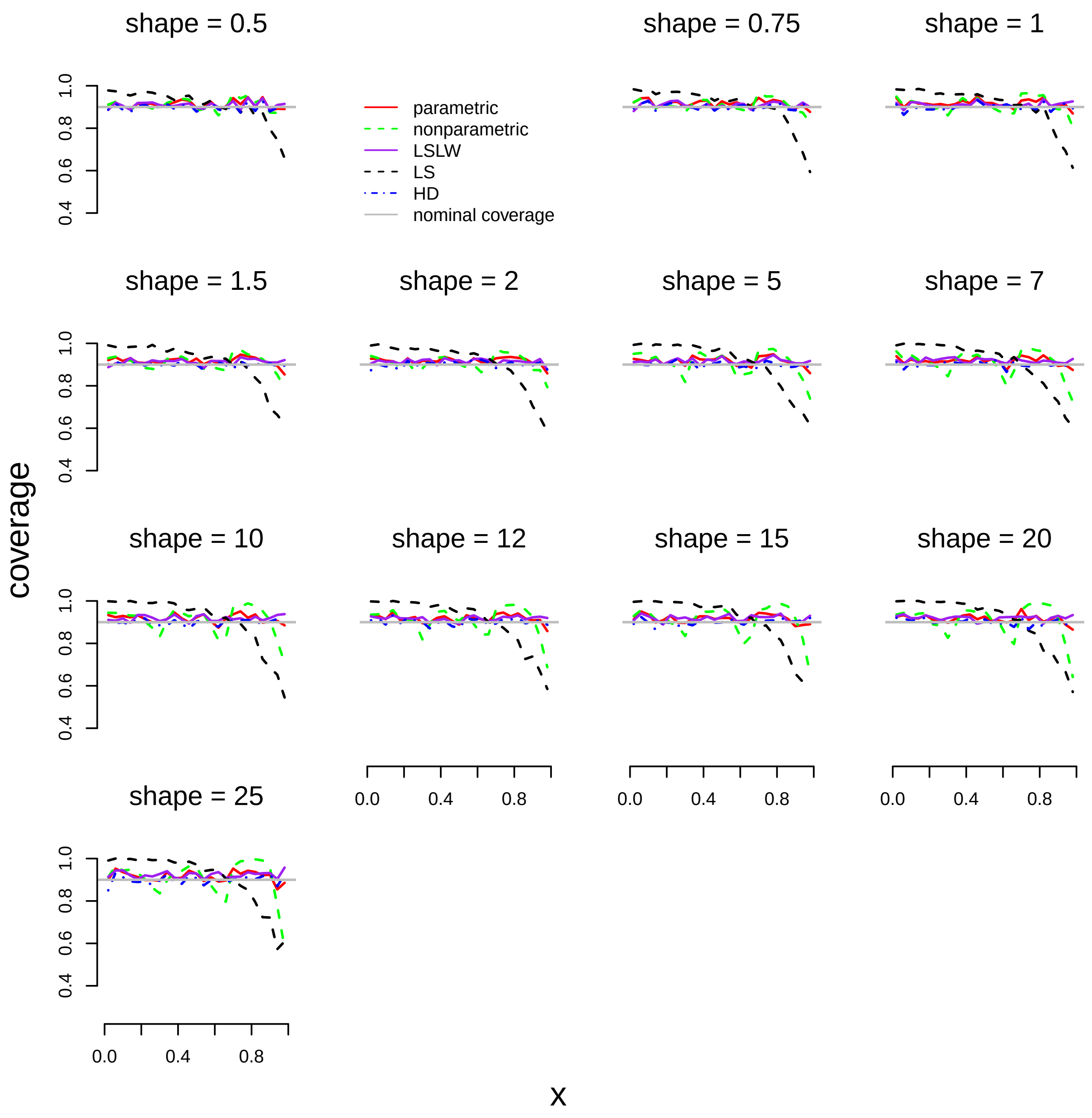


Figure 8: Plot of the estimated coverage probabilities of prediction regions across x and shape parameter values when the model is correctly specified, $n = 250$, and the number of bins is equal to 3.

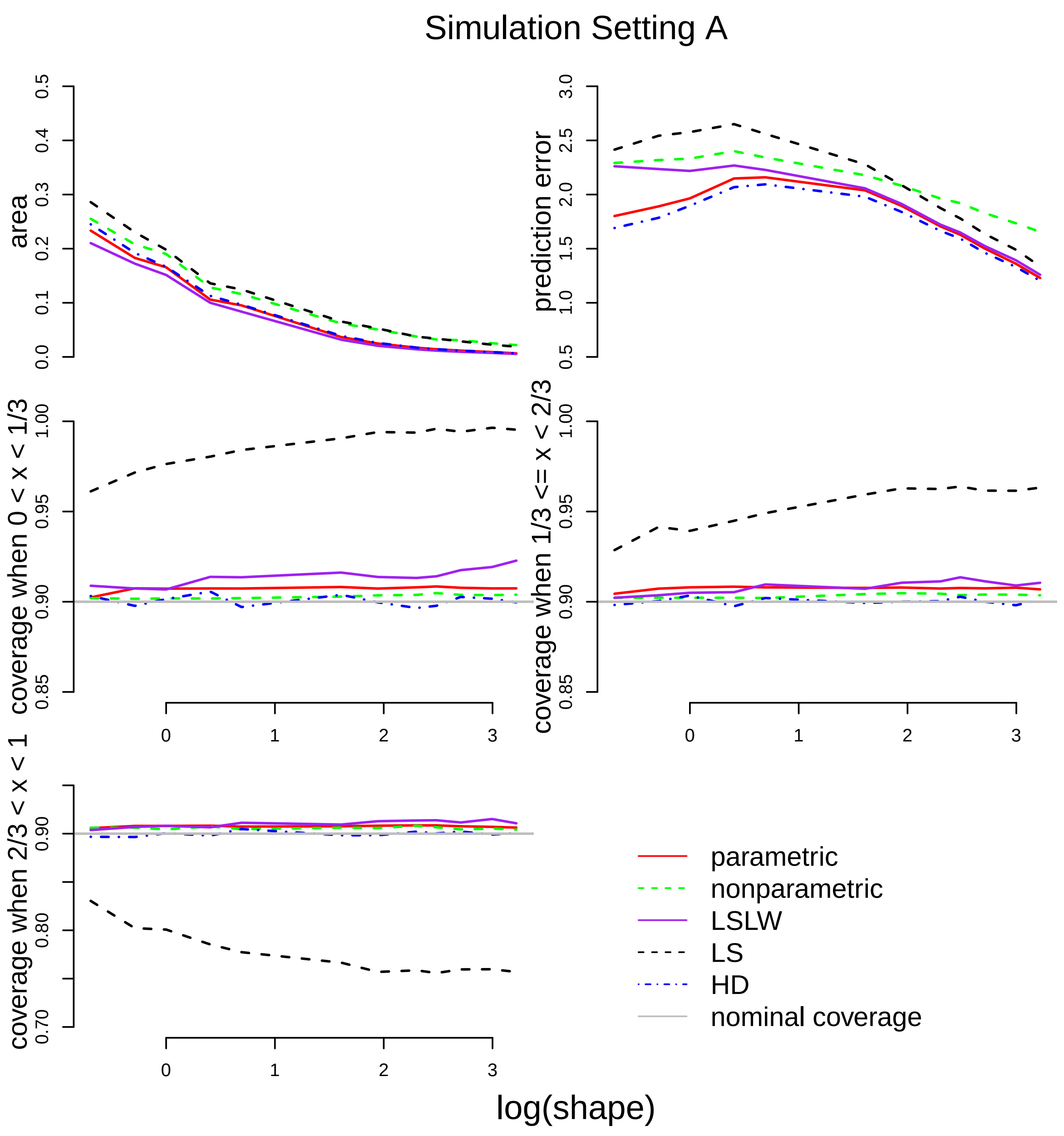


Figure 9: Diagnostic plots for prediction regions when the model is correctly specified, $n = 500$, and the number of bins is equal to 3.

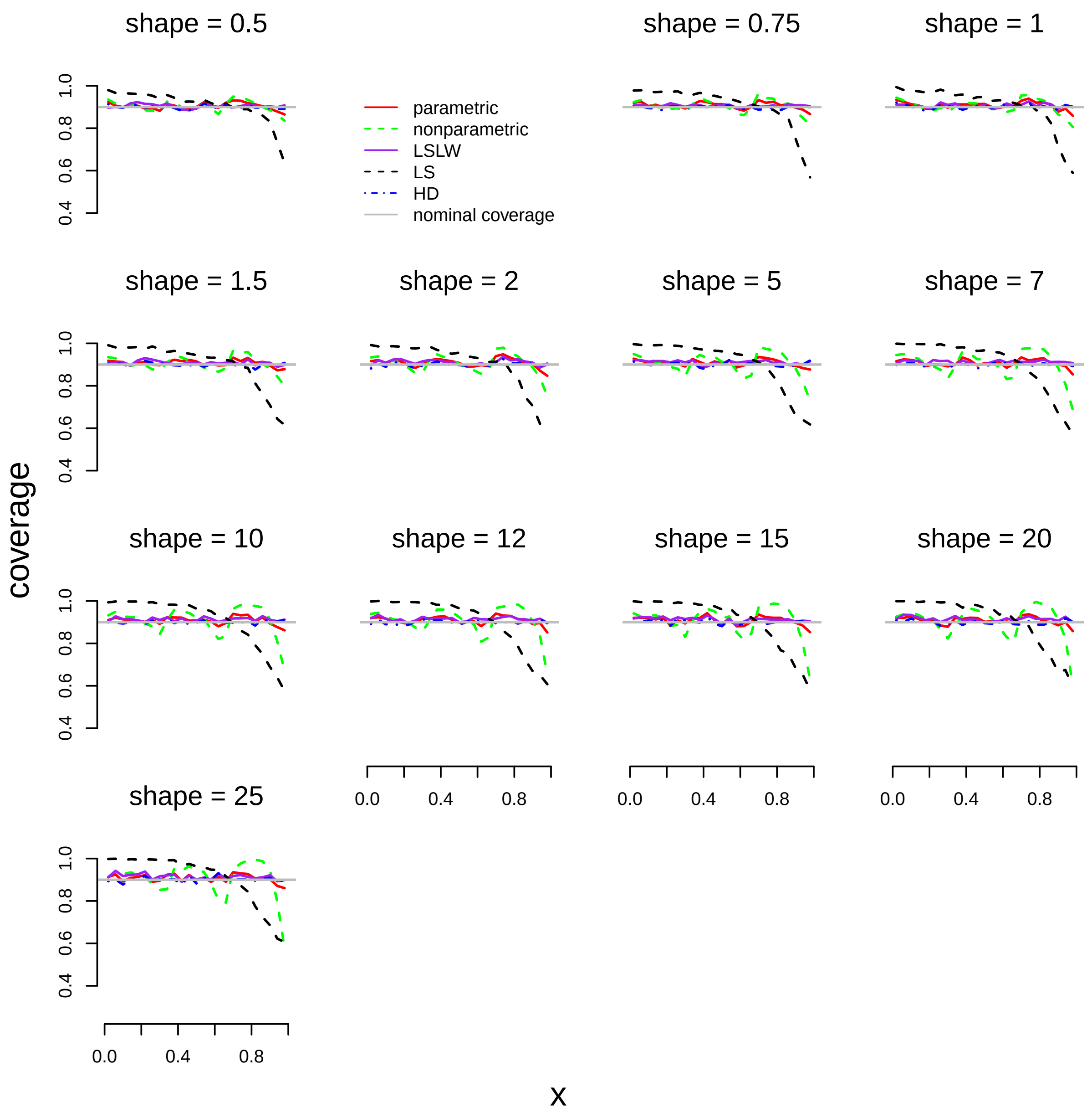


Figure 10: This figure compares the performance of the parametric, nonparametric, least squares, and least squares locally weighted conformal prediciton region and the highest density prediction region when $n = 550$ and the number of bins equals 3. The specific diagnostics used to compare these prediciton regions is the area (top-left panel), prediction error (top-right panel), and the coverage probability with respect to binning (bottom row) across shape parameter values. The average of 50 Monte Carlo samples at each shape parameter value in these simulation settings form the lines that are depicted in this figure.

6 Gamma-Gaussian model misspecification

In this Section, we compare the parametric conformal prediction region, the nonparametric conformal prediction region, the LSLW conformal prediction region, the LS conformal prediction region, and the HD prediction region under model misspecification. The data generating process is Gamma with an inverse link function and we set $\beta = (0.5, 1)^T$. We consider sample sizes of $n \in \{150, 250, 500\}$ and shape parameter values of $\{5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$ when $n = 150$ and shape parameter values of $\{5, 20, 40, 60, 80, 100\}$ when $n = 250, 500$. This value of β and these shape parameter values are chosen so that errors about a cubic regression model appear to be almost symmetric. In this analysis the parametric, LSLW, and LS conformal prediction regions and the HD prediction region are fit assuming this misspecified cubic regression model with homoscedastic normal errors. When $n = 150$ we build the parametric and nonparametric conformal prediction regions using 2 bins. When $n = 250, 500$ we build the parametric and nonparametric conformal prediction regions using 3 bins. These number of bin choices correspond to the bin width asymptotics of Lei and Wasserman [2014].

6.1 Simulations

The following function computes our diagnostic measures for the five prediction regions under investigation in the univariate case where data is assumed to be Gamma with inverse link function and the fitted model is Gaussian with a cubic fit.

```r
misspec.simulator <- function(n = 500, alpha = 0.10, beta,
  shape, bins = 3, family = "Gamma", link = "inverse",
  confamily = "gaussian", parametric = TRUE,
  nonparametric = TRUE, LS = TRUE, LSLW = TRUE, HD = TRUE,
  oracle = FALSE, cores = 6){

  p <- d <- length(beta) - 1
  x <- matrix(runif(n), ncol = p)
  y <- rep(0,n)
  data <- NULL

  ## set up partition
  if(class(bins) == "NULL"){
    wn <- min(1/ floor(1 / (log(n)/n)^(1/(d+3))), 1/2)
    bins <- 1 / wn
  }

  ## generate the data (has functionality for different
  ## families and link functions)
  if(family == "Gamma"){
    if(link == "identity"){
      rate <- (1 / (cbind(1, x) %*% beta)) * shape
      y <- rgamma(n = n, shape = shape, rate = rate)
      y <- y / sd(y)
      data <- data.frame(y = y, x = x)
      colnames(data)[2:(p+1)] <- paste("x", 1:p, sep = "")
    }
    if(link == "inverse"){
      rate <- (cbind(1, x) %*% beta) * shape
      y <- rgamma(n = n, shape = shape, rate = rate)
      y <- y / sd(y)
      data <- data.frame(y = y, x = x)
      colnames(data)[2:(p+1)] <- paste("x", 1:p, sep = "")
    }
    if(link == "log"){
      rate <- (1 / exp(cbind(1, x) %*% beta)) * shape
      y <- rgamma(n = n, shape = shape, rate = rate)
      y <- y / sd(y)
      data <- data.frame(y = y, x = x)
      colnames(data)[2:(p+1)] <- paste("x", 1:p, sep = "")
    }
  }

  if(family == "gaussian"){
    mu <- cbind(1, x) %*% beta
    y <- rnorm(n = n, mean = mu, sd = sd)
    data <- data.frame(y = y, x = x)
    colnames(data)[2:(p+1)] <- paste("x", 1:p, sep = "")
  }

  if(family == "inverse.gaussian"){
    mu = 1 / sqrt(cbind(1, x) %*% beta)
    y <- rinvgauss(n = n, mean = mu)
    y <- y / sd(y)
    data <- data.frame(y = y, x = x)
    colnames(data)[2:(p+1)] <- paste("x", 1:p, sep = "")
  }

  ## fit the misspecified cubic regression model
  fit <- glm(y ~ x1 + I(x1^2) + I(x1^3), family = confamily,
    data = data)
  paraCI <- nonparaCI <- LSCI <- LSLWCI <- HDCI <-
    trueHDCI <- NULL
  formula <- fit$formula
  newdata <- data
  respname <- all.vars(formula)[1]
  newdata <- newdata[, !(colnames(data) %in% respname)]
  newdata <- as.matrix(newdata)

  ## obtain the prediction regions
  if(parametric){
    cpred <- conformal.glm(fit, parametric = TRUE,
      nonparametric = FALSE, alpha = alpha,
      bins = bins, cores = cores)
    paraCI <- cpred$paraconformal
  }
  if(nonparametric){
    cpred <- conformal.glm(fit, parametric = FALSE,
      nonparametric = TRUE, alpha = alpha,
      bins = bins, cores = cores)
    nonparaCI <- cpred$nonparaconformal
  }
  if(LS){
    p1.tibs <- conformal.pred(x = cbind(x,x^2,x^3), y = y,
      x0 = cbind(x,x^2,x^3),
      train.fun = train.fun, predict.fun = predict.fun,
      alpha = alpha)
    LSCI <- cbind(p1.tibs$lo, p1.tibs$up)
  }
  if(LSLW){
    cubic.model <- lm(y ~ x + I(x^2) + I(x^3))
    abs.resid <- abs(cubic.model$resid)
    smooth.call <- smooth.spline(x, abs.resid,
      nknots = 10)
    lambda <- smooth.call$lambda
    df <- smooth.call$df
    mad.train.fun <- function(x, y, out = NULL){
      smooth.spline(x[, 1], y, lambda = lambda,
      df = df, nknots = 10)
    }
    p2.tibs <- conformal.pred(x = cbind(x,x^2,x^3), y = y,
      x0 = cbind(x,x^2,x^3),
      train.fun = train.fun, predict.fun = predict.fun,
      mad.train.fun = mad.train.fun,
      mad.predict.fun = mad.predict.fun,
      alpha = alpha)
    LSLWCI <- cbind(p2.tibs$lo, p2.tibs$up)
  }
  if(HD){
    if(oracle){
      fit.gamma <- glm(y ~ x1, family = "Gamma", data = data)
      betaMLE <- coefficients(fit.gamma)
      shapeMLE <- as.numeric(gamma.shape(fit.gamma)[1])
      rateMLE <- cbind(1, newdata) %*% betaMLE * shapeMLE
      trueHDCI <- do.call(rbind, lapply(1:nrow(newdata), function(j){
        hdi(qgamma, 1 - alpha, shape = shapeMLE, rate = rateMLE[j, 1])
      }))
    }
    fit = lm(y ~ x1 + I(x1^2) + I(x1^3), data = data)
    betaMLE <- coefficients(fit)
    sdMLE <- summary(fit)$sigma
```

```r
    meanMLE <- as.numeric(cbind(1, x, x^2, x^3) %*% betaMLE)
    HDCI <- do.call(rbind, lapply(1:nrow(newdata), function(j){
      hdi(qnorm, 1 - alpha, sd = sdMLE, mean = meanMLE[j])
    }))
  }

  ## local coverage prediction regions
  output.parametric <- output.nonparametric <-
    output.LS <- output.LSLW <- output.HD <-
    output.trueHD <- rep(NA, bins + 1)
  if(parametric){
    marginal.parametric <- local.coverage(region = paraCI,
      data = data, d = p, bins = 1, at.data = "TRUE")
    local.parametric <- local.coverage(region = paraCI,
      data = data, d = p, bins = bins, at.data = "TRUE")
    local.inx.parametric <- local.coverage(region = paraCI,
      data = data, d = p, bins = 25, at.data = "TRUE")
    output.parametric <- c(marginal.parametric, local.parametric,
      local.inx.parametric,
      mean(apply(paraCI, 1, diff)),
      absolute.error(y = y, region = paraCI))
  }
  if(nonparametric){
    marginal.nonparametric <- local.coverage(region = nonparaCI,
      nonparametric = "TRUE", data = data, d = p, bins = 1,
      at.data = "TRUE")
    local.nonparametric <- local.coverage(region = nonparaCI,
      nonparametric = "TRUE", data = data, d = p, bins = bins,
      at.data = "TRUE")
    local.inx.nonparametric <- local.coverage(region = nonparaCI,
      nonparametric = "TRUE", data = data, d = p, bins = 25,
      at.data = "TRUE")
    output.nonparametric <-
      c(marginal.nonparametric, local.nonparametric,
        local.inx.nonparametric,
        area.nonparametric(nonparaCI),
        absolute.error.nonparametric(data = data,
          region = nonparaCI))
  }
  if(LS){
    marginal.LS <- local.coverage(region = LSCI,
      data = data, d = p, bins = 1, at.data = "TRUE")
    local.LS <- local.coverage(region = LSCI,
      data = data, d = p, bins = bins, at.data = "TRUE")
    local.inx.LS <- local.coverage(region = LSCI,
      data = data, d = p, bins = 25, at.data = "TRUE")
    output.LS <- c(marginal.LS, local.LS, local.inx.LS,
      mean(apply(LSCI, 1, diff)),
      absolute.error(y = y, region = LSCI))
  }
  if(LSLW){
    marginal.LSLW <- local.coverage(region = LSLWCI,
      data = data, d = p, bins = 1, at.data = "TRUE")
    local.LSLW <- local.coverage(region = LSLWCI,
      data = data, d = p, bins = bins, at.data = "TRUE")
    local.inx.LSLW <- local.coverage(region = LSLWCI,
      data = data, d = p, bins = 25, at.data = "TRUE")
    output.LSLW <- c(marginal.LSLW, local.LSLW, local.inx.LSLW,
      mean(apply(LSLWCI, 1, diff)),
      absolute.error(y = y, region = LSLWCI))
  }
  if(HD){
    marginal.HD <- local.coverage(region = HDCI,
      data = data, d = p, bins = 1, at.data = "TRUE")
    local.HD <- local.coverage(region = HDCI,
      data = data, d = p, bins = bins, at.data = "TRUE")
    local.inx.HD <- local.coverage(region = HDCI,
      data = data, d = p, bins = 25, at.data = "TRUE")
    output.HD <- c(marginal.HD, local.HD, local.inx.HD,
      mean(apply(HDCI, 1, diff)),
      absolute.error(y = y, region = HDCI))
    if(oracle){
      marginal.trueHD <- local.coverage(region = trueHDCI,
        data = data, d = p, bins = 1, at.data = "TRUE")
      local.trueHD <- local.coverage(region = trueHDCI,
        data = data, d = p, bins = bins, at.data = "TRUE")
      local.inx.trueHD <- local.coverage(region = trueHDCI,
        data = data, d = p, bins = 25, at.data = "TRUE")
      output.trueHD <- c(marginal.trueHD, local.trueHD,
        local.inx.trueHD, mean(apply(trueHDCI, 1, diff)),
        absolute.error(y = y, region = trueHDCI))
    }
  }

  output <- list(output.parametric = output.parametric,
    output.nonparametric = output.nonparametric,
    output.LS = output.LS,
    output.LSLW = output.LSLW,
    output.HD = output.HD,
    output.trueHD = output.trueHD)
  output

}
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 0.75.

```r
set.seed(13)
beta <- c(0.5, 1)
n <- 150
bins <- 2
B <- 250
system.time(out.misspec.150.2.0.75 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      oracle = TRUE, bins = bins, shape = 0.75))
})))

##     user   system  elapsed
## 9229.657   76.507 4936.555
```

```r
misspec.150.2.0.75 <- cbind(
  rowMeans(out.misspec.150.2.0.75, na.rm = TRUE),
  apply(out.misspec.150.2.0.75, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 1.

```r
system.time(out.misspec.150.2.1 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      oracle = TRUE, bins = bins, shape = 1))
})))

##     user   system  elapsed
## 7795.038   68.079 4360.653
```

```r
misspec.150.2.1 <- cbind(
  rowMeans(out.misspec.150.2.1, na.rm = TRUE),
  apply(out.misspec.150.2.1, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 2.

```
system.time(out.misspec.150.2.2 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      oracle = TRUE, bins = bins, shape = 2))
})))

##      user   system  elapsed
## 7412.152   68.397 4419.657
```

```
misspec.150.2.2 <- cbind(
  rowMeans(out.misspec.150.2.2, na.rm = TRUE),
  apply(out.misspec.150.2.2, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 5.

```
system.time(out.misspec.150.2.5 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      oracle = TRUE, bins = bins, shape = 5))
})))

##      user   system  elapsed
## 6511.338   63.174 3930.743
```

```
misspec.150.2.5 <- cbind(
  rowMeans(out.misspec.150.2.5, na.rm = TRUE),
  apply(out.misspec.150.2.5, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 7.

```r
system.time(out.misspec.150.2.7 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      oracle = TRUE, bins = bins, shape = 7))
})))

##     user   system  elapsed
## 6288.464   60.822 3775.383
```

```r
misspec.150.2.7 <- cbind(
  rowMeans(out.misspec.150.2.7, na.rm = TRUE),
  apply(out.misspec.150.2.7, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 10.

```r
system.time(out.misspec.150.2.10 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      oracle = TRUE, bins = bins, shape = 10))
})))

##     user   system  elapsed
## 6290.455   60.924 3765.440
```

```r
misspec.150.2.10 <- cbind(
  rowMeans(out.misspec.150.2.10, na.rm = TRUE),
  apply(out.misspec.150.2.10, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 12.

```r
system.time(out.misspec.150.2.12 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      oracle = TRUE, bins = bins, shape = 12))
})))
```

```
##      user   system  elapsed
## 6282.843   61.404 3741.541
```

```
misspec.150.2.12 <- cbind(
  rowMeans(out.misspec.150.2.12, na.rm = TRUE),
  apply(out.misspec.150.2.12, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 15.

```
system.time(out.misspec.150.2.15 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      oracle = TRUE, bins = bins, shape = 15))
})))

##      user   system  elapsed
## 6348.338   62.052 3746.669
```

```
misspec.150.2.15 <- cbind(
  rowMeans(out.misspec.150.2.15, na.rm = TRUE),
  apply(out.misspec.150.2.15, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 20.

```
system.time(out.misspec.150.2.20 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      oracle = TRUE, bins = bins, shape = 20))
})))

##      user   system  elapsed
## 6739.892   64.271 3907.575
```

```
misspec.150.2.20 <- cbind(
  rowMeans(out.misspec.150.2.20, na.rm = TRUE),
  apply(out.misspec.150.2.20, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 25.

```
system.time(out.misspec.150.2.25 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      oracle = TRUE, bins = bins, shape = 25))
})))

##     user   system  elapsed
## 6520.551   63.804 3821.963
```

```
misspec.150.2.25 <- cbind(
  rowMeans(out.misspec.150.2.25, na.rm = TRUE),
  apply(out.misspec.150.2.25, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 30.

```
system.time(out.misspec.150.2.30 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      oracle = TRUE, bins = bins, shape = 30))
})))

##     user   system  elapsed
## 6682.500   63.427 3852.692
```

```
misspec.150.2.30 <- cbind(
  rowMeans(out.misspec.150.2.30, na.rm = TRUE),
  apply(out.misspec.150.2.30, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 40.

```
system.time(out.misspec.150.2.40 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      oracle = TRUE, bins = bins, shape = 40))
})))

##     user   system  elapsed
## 6969.211   64.157 3889.455
```

```
misspec.150.2.40 <- cbind(
  rowMeans(out.misspec.150.2.40, na.rm = TRUE),
  apply(out.misspec.150.2.40, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 50.

```
system.time(out.misspec.150.2.50 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      oracle = TRUE, bins = bins, shape = 50))
})))

##     user   system  elapsed
## 6999.379   64.950 3894.541
```

```
misspec.150.2.50 <- cbind(
  rowMeans(out.misspec.150.2.50, na.rm = TRUE),
  apply(out.misspec.150.2.50, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 60.

```r
system.time(out.misspec.150.2.60 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      oracle = TRUE, bins = bins, shape = 60))
})))

##     user   system  elapsed
## 6974.508   64.703 3881.183
```

```r
misspec.150.2.60 <- cbind(
  rowMeans(out.misspec.150.2.60, na.rm = TRUE),
  apply(out.misspec.150.2.60, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 70.

```r
system.time(out.misspec.150.2.70 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      oracle = TRUE, bins = bins, shape = 70))
})))

##     user   system  elapsed
## 7194.882   65.152 3919.274
```

```r
misspec.150.2.70 <- cbind(
  rowMeans(out.misspec.150.2.70, na.rm = TRUE),
  apply(out.misspec.150.2.70, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 80.

```r
system.time(out.misspec.150.2.80 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      oracle = TRUE, bins = bins, shape = 80))
})))

##      user   system  elapsed
## 7235.154   64.228 3921.615
```

```r
misspec.150.2.80 <- cbind(
  rowMeans(out.misspec.150.2.80, na.rm = TRUE),
  apply(out.misspec.150.2.80, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 90.

```r
system.time(out.misspec.150.2.90 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      oracle = TRUE, bins = bins, shape = 90))
})))

##      user   system  elapsed
## 7230.128   64.696 3912.669
```

```r
misspec.150.2.90 <- cbind(
  rowMeans(out.misspec.150.2.90, na.rm = TRUE),
  apply(out.misspec.150.2.90, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 250$ iterations when $n = 150$ and shape = 100.

```r
system.time(out.misspec.150.2.100 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      oracle = TRUE, bins = bins, shape = 100))
})))
```

```
##      user   system  elapsed
## 7222.524   64.756 3905.972
```

```
misspec.150.2.100 <- cbind(
  rowMeans(out.misspec.150.2.100, na.rm = TRUE),
  apply(out.misspec.150.2.100, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

Reorganize the output.

```
para.area.misspec.150 <- nonpara.area.misspec.150 <-
  LS.area.misspec.150 <- LSLW.area.misspec.150 <-
  HD.area.misspec.150 <- trueHD.area.misspec.150 <- NULL
para.error.misspec.150 <- nonpara.error.misspec.150 <-
  LS.error.misspec.150 <- LSLW.error.misspec.150 <-
  HD.error.misspec.150 <- trueHD.error.misspec.150 <- NULL
para.marginal.misspec.150 <- nonpara.marginal.misspec.150 <-
  LS.marginal.misspec.150 <- LSLW.marginal.misspec.150 <-
  HD.marginal.misspec.150 <- trueHD.marginal.misspec.150 <- NULL
para.local.misspec.150 <- nonpara.local.misspec.150 <-
  LS.local.misspec.150 <- LSLW.local.misspec.150 <-
  HD.local.misspec.150 <- trueHD.local.misspec.150 <- NULL
para.inx.misspec.150 <- nonpara.inx.misspec.150 <-
  LS.inx.misspec.150 <- LSLW.inx.misspec.150 <-
  HD.inx.misspec.150 <- trueHD.inx.misspec.150 <- NULL
#shapes <- c(0.5, 0.75, 1, 2, 5, 7, 10, 12, 15, 20, 25,
#  30, 40, 50, 60, 70, 80, 90, 100)
shapes <- c(0.75, 1, 2, 5, 7, 10, 12, 15, 20, 25,
  30, 40, 50, 60, 70, 80, 90, 100)
for(j in shapes ){
  internal.output <- eval(parse(text=paste("misspec.150.2", j, sep = ".")))
  k <- which(shapes == j)
  para.marginal.misspec.150[k] <- as.numeric(internal.output[1, 1])
  para.local.misspec.150[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[2:3, 1])
  para.inx.misspec.150[c((25*(k-1)+1):(25*k))] <- as.numeric(internal.output[4:28,
  para.error.misspec.150[k] <- as.numeric(internal.output[29, 1])
  para.area.misspec.150[k] <- as.numeric(internal.output[30, 1])
  nonpara.marginal.misspec.150[k] <- as.numeric(internal.output[31, 1])
  nonpara.local.misspec.150[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[32:33, 1])
  nonpara.inx.misspec.150[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[34:58, 1])
  nonpara.error.misspec.150[k] <- as.numeric(internal.output[59, 1])
  nonpara.area.misspec.150[k] <- as.numeric(internal.output[60, 1])
  LS.marginal.misspec.150[k] <- as.numeric(internal.output[61, 1])
  LS.local.misspec.150[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[62:63, 1])
  LS.inx.misspec.150[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[64:88, 1])
  LS.error.misspec.150[k] <- as.numeric(internal.output[89, 1])
  LS.area.misspec.150[k] <- as.numeric(internal.output[90, 1])
  LSLW.marginal.misspec.150[k] <- as.numeric(internal.output[91, 1])
  LSLW.local.misspec.150[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[92:93, 1])
  LSLW.inx.misspec.150[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[94:118, 1])
  LSLW.error.misspec.150[k] <- as.numeric(internal.output[119, 1])
  LSLW.area.misspec.150[k] <- as.numeric(internal.output[120, 1])
  HD.marginal.misspec.150[k] <- as.numeric(internal.output[121, 1])
  HD.local.misspec.150[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[122:123, 1])
  HD.inx.misspec.150[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[124:148, 1])
  HD.error.misspec.150[k] <- as.numeric(internal.output[149, 1])
  HD.area.misspec.150[k] <- as.numeric(internal.output[150, 1])
  trueHD.marginal.misspec.150[k] <- as.numeric(internal.output[151, 1])
  trueHD.local.misspec.150[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[152:153, 1])
  trueHD.inx.misspec.150[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[154:178, 1])
  trueHD.error.misspec.150[k] <- as.numeric(internal.output[179, 1])
  trueHD.area.misspec.150[k] <- as.numeric(internal.output[180, 1])
}
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 250$ and shape = 5.

```
set.seed(13)
n <- 250
bins <- 3
system.time(out.misspec.250.3.5 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      bins = bins, shape = 5))
})))

##     user   system  elapsed
## 2392.483   14.124 1461.153
```

```
misspec.250.3.5 <- cbind(
  rowMeans(out.misspec.250.3.5, na.rm = TRUE),
  apply(out.misspec.250.3.5, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 250$ and shape = 20.

```
system.time(out.misspec.250.3.20 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      bins = bins, shape = 20))
})))

##     user   system  elapsed
## 2296.508   14.109 1429.061
```

```
misspec.250.3.20 <- cbind(
  rowMeans(out.misspec.250.3.20, na.rm = TRUE),
  apply(out.misspec.250.3.20, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 250$ and shape = 40.

```
system.time(out.misspec.250.3.40 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      bins = bins, shape = 40))
})))

##     user   system  elapsed
## 2282.950   13.735 1413.949
```

```
misspec.250.3.40 <- cbind(
  rowMeans(out.misspec.250.3.40, na.rm = TRUE),
  apply(out.misspec.250.3.40, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 250$ and shape = 60.

```
system.time(out.misspec.250.3.60 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      bins = bins, shape = 60))
})))

##     user   system  elapsed
## 2414.106   13.831 1435.658
```

```
misspec.250.3.60 <- cbind(
  rowMeans(out.misspec.250.3.60, na.rm = TRUE),
  apply(out.misspec.250.3.60, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 250$ and shape = 80.

```
system.time(out.misspec.250.3.80 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      bins = bins, shape = 80))
})))

##     user   system  elapsed
## 2395.971   13.535 1425.224
```

```
misspec.250.3.80 <- cbind(
  rowMeans(out.misspec.250.3.80, na.rm = TRUE),
  apply(out.misspec.250.3.80, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 250$ and shape = 100.

```
system.time(out.misspec.250.3.100 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      bins = bins, shape = 100))
})))

##     user   system  elapsed
## 2391.049   12.727 1422.181
```

```
misspec.250.3.100 <- cbind(
  rowMeans(out.misspec.250.3.100, na.rm = TRUE),
  apply(out.misspec.250.3.100, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

Reorganize the output.

```
para.area.misspec.250 <- nonpara.area.misspec.250 <-
  LS.area.misspec.250 <- LSLW.area.misspec.250 <-
  HD.area.misspec.250 <- NULL
para.error.misspec.250 <- nonpara.error.misspec.250 <-
  LS.error.misspec.250 <- LSLW.error.misspec.250 <-
  HD.error.misspec.250 <- NULL
para.marginal.misspec.250 <- nonpara.marginal.misspec.250 <-
  LS.marginal.misspec.250 <- LSLW.marginal.misspec.250 <-
  HD.marginal.misspec.250 <- NULL
para.local.misspec.250 <- nonpara.local.misspec.250 <-
  LS.local.misspec.250 <- LSLW.local.misspec.250 <-
  HD.local.misspec.250 <- NULL
para.inx.misspec.250 <- nonpara.inx.misspec.250 <-
  LS.inx.misspec.250 <- LSLW.inx.misspec.250 <-
  HD.inx.misspec.250 <- NULL
shapes <- c(5, 20, 40, 60, 80, 100)
for(j in shapes ){
  internal.output <- eval(parse(text=paste("misspec.250.3", j, sep = ".")))
  k <- which(shapes == j)
  para.marginal.misspec.250[k] <- as.numeric(internal.output[1, 1])
  para.local.misspec.250[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[2:4, 1])
  para.inx.misspec.250[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[5:29, 1])
  para.error.misspec.250[k] <- as.numeric(internal.output[30, 1])
  para.area.misspec.250[k] <- as.numeric(internal.output[31, 1])
  nonpara.marginal.misspec.250[k] <- as.numeric(internal.output[32, 1])
  nonpara.local.misspec.250[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[33:35, 1])
  nonpara.inx.misspec.250[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[36:60, 1])
  nonpara.error.misspec.250[k] <- as.numeric(internal.output[61, 1])
  nonpara.area.misspec.250[k] <- as.numeric(internal.output[62, 1])
  LS.marginal.misspec.250[k] <- as.numeric(internal.output[63, 1])
  LS.local.misspec.250[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[64:66, 1])
  LS.inx.misspec.250[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[67:91, 1])
  LS.error.misspec.250[k] <- as.numeric(internal.output[92, 1])
  LS.area.misspec.250[k] <- as.numeric(internal.output[93, 1])
  LSLW.marginal.misspec.250[k] <- as.numeric(internal.output[94, 1])
  LSLW.local.misspec.250[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[95:97, 1])
  LSLW.inx.misspec.250[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[98:122, 1])
  LSLW.error.misspec.250[k] <- as.numeric(internal.output[123, 1])
  LSLW.area.misspec.250[k] <- as.numeric(internal.output[124, 1])
  HD.marginal.misspec.250[k] <- as.numeric(internal.output[125, 1])
  HD.local.misspec.250[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[126:128, 1])
  HD.inx.misspec.250[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[129:153, 1])
  HD.error.misspec.250[k] <- as.numeric(internal.output[154, 1])
  HD.area.misspec.250[k] <- as.numeric(internal.output[155, 1])
}
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 500$ and shape = 5.

```
set.seed(13)
n <- 500
bins <- 3
system.time(out.misspec.500.3.5 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      bins = bins, shape = 5))
})))

##     user   system  elapsed
## 6925.010   19.527 4348.905
```

```
misspec.500.3.5 <- cbind(
  rowMeans(out.misspec.500.3.5, na.rm = TRUE),
  apply(out.misspec.500.3.5, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 500$ and shape = 20.

```
system.time(out.misspec.500.3.20 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      bins = bins, shape = 20))
})))

##     user   system  elapsed
## 6726.821   19.619 4257.396
```

```
misspec.500.3.20 <- cbind(
  rowMeans(out.misspec.500.3.20, na.rm = TRUE),
  apply(out.misspec.500.3.20, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 500$ and shape = 40.

```
system.time(out.misspec.500.3.40 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      bins = bins, shape = 40))
})))

##     user   system  elapsed
## 6859.403   19.266 4252.117
```

```
misspec.500.3.40 <- cbind(
  rowMeans(out.misspec.500.3.40, na.rm = TRUE),
  apply(out.misspec.500.3.40, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 500$ and shape = 60.

```
system.time(out.misspec.500.3.60 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      bins = bins, shape = 60))
})))

##     user   system  elapsed
## 7253.875   19.272 4346.840
```

```
misspec.500.3.60 <- cbind(
  rowMeans(out.misspec.500.3.60, na.rm = TRUE),
  apply(out.misspec.500.3.60, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 500$ and shape = 80.

```
system.time(out.misspec.500.3.80 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      bins = bins, shape = 80))
})))

##     user   system  elapsed
## 7358.336   19.303 4335.417
```

```
misspec.500.3.80 <- cbind(
  rowMeans(out.misspec.500.3.80, na.rm = TRUE),
  apply(out.misspec.500.3.80, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 500$ and shape = 100.

```
system.time(out.misspec.500.3.100 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(misspec.simulator(beta = beta, n = n,
      bins = bins, shape = 100))
})))

##     user   system  elapsed
## 7433.921   19.266 4345.724
```

```
misspec.500.3.100 <- cbind(
  rowMeans(out.misspec.500.3.100, na.rm = TRUE),
  apply(out.misspec.500.3.100, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

Reorganize the output.

```
para.area.misspec.500 <- nonpara.area.misspec.500 <-
  LS.area.misspec.500 <- LSLW.area.misspec.500 <-
  HD.area.misspec.500 <- NULL
para.error.misspec.500 <- nonpara.error.misspec.500 <-
  LS.error.misspec.500 <- LSLW.error.misspec.500 <-
  HD.error.misspec.500 <- NULL
para.marginal.misspec.500 <- nonpara.marginal.misspec.500 <-
  LS.marginal.misspec.500 <- LSLW.marginal.misspec.500 <-
  HD.marginal.misspec.500 <- NULL
para.local.misspec.500 <- nonpara.local.misspec.500 <-
  LS.local.misspec.500 <- LSLW.local.misspec.500 <-
  HD.local.misspec.500 <- NULL
para.inx.misspec.500 <- nonpara.inx.misspec.500 <-
  LS.inx.misspec.500 <- LSLW.inx.misspec.500 <-
  HD.inx.misspec.500 <- NULL
shapes <- c(5, 20, 40, 60, 80, 100)
for(j in shapes ){
  internal.output <- eval(parse(text=paste("misspec.500.3", j, sep = ".")))
  k <- which(shapes == j)
  para.marginal.misspec.500[k] <- as.numeric(internal.output[1, 1])
  para.local.misspec.500[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[2:4, 1])
  para.inx.misspec.500[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[5:29, 1])
  para.error.misspec.500[k] <- as.numeric(internal.output[30, 1])
  para.area.misspec.500[k] <- as.numeric(internal.output[31, 1])
  nonpara.marginal.misspec.500[k] <- as.numeric(internal.output[32, 1])
  nonpara.local.misspec.500[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[33:35, 1])
  nonpara.inx.misspec.500[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[36:60, 1])
  nonpara.error.misspec.500[k] <- as.numeric(internal.output[61, 1])
  nonpara.area.misspec.500[k] <- as.numeric(internal.output[62, 1])
  LS.marginal.misspec.500[k] <- as.numeric(internal.output[63, 1])
  LS.local.misspec.500[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[64:66, 1])
  LS.inx.misspec.500[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[67:91, 1])
  LS.error.misspec.500[k] <- as.numeric(internal.output[92, 1])
  LS.area.misspec.500[k] <- as.numeric(internal.output[93, 1])
  LSLW.marginal.misspec.500[k] <- as.numeric(internal.output[94, 1])
  LSLW.local.misspec.500[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[95:97, 1])
  LSLW.inx.misspec.500[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[98:122, 1])
  LSLW.error.misspec.500[k] <- as.numeric(internal.output[123, 1])
  LSLW.area.misspec.500[k] <- as.numeric(internal.output[124, 1])
  HD.marginal.misspec.500[k] <- as.numeric(internal.output[125, 1])
  HD.local.misspec.500[c((bins*(k-1)+1):(bins*k))] <-
    as.numeric(internal.output[126:128, 1])
  HD.inx.misspec.500[c((25*(k-1)+1):(25*k))] <-
    as.numeric(internal.output[129:153, 1])
  HD.error.misspec.500[k] <- as.numeric(internal.output[154, 1])
  HD.area.misspec.500[k] <- as.numeric(internal.output[155, 1])
}
```

6.2 Results

Results form our simulations are depicted in Figures 11-16. For all five prediction regions we depict the estimatated area, prediction error, and local coverage probabilities with respect to binning in Figures 11, 13, and 15 for $n = 150$, 250, and 500 respectively. For all five prediction regions we depict the local coverage probabilities across x in Figures 12, 14, and 16 for $n = 150$, 250, and 500 respectively.

In these simulations, we expect for the LSLW conformal prediction region to perform well. The model misspecification is modest, the Gamma data appears to be almost symmetric, albeit heterogenous, about a cubic mean function. From these simulations we see that the parametric conformal prediction region is similar to the LSLW prediciton region in area and prediction error. Both the parametric and LSLW conformal prediction regions possess finite-sample, albeit slightly conservative, local validity with respect to binning and they both give almost nominal finite-sample conditional validity across the support. Moreover, these prediciton regions visually fit the data well as seen in Section 8.2. The nonparametric conformal prediction region gives closer to nominal coverage than both the parametric and LSLW conformal prediction regions, and it possesses finite-sample local validity with respect to binning. However, this prediction region is larger, gives higher prediciton error than the parametric and LSLW conformal prediction regions, and does not visually fit the data well for most simulation settings as seen in Section 8.2. The misspecified HD and LS conformal prediction regions provide extreme undercoverage at small values of x and extreme overcoverage at large values of x. These prediction regions are also larger, they give higher prediciton error than the parametric and LSLW conformal prediction regions, and the LS conformal prediction region does not visually fit the data well for most simulation settings as seen in Section 8.2.

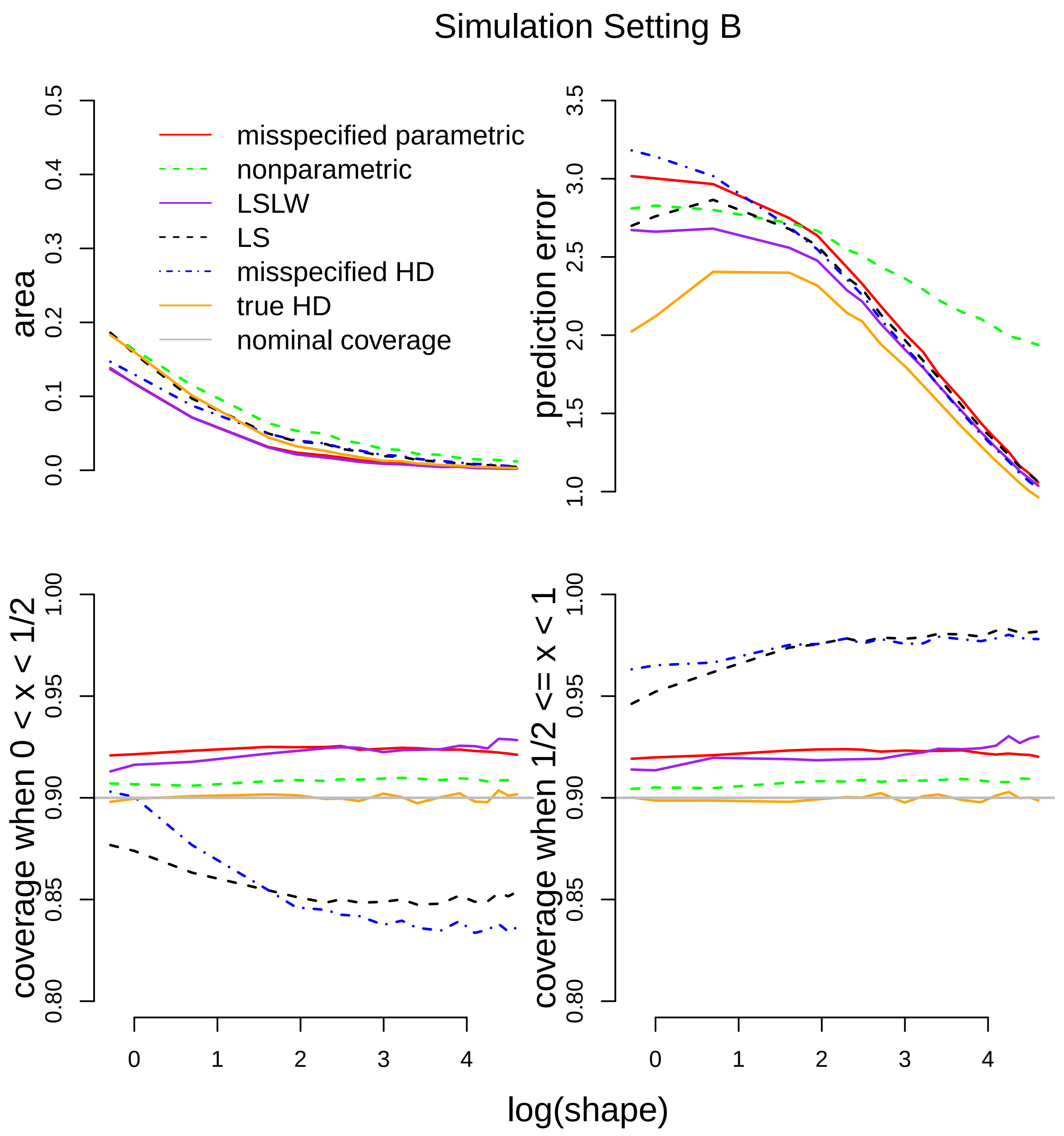


Figure 11: This figure compares the performance of the misspecified parametric, nonparametric, least squares, and least squares locally weighted conformal prediciton region and the misspecified and correctly specified highest density prediction region when $n = 150$ and the number of bins equals 2. The specific diagnostics used to compare these prediciton regions is the area (top-left panel), prediction error (top-right panel), and the coverage probability with respect to binning (bottom row) across shape parameter values. The average of 250 Monte Carlo samples at each shape parameter value in these simulation settings form the lines that are depicted in this figure.

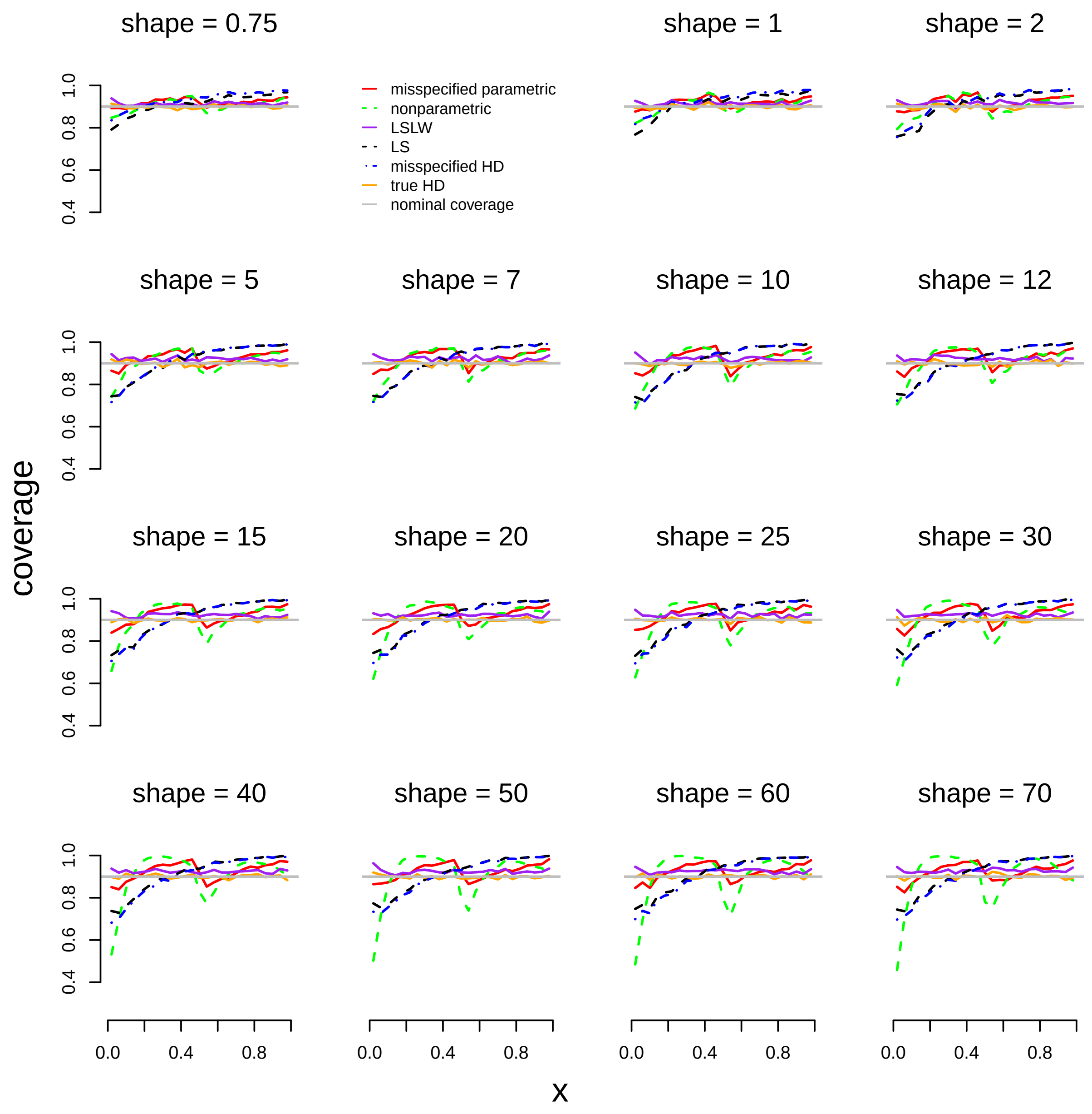


Figure 12: Plot of the estimated coverage probabilities of prediction regions across x and shape parameter values when the model is misspecified, $n = 150$, and the number of bins is equal to 2.

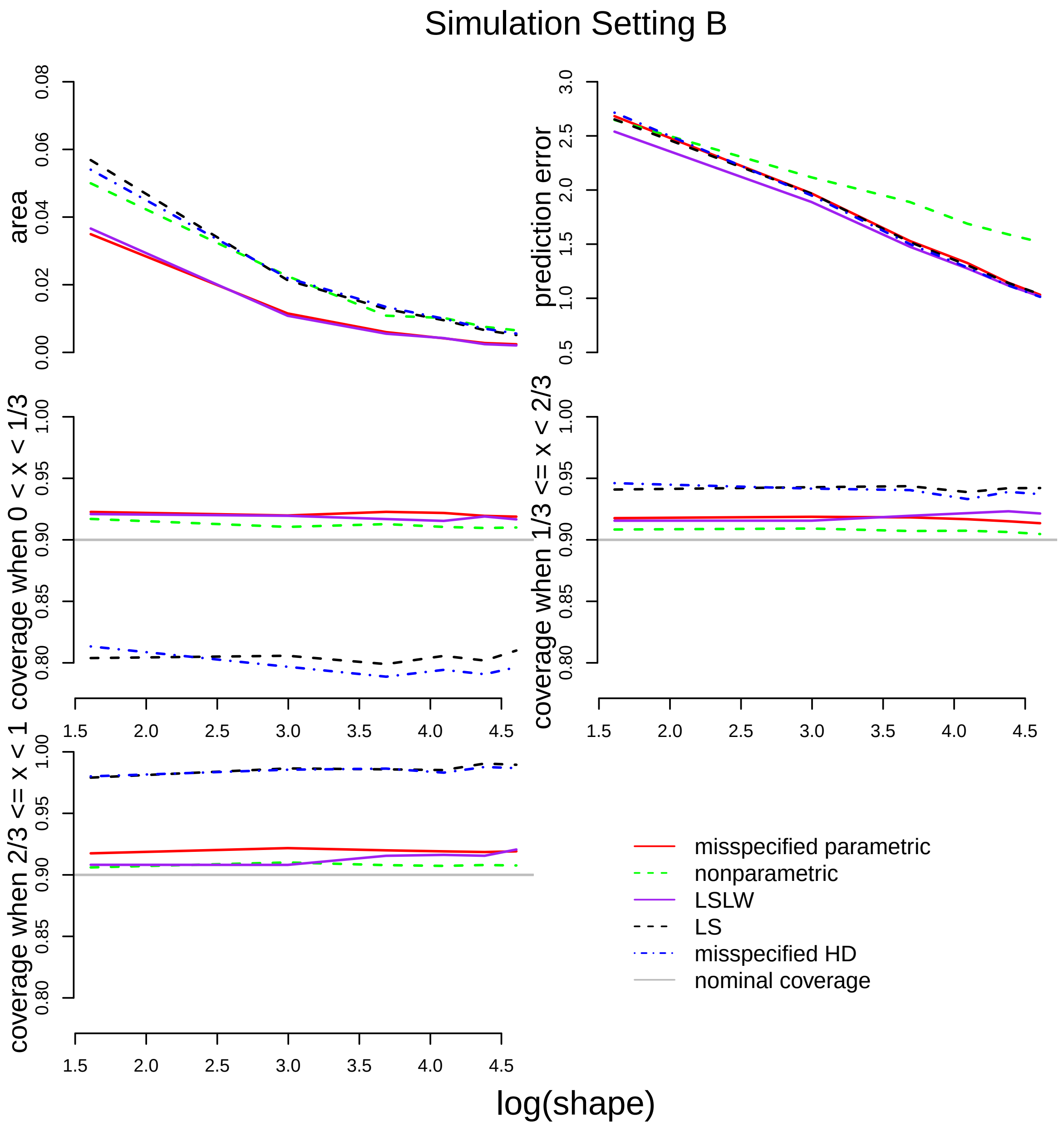


Figure 13: This figure compares the performance of the misspecified parametric, nonparametric, least squares, and least squares locally weighted conformal prediciton region and the misspecified highest density prediction region when $n = 250$ and the number of bins equals 3. The specific diagnostics used to compare these prediciton regions is the area (top-left panel), prediction error (top-right panel), and the coverage probability with respect to binning (bottom row) across shape parameter values. The average of 50 Monte Carlo samples at each shape parameter value in these simulation settings form the lines that are depicted in this figure.

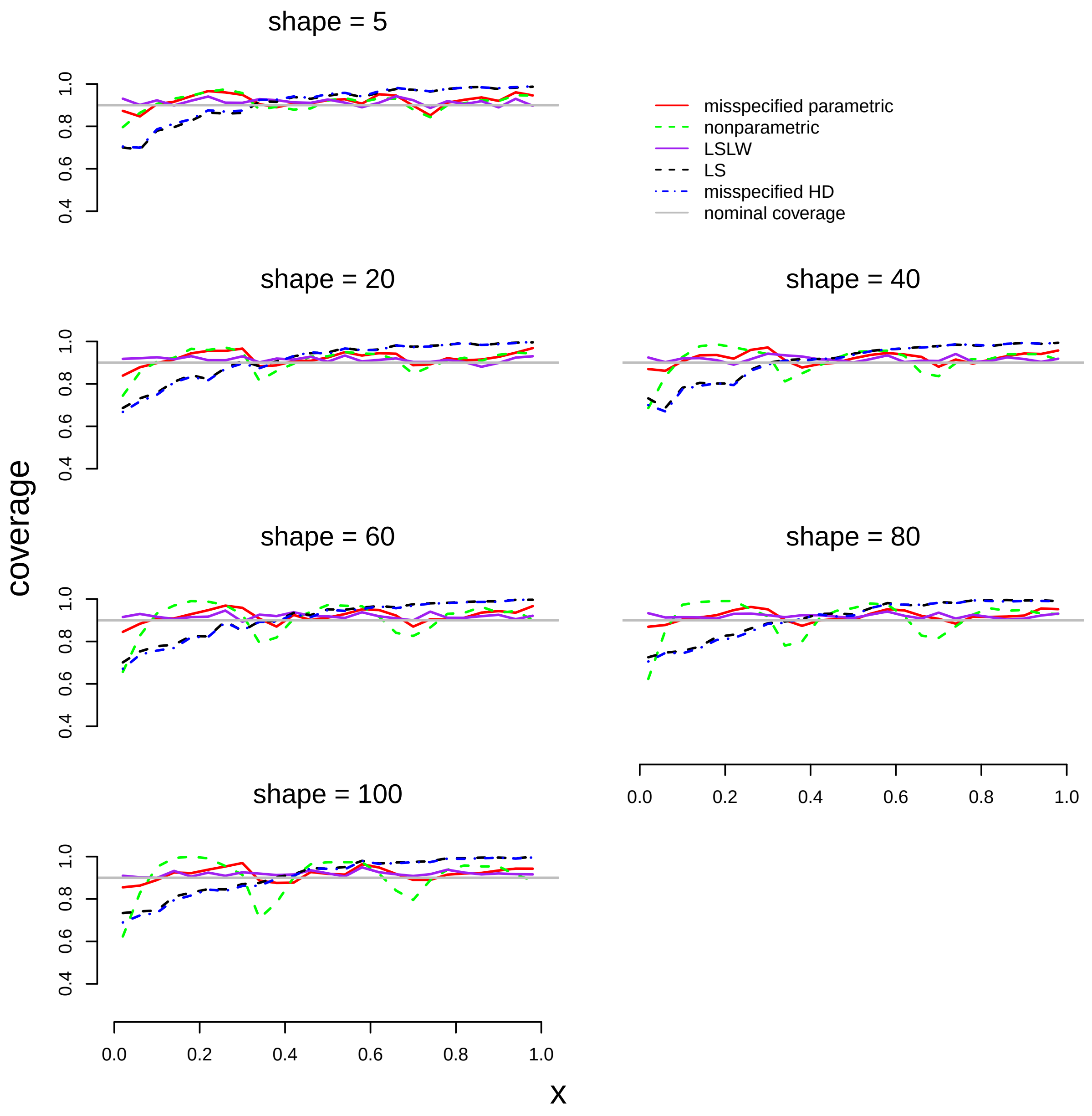


Figure 14: Plot of the estimated coverage probabilities of prediction regions across x and shape parameter values when the model is misspecified, $n = 250$, and the number of bins is equal to 3.

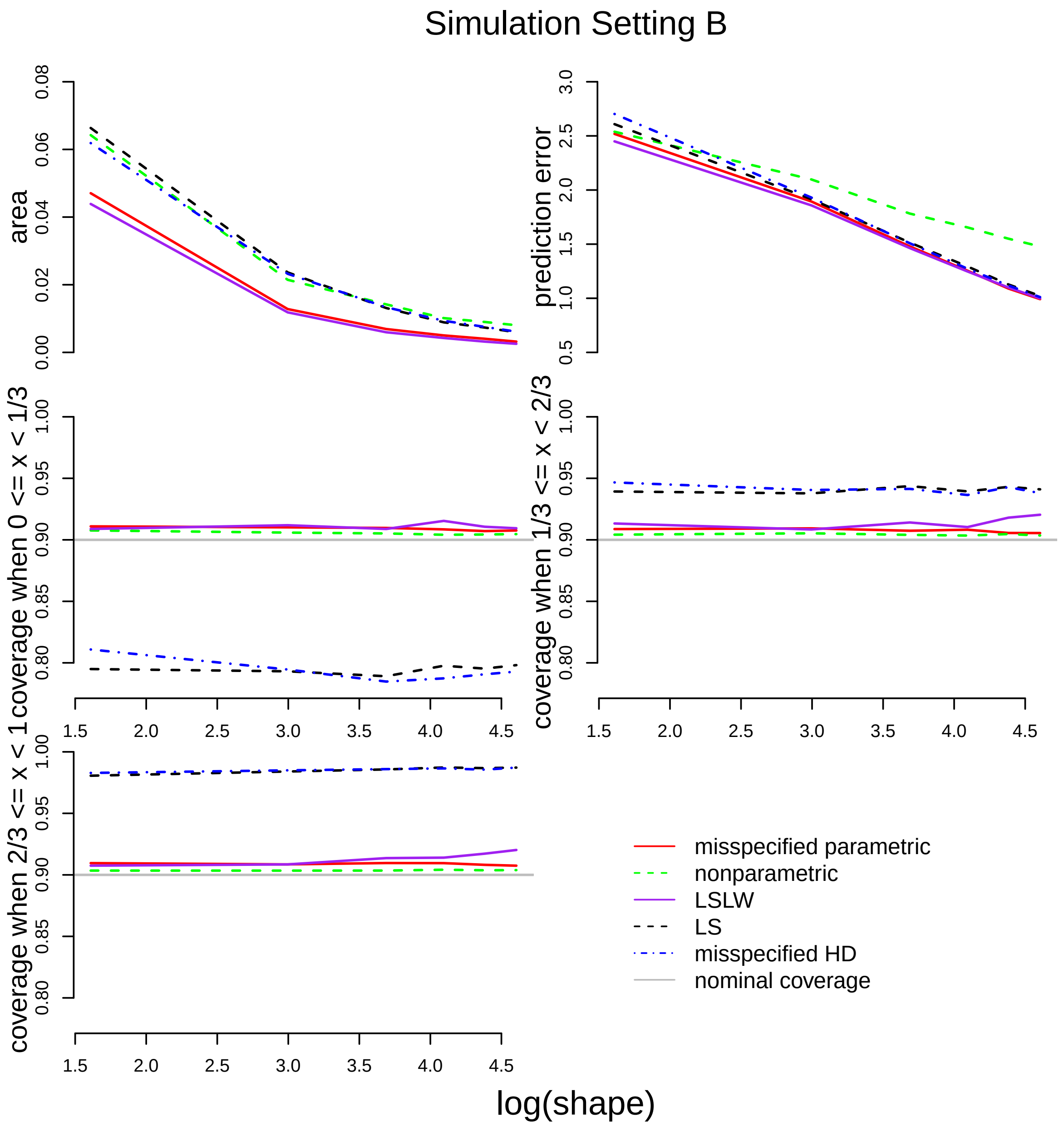


Figure 15: This figure compares the performance of the misspecified parametric, nonparametric, least squares, and least squares locally weighted conformal prediciton region and the misspecified highest density prediction region when $n = 500$ and the number of bins equals 3. The specific diagnostics used to compare these prediciton regions is the area (top-left panel), prediction error (top-right panel), and the coverage probability with respect to binning (bottom row) across shape parameter values. The average of 50 Monte Carlo samples at each shape parameter value in these simulation settings form the lines that are depicted in this figure.

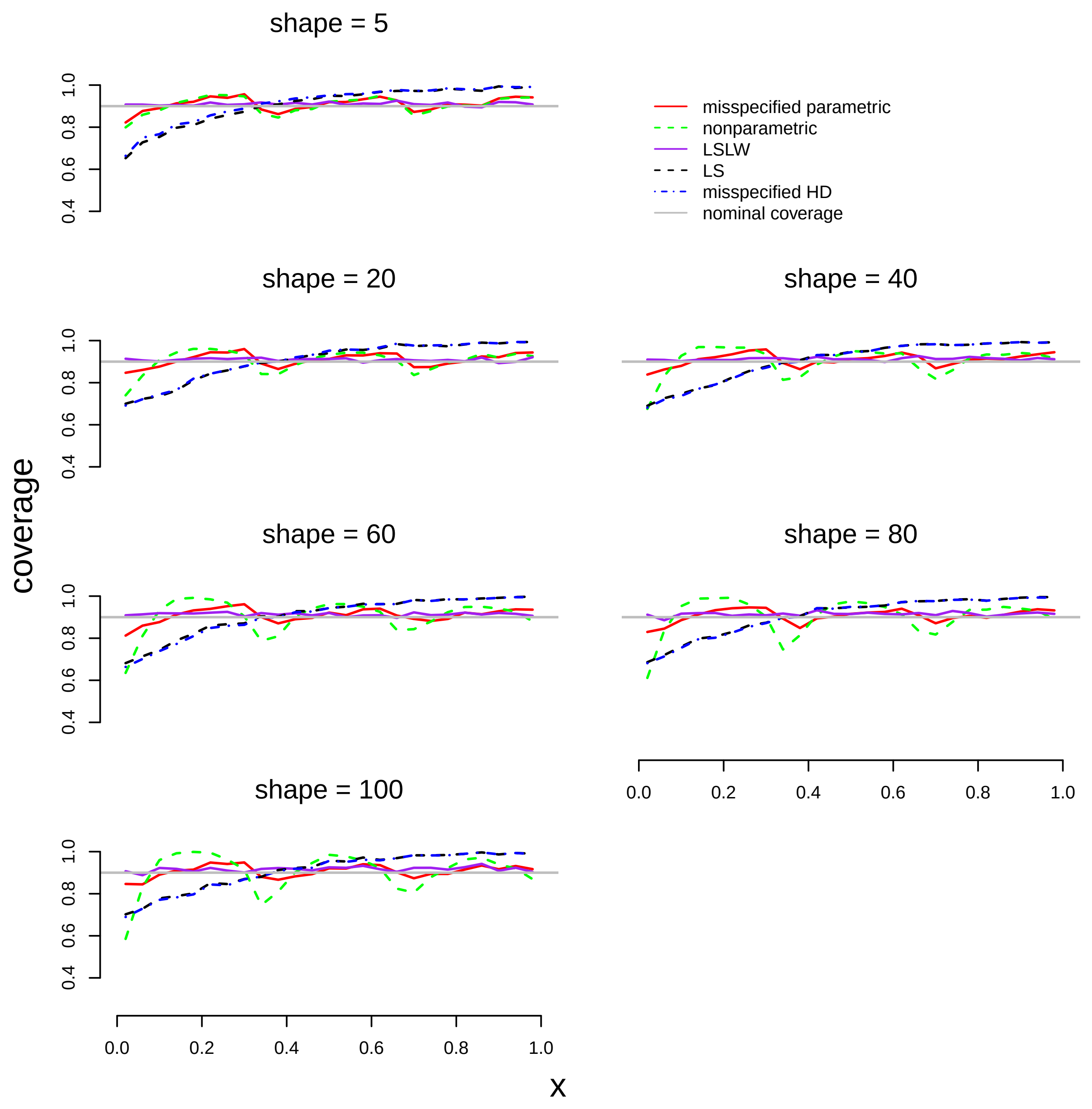


Figure 16: Plot of the estimated coverage probabilities of prediction regions across x and shape parameter values when the model is misspecified, $n = 500$, and the number of bins is equal to 3.

7 Linear Regression Simulations

In this Section, we compare the parametric conformal prediction region, the nonparametric conformal prediction region, the LSLW conformal prediction region, the LS conformal prediction region, and the HD prediction region for linear regression models with normal errors and constant variance. We specify that $\beta = (2, 5)^T$ and the standard deviation of the errors about the mean function as $\sigma = 1$. We consider sample sizes of $n \in \{150, 250, 500\}$. When $n = 150$ we build the parametric and nonparametric conformal prediction regions using 2 bins. When $n = 250, 500$ we build the parametric and nonparametric conformal prediction regions using 3 bins. These number of bin choices correspond to the bin width asymptotics of Lei and Wasserman [2014].

7.1 Simulations

The following function computes our diagnostic measures for the five prediction regions under investigation in the univariate case where data is generated from a Gaussian regression model, i.e. $Y \sim N(X'\beta, \sigma^2)$ and $X \sim U(0, 1)$.

```r
regression.simulator <- function(n = 500, alpha = 0.10, beta,
  bins = 3, sd = 1, parametric = TRUE, nonparametric = TRUE,
  LS = TRUE, LSLW = TRUE, HD = TRUE, cores = 6){

  p <- d <- length(beta) - 1
  x <- matrix(runif(n), ncol = p)
  y <- rep(0,n)
  data <- NULL

  ## set up partition
  if(class(bins) == "NULL"){
    wn <- min(1/ floor(1 / (log(n)/n)^(1/(d+3))), 1/2)
    bins <- 1 / wn
  }

  ## generate the data
  mu <- cbind(1, x) %*% beta
  y <- rnorm(n = n, mean = mu, sd = sd)
  data <- data.frame(y = y, x = x)
  colnames(data)[2:(p+1)] <- paste("x", 1:p, sep = "")

  ## fit the linear regression model
  fit <- glm(y ~ x1, family = "gaussian", data = data)
  paraCI <- nonparaCI <- LSCI <- LSLWCI <- HDCI <- NULL
  formula <- fit$formula
  newdata <- data
  respname <- all.vars(formula)[1]
  newdata <- newdata[, !(colnames(data) %in% respname)]
  newdata <- as.matrix(newdata)

  ## obtain the prediction regions
  if(parametric){
    cpred <- conformal.glm(fit, parametric = TRUE,
      nonparametric = FALSE, alpha = alpha,
      bins = bins, cores = cores)
    paraCI <- cpred$paraconformal
  }
  if(nonparametric){
    cpred <- conformal.glm(fit, parametric = FALSE,
      nonparametric = TRUE, alpha = alpha,
      bins = bins, cores = cores)
    nonparaCI <- cpred$nonparaconformal
  }
  if(LS){
    p1.tibs <- conformal.pred(x = x, y = y, x0 = x,
      train.fun = train.fun, predict.fun = predict.fun,
      alpha = alpha)
    LSCI <- cbind(p1.tibs$lo, p1.tibs$up)
  }
  if(LSLW){
    regression.model <- lm(y ~ x)
    abs.resid <- abs(regression.model$resid)
    smooth.call <- smooth.spline(x, abs.resid,
      nknots = 10)
    lambda <- smooth.call$lambda
    df <- smooth.call$df
    mad.train.fun <- function(x, y, out = NULL){
      smooth.spline(x[, 1], y, lambda = lambda,
      df = df, nknots = 10)
    }
    p2.tibs <- conformal.pred(x = x, y = y, x0 = x,
      train.fun = train.fun, predict.fun = predict.fun,
      mad.train.fun = mad.train.fun,
      mad.predict.fun = mad.predict.fun,
      alpha = alpha)
    LSLWCI <- cbind(p2.tibs$lo, p2.tibs$up)
  }
  if(HD){
    fit = lm(y ~ x, data = data)
    betaMLE <- coefficients(fit)
    sdMLE <- summary(fit)$sigma
    meanMLE <- as.numeric(cbind(1, x) %*% betaMLE)
    HDCI <- do.call(rbind, lapply(1:nrow(newdata), function(j){
      hdi(qnorm, 1 - alpha, sd = sdMLE, mean = meanMLE[j])
    }))
  }

  ## local coverage prediction regions
```

```
output.parametric <- output.nonparametric <-
  output.LS <- output.LSLW <- output.HD <- rep(NA, bins + 1)
if(parametric){
  marginal.parametric <- local.coverage(region = paraCI,
    data = data, d = p, bins = 1, at.data = "TRUE")
  local.parametric <- local.coverage(region = paraCI,
    data = data, d = p, bins = bins, at.data = "TRUE")
  local.inx.parametric <- local.coverage(region = paraCI,
    data = data, d = p, bins = 25, at.data = "TRUE")
  output.parametric <- c(marginal.parametric, local.parametric,
    local.inx.parametric,
    mean(apply(paraCI, 1, diff)),
    absolute.error(y = y, region = paraCI))
}
if(nonparametric){
  marginal.nonparametric <- local.coverage(region = nonparaCI,
    nonparametric = "TRUE", data = data, d = p, bins = 1,
    at.data = "TRUE")
  local.nonparametric <- local.coverage(region = nonparaCI,
    nonparametric = "TRUE", data = data, d = p, bins = bins,
    at.data = "TRUE")
  local.inx.nonparametric <- local.coverage(region = nonparaCI,
    nonparametric = "TRUE", data = data, d = p, bins = 25,
    at.data = "TRUE")
  output.nonparametric <-
    c(marginal.nonparametric, local.nonparametric,
      local.inx.nonparametric,
      area.nonparametric(nonparaCI),
      absolute.error.nonparametric(data = data,
        region = nonparaCI))
}
if(LS){
  marginal.LS <- local.coverage(region = LSCI,
    data = data, d = p, bins = 1, at.data = "TRUE")
  local.LS <- local.coverage(region = LSCI,
    data = data, d = p, bins = bins, at.data = "TRUE")
  local.inx.LS <- local.coverage(region = LSCI,
    data = data, d = p, bins = 25, at.data = "TRUE")
  output.LS <- c(marginal.LS, local.LS, local.inx.LS,
    mean(apply(LSCI, 1, diff)),
    absolute.error(y = y, region = LSCI))
}
if(LSLW){
  marginal.LSLW <- local.coverage(region = LSLWCI,
    data = data, d = p, bins = 1, at.data = "TRUE")
  local.LSLW <- local.coverage(region = LSLWCI,
    data = data, d = p, bins = bins, at.data = "TRUE")
    local.inx.LSLW <- local.coverage(region = LSLWCI,
      data = data, d = p, bins = 25, at.data = "TRUE")
    output.LSLW <- c(marginal.LSLW, local.LSLW, local.inx.LSLW,
      mean(apply(LSLWCI, 1, diff)),
      absolute.error(y = y, region = LSLWCI))
  }
  if(HD){
    marginal.HD <- local.coverage(region = HDCI,
      data = data, d = p, bins = 1, at.data = "TRUE")
    local.HD <- local.coverage(region = HDCI,
      data = data, d = p, bins = bins, at.data = "TRUE")
    local.inx.HD <- local.coverage(region = HDCI,
      data = data, d = p, bins = 25, at.data = "TRUE")
    output.HD <- c(marginal.HD, local.HD, local.inx.HD,
      mean(apply(HDCI, 1, diff)),
      absolute.error(y = y, region = HDCI))
  }

  output <- list(output.parametric = output.parametric,
    output.nonparametric = output.nonparametric,
    output.LS = output.LS,
    output.LSLW = output.LSLW,
    output.HD = output.HD)
  output

}
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 150$.

```
set.seed(13)
beta <- c(2, 5)
n <- 150
bins <- 2
B <- 50
system.time(out.regression.150.2 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(regression.simulator(beta = beta, n = n,
      bins = bins))
})))

##     user   system  elapsed
## 1509.298   13.117  901.467
```

```
regression.150.2 <- cbind(
  rowMeans(out.regression.150.2, na.rm = TRUE),
  apply(out.regression.150.2, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 150$.

```
n <- 250
bins <- 3
system.time(out.regression.250.3 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(regression.simulator(beta = beta, n = n,
      bins = bins))
})))

##     user   system  elapsed
## 2945.050   18.113 1846.314
```

```
regression.250.3 <- cbind(
  rowMeans(out.regression.250.3, na.rm = TRUE),
  apply(out.regression.250.3, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

The following performs our Monte Carlo simulation of $B = 50$ iterations when $n = 500$.

```
n <- 500
system.time(out.regression.500.3 <- do.call(cbind,
  lapply(1:B, FUN = function(j){
    unlist(regression.simulator(beta = beta, n = n,
      bins = bins))
})))

##     user   system  elapsed
## 7285.245   19.752 4517.354
```

```
regression.500.3 <- cbind(
  rowMeans(out.regression.500.3, na.rm = TRUE),
  apply(out.regression.500.3, 1,
  FUN = function(x){
    sds <- sd(x, na.rm = TRUE)
    lengths <- length(which(!is.na(x)))
    sds / sqrt(lengths)
  }))
```

7.2 Results

Results form our simulations are depicted in Table 2 and Figure 17. This table and figure depicts the estimatated area, prediction error, and local coverage probabilities for all five considered prediction regions.

In these simulations, errors about the estimated mean function are symmetric and homogeneous across the support. We therefore expect for the LS and LSLW conformal prediction regions to perform nearly as well as the oracle HD prediction region. These prediction regions also exhibit finite-sample marginal validity, local validity with respect to binning, and near conditional validity across the support. The parametric conformal prediction region is similar to the LS and LSLW conformal prediciton regions and the HD prediction region in area, prediction error, finite-sample coverage properties, and appearance. However, the parametric conformal prediciton region is slightly larger and gives more conservative coverage that these other prediction regions. The nonparametric conformal prediciton region is larger and gives larger prediciton errors than the other prediciton regions. It also appears to not visually fit the data well while the others do as seen in Section 8.3.

		parametric conformal	nonparametric conformal	LS conformal	LSLW conformal	HD region
$n = 150$	marginal coverage	0.922 (0.0007)	0.916 (0.001)	0.913 (0.001)	0.918(0.0011)	0.904 (0.0023)
	local coverage when $0 < x < 1/2$	0.922 (0.0011)	0.916 (0.0014)	0.912 (0.0035)	0.921 (0.0029)	0.903 (0.0042)
	local coverage when $1/2 \leq x < 1$	0.922 (0.001)	0.916 (0.0016)	0.914 (0.0036)	0.915 (0.003)	0.904 (0.0036)
	area	3.521 (0.0366)	4.258 (0.0466)	3.361 (0.0291)	3.385 (0.0322)	3.28 (0.0263)
	prediction error	0.021 (0.0011)	0.033 (0.0023)	0.027 (0.0015)	0.023 (0.0013)	0.03 (0.0013)
$n = 250$	marginal coverage	0.918 (0.0006)	0.915 (0.0007)	0.908 (0.0006)	0.911 (0.0007)	0.901 (0.0014)
	local coverage when $0 < x < 1/3$	0.919 (0.001)	0.915 (0.0014)	0.91 (0.0036)	0.912 (0.0034)	0.905 (0.0039)
	local coverage when $1/3 \leq x < 2/3$	0.917 (0.0007)	0.917 (0.0014)	0.912 (0.0035)	0.913 (0.0028)	0.907 (0.0039)
	local coverage when $2/3 \leq x < 1$	0.918 (0.001)	0.915 (0.0015)	0.904 (0.0042)	0.907 (0.0032)	0.893 (0.0046)
	area	3.518 (0.0291)	3.822 (0.0338)	3.36 (0.0233)	3.363 (0.0261)	3.296 (0.0218)
	prediction error	0.021 (0.0013)	0.028 (0.0017)	0.027 (0.0013)	0.025 (0.0013)	0.029 (0.0013)
$n = 500$	marginal coverage	0.908 (0.0002)	0.907 (0.0004)	0.905 (0.0005)	0.906 (0.0005)	0.903 (0.0013)
	local coverage when $0 < x < 1/3$	0.909 (0.0004)	0.907 (0.0008)	0.907 (0.0032)	0.906 (0.0025)	0.905 (0.0034)
	local coverage when $1/3 \leq x < 2/3$	0.908 (0.0004)	0.906 (0.0006)	0.907 (0.0026)	0.906 (0.0021)	0.907 (0.0029)
	local coverage when $2/3 \leq x < 1$	0.909 (0.0005)	0.908 (0.0007)	0.901 (0.0024)	0.905 (0.0021)	0.898 (0.0028)
	area	3.369 (0.0211)	3.715 (0.0187)	3.305 (0.0206)	3.314 (0.0202)	3.287 (0.0169)
	prediction error	0.026 (0.0013)	0.033 (0.0013)	0.029 (0.0014)	0.028 (0.0014)	0.03 (0.0011)

Table 2: Diagnostics for conformal prediction regions for linear regression models with normal errors and constant variance. Local and marginal coverage properties, areas, and prediction errors are presented for the parametric conformal prediction region (third column), nonparametric conformal prediction region (fourth column), LS conformal prediction region (fifth column), LSLW conformal prediction region (sixth column), and HD prediction region (seventh column). Standard errors are in parentheses.

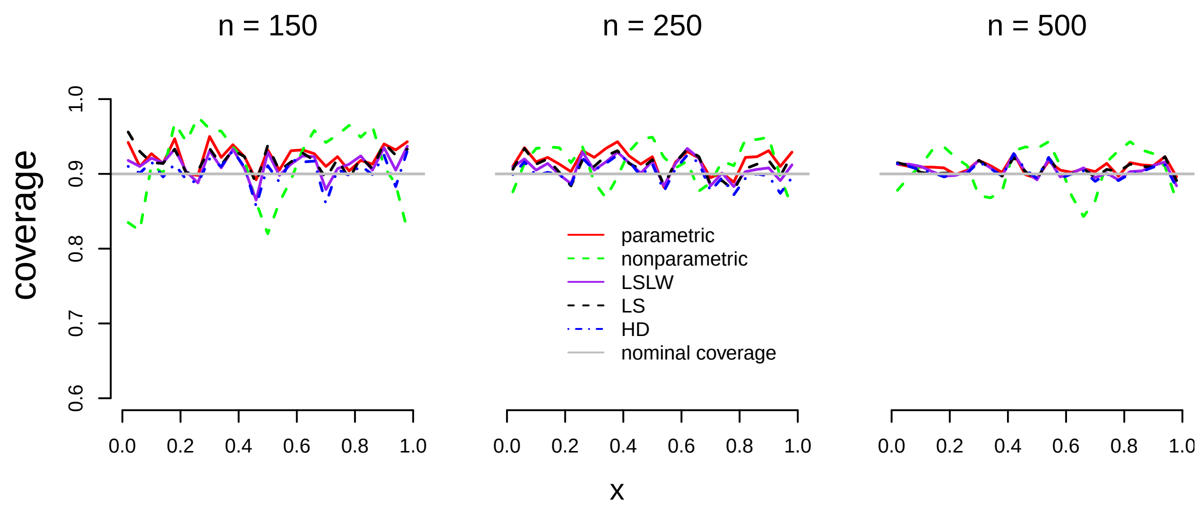


Figure 17: Plot of the estimated coverage probabilities of prediction regions across x and sample sizes.

8 Example plots of prediction regions

In this section we construct prediction regions corresponding to the simulations and results in Sections 5 through 8.3. In Gamma analyses we generate a dataset for each shape parameter considered with $n = 150$ and in regression analyses we generate a dataset for all sample sizes considered. For each of these datasets we depict the parametric, nonparametric, LS, LSLW conformal prediction regions over the observed data to visually assess the appropriateness of each prediction region. The findings from these figures are consistent with the findings from our numerical diagnostics. The parametric conformal prediction region gives visually natural bounds for the observed data when the model is correctly specified in small to moderate sample sizes and is appropriate when the model is misspecified. The LSLW conformal prediction region gives visually natural bounds for the observed data when the model is correctly specified, is appropriate under mild model misspecification, and is ill-fitting in settings where deviations about an estimated mean function are clearly not symmetric. The nonparametric conformal prediction region is coarse and is larger than necessary when the mean function is steep relative to its variability. The LS conformal prediction performs well when deviations about the estimated mean function are symmetirc and is sensitive to mild departures from that setting. This prediciton region is seen to provide overcoverage (undercoverage) in regions of the predictor space where variability about the estimated mean function is relatively small (large).

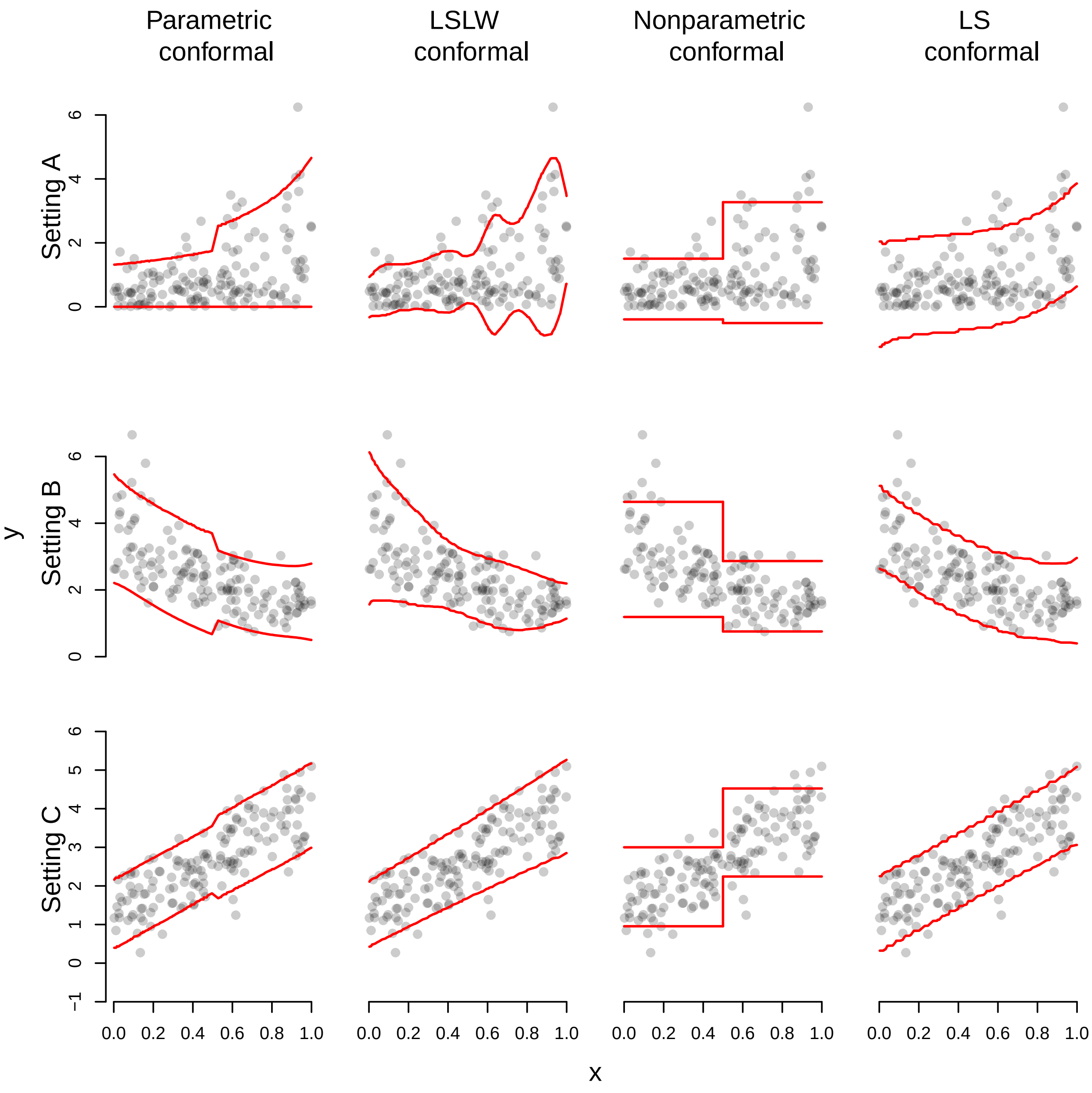


Figure 18: The depiction of conformal prediction regions when $n = 150$ that appears in Eck et al. [2019]. The rows display conformal prediction regions across simulation settings. The columns display the different conformal prediction regions. The top, middle, and bottom rows correspond to simulation setting a with shape parameter equal to 1, simulation setting b with shape parameter equal to 10, and simulation setting c respectively. The first column displays the parametric conformal prediction region which is misspecified in row 2, the second column displays the least squares locally weighted conformal prediction region, the third column displays the nonparametric conformal prediction region, and the fourth column displays the least squares conformal prediction region.

8.1 Plots corresponding to Section 5

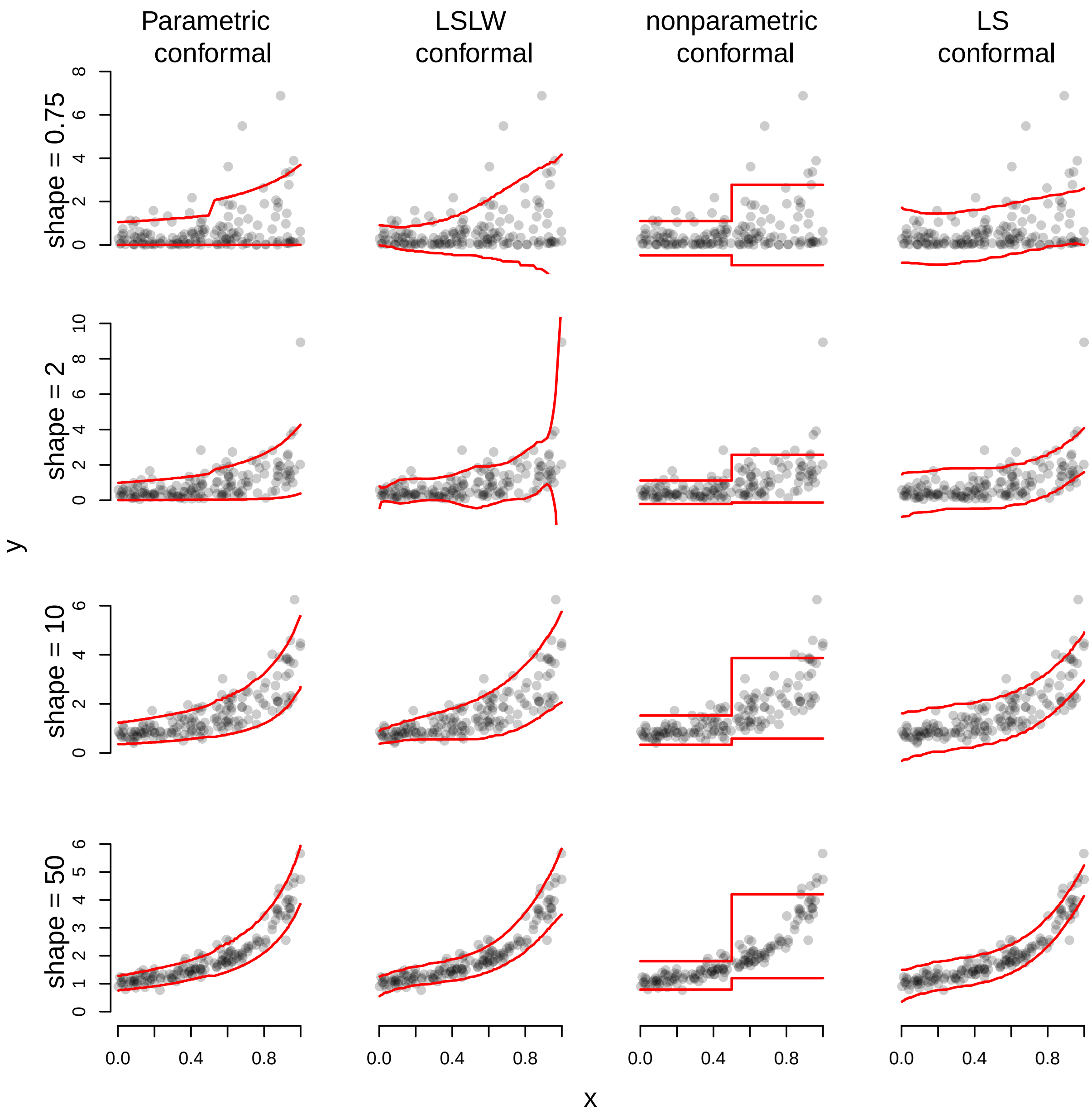


Figure 19: The depiction of conformal prediction regions under simulation setting A when $n = 150$ and the number of bins equals 2.

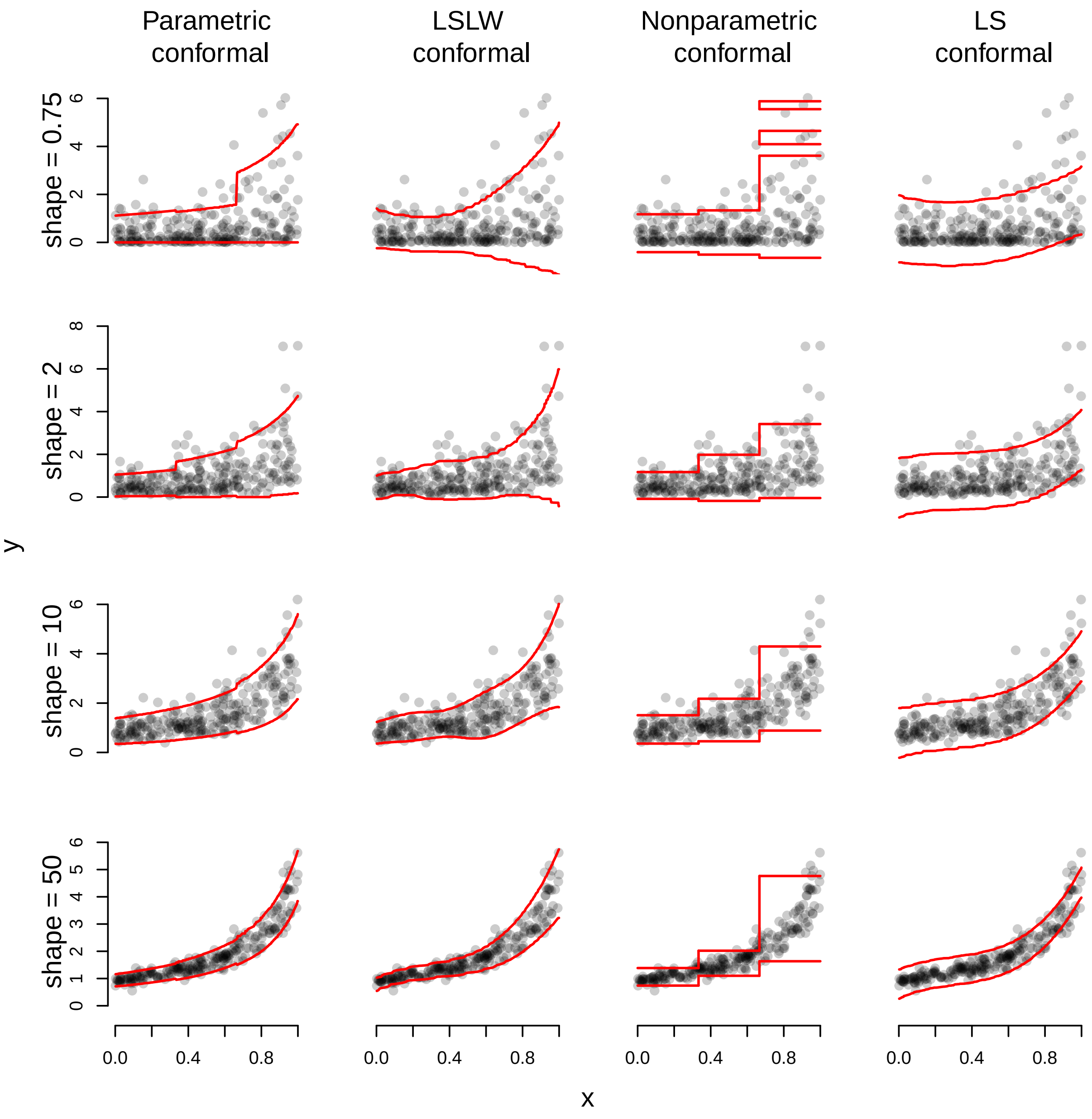


Figure 20: The depiction of conformal prediction regions under simulation setting A when $n = 250$ and the number of bins equals 3.

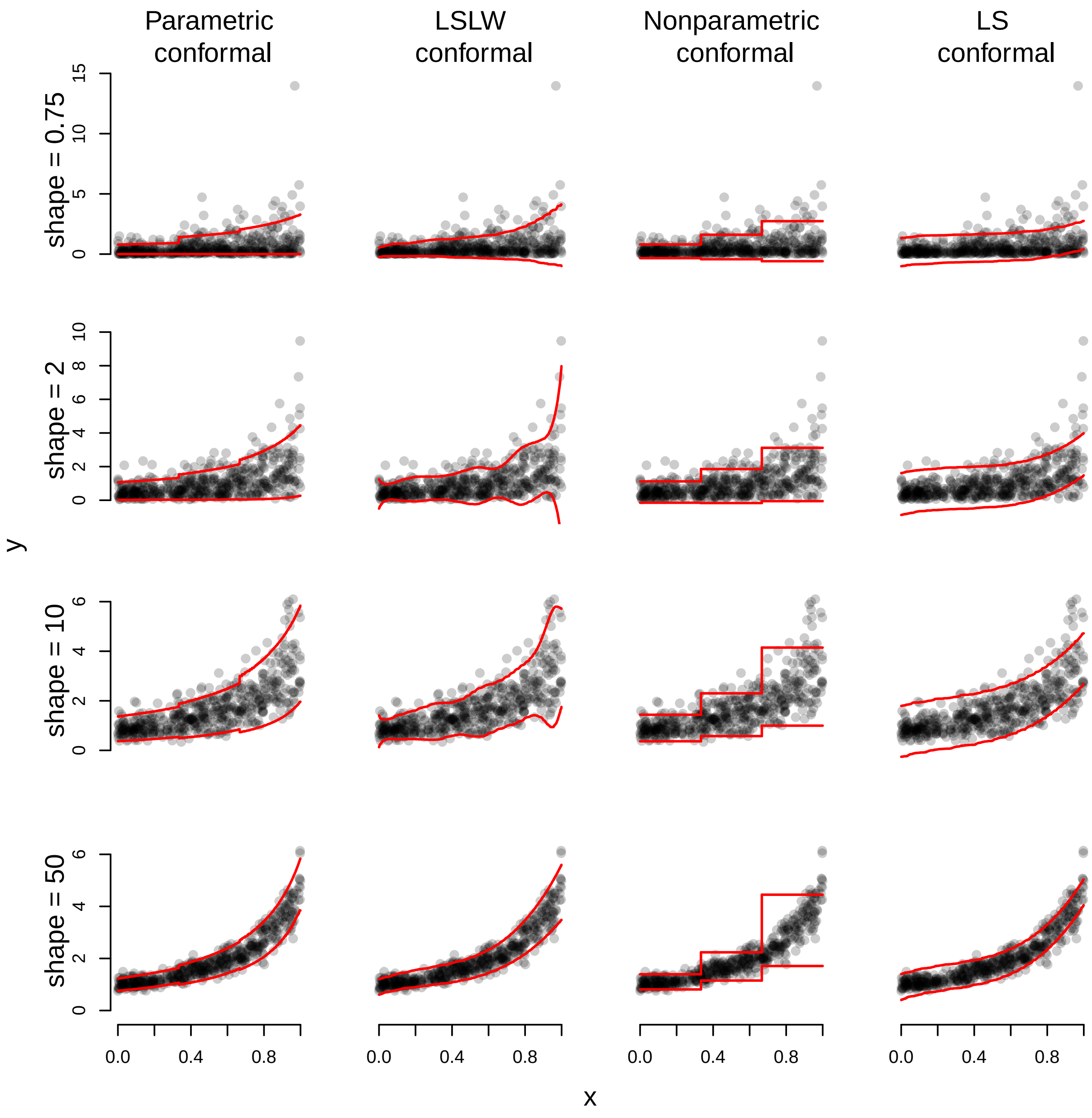


Figure 21: The depiction of conformal prediction regions under simulation setting A when $n = 500$ and the number of bins equals 3.

8.2 Plots corresponding to Section 6

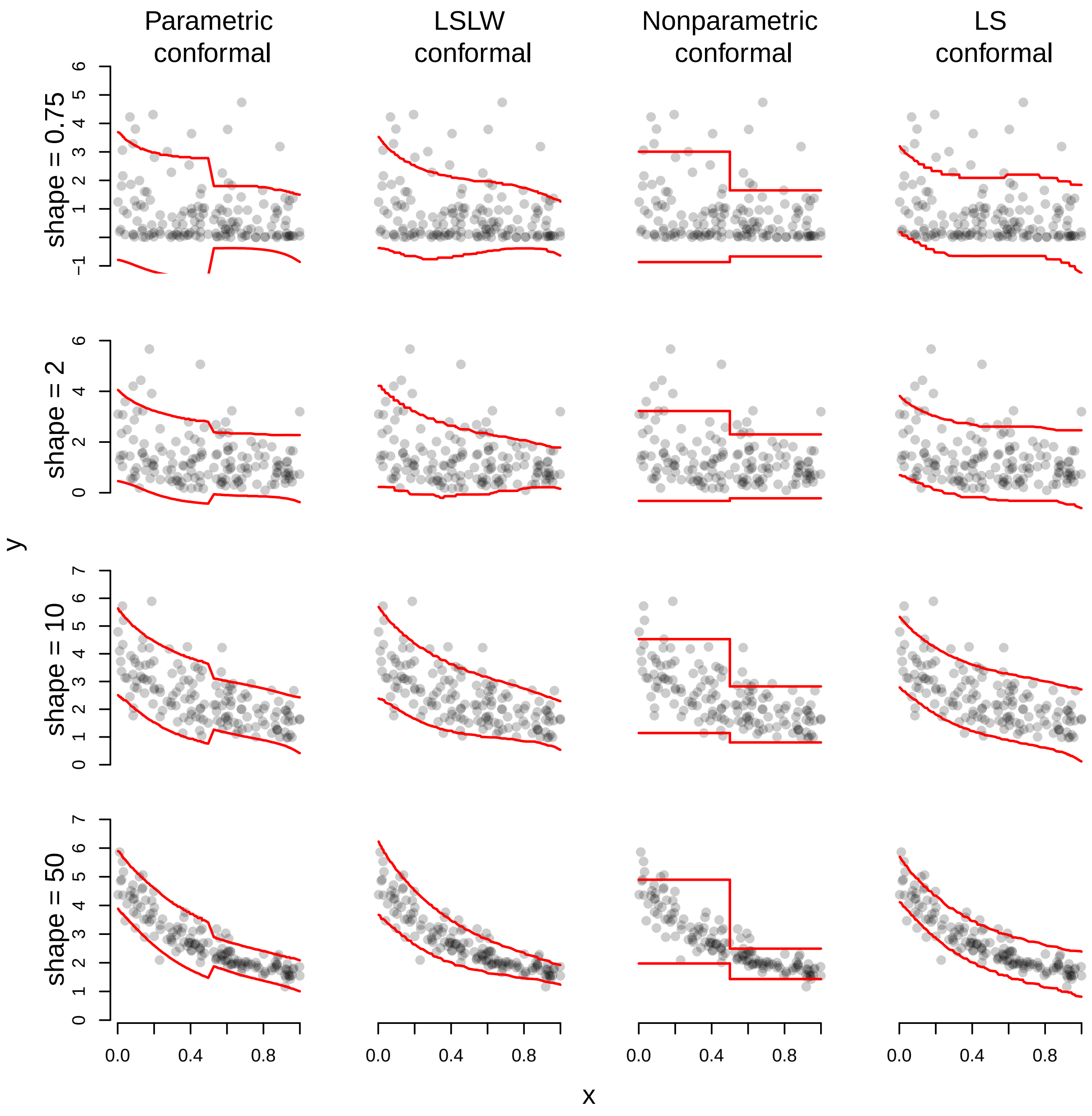


Figure 22: The depiction of conformal prediction regions under simulation setting B when $n = 150$ and the number of bins equals 2.

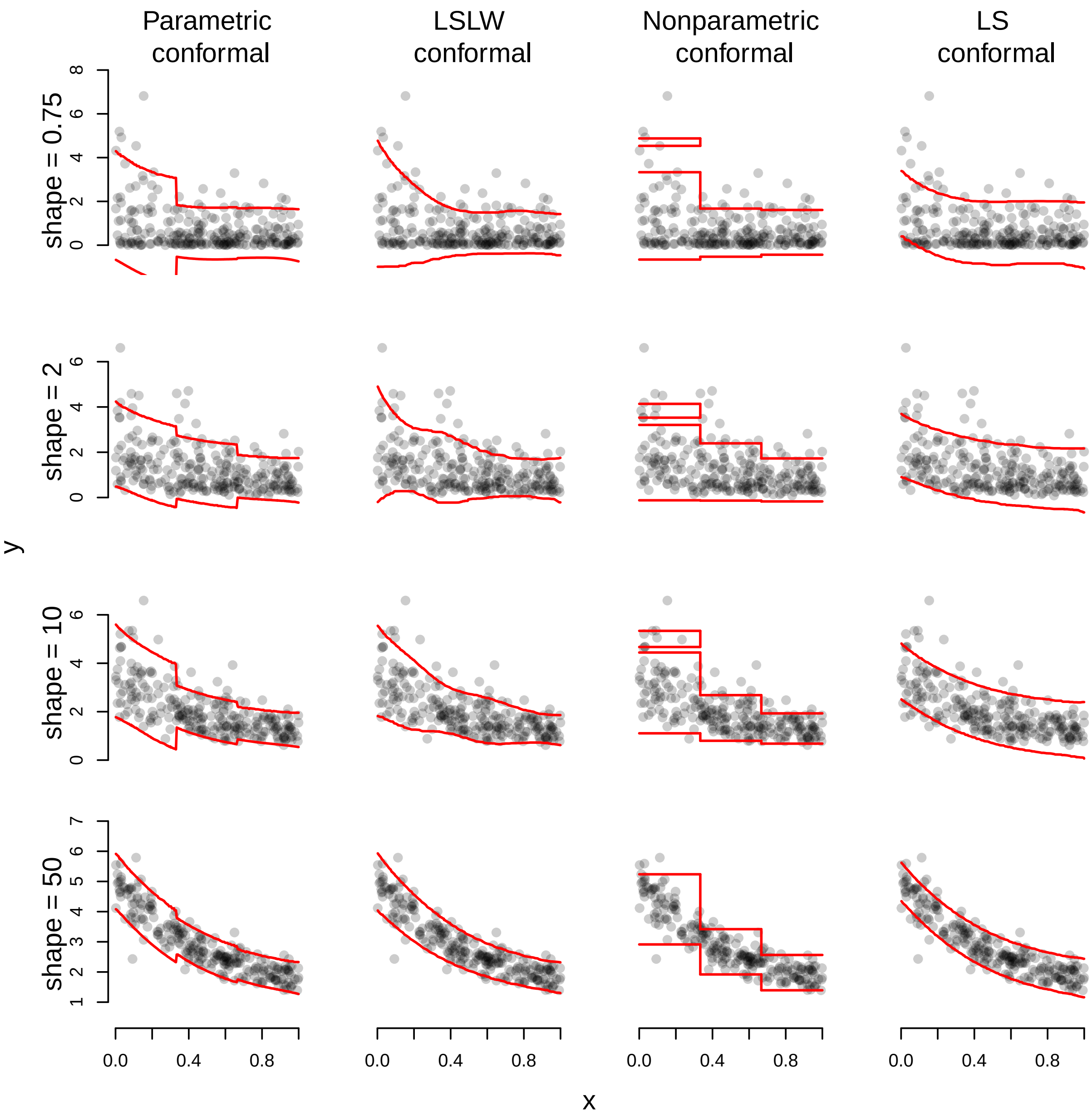


Figure 23: The depiction of conformal prediction regions under simulation setting B when $n = 250$ and the number of bins equals 3.

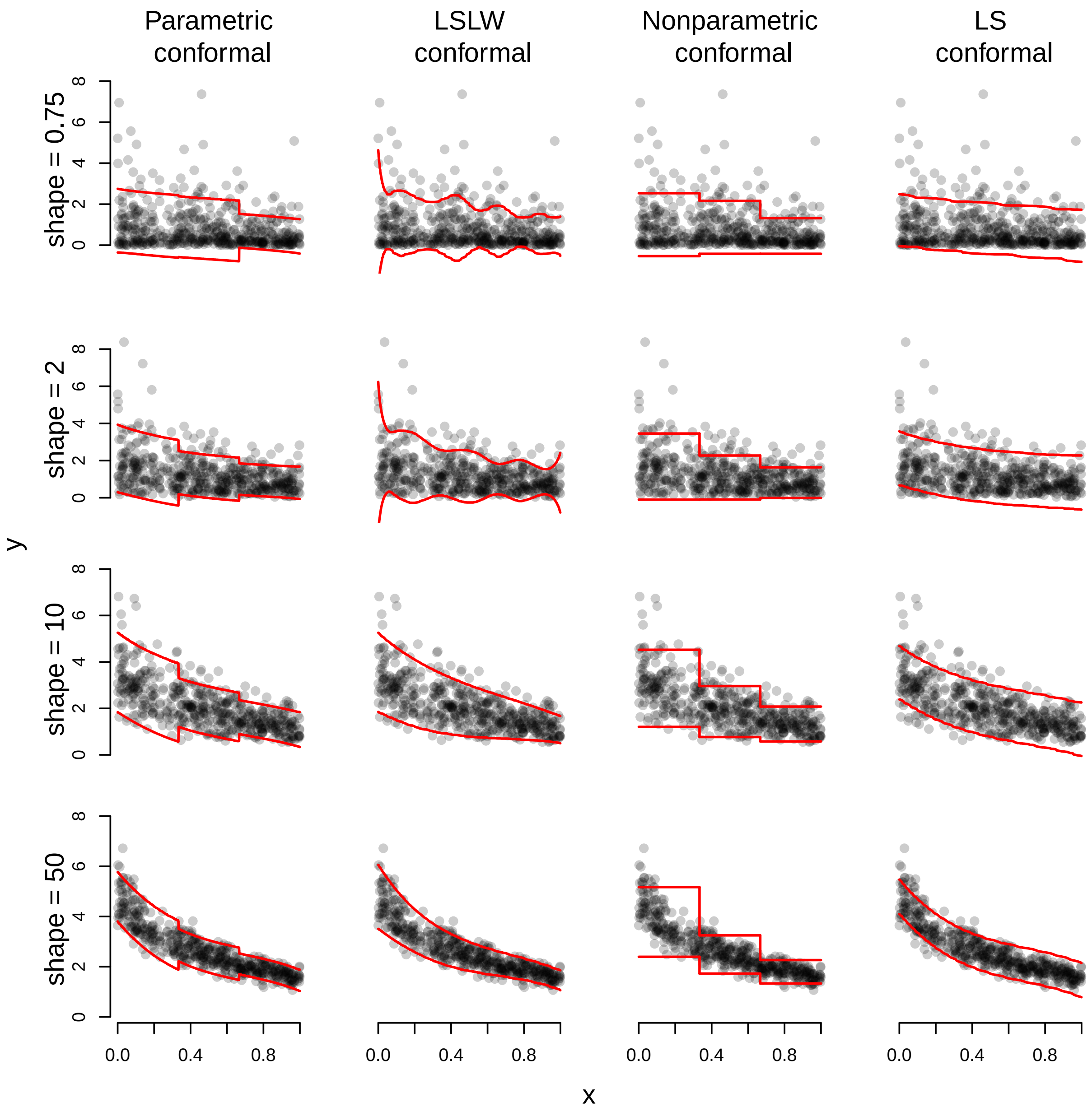


Figure 24: The depiction of conformal prediction regions under simulation setting B when $n = 500$ and the number of bins equals 3.

8.3 Plots corresponding to Section 7

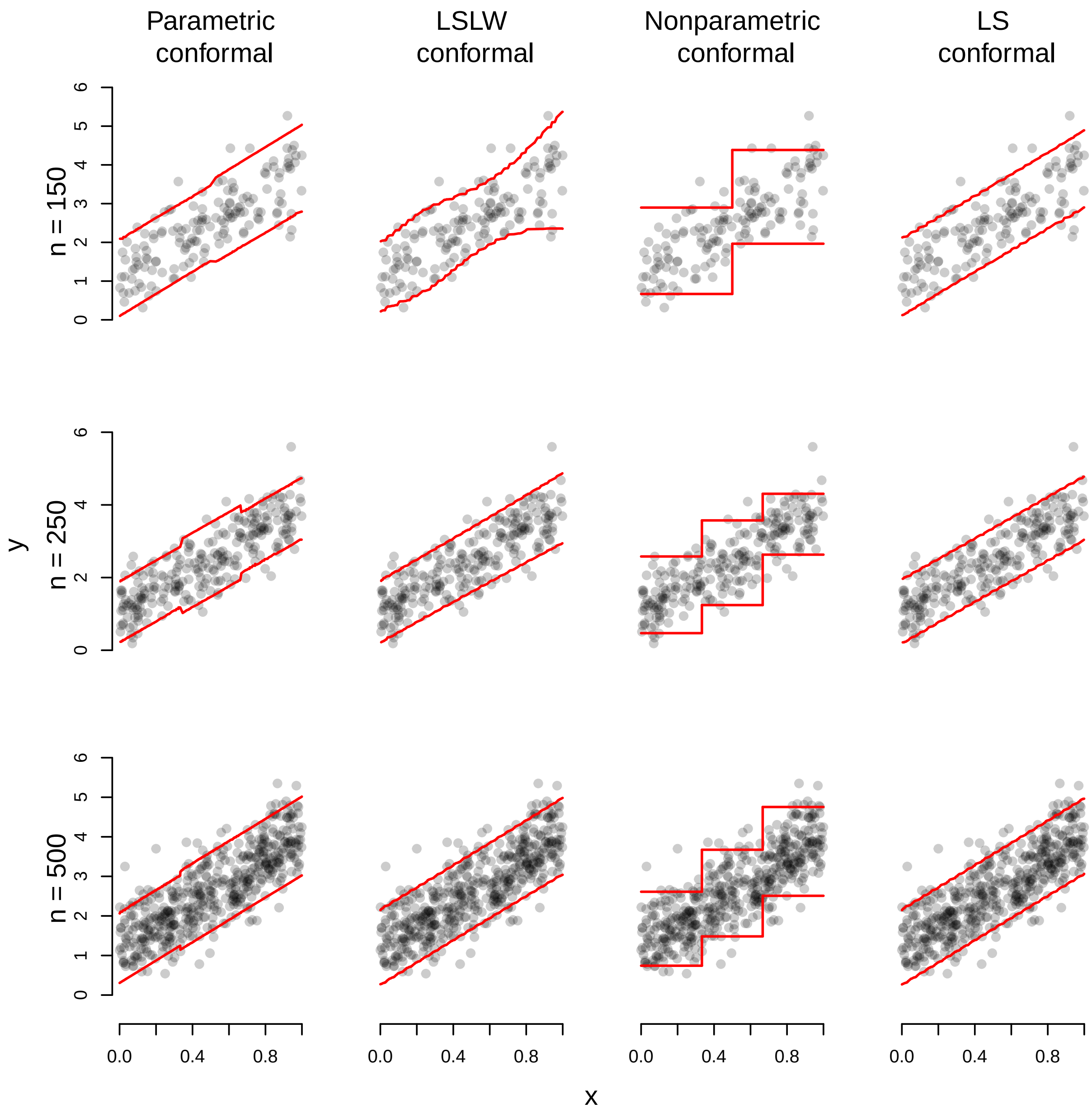


Figure 25: The depiction of conformal prediction regions under simulation setting C.

9 Creating this Document

The purpose of this document is to provide a completely reproducible exploration and motivation of conformal prediction. All of the R code presented in this document (or the corresponding .Rnw file) is run when this document is compiled using the linux terminal.

This document is created from its source file `supplement-conformal.Rnw` using `knitr` and the `pdflatex` command in the linux terminal. The `knitr` R package needs to be installed before this document can be compiled. Open R and install this package if you have not previously done so. To compile this document in the linux terminal, enter the command

```
Rscript -e "library(knitr); knit('supplement-conformal.Rnw')"
```

This produces the LaTex .tex file with the name `supplement-conformal.tex`. To get a corresponding .pdf file, enter the command

```
pdflatex supplement-conformal.tex
```

You may want to run the previous command twice in order to get labels and references right.

References

Daniel J. Eck. *conformal.glm: Conformal Prediction for Generalized Linear Regression Models*, 2018. `https://github.com/DEck13/conformal.glm`.

Daniel J. Eck, Forrest W. Crawford, and Peter M. Aronow. Conformal inference for exponential families and generalized linear models. *preprint*, 2019.

Julian Faraway. *faraway: Functions and Datasets for Books by Julian Faraway*, 2016. `https://cran.r-project.org/web/packages/faraway/index.html`.

Jing Lei and Larry Wasserman. Distribution-free prediction bands for nonparametric regression. *Journal of the Royal Statistical Society series B*, 76, 1:71–96, 2014.

Jing Lei, Max GSell, Alessandro Rinaldo, Ryan J Tibshirani, and Larry Wasserman. Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113(523):1094–1111, 2018.

Mike Meredith and John Kruschke. *HDInterval: Highest (Posterior) Density Intervals*, 2018. `https://cran.r-project.org/web/packages/HDInterval/`.

Ryan Tibshirani. *conformalInference: Tools for conformal inference in regression*, 2016. `https://github.com/ryantibs/conformal`.

James P Willems, J Terry Saunders, Dawn E Hunt, and John B Schorling. Prevalence of coronary heart disease risk factors among rural blacks: a community-based study. *Southern medical journal*, 90(8): 814–820, 1997.

World Health Organization. Use of glycated haemoglobin (hba1c) in diagnosis of diabetes mellitus: abbreviated report of a who consultation. Technical report, Geneva: World Health Organization, 2011.