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

Daniel J. Eck

September 20, 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 **??**, 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 similarily 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 **??**.

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
## 589.252   1.133  96.739

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
##  70.651   0.000  70.652

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.

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.

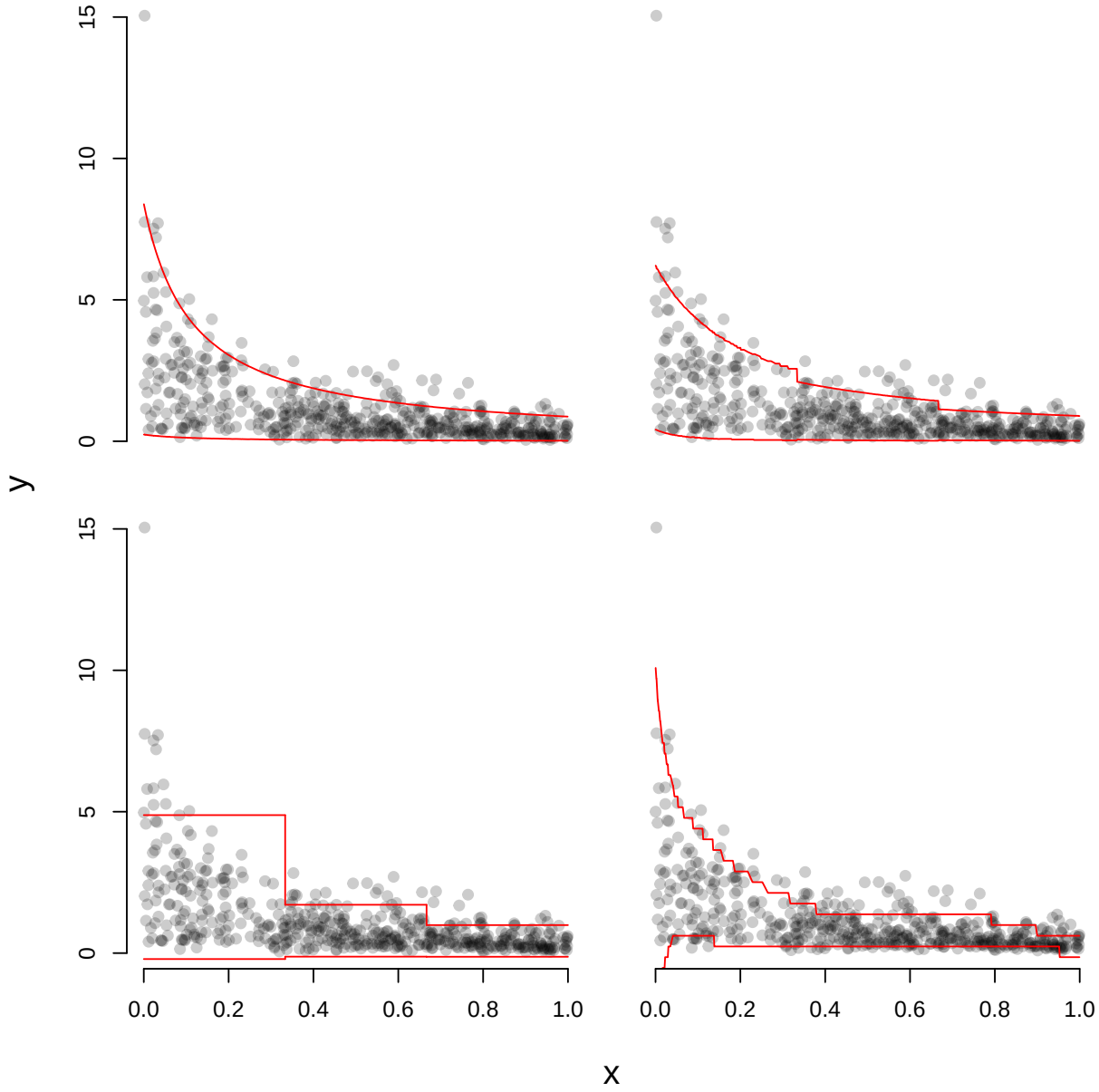
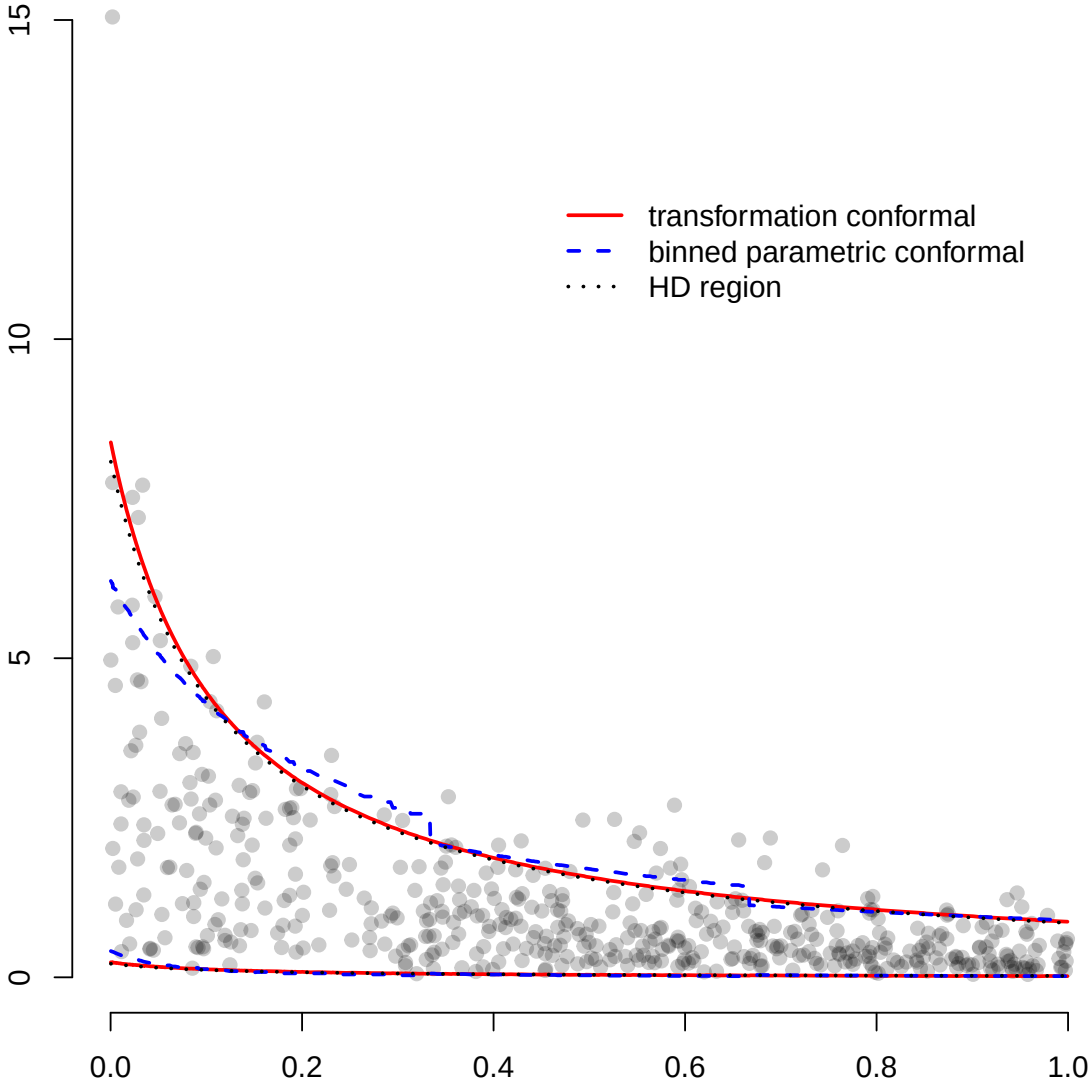


Figure 1: Sim setting: $n = 500$, shape = 2, bins = 3. The regions are depicted as follows: transformation based parametric conformal prediction (top-left panel), binned parametric conformal prediction region (top-right panel), nonparametric conformal prediction region (bottom-left panel), and LSLW conformal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

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

# 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	2.02	2.21
marginal coverage	0.91	0.91	0.91	0.77	0.90
bin 1 coverage	0.91	0.91	0.93	0.81	0.90
bin 2 coverage	0.89	0.90	0.90	0.75	0.88
bin 3 coverage	0.93	0.91	0.90	0.74	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

```
dat <- data.frame(y = y, x1 = x)
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 <- data.frame(y = y, x = x)
  colnames(dat)[2:(p+1)] <- paste("x", 1:p, sep = "")

  ## 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)
      dat$y <- y / sd(y)
    }
    if(link == "inverse"){
      rate <- (cbind(1, x) %*% beta) * shape
      y <- rgamma(n = n, shape = shape, rate = rate)
      dat$y <- y / sd(y)
    }
    if(link == "log"){
```

```
    rate <- (1 / exp(cbind(1, x) %*% beta)) * shape
    y <- rgamma(n = n, shape = shape, rate = rate)
    dat$y <- y / sd(y)
  }
}
if(family == "gaussian"){
  mu <- cbind(1, x) %*% beta
  y <- rnorm(n = n, mean = mu, sd = sd)
  dat$y <- y / sd(y)
}
if(family == "inverse.gaussian"){
  mu = 1 / sqrt(cbind(1, x) %*% beta)
  y <- rinvgauss(n = n, mean = mu)
  dat$y <- y / sd(y)
}

## fit the Gamma regression model
fit <- glm(y ~ x1, family = "Gamma", data = dat)
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)
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")
  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.

```
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
## 20924.373    90.547  6480.127
```

```
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

	Parametric Conformal	Transformation Conformal	Nonparametric Conformal	LS Conformal	LSLW Conformal	HD Region
marginal coverage	0.909 (0.0005)	0.912 (0.0006)	0.901 (0.0005)	0.893 (0.0016)	0.864 (0.0047)	0.9 (0.0012)
local coverage when $0 < x < 1/3$	0.909 (0.001)	0.911 (0.0023)	0.903 (0.0011)	0.706 (0.0048)	0.914 (0.0044)	0.902 (0.0026)
local coverage when $1/3 \leq x < 2/3$	0.908 (0.0006)	0.913 (0.0026)	0.9 (0.0006)	0.975 (0.0023)	0.867 (0.0054)	0.903 (0.003)
local coverage when $2/3 \leq x < 1$	0.909 (0.0008)	0.912 (0.0028)	0.898 (0.0007)	0.995 (0.0011)	0.81 (0.0082)	0.895 (0.0028)
area	1.832 (0.0203)	1.853 (0.0186)	2.035 (0.0231)	2.498 (0.034)	2.007 (0.0248)	1.79 (0.017)
prediction error	0.23 (0.0233)	0.187 (0.0179)	0.137 (0.0077)	0.217 (0.0138)	0.096 (0.0065)	0.198 (0.0184)

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, transformation, nonparametric, LS, and LSLW conformal prediction regions and the HD prediction region.

(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.

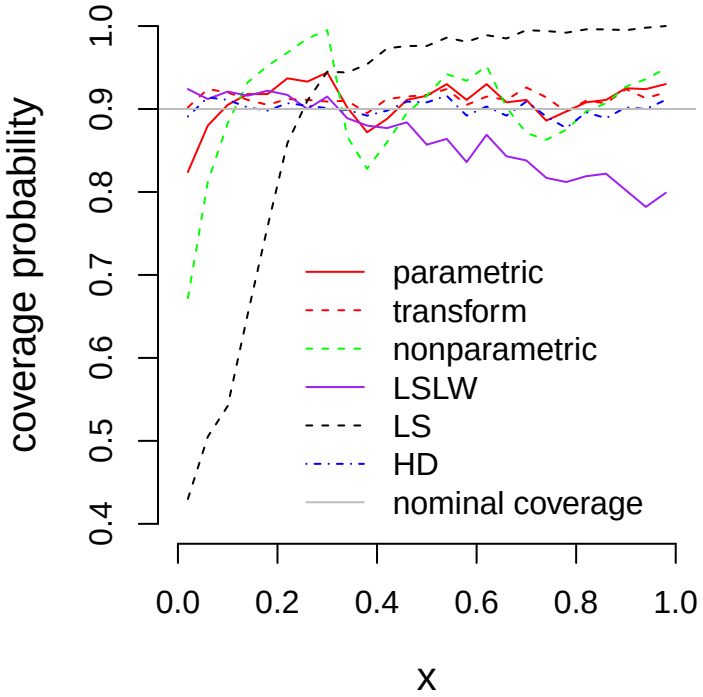


Figure 2: Plot of the estimated coverage probabilities of prediction regions across x.

4 Predicting the risk of diabetes

In this Section we reproduce the data analysis that appears in Section 6 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].

```
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
```

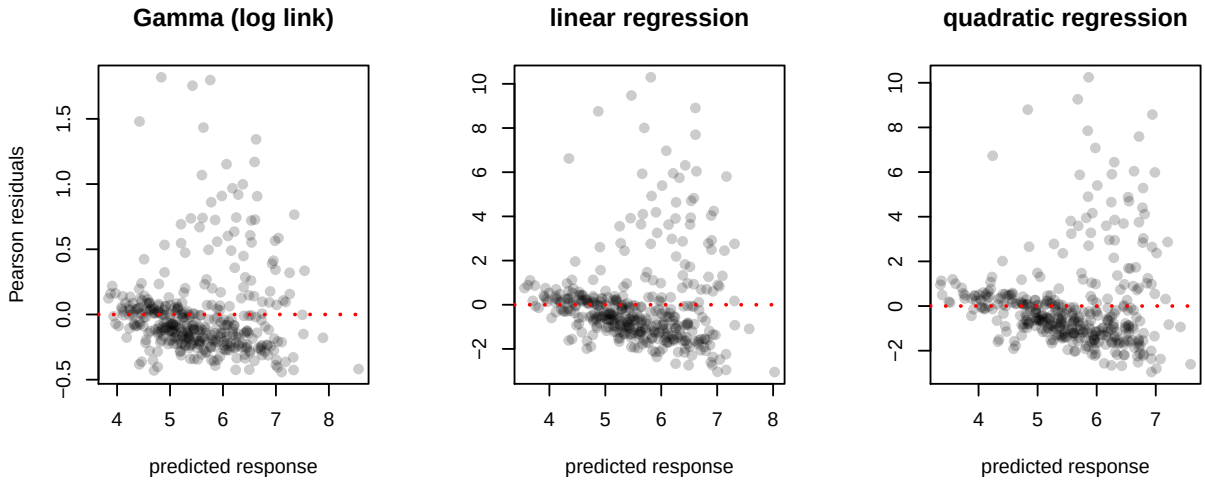


Figure 3: Predictive performance of candidate models.

```
## 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.

Six conformal prediction regions are considered for predicting glycosylated hemoglobin percentage. These conformal prediction regions are the binned and transformation parametric conformal prediction region with a Gamma model fit, the binned and transformation 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 binned 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,
    method = "both"))

##    user  system elapsed
## 805.660   0.913 151.499

gamma.paraconfbin <- conformal.gamma$paraconfbin
gamma.transform <- conformal.gamma$transformconf
head(gamma.paraconfbin)

##         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

head(gamma.transform)

##          lwr       upr
## [1,] 3.112995  7.807848
## [2,] 3.186762  8.064868
## [3,] 4.253349 10.786782
## [4,] 3.707534  9.338419
## [5,] 4.051886 10.160153
## [6,] 3.226689  8.169503

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

##    user  system elapsed
## 413.969   0.888  73.228

gaussian.paraconfbin <- conformal.gaussian$paraconfbin
gaussian.transform <- conformal.gaussian$transformconf
head(gaussian.paraconfbin)

##        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

head(gaussian.transform)

##          lwr       upr
## [1,] 2.569146  9.189792
## [2,] 2.748739  9.399384
## [3,] 4.349760 11.010405
## [4,] 3.535198 10.170843
## [5,] 4.033451 10.649097
## [6,] 2.801680  9.462326

## 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
##   4.433   0.000   4.432
```

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

```
##           lwr      upr
## [1,] 2.237500 7.526137
## [2,] 2.644318 7.526137
## [3,] 3.864773 9.153409
## [4,] 3.051136 8.339773
## [5,] 3.864773 9.153409
## [6,] 2.644318 7.932955
```

```
## 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
##  49.123   0.000  49.125
```

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

```
##           lwr       upr
## [1,] 3.051136  6.712500
## [2,] 3.051136  7.119318
## [3,] 2.237500 11.187500
## [4,] 2.644318  9.153409
## [5,] 2.237500 10.373864
## [6,] 3.051136  7.119318
```

We also consider the highest density prediction region under the Gamma model, this region is constructed below.

```
## HD region
library(HDInterval)
betaMLE <- coefficients(m.gamma.log)
shapeMLE <- as.numeric(gamma.shape(m.gamma.log)[1])
rateMLE <- (1 / exp(X %*% betaMLE)) * shapeMLE
#rateMLE <- cbind(1, x) %*% betaMLE * shapeMLE
HD <- do.call(rbind,
  lapply(1:nrow(X), function(j){
    hdi(qgamma, 0.90, shape = shapeMLE, rate = rateMLE[j, 1])
  }))
colnames(HD) <- c("lwr", "upr")
head(HD)

##           lwr      upr
## [1,] 2.556343 7.182955
## [2,] 2.628804 7.386560
## [3,] 3.510997 9.865393
## [4,] 3.050565 8.571645
## [5,] 3.328264 9.351939
## [6,] 2.659659 7.473256
```

4.1 Results in Eck et al. [2019]

Diagnostics from the six conformal prediction regions are depicted in Table 2 of the manuscript Eck et al. [2019]. The error tolerance for all prediction regions was set at $\alpha = 0.10$. We see that parametric conformal prediction regions maintain their advertised finite sample marginal validity for the predictions of glycosylated hemoglobin. These prediction regions provide a balance between marginal coverage, size, prediction error, and average conditional coverage (the average of the coverage probabilities taken over small subregions of the predictor space). The transformation Gamma conformal prediction region balances these criteria particularly nicely. This prediction region is relatively small, has relatively small prediction error, and it gives near nominal desired coverage. This finding is expected when the underlying estimated density used as the parametric conformity measure is a good approximation of the data generating model.

We now compute the diagnostics of these six conformal prediction regions and the highest density prediction region, and we 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.trans <- local.coverage(region = gamma.transform,
  data = dat, bins = 1, d = 2)
marginal.gaussian.trans <- local.coverage(
  region = gaussian.transform, data = dat, bins = 1, d = 2)
marginal.gamma.bin <- local.coverage(region = gamma.paraconfbin,
  data = dat, bins = 1, d = 2)
marginal.gaussian.bin <- local.coverage(region = gaussian.paraconfbin,
```

```
  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)
marginal.HD <- local.coverage(region = HD, data = dat, bins = 1, d = 2)
table.marginal <- matrix(c(marginal.gamma.trans, marginal.gaussian.trans,
  marginal.gamma.bin, marginal.gaussian.bin, marginal.LSLW, marginal.LS,
  marginal.HD), ncol = 7)
table.marginal <- round(table.marginal, 3)
colnames(table.marginal) <- c("gamma trans", "gaussian trans", "gamma bin",
  "gaussian bin", "LSLW", "LS", "HD")
rownames(table.marginal) <- "marginal coverage"

## conditional coverages
d <- ncol(dat) - 2
conditional.gamma.trans <- local.coverage(region = gamma.transform,
  data = dat, bins = 3, d = d)
conditional.gaussian.trans <- local.coverage(
  region = gaussian.transform, data = dat, bins = 3, d = d)
conditional.gamma.bin <- local.coverage(region = gamma.paraconfbin,
  data = dat, bins = 3, d = d)
conditional.gaussian.bin <- local.coverage(region = gaussian.paraconfbin,
  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)
conditional.HD <- local.coverage(region = HD, data = dat, bins = 3, d = d)
table.conditional <- matrix(cbind(conditional.gamma.trans,
  conditional.gaussian.trans, conditional.gamma.bin, conditional.gaussian.bin,
  conditional.LSLW, conditional.LS, conditional.HD), ncol = 7)
table.conditional <- round(table.conditional, 3)
colnames(table.conditional) <- c("gamma trans", "gaussian trans",
  "gamma bin", "gaussian bin", "LSLW", "LS", "HD")
avg.cond.coverage <- colMeans(table.conditional,na.rm = TRUE)
```

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

```
pred.error.gamma.trans <- absolute.error(y = dat$glyhb, region = gamma.transform)
pred.error.gaussian.trans <- absolute.error(y = y, region = gaussian.transform)
pred.error.gamma.bin <- absolute.error(y = dat$glyhb, region = gamma.paraconfbin)
pred.error.gaussian.bin <- absolute.error(y = y, region = gaussian.paraconfbin)
pred.error.LSLW <- absolute.error(y = y, region = LSLW)
pred.error.LS <- absolute.error(y = y, region = LS)
pred.error.HD <- absolute.error(y = y, region = HD)
table.pred.error <- matrix(c(pred.error.gamma.trans, pred.error.gaussian.trans,
  pred.error.gamma.bin, pred.error.gaussian.bin, pred.error.LSLW, pred.error.LS,
  pred.error.HD), ncol = 7)
colnames(table.pred.error) <- c("gamma trans", "gaussian trans", "gamma bin",
  "gaussian bin ", "LSLW", "LS", "HD")
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.trans <- volume.regions(region = gamma.transform, preds = preds)
volume.gaussian.trans <- volume.regions(region = gaussian.transform, preds = preds)
volume.gamma.bin <- volume.regions(region = gamma.paraconfbin, preds = preds)
volume.gaussian.bin <- volume.regions(region = gaussian.paraconfbin, preds = preds)
volume.LSLW <- volume.regions(region = LSLW, preds = preds)
volume.LS <- volume.regions(region = LS, preds = preds)
volume.HD <- volume.regions(region = HD, preds = preds)
table.volume <- matrix(c(volume.gamma.trans, volume.gaussian.trans,
  volume.gamma.bin, volume.gaussian.bin, volume.LSLW, volume.LS, volume.HD), ncol =
colnames(table.volume) <- c("gamma trans", "gaussian trans", "gamma bin",
  "gaussian bin", "LSLW", "LS", "HD")
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(d
  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.trans <- volume.factors(region = gamma.transform,
  data = dat, object = m.gamma.log)
volume.gaussian.trans <- volume.factors(gaussian.transform,
  data = dat, object = m.gaussian)
volume.gamma.bin <- volume.factors(region = gamma.paraconfbin,
  data = dat, object = m.gamma.log)
volume.gaussian.bin <- volume.factors(gaussian.paraconfbin,
  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)
volume.HD <- volume.factors(HD,
```

```
  data = dat, object = m.gamma.log)

table.volume <- matrix(c(volume.gamma.trans, volume.gaussian.trans,
  volume.gamma.bin, volume.gaussian.bin, volume.LSLW, volume.LS,
  volume.HD), ncol = 7)
colnames(table.volume) <- c("gamma trans", "gaussian trans", "gamma bin",
  "gaussian bin", "LSLW", "LS", "HD")
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, 3, 3, 3))
```

	gamma trans	gaussian trans	gamma bin	gaussian bin	LSLW	LS	HD
marginal coverage	0.901	0.924	0.909	0.906	0.880	0.888	0.906
volume	7.656	8.560	7.349	7.730	8.574	7.103	7.294
pred error	0.653	0.425	0.931	0.849	0.803	1.012	0.948
avg.cond.coverage	0.856	0.888	0.874	0.849	0.863	0.827	0.873

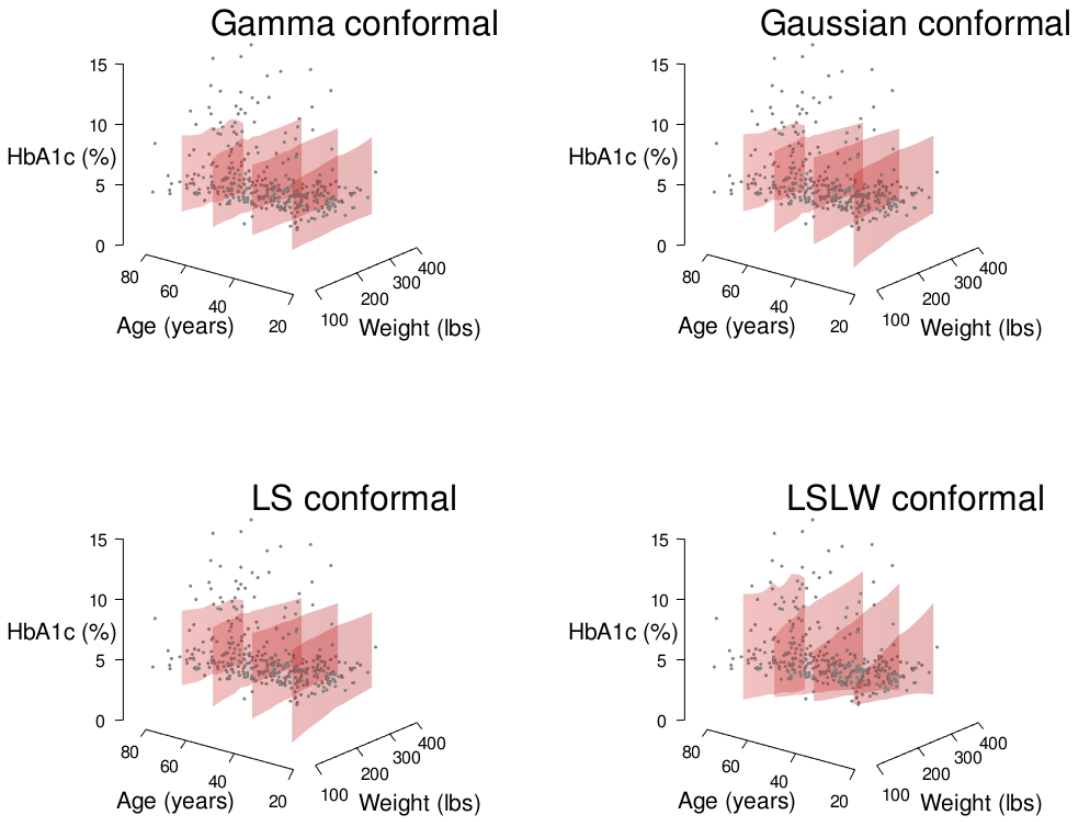


Figure 4: Figure 4 in Eck et al. [2019]. Conformal prediction regions for glycosylated hemoglobin projected onto the age and weight predictor axes. Upper and lower bounds of the conformal prediction region are loess smoothed for visual appearance. The code that constructed this figure is displayed in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

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

We perform Monte Carlo simulations to investigate the coverage properties, size, and prediction error of the Gamma conformal prediction region under transformation...

```
beta <- c(1.25, -1)
gamma_simulator <- function(beta, n = 150, shape = 0.5,
  bins = 2, alpha = 0.10, B = 2){

  p <- d <- length(beta) - 1

  models <- lapply(1:B, FUN = function(j){
    x <- matrix(runif(n*p), ncol = p)
    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 = "")
    fit <- glm(y ~ x1, family = "Gamma", data = dat,
      x = TRUE, y = TRUE)
    output = list(model = fit, dat = dat)
    output
  })

  funs <- lm.funs(intercept = TRUE)
  train.fun <- funs$train.fun
  predict.fun <- funs$predict.fun
  conformal_regions <- lapply(models, FUN = function(obj){

    model <- obj$model
    dat <- obj$dat
    model$data <- dat
```

```
  x <- model$x[, -1]
  y <- model$y
  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)

  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$df
  mad.train.fun <- function(x, y, out = NULL){
    smooth.spline(x[, 1], y, lambda = lambda,
    df = df, nknots = 10)
  }
  mad.predict.fun <- function(out, newx){
    predict(out, as.data.frame(newx))$y[, 1]
  }
  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)

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

  conf <- conformal.glm(model, nonparametric = TRUE, method = "both",
    bins = bins, cores = 7)
  parabinCI <- conf$paraconfbin
  transformCI <- conf$transformconf
  nonparabinCI <- conf$nonparaconfbin

  out = list(parabinCI = parabinCI, transformCI = transformCI,
    nonparabinCI = nonparabinCI, LSLWCI = LSLWCI, LSCI = LSCI,
    HDCI = HDCI)
  out
})
```

```
  diagnostics_regions <- lapply(1:B, FUN = function(j){

    obj <- conformal_regions[[j]]
    dat <- as.data.frame(models[[j]]$dat)
    y <- as.numeric(dat[, 1])

    diagnostics <- lapply(obj, FUN = function(region){

      output <- NULL
      if(class(region) == "matrix"){
        marginal.coverage <- local.coverage(region = region,
          data = dat, d = p, bins = 1, at.data = "TRUE")
        local.coverage <- local.coverage(region = region,
          data = dat, d = p, bins = bins, at.data = "TRUE")
        local.inx.coverage <- local.coverage(region = region,
          data = dat, d = p, bins = 25, at.data = "TRUE")
        output <- c(marginal.coverage, local.coverage,
          local.inx.coverage, mean(apply(region, 1, diff)),
          absolute.error(y = y, region = region))
      }
      else{
        marginal.coverage <- local.coverage(region = region,
          nonparametric = "TRUE", data = dat, d = p, bins = 1,
          at.data = "TRUE")
        local.coverage <- local.coverage(region = region,
          nonparametric = "TRUE", data = dat, d = p, bins = bins,
          at.data = "TRUE")
        local.inx.coverage <- local.coverage(region = region,
          nonparametric = "TRUE", data = dat, d = p, bins = 25,
          at.data = "TRUE")
        output <- c(marginal.coverage, local.coverage,
          local.inx.coverage, area.nonparametric(region),
          absolute.error.nonparametric(data = dat,
            region = region))
      }
      output
    })
    do.call(rbind, diagnostics)

  })

  diagnostics_regions
}
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 0.5.

```
set.seed(13)
beta <- c(1.25, -1)
bins <- 2
B <- 200
n <- 150
system.time(out.gamma.150.2.0.5 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 0.5, bins = bins, B = B)))

##      user    system   elapsed
## 16345.387   204.330  7011.362

diagnostics.150.2.0.5 <- do.call(rbind, lapply(split(out.gamma.150.2.0.5,
  f = as.factor(rownames(out.gamma.150.2.0.5))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 0.75.

```
set.seed(13)
system.time(out.gamma.150.2.0.75 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 0.75, bins = bins, B = B)))

## Warning in log(ifelse(y == 0, 1, y/mu)): NaNs produced
## Error: no valid set of coefficients has been found: please supply starting
values
## Timing stopped at: 0.239 0 0.24

diagnostics.150.2.0.75 <- do.call(rbind, lapply(split(out.gamma.150.2.0.75,
  f = as.factor(rownames(out.gamma.150.2.0.75))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))

## Error in split(out.gamma.150.2.0.75, f = as.factor(rownames(out.gamma.150.2.0.75
object 'out.gamma.150.2.0.75' not found
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 1.

```
set.seed(13)
system.time(out.gamma.150.2.1 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 1, bins = bins, B = B)))

##      user    system   elapsed
## 16618.992   207.724  7035.463

diagnostics.150.2.1 <- do.call(rbind, lapply(split(out.gamma.150.2.1,
  f = as.factor(rownames(out.gamma.150.2.1))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 1.5.

```
set.seed(13)
system.time(out.gamma.150.2.1.5 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 1.5, bins = bins, B = B)))

##      user    system   elapsed
## 17682.956   214.156  7197.812

diagnostics.150.2.1.5 <- do.call(rbind, lapply(split(out.gamma.150.2.1.5,
  f = as.factor(rownames(out.gamma.150.2.1.5))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 2.

```
set.seed(13)
system.time(out.gamma.150.2.2 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 2, bins = bins, B = B)))

##      user    system   elapsed
## 19198.965   220.455  7458.240

diagnostics.150.2.2 <- do.call(rbind, lapply(split(out.gamma.150.2.2,
  f = as.factor(rownames(out.gamma.150.2.2))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 5.

```
set.seed(13)
system.time(out.gamma.150.2.5 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 5, bins = bins, B = B)))

##      user    system   elapsed
## 20250.447   223.468  7670.782

diagnostics.150.2.5 <- do.call(rbind, lapply(split(out.gamma.150.2.5,
  f = as.factor(rownames(out.gamma.150.2.5))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 7.

```
set.seed(13)
system.time(out.gamma.150.2.7 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 7, bins = bins, B = B)))

##      user    system   elapsed
## 19494.677   224.071  7521.206
```

```
diagnostics.150.2.7 <- do.call(rbind, lapply(split(out.gamma.150.2.7,
  f = as.factor(rownames(out.gamma.150.2.7))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 10.

```
set.seed(13)
system.time(out.gamma.150.2.10 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 10, bins = bins, B = B)))

##      user    system   elapsed
## 18152.953   217.606  7289.143

diagnostics.150.2.10 <- do.call(rbind, lapply(split(out.gamma.150.2.10,
  f = as.factor(rownames(out.gamma.150.2.10))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 12.

```
set.seed(13)
system.time(out.gamma.150.2.12 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 12, bins = bins, B = B)))

##      user    system   elapsed
## 17510.891   215.420  7175.219

diagnostics.150.2.12 <- do.call(rbind, lapply(split(out.gamma.150.2.12,
  f = as.factor(rownames(out.gamma.150.2.12))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 15.

```
set.seed(13)
system.time(out.gamma.150.2.15 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 15, bins = bins, B = B)))

##      user    system   elapsed
## 16722.807   211.209  7049.776

diagnostics.150.2.15 <- do.call(rbind, lapply(split(out.gamma.150.2.15,
  f = as.factor(rownames(out.gamma.150.2.15))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 20.

```
set.seed(13)
system.time(out.gamma.150.2.20 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 20, bins = bins, B = B)))

##      user    system   elapsed
## 16014.976   209.719  6938.588

diagnostics.150.2.20 <- do.call(rbind, lapply(split(out.gamma.150.2.20,
  f = as.factor(rownames(out.gamma.150.2.20))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 25.

```
set.seed(13)
system.time(out.gamma.150.2.25 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 25, bins = bins, B = B)))

##      user    system   elapsed
## 15321.230   228.135  6826.116

diagnostics.150.2.25 <- do.call(rbind, lapply(split(out.gamma.150.2.25,
  f = as.factor(rownames(out.gamma.150.2.25))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 30.

```
set.seed(13)
system.time(out.gamma.150.2.30 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 30, bins = bins, B = B)))

##      user    system   elapsed
## 18924.091   283.772  8870.716

diagnostics.150.2.30 <- do.call(rbind, lapply(split(out.gamma.150.2.30,
  f = as.factor(rownames(out.gamma.150.2.30))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 40.

```
set.seed(13)
system.time(out.gamma.150.2.40 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 40, bins = bins, B = B)))

##      user    system   elapsed
## 15480.968   239.581  7165.747
```

```
diagnostics.150.2.40 <- do.call(rbind, lapply(split(out.gamma.150.2.40,
  f = as.factor(rownames(out.gamma.150.2.40))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 50.

```
set.seed(13)
system.time(out.gamma.150.2.50 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 50, bins = bins, B = B)))

##      user    system   elapsed
## 14341.742   221.524  6656.901

diagnostics.150.2.50 <- do.call(rbind, lapply(split(out.gamma.150.2.50,
  f = as.factor(rownames(out.gamma.150.2.50))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 60.

```
set.seed(13)
system.time(out.gamma.150.2.60 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 60, bins = bins, B = B)))

##      user    system   elapsed
## 14248.439   221.815  6637.738

diagnostics.150.2.60 <- do.call(rbind, lapply(split(out.gamma.150.2.60,
  f = as.factor(rownames(out.gamma.150.2.60))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 70.

```
set.seed(13)
system.time(out.gamma.150.2.70 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 70, bins = bins, B = B)))

##      user    system   elapsed
## 14017.465   216.953  6649.866

diagnostics.150.2.70 <- do.call(rbind, lapply(split(out.gamma.150.2.70,
  f = as.factor(rownames(out.gamma.150.2.70))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 80.

```
set.seed(13)
system.time(out.gamma.150.2.80 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 80, bins = bins, B = B)))

##      user    system   elapsed
## 14773.413   229.631  6915.396

diagnostics.150.2.80 <- do.call(rbind, lapply(split(out.gamma.150.2.80,
  f = as.factor(rownames(out.gamma.150.2.80))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 90.

```
set.seed(13)
system.time(out.gamma.150.2.90 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 90, bins = bins, B = B)))

##      user    system   elapsed
## 14279.957   227.990  6826.982

diagnostics.150.2.90 <- do.call(rbind, lapply(split(out.gamma.150.2.90,
  f = as.factor(rownames(out.gamma.150.2.90))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 100.

```
set.seed(13)
system.time(out.gamma.150.2.100 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 100, bins = bins, B = B)))

##      user    system   elapsed
## 14015.261   216.283  6778.074

diagnostics.150.2.100 <- do.call(rbind, lapply(split(out.gamma.150.2.100,
  f = as.factor(rownames(out.gamma.150.2.100))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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
B <- 50
n <- 150
system.time(out.gamma.250.3.0.5 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 0.5, bins = bins, B = B)))
```

```
##     user   system  elapsed
## 4333.197   56.968 1827.720

diagnostics.250.3.0.5 <- do.call(rbind, lapply(split(out.gamma.250.3.0.5,
  f = as.factor(rownames(out.gamma.250.3.0.5))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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

```
set.seed(13)
system.time(out.gamma.250.3.0.75 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 0.75, bins = bins, B = B)))

##     user   system  elapsed
## 4347.287   58.482 1832.657

diagnostics.250.3.0.75 <- do.call(rbind, lapply(split(out.gamma.250.3.0.75,
  f = as.factor(rownames(out.gamma.250.3.0.75))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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

```
set.seed(13)
system.time(out.gamma.250.3.1 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 1, bins = bins, B = B)))

##     user   system  elapsed
## 4236.173   58.476 1809.167

diagnostics.250.3.1 <- do.call(rbind, lapply(split(out.gamma.250.3.1,
  f = as.factor(rownames(out.gamma.250.3.1))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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

```
set.seed(13)
system.time(out.gamma.250.3.1.5 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 1.5, bins = bins, B = B)))

##     user   system  elapsed
## 4568.366   60.385 1858.704

diagnostics.250.3.1.5 <- do.call(rbind, lapply(split(out.gamma.250.3.1.5,
  f = as.factor(rownames(out.gamma.250.3.1.5))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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

```
set.seed(13)
system.time(out.gamma.250.3.2 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 2, bins = bins, B = B)))

##     user   system  elapsed
## 4761.727   61.526 1891.619

diagnostics.250.3.2 <- do.call(rbind, lapply(split(out.gamma.250.3.2,
  f = as.factor(rownames(out.gamma.250.3.2))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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

```
set.seed(13)
system.time(out.gamma.250.3.5 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 5, bins = bins, B = B)))

##     user   system  elapsed
## 4950.414   61.890 1912.842

diagnostics.250.3.5 <- do.call(rbind, lapply(split(out.gamma.250.3.5,
  f = as.factor(rownames(out.gamma.250.3.5))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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

```
set.seed(13)
system.time(out.gamma.250.3.7 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 7, bins = bins, B = B)))

##     user   system  elapsed
## 4664.964   59.634 1875.774

diagnostics.250.3.7 <- do.call(rbind, lapply(split(out.gamma.250.3.7,
  f = as.factor(rownames(out.gamma.250.3.7))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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

```
set.seed(13)
system.time(out.gamma.250.3.10 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 10, bins = bins, B = B)))

##     user   system  elapsed
## 4380.695   61.199 1823.847

diagnostics.250.3.10 <- do.call(rbind, lapply(split(out.gamma.250.3.10,
  f = as.factor(rownames(out.gamma.250.3.10))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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

```
set.seed(13)
system.time(out.gamma.250.3.12 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 12, bins = bins, B = B)))

##     user   system  elapsed
## 4256.742   60.729 1800.828

diagnostics.250.3.12 <- do.call(rbind, lapply(split(out.gamma.250.3.12,
  f = as.factor(rownames(out.gamma.250.3.12))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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

```
set.seed(13)
system.time(out.gamma.250.3.15 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 15, bins = bins, B = B)))

##     user   system  elapsed
## 3979.282   59.681 1755.243

diagnostics.250.3.15 <- do.call(rbind, lapply(split(out.gamma.250.3.15,
  f = as.factor(rownames(out.gamma.250.3.15))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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

```
set.seed(13)
system.time(out.gamma.250.3.20 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 20, bins = bins, B = B)))

##     user   system  elapsed
## 3752.405   59.992 1722.503

diagnostics.250.3.20 <- do.call(rbind, lapply(split(out.gamma.250.3.20,
  f = as.factor(rownames(out.gamma.250.3.20))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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

```
set.seed(13)
system.time(out.gamma.250.3.25 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 25, bins = bins, B = B)))
```

```
##     user   system  elapsed
## 3653.899   58.167 1709.166

diagnostics.250.3.25 <- do.call(rbind, lapply(split(out.gamma.250.3.25,
  f = as.factor(rownames(out.gamma.250.3.25))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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
set.seed(13)
system.time(out.gamma.500.3.0.5 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 0.5, bins = bins, B = B)))

##      user    system   elapsed
## 25396.226    87.914  9592.865

diagnostics.500.3.0.5 <- do.call(rbind, lapply(split(out.gamma.500.3.0.5,
  f = as.factor(rownames(out.gamma.500.3.0.5))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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

```
set.seed(13)
system.time(out.gamma.500.3.0.75 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 0.75, bins = bins, B = B)))

##      user    system   elapsed
## 24360.489    86.915  9359.060

diagnostics.500.3.0.75 <- do.call(rbind, lapply(split(out.gamma.500.3.0.75,
  f = as.factor(rownames(out.gamma.500.3.0.75))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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

```
set.seed(13)
system.time(out.gamma.500.3.1 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 1, bins = bins, B = B)))

##      user    system   elapsed
## 25001.896    86.408  9495.724
```

```
diagnostics.500.3.1 <- do.call(rbind, lapply(split(out.gamma.500.3.1,
  f = as.factor(rownames(out.gamma.500.3.1))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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

```
set.seed(13)
system.time(out.gamma.500.3.1.5 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 1.5, bins = bins, B = B)))

##      user    system   elapsed
## 26471.771    88.401  9697.074

diagnostics.500.3.1.5 <- do.call(rbind, lapply(split(out.gamma.500.3.1.5,
  f = as.factor(rownames(out.gamma.500.3.1.5))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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

```
set.seed(13)
system.time(out.gamma.500.3.2 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 2, bins = bins, B = B)))

##      user    system   elapsed
## 29889.860    87.406 10213.192

diagnostics.500.3.2 <- do.call(rbind, lapply(split(out.gamma.500.3.2,
  f = as.factor(rownames(out.gamma.500.3.2))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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

```
set.seed(13)
system.time(out.gamma.500.3.5 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 5, bins = bins, B = B)))

##      user    system   elapsed
## 29549.683    85.437 10150.362

diagnostics.500.3.5 <- do.call(rbind, lapply(split(out.gamma.500.3.5,
  f = as.factor(rownames(out.gamma.500.3.5))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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

```
set.seed(13)
system.time(out.gamma.500.3.7 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 7, bins = bins, B = B)))
```

```
##      user    system   elapsed
## 28462.132    85.665  9986.717

diagnostics.500.3.7 <- do.call(rbind, lapply(split(out.gamma.500.3.7,
  f = as.factor(rownames(out.gamma.500.3.7))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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

```
set.seed(13)
system.time(out.gamma.500.3.10 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 10, bins = bins, B = B)))

##      user    system   elapsed
## 27569.663    84.856  9847.450

diagnostics.500.3.10 <- do.call(rbind, lapply(split(out.gamma.500.3.10,
  f = as.factor(rownames(out.gamma.500.3.10))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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

```
set.seed(13)
system.time(out.gamma.500.3.12 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 12, bins = bins, B = B)))

##      user    system   elapsed
## 26384.662    83.713  9678.140

diagnostics.500.3.12 <- do.call(rbind, lapply(split(out.gamma.500.3.12,
  f = as.factor(rownames(out.gamma.500.3.12))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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

```
set.seed(13)
system.time(out.gamma.500.3.15 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 15, bins = bins, B = B)))

##      user    system   elapsed
## 28110.812    84.563  9991.120

diagnostics.500.3.15 <- do.call(rbind, lapply(split(out.gamma.500.3.15,
  f = as.factor(rownames(out.gamma.500.3.15))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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

```
set.seed(13)
system.time(out.gamma.500.3.20 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 20, bins = bins, B = B)))

##      user    system   elapsed
## 24238.182    84.149  9356.842

diagnostics.500.3.20 <- do.call(rbind, lapply(split(out.gamma.500.3.20,
  f = as.factor(rownames(out.gamma.500.3.20))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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

```
set.seed(13)
system.time(out.gamma.500.3.25 <- do.call(rbind,
  gamma_simulator(beta = beta, n = n, shape = 25, bins = bins, B = B)))

##      user    system   elapsed
## 23086.432    83.901  9206.594

diagnostics.500.3.25 <- do.call(rbind, lapply(split(out.gamma.500.3.25,
  f = as.factor(rownames(out.gamma.500.3.25))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B))))
```

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 **??**). 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 **??**. 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.

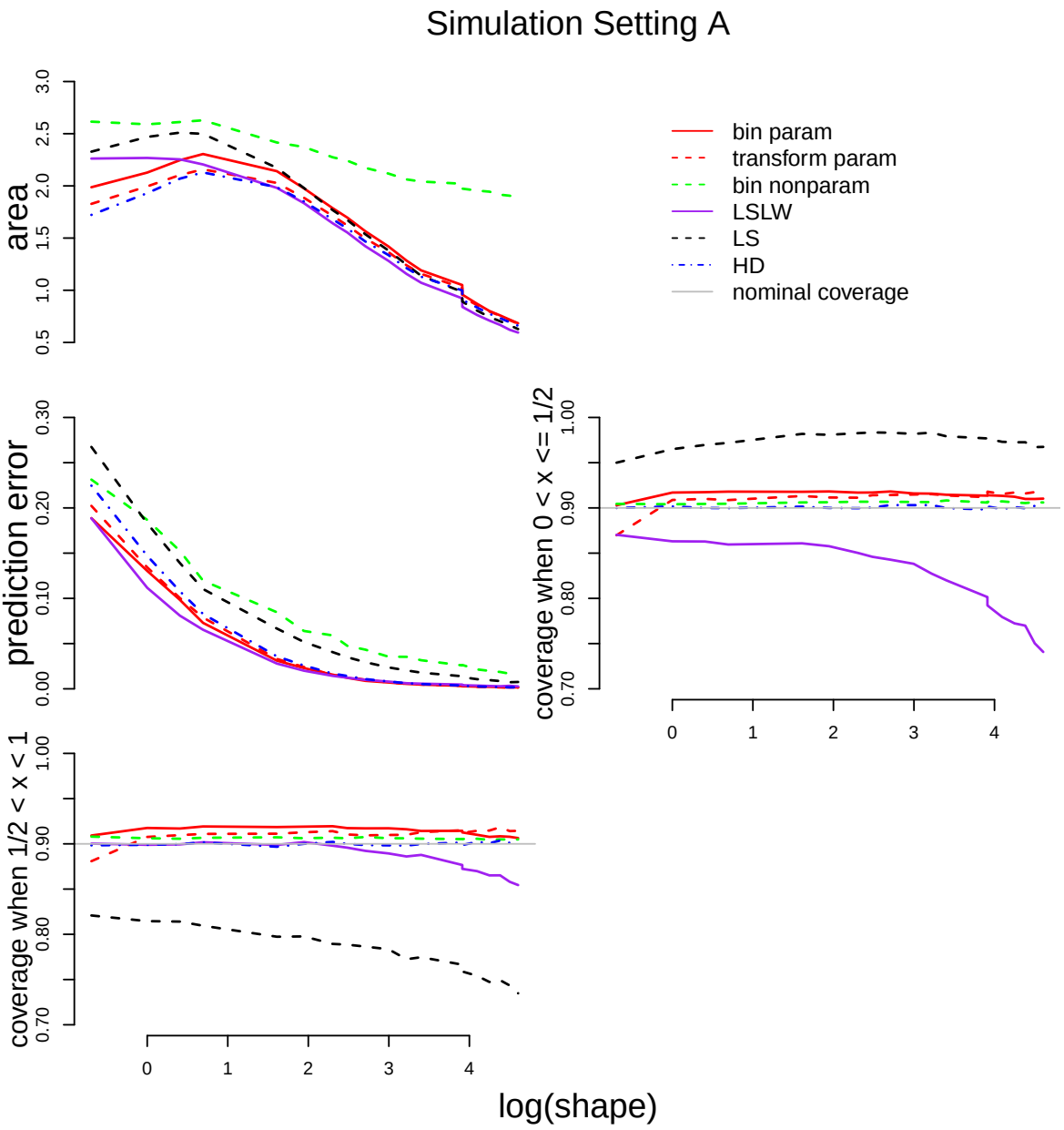


Figure 5: This figure compares the performance of the binned parametric, transformation, binned nonparametric, least squares, and least squares locally weighted conformal prediciton region and the highest density predcition region when $n = 150$ and the number of bins equals 2. The specific diagnostics used to compare these prediciton regions is the area, prediction error, and the coverage probability with respect to binning across shape parameter values. The average of 200 Monte Carlo samples at each shape parameter value in these simulation settings form the lines that are depicted in this figure.

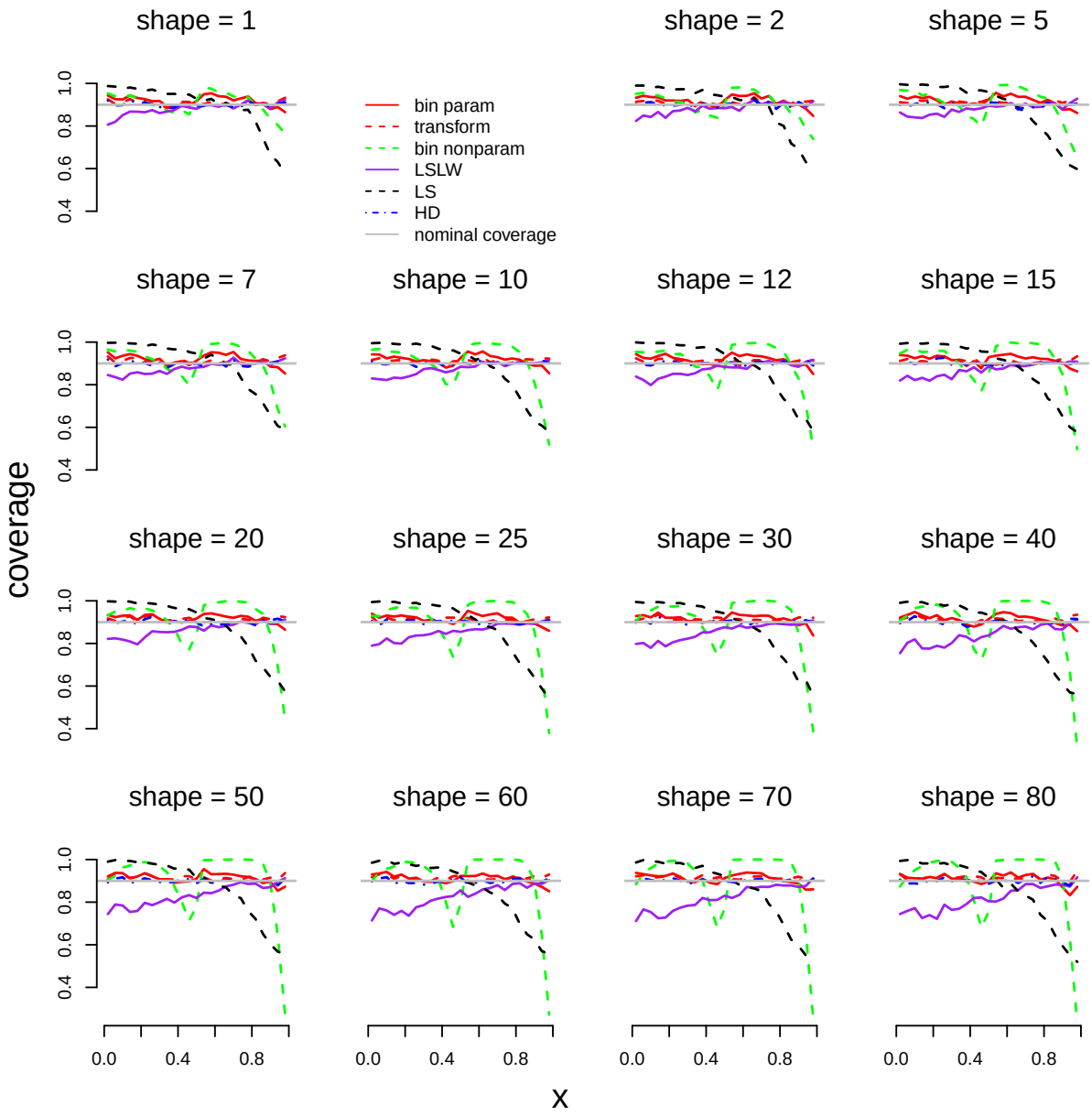


Figure 6: Plot of the estimated coverage probabilities of prediction regions across x and shape parameter values when the model is correctly specified, $n = 150$, and the number of bins is equal to 2.

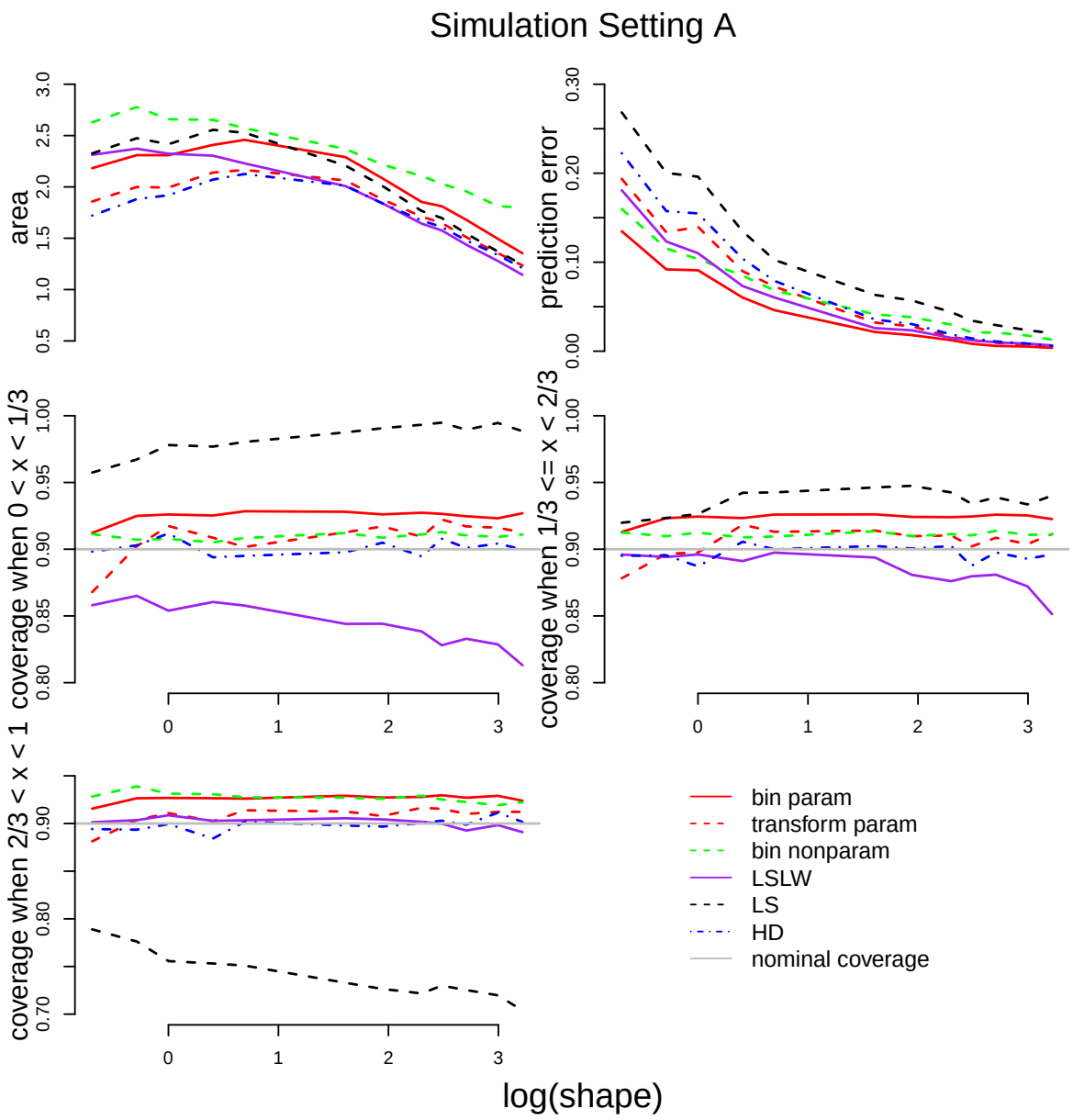


Figure 7: This figure compares the performance of the binned parametric, transformation, binned nonparametric, least squares, and least squares locally weighted conformal prediciton region and the highest density prediction region when $n = 250$ and the number of bins equals 3. The specific diagnostics 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.

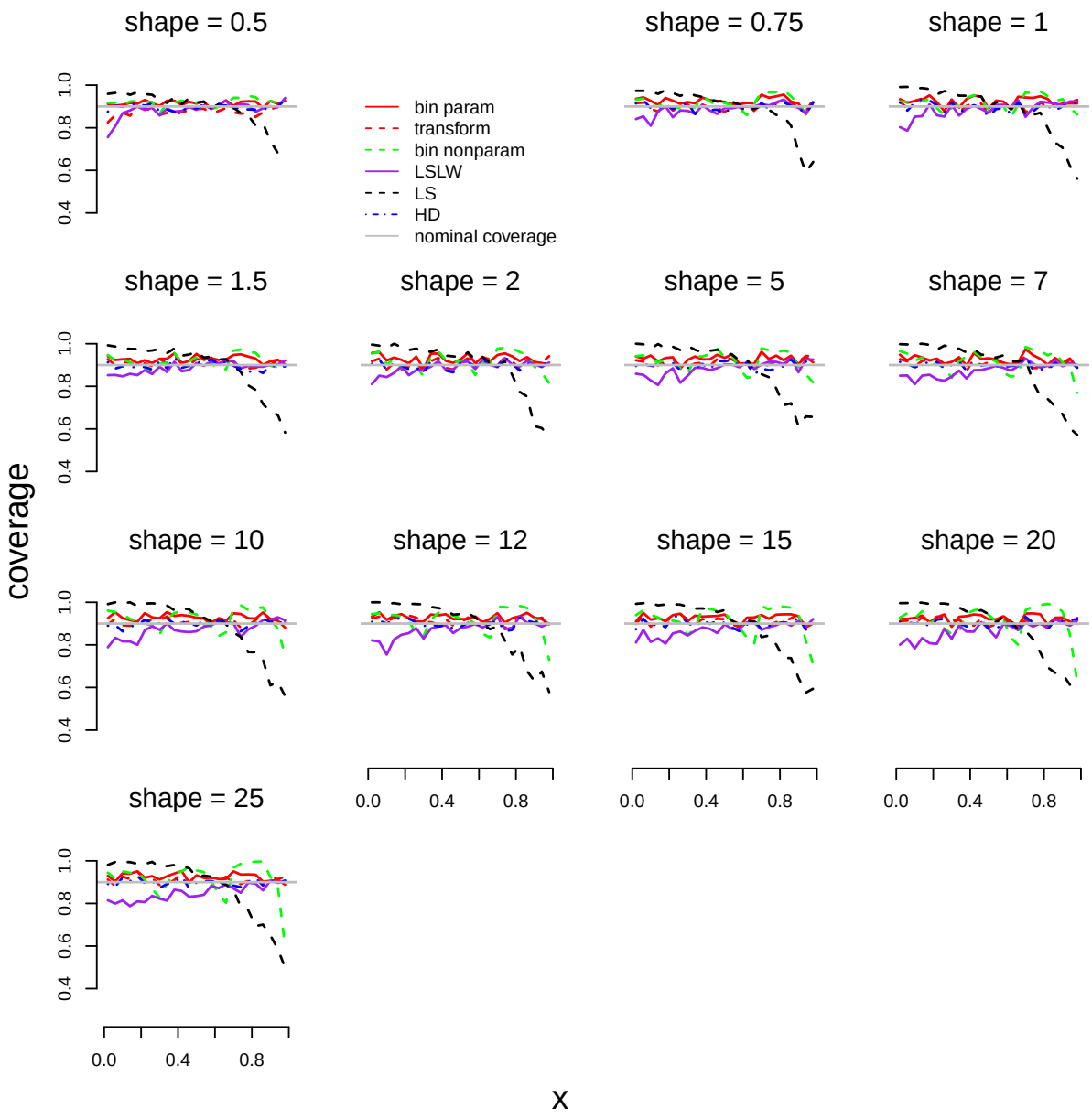


Figure 8: Plot of the estimated coverage probabilities of prediction regions across x and shape parameter values when the model is correctly specified, $n = 250$, and the number of bins is equal to 3.

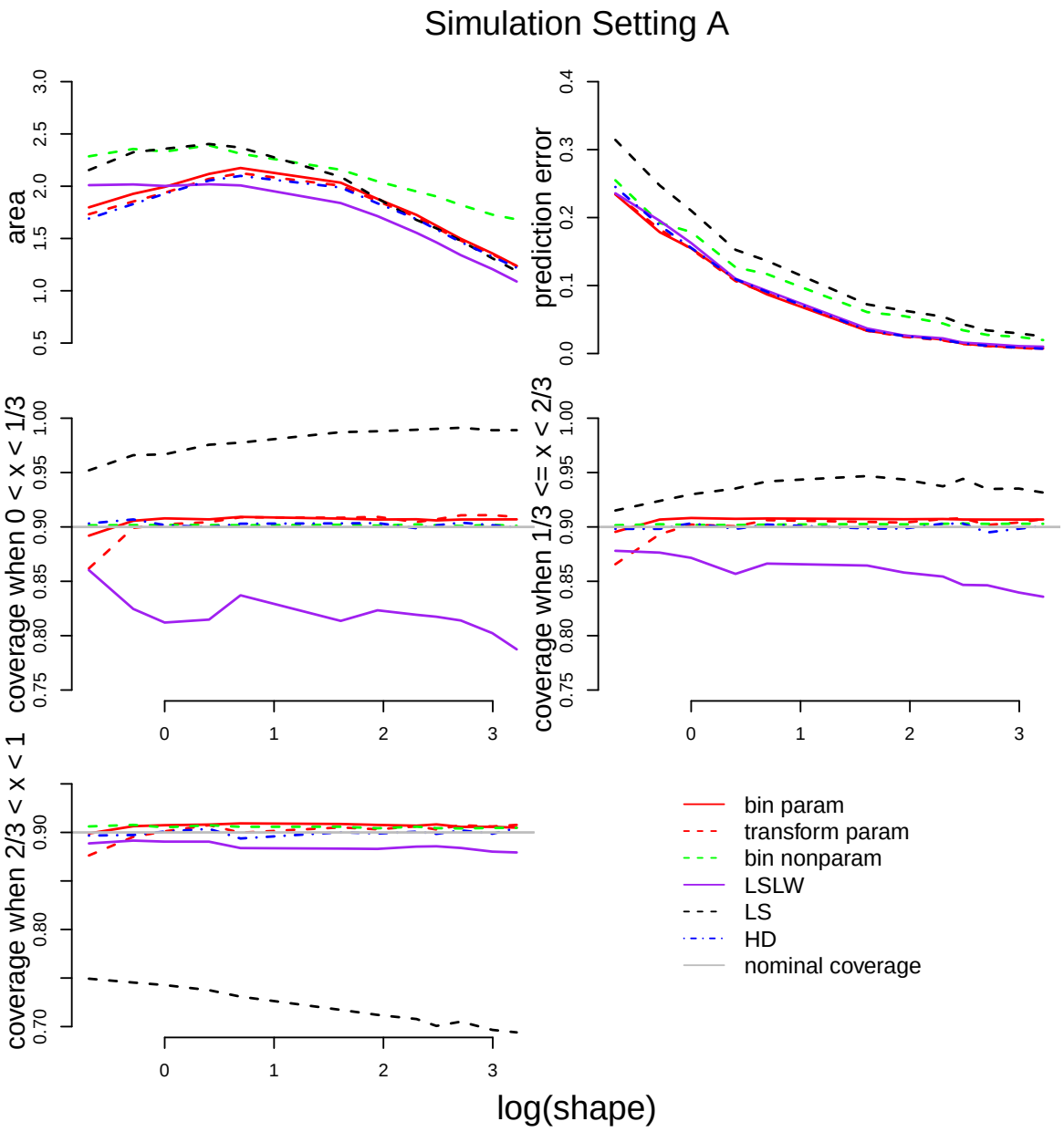


Figure 9: Diagnostic plots for prediction regions when the model is correctly specified, $n = 500$, and the number of bins is equal to 3.

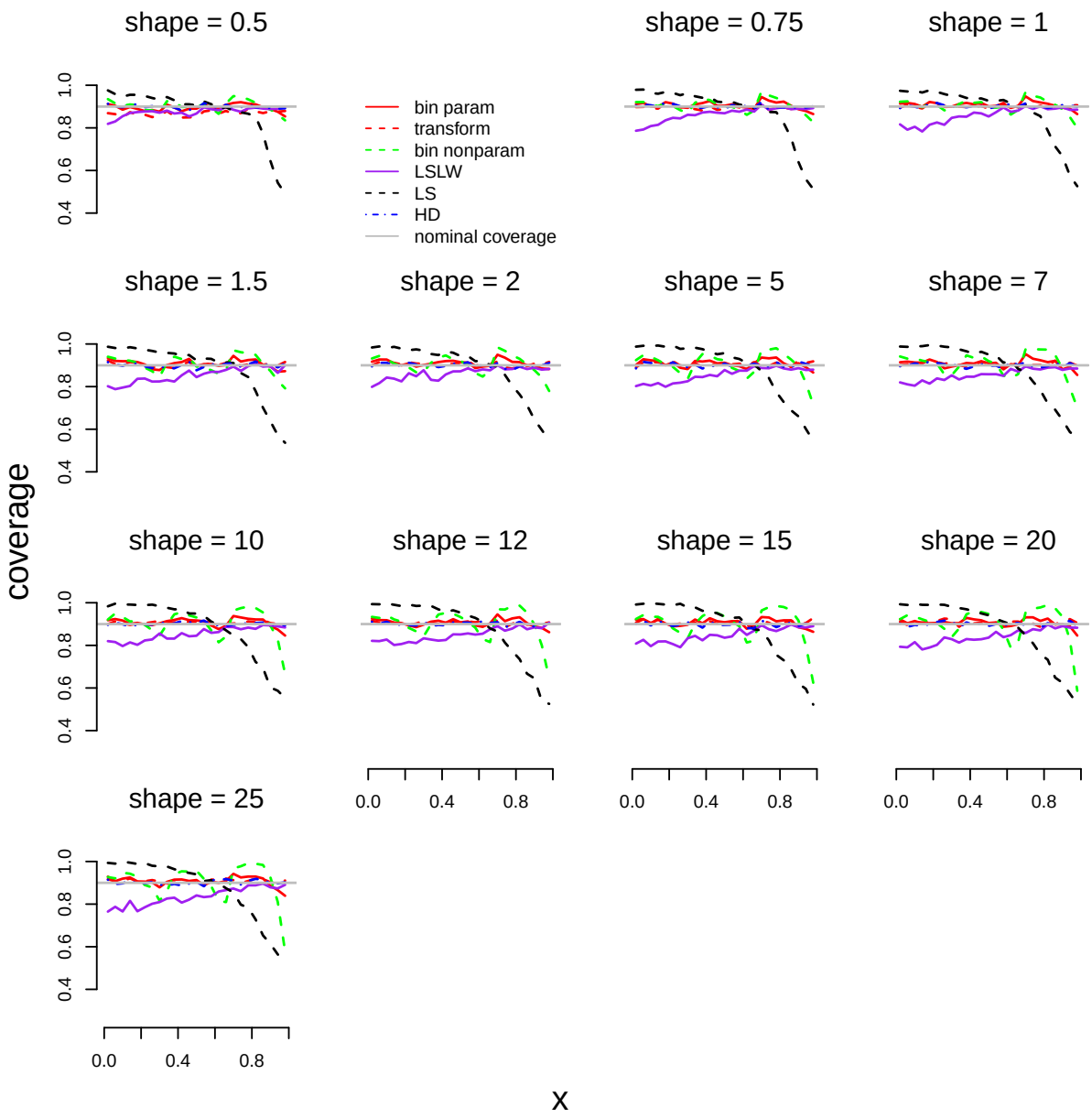


Figure 10: This figure compares the performance of the binned parametric, transformation, binned nonparametric, least squares, and least squares locally weighted conformal prediciton region and the highest density prediction region when $n = 500$ and the number of bins equals 3. The specific diagnostics 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.

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.

```
beta <- c(0.5, 1)
misspec_simulator <- function(beta, n = 150, shape = 0.5,
  bins = 2, alpha = 0.10, B = 2){

  p <- d <- length(beta) - 1

  models <- lapply(1:B, FUN = function(j){
    x <- matrix(runif(n*p), ncol = p)
    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 = "")
    fit <- glm(y ~ x1 + I(x1^2) + I(x1^3), family = "gaussian",
      data = dat, x = TRUE, y = TRUE)
    fit.true <- glm(y ~ x, family = "Gamma",
      data = dat, x = TRUE, y = TRUE)
    output = list(model = fit, model.true = fit.true, dat = dat)
    output
  })

  funs <- lm.funs(intercept = TRUE)
  train.fun <- funs$train.fun
  predict.fun <- funs$predict.fun
  conformal_regions <- lapply(models, FUN = function(obj){

    model <- obj$model
```

```
dat <- obj$dat
model$data <- dat
x <- model$x[, -1]
y <- model$y
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)

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$df
mad.train.fun <- function(x, y, out = NULL){
  smooth.spline(x[, 1], y, lambda = lambda,
  df = df, nknots = 10)
}
mad.predict.fun <- function(out, newx){
  predict(out, as.data.frame(newx))$y[, 1]
}
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)

fit.int = lm(y ~ x1 + I(x1^2) + I(x1^3), data = dat)
betaMLE <- coefficients(fit.int)
sdMLE <- summary(fit.int)$sigma
meanMLE <- as.numeric(cbind(1, x, x^2, x^3) %*% betaMLE)
HDCI <- do.call(rbind, lapply(1:n, function(j){
  hdi(qnorm, 1 - alpha, mean = meanMLE, sd = sdMLE)
}))

betaMLE.true <- coefficients(model.true)
shapeMLE.true <- as.numeric(gamma.shape(model.true)[1])
rateMLE.true <- cbind(1, x) %*% betaMLE.true * shapeMLE.true
HDCI.true <- do.call(rbind, lapply(1:n, function(j){
  hdi(qgamma, 1 - alpha, shape = shapeMLE.true, rate = rateMLE.true[j, 1])
}))

conf <- conformal.glm(model, nonparametric = TRUE, method = "both",
```

```
    bins = bins, cores = 7)
  parabinCI <- conf$paraconfbin
  transformCI <- conf$transformconf
  nonparabinCI <- conf$nonparaconfbin

  out = list(parabinCI = parabinCI, transformCI = transformCI,
    nonparabinCI = nonparabinCI, LSLWCI = LSLWCI, LSCI = LSCI,
    HDCI = HDCI, HDCI.true = HDCI.true)
  out
})

diagnostics_regions <- lapply(1:B, FUN = function(j){

  obj <- conformal_regions[[j]]
  dat <- as.data.frame(models[[j]]$dat)
  y <- as.numeric(dat[, 1])

  diagnostics <- lapply(obj, FUN = function(region){

    output <- NULL
    if(class(region) == "matrix"){
      marginal.coverage <- local.coverage(region = region,
        data = dat, d = p, bins = 1, at.data = "TRUE")
      local.coverage <- local.coverage(region = region,
        data = dat, d = p, bins = bins, at.data = "TRUE")
      local.inx.coverage <- local.coverage(region = region,
        data = dat, d = p, bins = 25, at.data = "TRUE")
      output <- c(marginal.coverage, local.coverage,
        local.inx.coverage, mean(apply(region, 1, diff)),
        absolute.error(y = y, region = region))
    }
    else{
      marginal.coverage <- local.coverage(region = region,
        nonparametric = "TRUE", data = dat, d = p, bins = 1,
        at.data = "TRUE")
      local.coverage <- local.coverage(region = region,
        nonparametric = "TRUE", data = dat, d = p, bins = bins,
        at.data = "TRUE")
      local.inx.coverage <- local.coverage(region = region,
        nonparametric = "TRUE", data = dat, d = p, bins = 25,
        at.data = "TRUE")
      output <- c(marginal.coverage, local.coverage,
        local.inx.coverage, area.nonparametric(region),
        absolute.error.nonparametric(data = dat,
          region = region))
    }
    output
```

```
    })
    do.call(rbind, diagnostics)

  })

  diagnostics_regions
}
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 0.75.

```
set.seed(13)
n <- 150
bins <- 2
B <- 200
system.time(out.misspec.150.2.0.75 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 0.75, bins = bins, B = B)))

## Error in chol.default(crossprod(x) + lambda[j] * diag(v)):  the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.887 0.004 0.892

diagnostics.misspec.150.2.0.75 <- do.call(rbind, lapply(split(out.misspec.150.2.0.7
  f = as.factor(rownames(out.misspec.150.2.0.75))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))

## Error in split(out.misspec.150.2.0.75, f = as.factor(rownames(out.misspec.150.2.
object 'out.misspec.150.2.0.75' not found
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 1.

```
set.seed(13)
system.time(out.misspec.150.2.1 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 1, bins = bins, B = B)))

## Error in chol.default(crossprod(x) + lambda[j] * diag(v)):  the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.798 0 0.798

diagnostics.misspec.150.2.1 <- do.call(rbind, lapply(split(out.misspec.150.2.1,
  f = as.factor(rownames(out.misspec.150.2.1))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))

## Error in split(out.misspec.150.2.1, f = as.factor(rownames(out.misspec.150.2.1))
object 'out.misspec.150.2.1' not found
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 2.

```
set.seed(13)
system.time(out.misspec.150.2.2 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 2, bins = bins, B = B)))

## Error in chol.default(crossprod(x) + lambda[j] * diag(v)):  the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.698 0 0.698

diagnostics.misspec.150.2.2 <- do.call(rbind, lapply(split(out.misspec.150.2.2,
  f = as.factor(rownames(out.misspec.150.2.2))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))

## Error in split(out.misspec.150.2.2, f = as.factor(rownames(out.misspec.150.2.2))
object 'out.misspec.150.2.2' not found
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 5.

```
set.seed(13)
system.time(out.misspec.150.2.5 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 5, bins = bins, B = B)))

## Error in chol.default(crossprod(x) + lambda[j] * diag(v)):  the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.667 0 0.667

diagnostics.misspec.150.2.5 <- do.call(rbind, lapply(split(out.misspec.150.2.5,
  f = as.factor(rownames(out.misspec.150.2.5))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))

## Error in split(out.misspec.150.2.5, f = as.factor(rownames(out.misspec.150.2.5))
object 'out.misspec.150.2.5' not found
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 7.

```
set.seed(13)
system.time(out.misspec.150.2.7 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 7, bins = bins, B = B)))

## Error in chol.default(crossprod(x) + lambda[j] * diag(v)):  the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.764 0 0.763

diagnostics.misspec.150.2.7 <- do.call(rbind, lapply(split(out.misspec.150.2.7,
  f = as.factor(rownames(out.misspec.150.2.7))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))

## Error in split(out.misspec.150.2.7, f = as.factor(rownames(out.misspec.150.2.7))
object 'out.misspec.150.2.7' not found
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 10.

```
set.seed(13)
system.time(out.misspec.150.2.10 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 10, bins = bins, B = B)))

## Error in chol.default(crossprod(x) + lambda[j] * diag(v)):  the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.692 0 0.692

diagnostics.misspec.150.2.10 <- do.call(rbind, lapply(split(out.misspec.150.2.10,
  f = as.factor(rownames(out.misspec.150.2.10))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))

## Error in split(out.misspec.150.2.10, f = as.factor(rownames(out.misspec.150.2.10
object 'out.misspec.150.2.10' not found
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 12.

```
set.seed(13)
system.time(out.misspec.150.2.12 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 12, bins = bins, B = B)))

## Error in chol.default(crossprod(x) + lambda[j] * diag(v)):  the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.671 0 0.671

diagnostics.misspec.150.2.12 <- do.call(rbind, lapply(split(out.misspec.150.2.12,
  f = as.factor(rownames(out.misspec.150.2.12))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))

## Error in split(out.misspec.150.2.12, f = as.factor(rownames(out.misspec.150.2.12
object 'out.misspec.150.2.12' not found
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 15.

```
set.seed(13)
system.time(out.misspec.150.2.15 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 15, bins = bins, B = B)))

## Error in chol.default(crossprod(x) + lambda[j] * diag(v)):  the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.65 0 0.65

diagnostics.misspec.150.2.15 <- do.call(rbind, lapply(split(out.misspec.150.2.15,
  f = as.factor(rownames(out.misspec.150.2.15))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))
```

```
## Error in split(out.misspec.150.2.15, f = as.factor(rownames(out.misspec.150.2.15
object 'out.misspec.150.2.15' not found
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 20.

```
set.seed(13)
system.time(out.misspec.150.2.20 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 20, bins = bins, B = B)))

## Error in chol.default(crossprod(x) + lambda[j] * diag(v)): the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.665 0 0.665

diagnostics.misspec.150.2.20 <- do.call(rbind, lapply(split(out.misspec.150.2.20,
  f = as.factor(rownames(out.misspec.150.2.20))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))

## Error in split(out.misspec.150.2.20, f = as.factor(rownames(out.misspec.150.2.20
object 'out.misspec.150.2.20' not found
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 25.

```
set.seed(13)
system.time(out.misspec.150.2.25 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 25, bins = bins, B = B)))

## Error in chol.default(crossprod(x) + lambda[j] * diag(v)): the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.654 0 0.654

diagnostics.misspec.150.2.25 <- do.call(rbind, lapply(split(out.misspec.150.2.25,
  f = as.factor(rownames(out.misspec.150.2.25))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))

## Error in split(out.misspec.150.2.25, f = as.factor(rownames(out.misspec.150.2.25
object 'out.misspec.150.2.25' not found
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 30.

```
set.seed(13)
system.time(out.misspec.150.2.30 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 30, bins = bins, B = B)))

## Error in chol.default(crossprod(x) + lambda[j] * diag(v)): the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.656 0 0.656
```

```
diagnostics.misspec.150.2.30 <- do.call(rbind, lapply(split(out.misspec.150.2.30,
  f = as.factor(rownames(out.misspec.150.2.30))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))

## Error in split(out.misspec.150.2.30, f = as.factor(rownames(out.misspec.150.2.30
object 'out.misspec.150.2.30' not found
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 40.

```
set.seed(13)
system.time(out.misspec.150.2.40 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 40, bins = bins, B = B)))

## Error in chol.default(crossprod(x) + lambda[j] * diag(v)): the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.67 0 0.67

diagnostics.misspec.150.2.40 <- do.call(rbind, lapply(split(out.misspec.150.2.40,
  f = as.factor(rownames(out.misspec.150.2.40))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))

## Error in split(out.misspec.150.2.40, f = as.factor(rownames(out.misspec.150.2.40
object 'out.misspec.150.2.40' not found
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 50.

```
set.seed(13)
system.time(out.misspec.150.2.50 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 50, bins = bins, B = B)))

## Error in chol.default(crossprod(x) + lambda[j] * diag(v)): the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.716 0 0.716

diagnostics.misspec.150.2.50 <- do.call(rbind, lapply(split(out.misspec.150.2.50,
  f = as.factor(rownames(out.misspec.150.2.50))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))

## Error in split(out.misspec.150.2.50, f = as.factor(rownames(out.misspec.150.2.50
object 'out.misspec.150.2.50' not found
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 60.

```
set.seed(13)
system.time(out.misspec.150.2.60 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 60, bins = bins, B = B)))
```

```
## Error in chol.default(crossprod(x) + lambda[j] * diag(v)):  the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.717 0 0.717
```

```
diagnostics.misspec.150.2.60 <- do.call(rbind, lapply(split(out.misspec.150.2.60,
  f = as.factor(rownames(out.misspec.150.2.60))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))
```

```
## Error in split(out.misspec.150.2.60, f = as.factor(rownames(out.misspec.150.2.60
object 'out.misspec.150.2.60' not found
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 70.

```
set.seed(13)
system.time(out.misspec.150.2.70 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 70, bins = bins, B = B)))
```

```
## Error in chol.default(crossprod(x) + lambda[j] * diag(v)):  the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.702 0 0.702
```

```
diagnostics.misspec.150.2.70 <- do.call(rbind, lapply(split(out.misspec.150.2.70,
  f = as.factor(rownames(out.misspec.150.2.70))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))
```

```
## Error in split(out.misspec.150.2.70, f = as.factor(rownames(out.misspec.150.2.70
object 'out.misspec.150.2.70' not found
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 80.

```
set.seed(13)
system.time(out.misspec.150.2.80 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 80, bins = bins, B = B)))
```

```
## Error in chol.default(crossprod(x) + lambda[j] * diag(v)):  the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.712 0 0.712
```

```
diagnostics.misspec.150.2.80 <- do.call(rbind, lapply(split(out.misspec.150.2.80,
  f = as.factor(rownames(out.misspec.150.2.80))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))
```

```
## Error in split(out.misspec.150.2.80, f = as.factor(rownames(out.misspec.150.2.80
object 'out.misspec.150.2.80' not found
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 90.

```
set.seed(13)
system.time(out.misspec.150.2.90 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 40, bins = bins, B = B)))

## Error in chol.default(crossprod(x) + lambda[j] * diag(v)):  the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.72 0 0.719

diagnostics.misspec.150.2.90 <- do.call(rbind, lapply(split(out.misspec.150.2.90,
  f = as.factor(rownames(out.misspec.150.2.90))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))

## Error in split(out.misspec.150.2.90, f = as.factor(rownames(out.misspec.150.2.90
object 'out.misspec.150.2.90' not found
```

The following performs our Monte Carlo simulation of $B = 200$ iterations when $n = 150$ and shape = 100.

```
set.seed(13)
system.time(out.misspec.150.2.100 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 100, bins = bins, B = B)))

## Error in chol.default(crossprod(x) + lambda[j] * diag(v)):  the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.673 0 0.672

diagnostics.misspec.150.2.100 <- do.call(rbind, lapply(split(out.misspec.150.2.100,
  f = as.factor(rownames(out.misspec.150.2.100))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))

## Error in split(out.misspec.150.2.100, f = as.factor(rownames(out.misspec.150.2.1
object 'out.misspec.150.2.100' not found


## Error in eval(parse(text = paste("diagnostics.misspec.150.2", j, sep =
".")):  object 'diagnostics.misspec.150.2.0.75' not found
```

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

```
set.seed(13)
n <- 250
B <- 50
bins <- 3
system.time(out.misspec.250.3.5 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 5, bins = bins, B = B)))

## Error in chol.default(crossprod(x) + lambda[j] * diag(v)):  the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.199 0 0.2
```

```
diagnostics.misspec.250.3.5 <- do.call(rbind, lapply(split(out.misspec.250.3.5,
  f = as.factor(rownames(out.misspec.250.3.5))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))
```

```
## Error in split(out.misspec.250.3.5, f = as.factor(rownames(out.misspec.250.3.5))
object 'out.misspec.250.3.5' not found
```

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

```
set.seed(13)
system.time(out.misspec.250.3.20 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 20, bins = bins, B = B)))
```

```
## Error in chol.default(crossprod(x) + lambda[j] * diag(v)): the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.178 0 0.179
```

```
diagnostics.misspec.250.3.20 <- do.call(rbind, lapply(split(out.misspec.250.3.20,
  f = as.factor(rownames(out.misspec.250.3.20))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))
```

```
## Error in split(out.misspec.250.3.20, f = as.factor(rownames(out.misspec.250.3.20
object 'out.misspec.250.3.20' not found
```

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

```
set.seed(13)
system.time(out.misspec.250.3.40 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 40, bins = bins, B = B)))
```

```
## Error in chol.default(crossprod(x) + lambda[j] * diag(v)): the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.184 0 0.184
```

```
diagnostics.misspec.250.3.40 <- do.call(rbind, lapply(split(out.misspec.250.3.40,
  f = as.factor(rownames(out.misspec.250.3.40))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))
```

```
## Error in split(out.misspec.250.3.40, f = as.factor(rownames(out.misspec.250.3.40
object 'out.misspec.250.3.40' not found
```

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

```
set.seed(13)
system.time(out.misspec.250.3.60 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 60, bins = bins, B = B)))
```

```
## Error in chol.default(crossprod(x) + lambda[j] * diag(v)):  the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.183 0 0.183
```

```
diagnostics.misspec.250.3.60 <- do.call(rbind, lapply(split(out.misspec.250.3.60,
  f = as.factor(rownames(out.misspec.250.3.60))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))
```

```
## Error in split(out.misspec.250.3.60, f = as.factor(rownames(out.misspec.250.3.60
object 'out.misspec.250.3.60' not found
```

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

```
set.seed(13)
system.time(out.misspec.250.3.80 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 80, bins = bins, B = B)))
```

```
## Error in chol.default(crossprod(x) + lambda[j] * diag(v)):  the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.188 0 0.189
```

```
diagnostics.misspec.250.3.80 <- do.call(rbind, lapply(split(out.misspec.250.3.80,
  f = as.factor(rownames(out.misspec.250.3.80))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))
```

```
## Error in split(out.misspec.250.3.80, f = as.factor(rownames(out.misspec.250.3.80
object 'out.misspec.250.3.80' not found
```

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

```
set.seed(13)
system.time(out.misspec.250.3.100 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 100, bins = bins, B = B)))
```

```
## Error in chol.default(crossprod(x) + lambda[j] * diag(v)):  the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.199 0 0.2
```

```
diagnostics.misspec.250.3.100 <- do.call(rbind, lapply(split(out.misspec.250.3.100,
  f = as.factor(rownames(out.misspec.250.3.100))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))
```

```
## Error in split(out.misspec.250.3.100, f = as.factor(rownames(out.misspec.250.3.1
object 'out.misspec.250.3.100' not found
```

```
## Error in eval(parse(text = paste("diagnostics.misspec.250.3", j, sep =
".")[1])):  object 'diagnostics.misspec.250.3.5' not found
```

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

```
set.seed(13)
n <- 500
B <- 50
bins <- 3
system.time(out.misspec.500.3.5 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 5, bins = bins, B = B)))

## Error in chol.default(crossprod(x) + lambda[j] * diag(v)):  the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.262 0 0.261

diagnostics.misspec.500.3.5 <- do.call(rbind, lapply(split(out.misspec.500.3.5,
  f = as.factor(rownames(out.misspec.500.3.5))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))

## Error in split(out.misspec.500.3.5, f = as.factor(rownames(out.misspec.500.3.5))
object 'out.misspec.500.3.5' not found
```

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

```
set.seed(13)
system.time(out.misspec.500.3.20 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 20, bins = bins, B = B)))

## Error in chol.default(crossprod(x) + lambda[j] * diag(v)):  the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.219 0 0.218

diagnostics.misspec.500.3.20 <- do.call(rbind, lapply(split(out.misspec.500.3.20,
  f = as.factor(rownames(out.misspec.500.3.20))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))

## Error in split(out.misspec.500.3.20, f = as.factor(rownames(out.misspec.500.3.20
object 'out.misspec.500.3.20' not found
```

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

```
set.seed(13)
system.time(out.misspec.500.3.40 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 40, bins = bins, B = B)))

## Error in chol.default(crossprod(x) + lambda[j] * diag(v)):  the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.235 0 0.235
```

```
diagnostics.misspec.500.3.40 <- do.call(rbind, lapply(split(out.misspec.500.3.40,
  f = as.factor(rownames(out.misspec.500.3.40))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))
```

```
## Error in split(out.misspec.500.3.40, f = as.factor(rownames(out.misspec.500.3.40
object 'out.misspec.500.3.40' not found
```

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

```
set.seed(13)
system.time(out.misspec.500.3.60 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 60, bins = bins, B = B)))
```

```
## Error in chol.default(crossprod(x) + lambda[j] * diag(v)): the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.303 0 0.302
```

```
diagnostics.misspec.500.3.60 <- do.call(rbind, lapply(split(out.misspec.500.3.60,
  f = as.factor(rownames(out.misspec.500.3.60))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))
```

```
## Error in split(out.misspec.500.3.60, f = as.factor(rownames(out.misspec.500.3.60
object 'out.misspec.500.3.60' not found
```

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

```
set.seed(13)
system.time(out.misspec.500.3.80 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 80, bins = bins, B = B)))
```

```
## Error in chol.default(crossprod(x) + lambda[j] * diag(v)): the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.225 0 0.226
```

```
diagnostics.misspec.500.3.80 <- do.call(rbind, lapply(split(out.misspec.500.3.80,
  f = as.factor(rownames(out.misspec.500.3.80))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))
```

```
## Error in split(out.misspec.500.3.80, f = as.factor(rownames(out.misspec.500.3.80
object 'out.misspec.500.3.80' not found
```

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

```
set.seed(13)
system.time(out.misspec.500.3.100 <- do.call(rbind,
  misspec_simulator(beta = beta, n = n, shape = 100, bins = bins, B = B)))
```

```
## Error in chol.default(crossprod(x) + lambda[j] * diag(v)):  the leading
minor of order 5 is not positive definite
## Timing stopped at:  0.218 0 0.218
```

```
diagnostics.misspec.500.3.100 <- do.call(rbind, lapply(split(out.misspec.500.3.100,
  f = as.factor(rownames(out.misspec.500.3.100))),
  FUN = function(xx) colMeans(matrix(xx, nrow = B), na.rm = TRUE)))
```

```
## Error in split(out.misspec.500.3.100, f = as.factor(rownames(out.misspec.500.3.1
object 'out.misspec.500.3.100' not found
```

```
## Error in eval(parse(text = paste("diagnostics.misspec.500.3", j, sep =
".")):  object 'diagnostics.misspec.500.3.5' not found
```

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 **??**. 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 **??**. 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 **??**.

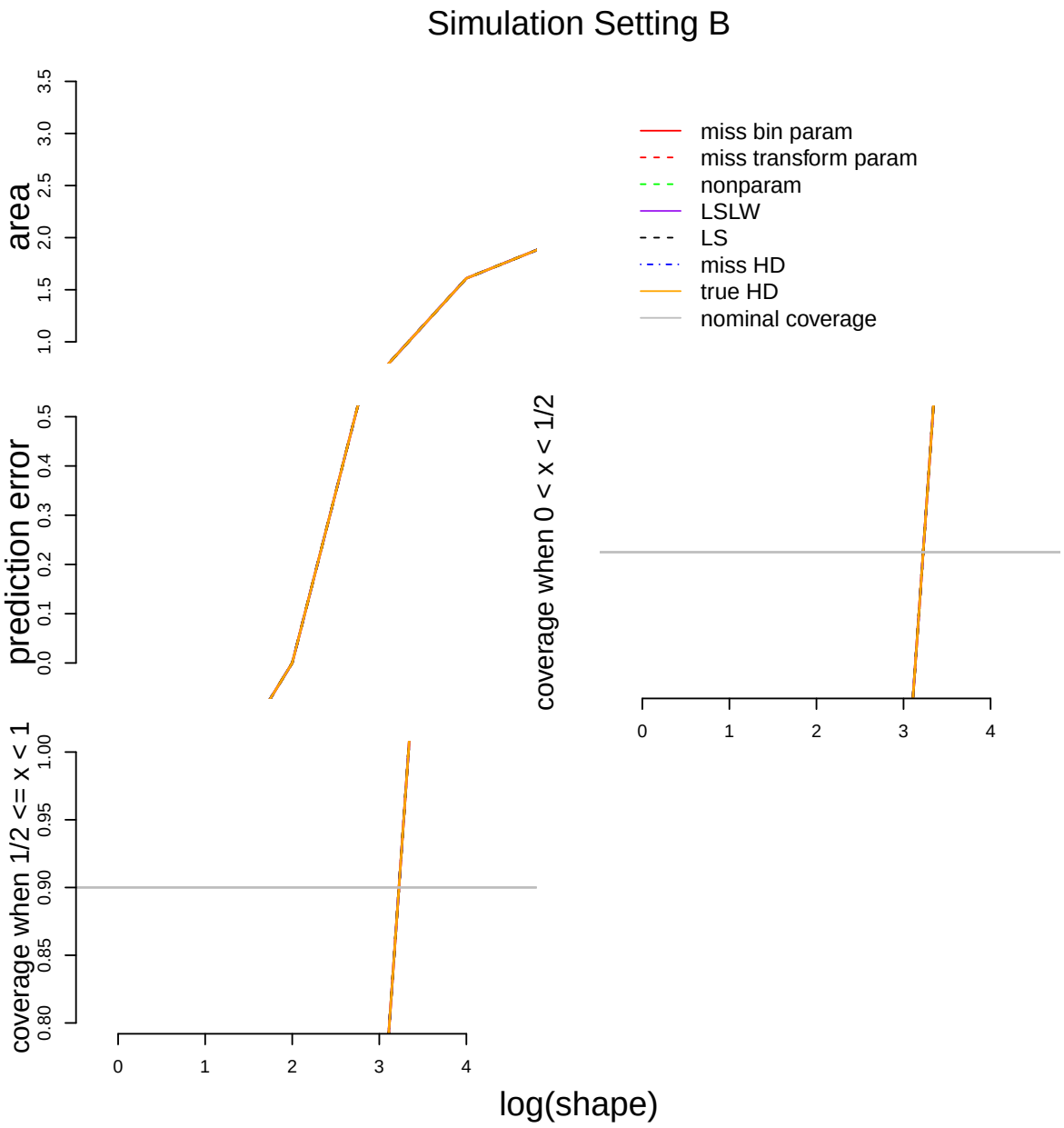


Figure 11: This figure compares the performance of the misspecified binned parametric, misspecified transformation, binned nonparametric, least squares, and least squares locally weighted conformal prediciton regions and the misspecified and correctly specified highest density prediction regions when $n = 150$ and the number of bins equals 2. The specific diagnostics used to compare these prediciton regions is the area, prediction error, and the coverage probability with respect to binning across shape parameter values. The average of 200 Monte Carlo samples at each shape parameter value in these simulation settings form the lines that are depicted in this figure.

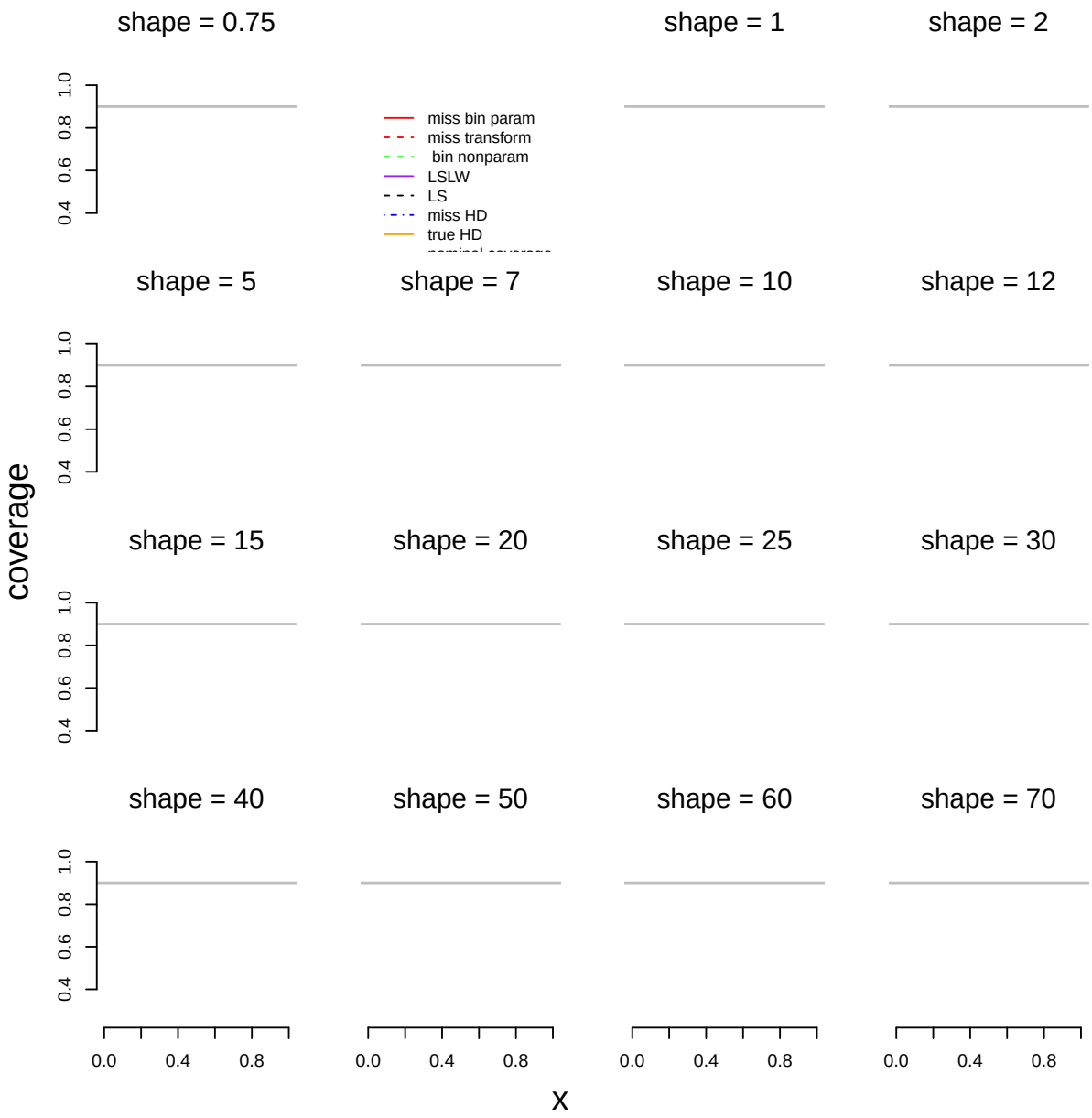


Figure 12: Plot of the estimated coverage probabilities of prediction regions across x and shape parameter values when the model is misspecified, $n = 150$, and the number of bins is equal to 2.

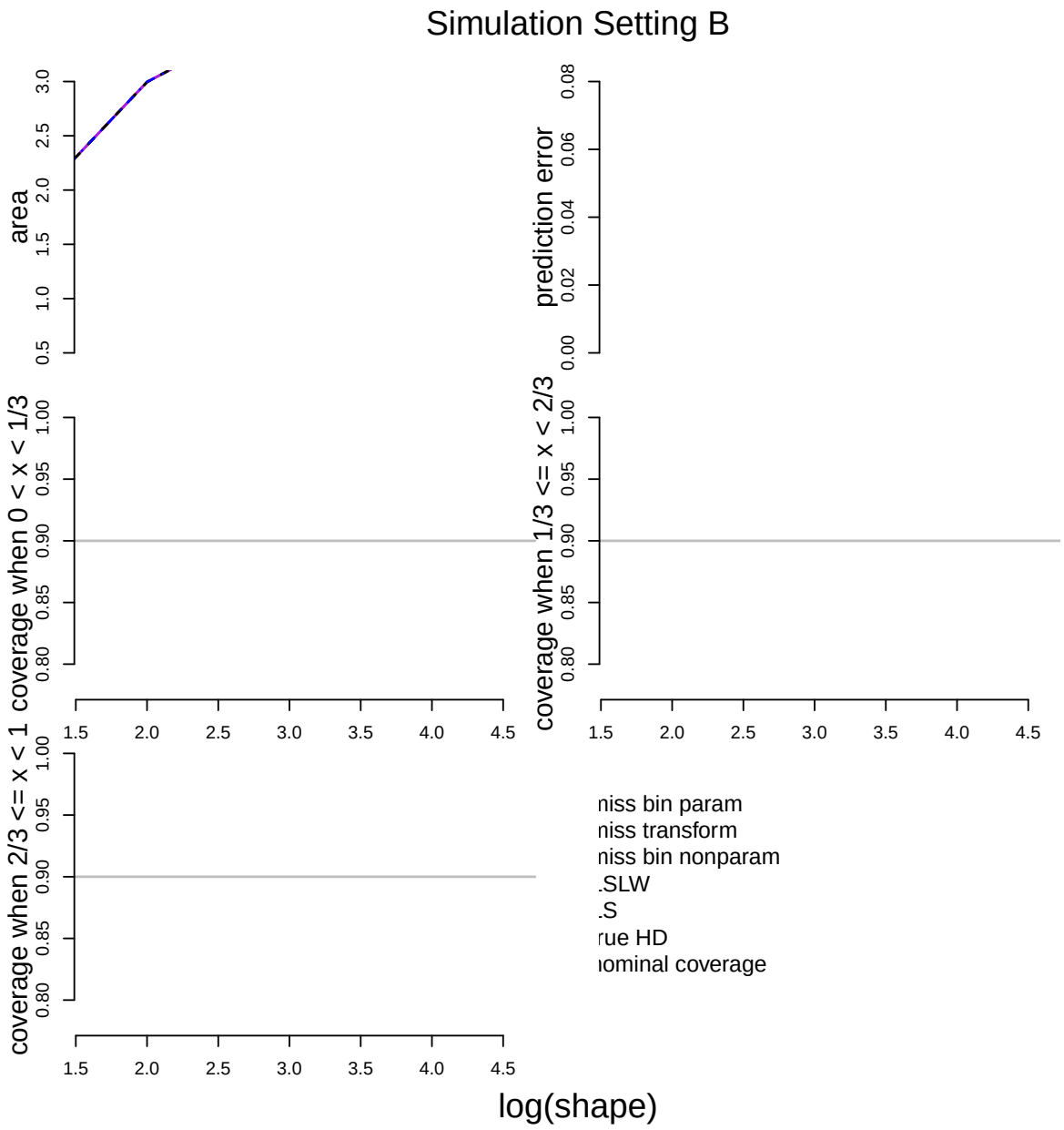


Figure 13: This figure compares the performance of the misspecified binned parametric, misspecified transformation, binned nonparametric, least squares, and least squares locally weighted conformal prediciton region and the correctly specified highest density prediction regions when $n = 250$ and the 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.

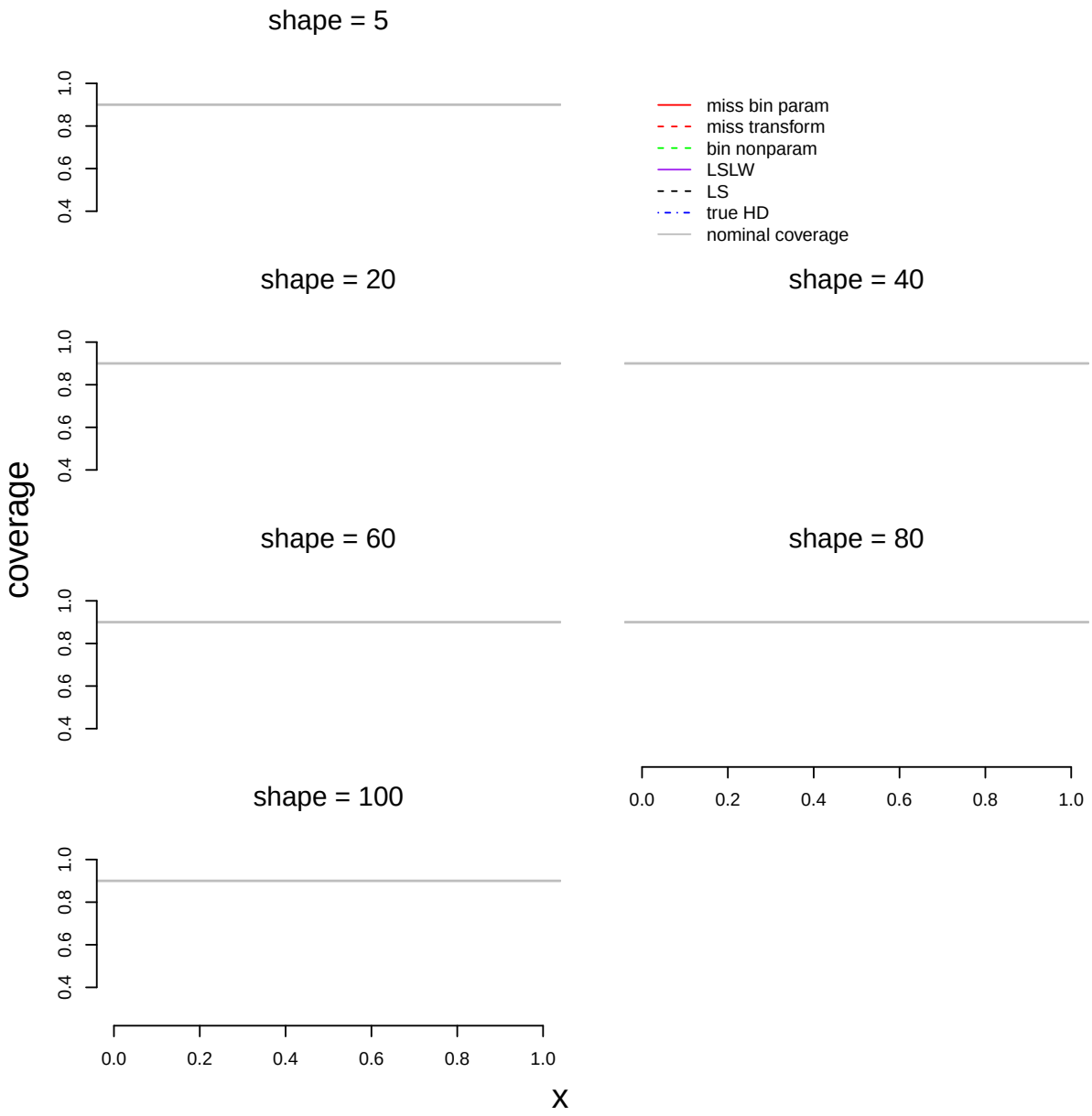


Figure 14: Plot of the estimated coverage probabilities of prediction regions across x and shape parameter values when the model is misspecified, $n = 250$, and the number of bins is equal to 3.

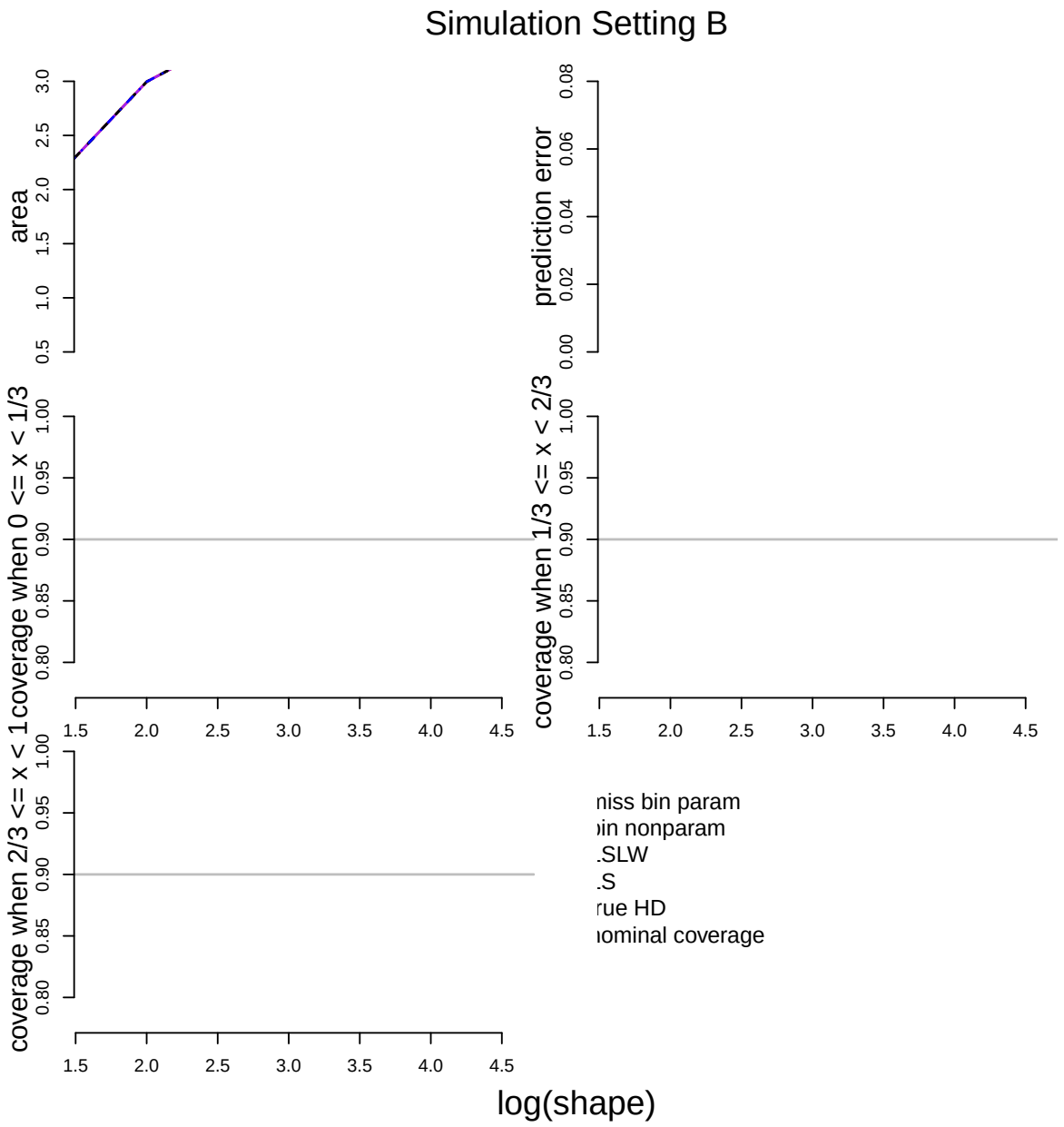


Figure 15: This figure compares the performance of the binned misspecified parametric, misspecified transformation, binned nonparametric, least squares, and least squares locally weighted conformal prediciton regions and the correctly specified highest density prediction region when $n = 500$ and the 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.

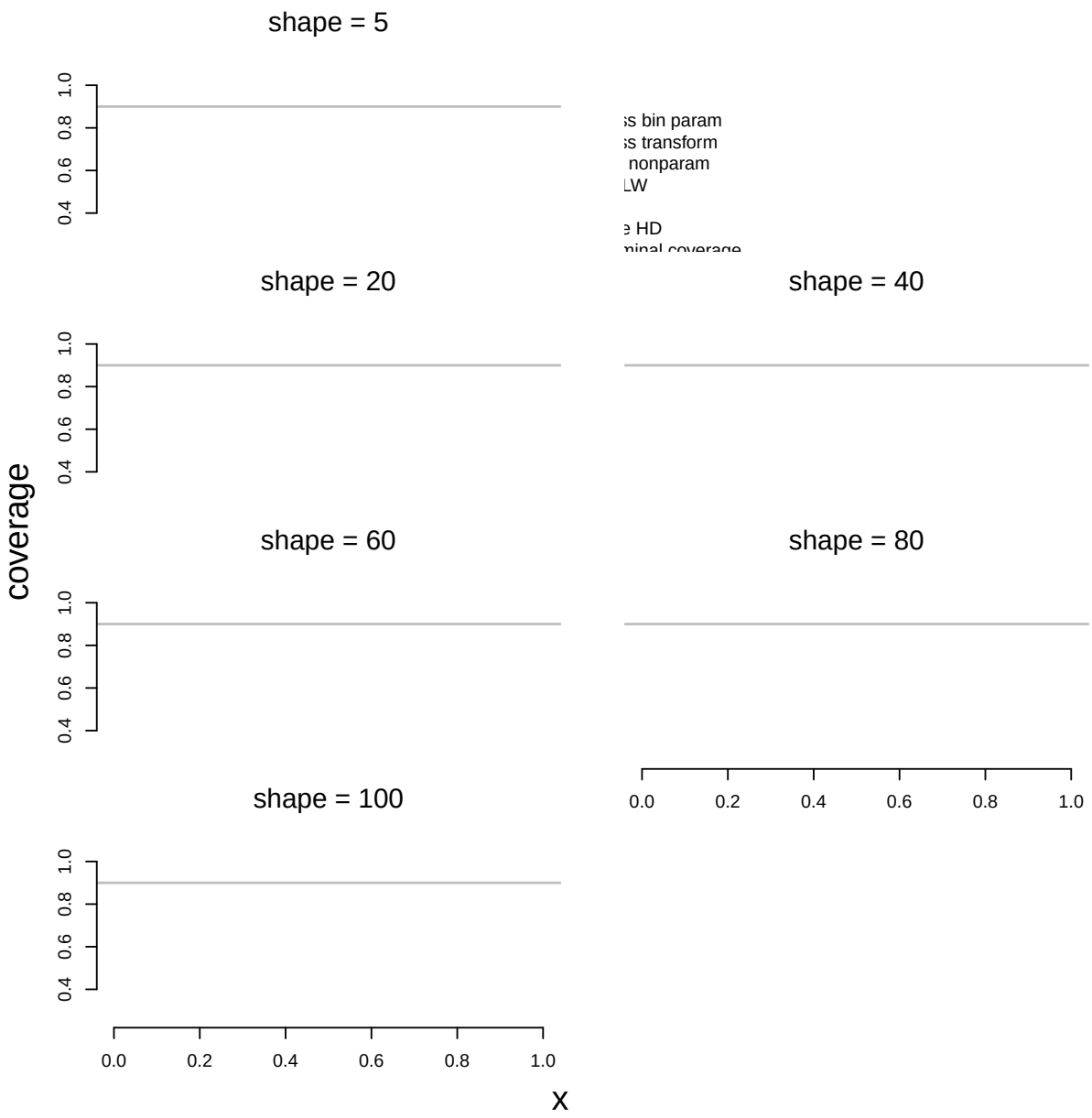


Figure 16: Plot of the estimated coverage probabilities of prediction regions across x and shape parameter values when the model is misspecified, $n = 500$, and the number of bins is equal to 3.

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 $X \sim U(0, 1)$.

```
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

}
```

8 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.