STAT 8830 HW5

1) Local linear regression (LOESS)

Here we define a function to perform LOESS regression on 1-dimensional data, accepting a bandwidth parameter and the number of (evenly spaced) points to predict over.

This notebook is ported more or less directly from `hw5_python.ipynb` since I developed solutions in Python first. Again much inspiration is taken here from this and this.

First define some utility functions, for:

- point normalization (rescaling)
- computing band indices from bandwidth
- computing band index weights

Jitter is no longer necessary because it's built into R.

```
# load dependencies
library(scales)
library(ggplot2)

# set the random seed
set.seed(42)

rescale_value = function(value, r_min, r_max, t_min=0, t_max=1) {
    (value - r_min) / (r_max - r_min) * (t_max - t_min) + t_min
}

get_band = function(distances, width) {
    min_i = which.min(distances)
    total = length(distances)
    band = c(min_i)

    # if the closest neighbor is at either the left or right bound, start the window there
    if (min_i == 1) return(seq(1, width))
    if (min_i == total) return(seq(total - width, total))

    # otherwise build it up iteratively
    while(length(band) < width) {
        min_i = head(band, n=1)
        max_i = tail(band, n=1)
        if (min_i == 1) band = c(band, c(max_i + 1))
        else if (max_i == total) band = c(c(min_i), band)
        else if (distances[min_i - 1] < distances[max_i + 1]) band = c(c(min_i - 1), band)
        else band = c(band, c(max_i + 1))
    }
```

```
    # return the band
    c(band)
}

get_weights = function(distances, band) {
    normed_ds = distances[band] / max(distances[band])
    bandwidth = length(band)
    mu = 0
    std = bandwidth * sd(normed_ds)
    x = seq(mu - std, mu + std, length.out=length(normed_ds))
    weights = dnorm(x, mu, std)
    weights
}
```

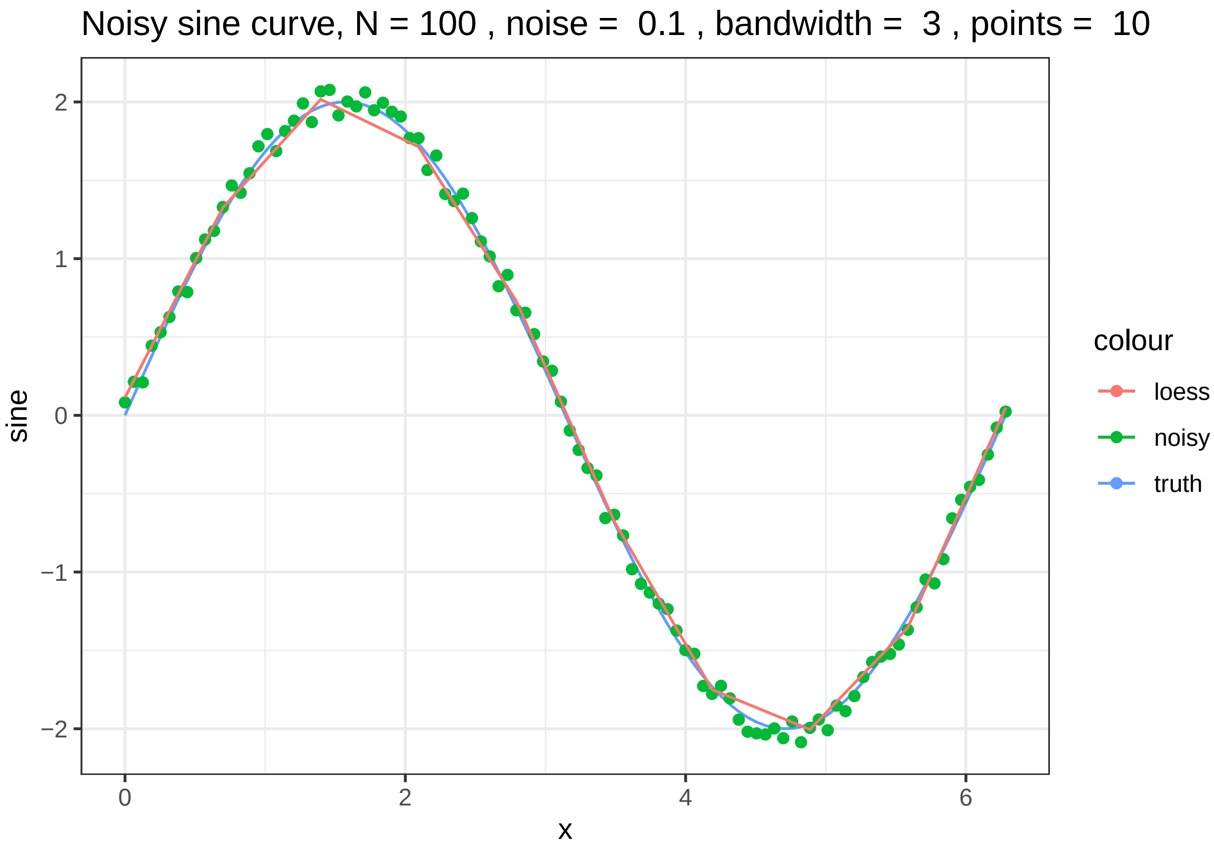
Define the function to perform LOESS regression.

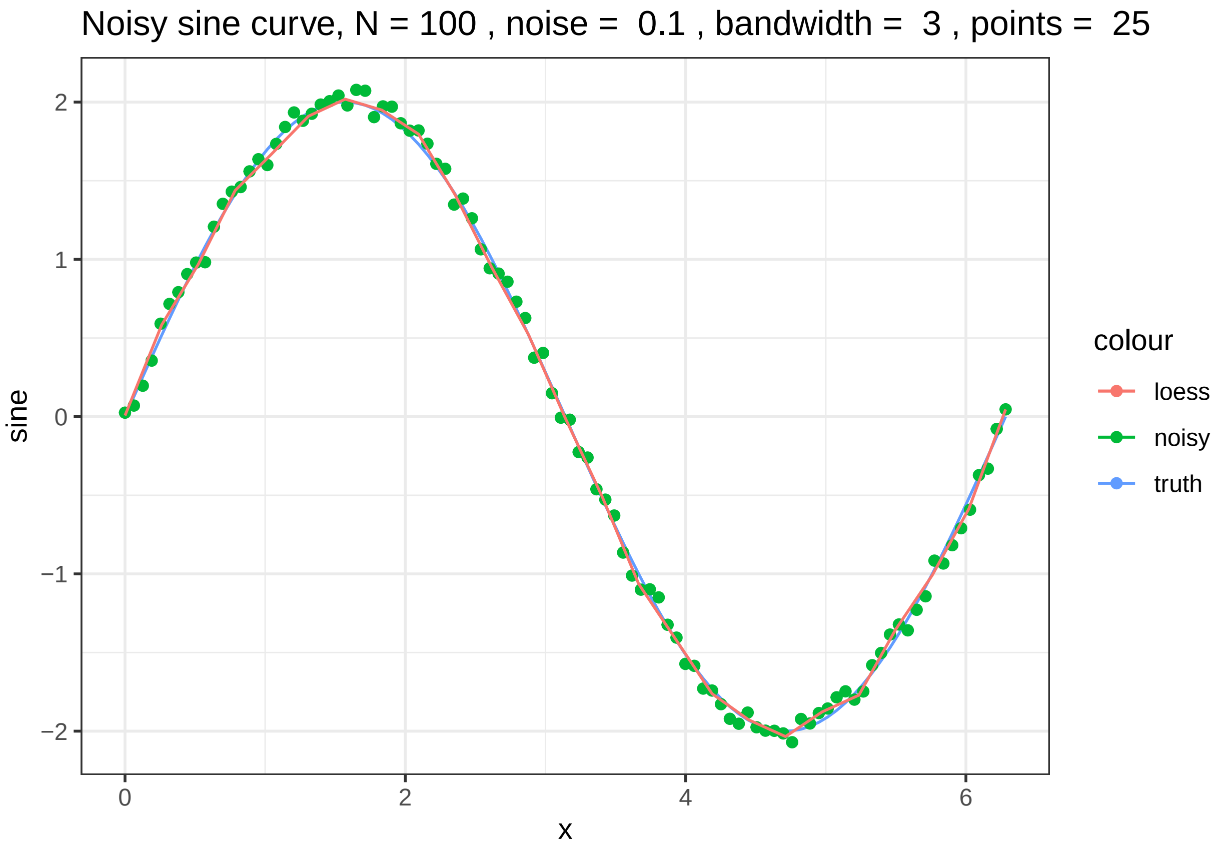
```
myloess = function(dat, bandwidth, num_pts) {
    # extract the predictor and response and compute their respective min and max values
    xs = dat$x
    x_min = min(xs)
    x_max = max(xs)

    # extract the response
    ys = dat$y
    y_min = min(ys)
    y_max = max(ys)

    # scale the predictor and response to unit interval
    normed_xs = rescale(xs, c(0, 1), c(x_min, x_max))
    normed_ys = rescale(ys, c(0, 1), c(y_min, y_max))

    # predict n evenly spaced points over the output range
    output_xs = seq(x_min, x_max, length.out=num_pts)
    output_ys = c()

    for (x in output_xs) {
        # rescale the value to the unit interval
        normed_x = rescale_value(x, x_min, x_max)

        # compute distances from this value to all other values
        distances = abs(normed_xs - normed_x)

        # get the indices of the band around the current points
        band_is = get_band(distances, bandwidth)

        # get weights for elements of the band
        weights = get_weights(distances, band_is)

        # get the subsets of the normed predictor and response corresponding to the band
        band_xs = normed_xs[band_is]
        band_ys = normed_ys[band_is]

        data = data.frame(x=band_xs, y=band_ys)
```

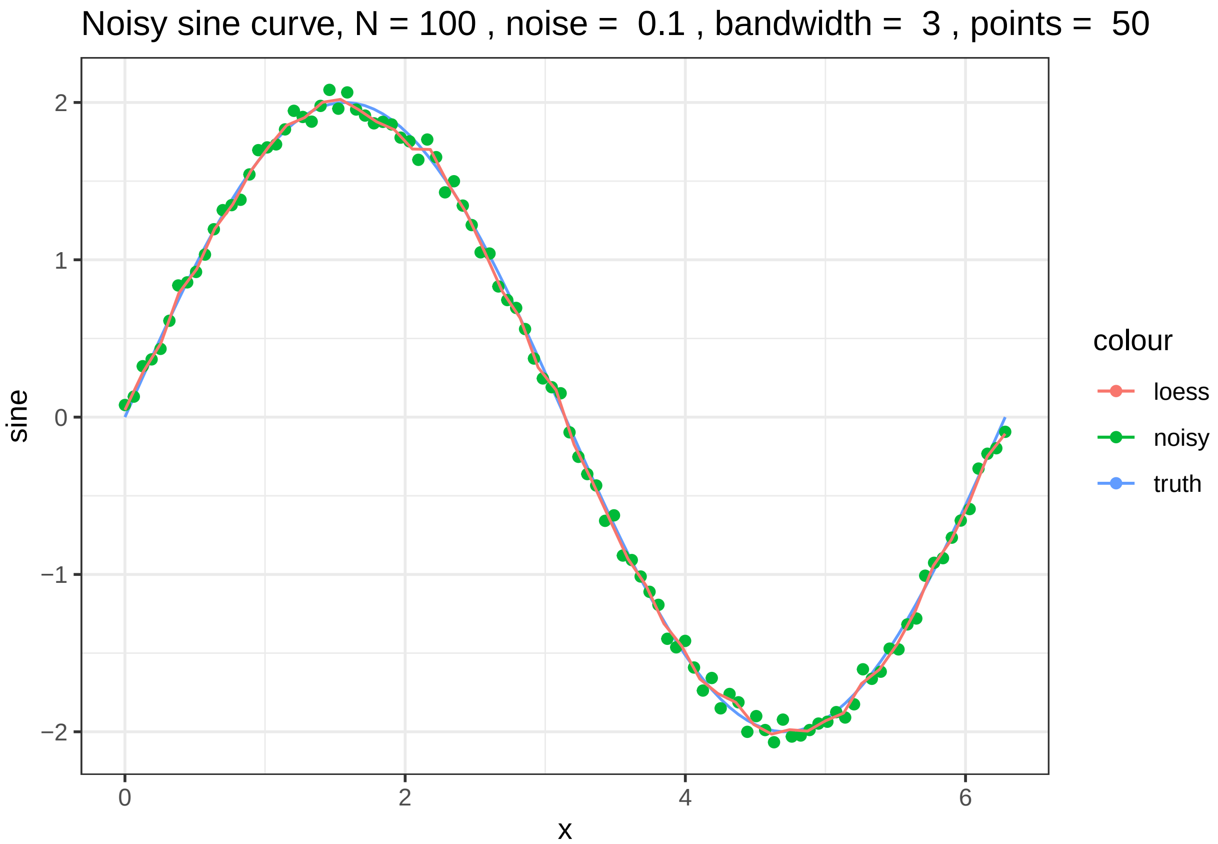
```r
        # fit weighted least squares model to the band
        wls = lm(y ~ x, data=data, weight=weights)
        wls_y = predict(wls, data.frame(x=normed_x))

        # save predicted value
        output_ys = c(output_ys, c(wls_y[[1]]))
    }

    # rescale the response to the original interval
    output_ys = rescale(output_ys, c(y_min, y_max), c(0, 1))

    data.frame(X=output_xs, Y=output_ys)
}
```

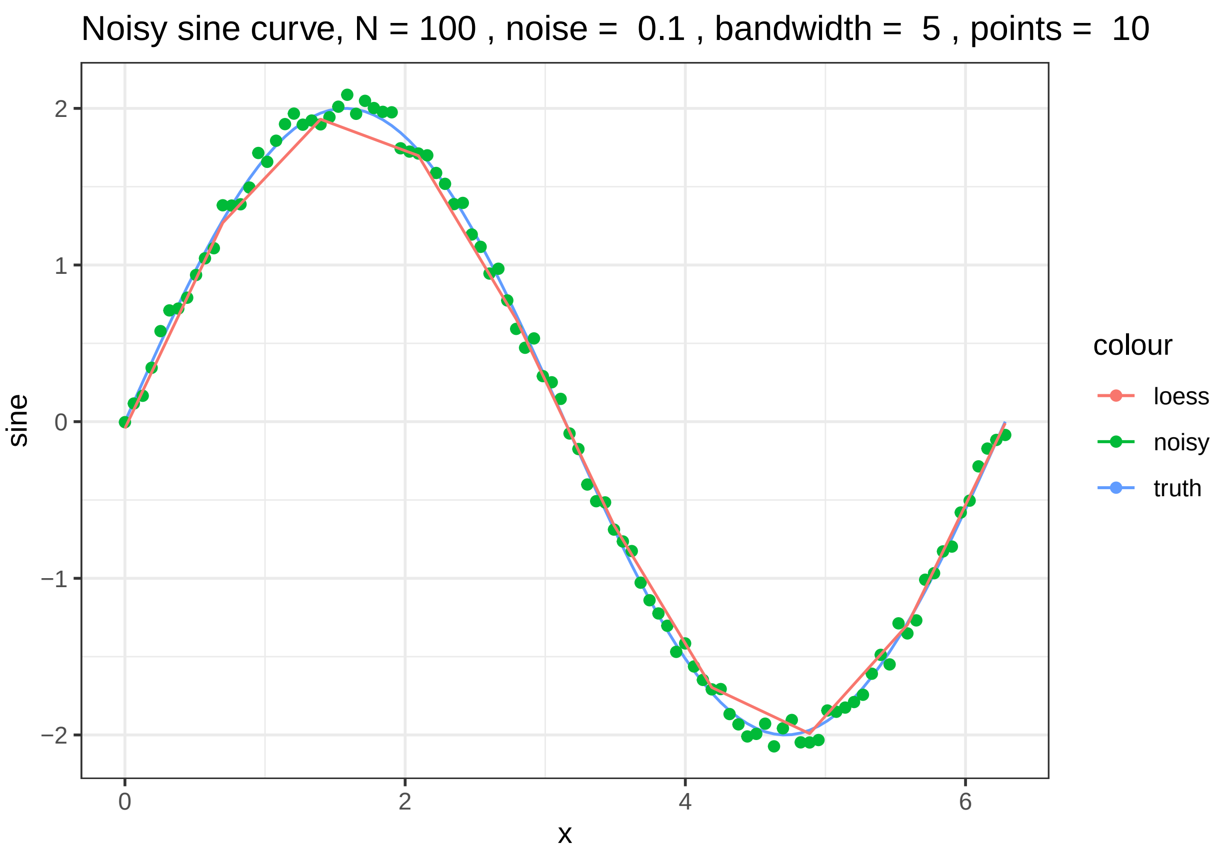
Generate a sine wave as a simulated dataset to attempt to fit, then test the LOESS function with various levels of noise and bandwidth values.

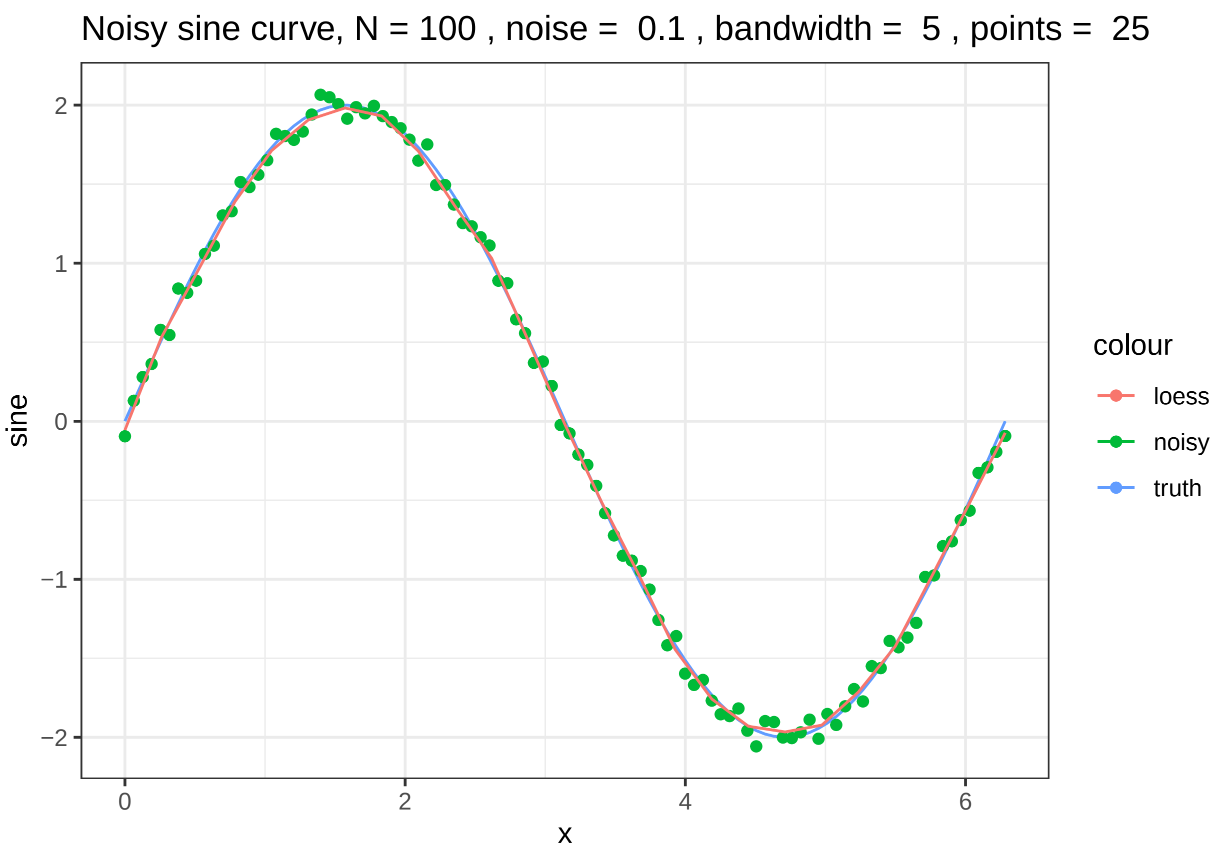
```r
n = 100
x = seq(0, 2*pi, length.out=n)
sine = 2*sin(x)
noises = c(.1, .5, 1)
lambdas = c(3, 5, 10, 20)
points = c(as.integer(n / 10), as.integer(n / 4), as.integer(n / 2))

# try the LOESS function with each combination of noise, bandwidth, and point coun
for (noise in noises) {
  for (l in lambdas) {
    for (pts in points) {
      # simulate noisy sampling
      y = sapply(sine, function(i) jitter(i, amount=noise))
      frame = data.frame(x=x, y=y)

      # apply LOESS
      result = myloess(frame, l, pts)

      # create plot
      p = ggplot(frame) +
        geom_line(aes(x=x, y=sine, color='truth')) +
        geom_point(aes(x=x, y=y, color='noisy')) +
        geom_line(data=result, aes(x=X, y=Y, color='loess')) +
        theme_bw() +
        labs(title=paste(
          "Noisy sine curve, N =", n,
          ", noise = ", noise,
          ", bandwidth = ", l,
          ", points = ", pts))
      print(p)
    }
  }
}
```

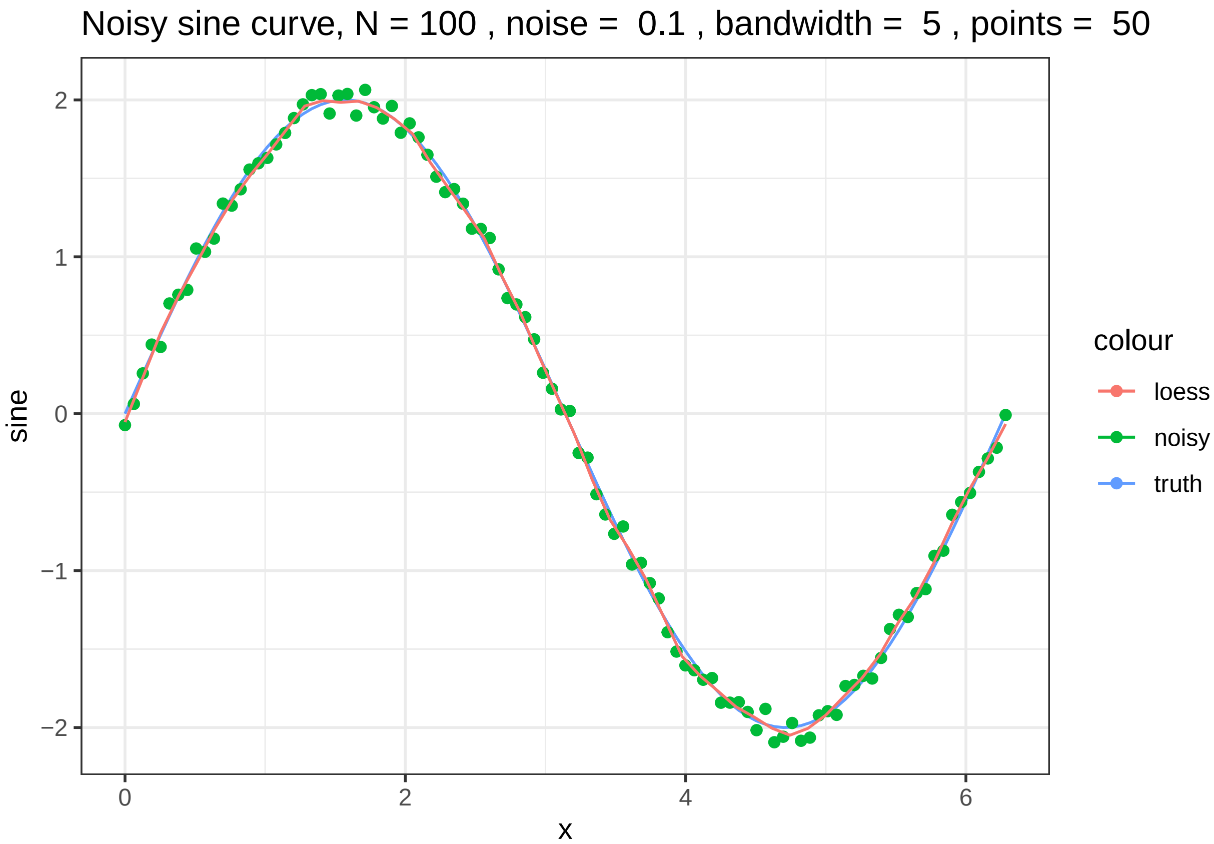

Noisy sine curve, N = 100 , noise = 0.1 , bandwidth = 3 , points = 10
sine
x
2
1
0
−1
−2
0
2
4
6
colour
loess
noisy
truth


Noisy sine curve, N = 100 , noise = 0.1 , bandwidth = 3 , points = 25
sine
x
2
1
0
−1
−2
0
2
4
6
colour
loess
noisy
truth

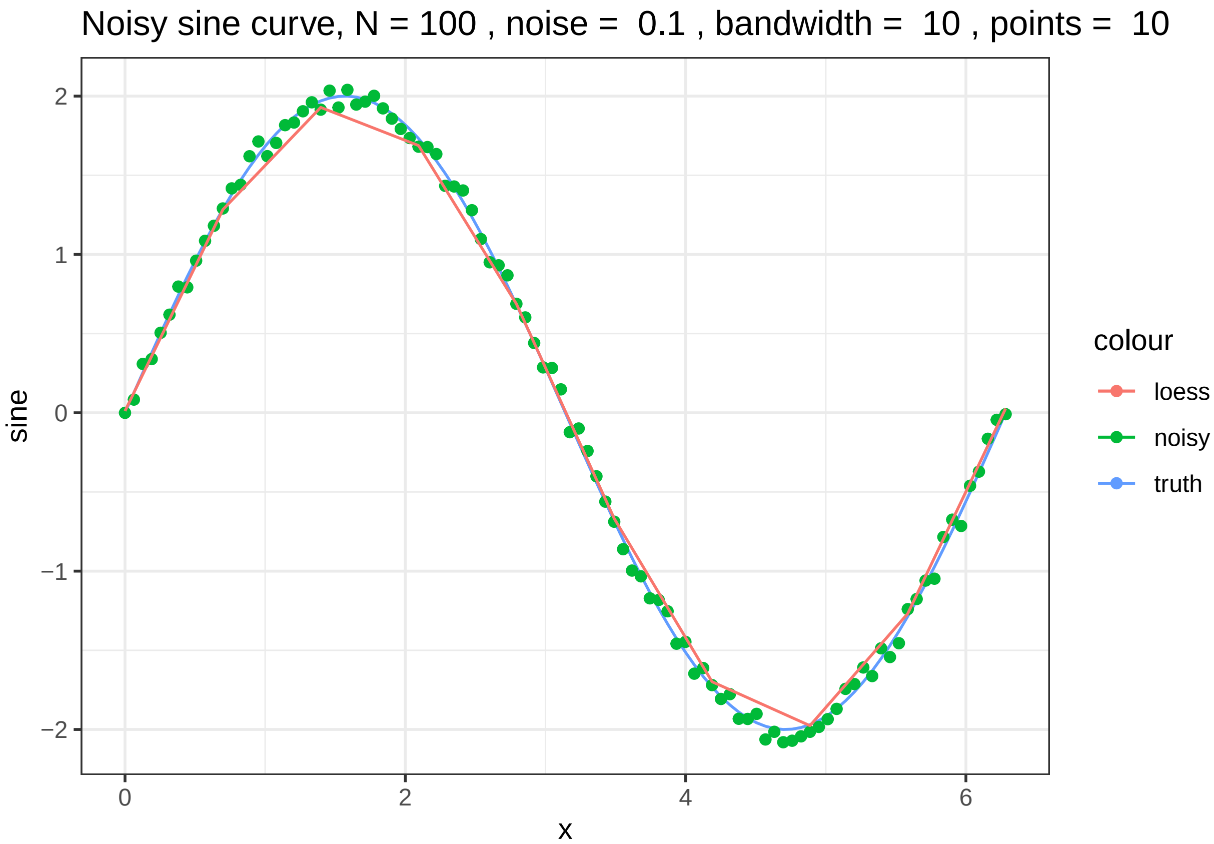

Noisy sine curve, N = 100 , noise = 0.1 , bandwidth = 3 , points = 50
sine
x
colour
loess
noisy
truth

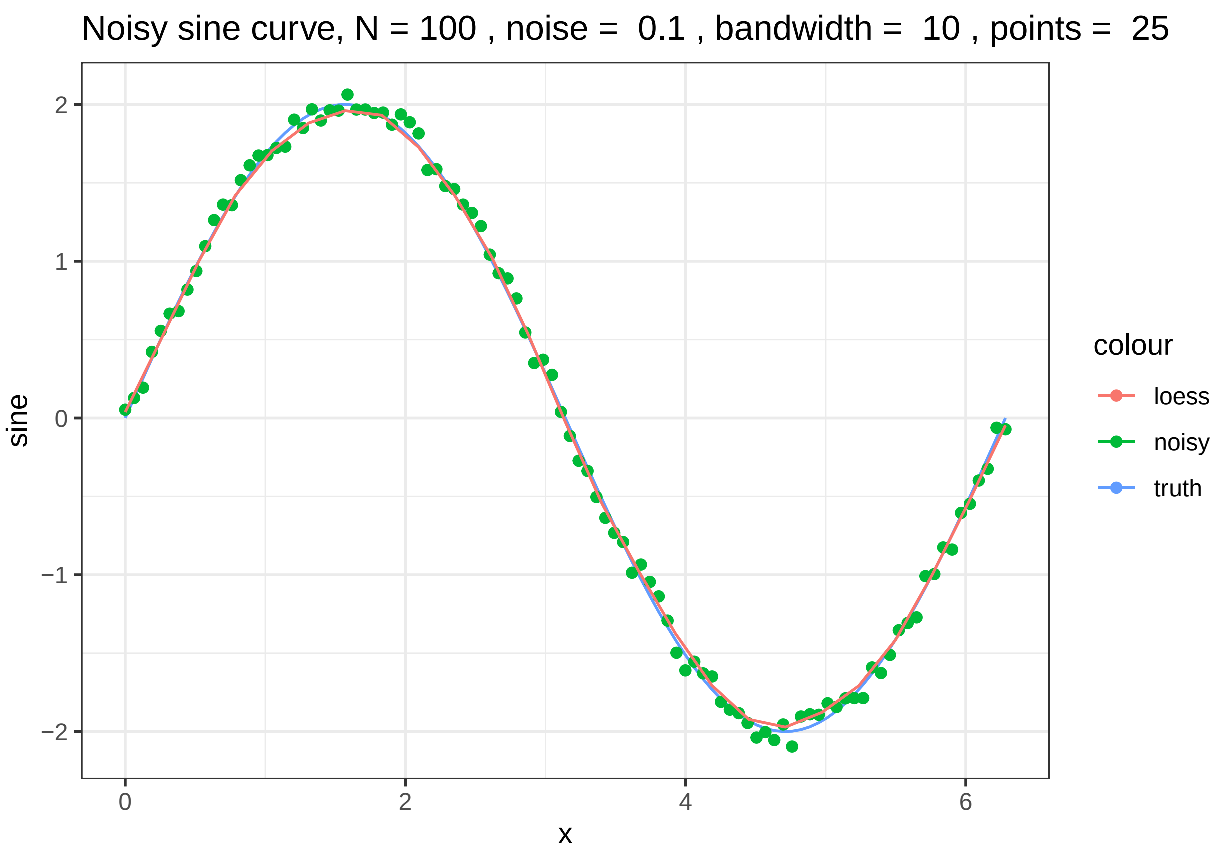

Noisy sine curve, N = 100 , noise = 0.1 , bandwidth = 5 , points = 10
sine
x
colour
loess
noisy
truth
2
1
0
−1
−2
0
2
4
6


Noisy sine curve, N = 100 , noise = 0.1 , bandwidth = 5 , points = 25
sine
x
colour
loess
noisy
truth

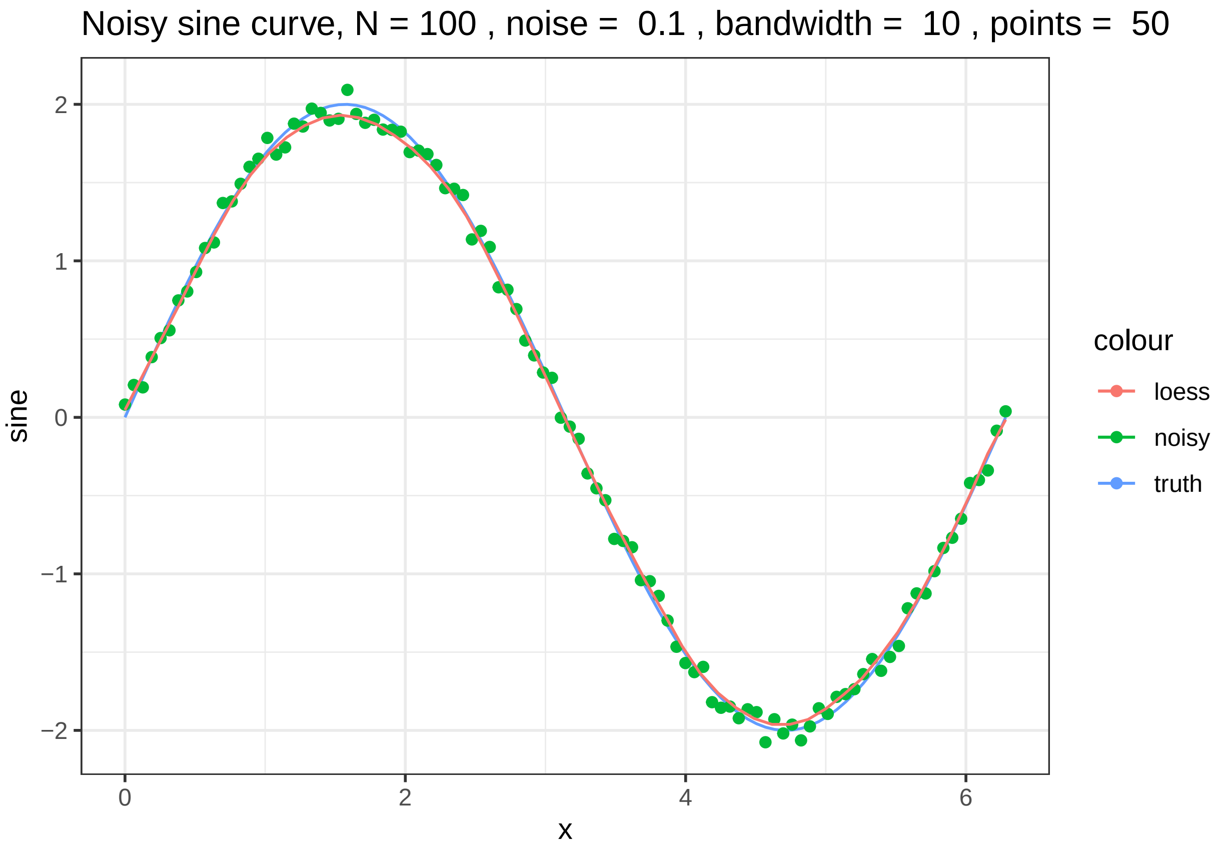


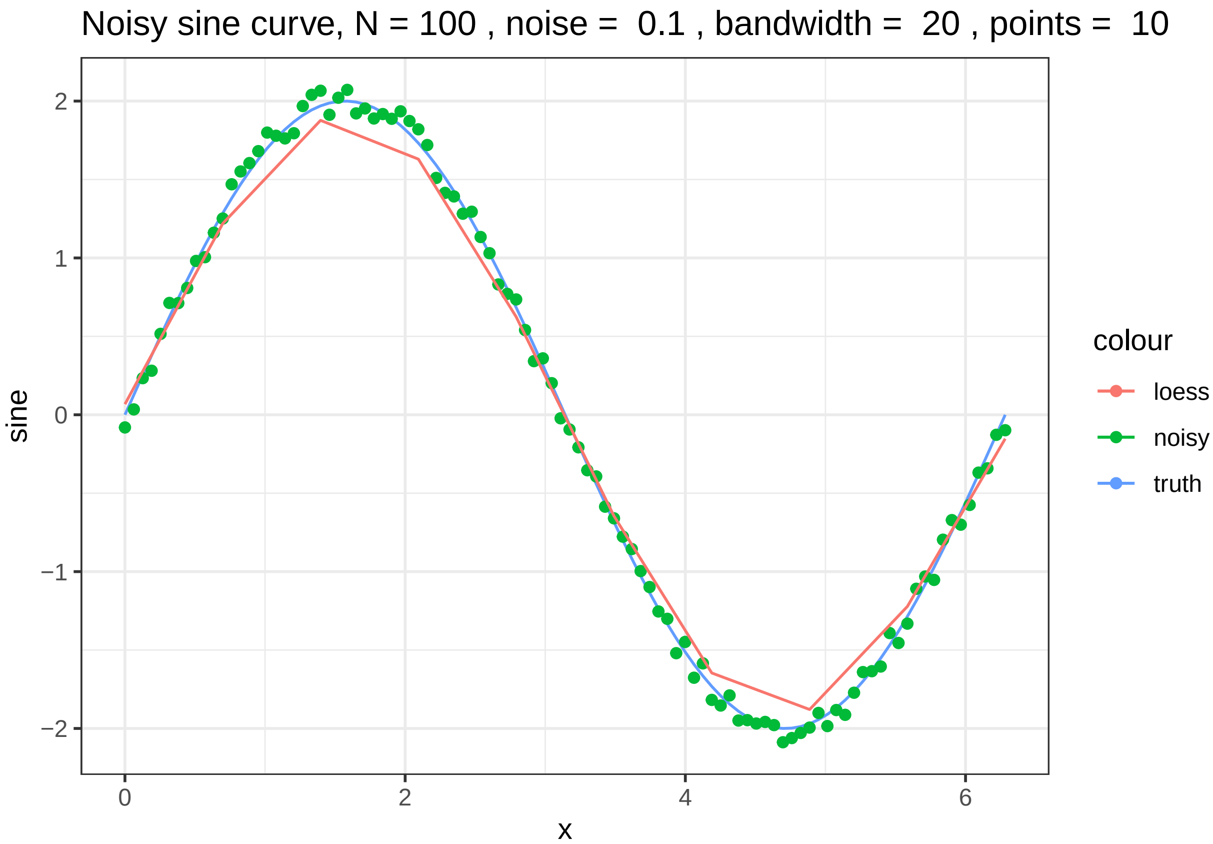
Noisy sine curve, N = 100 , noise = 0.1 , bandwidth = 5 , points = 50
sine
2
1
0
−1
−2
0
2
4
6
x
colour
loess
noisy
truth

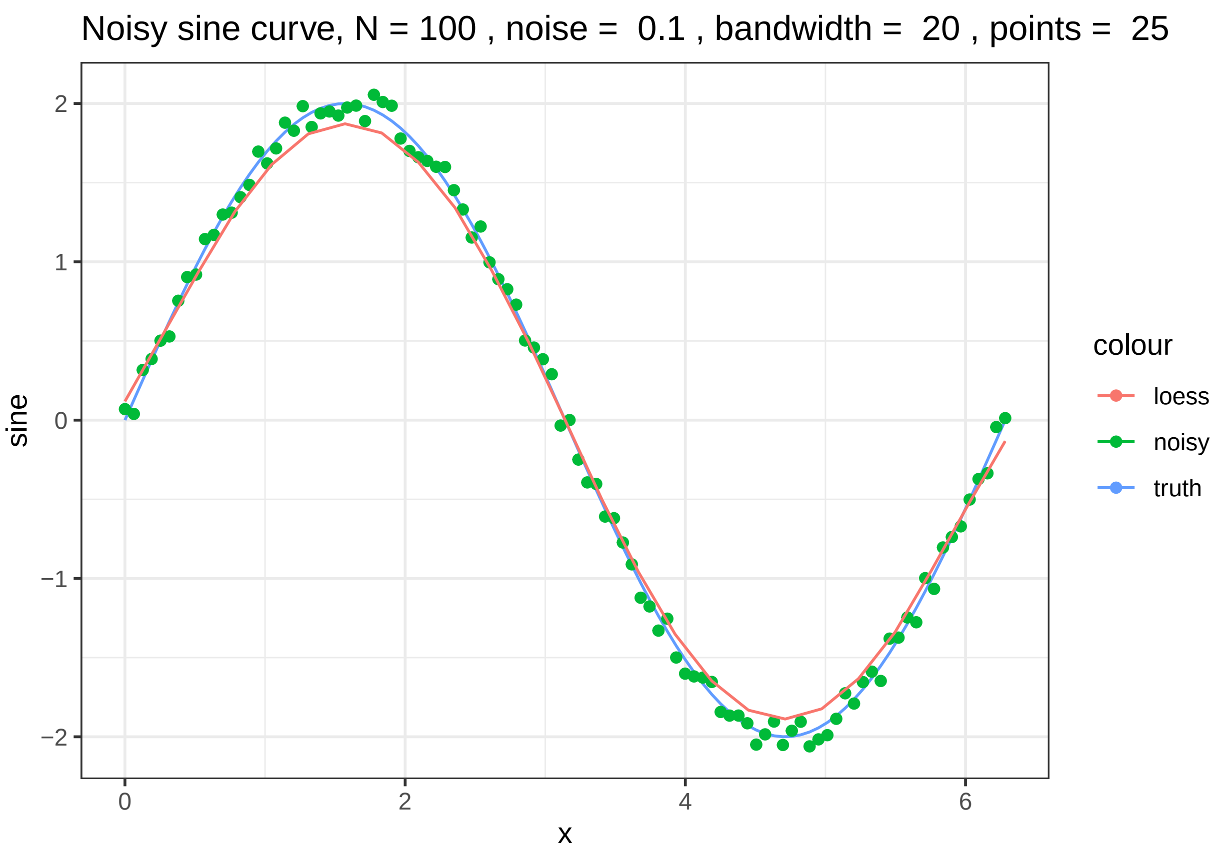

Noisy sine curve, N = 100 , noise = 0.1 , bandwidth = 10 , points = 10
sine
x
2
1
0
−1
−2
0
2
4
6
colour
loess
noisy
truth

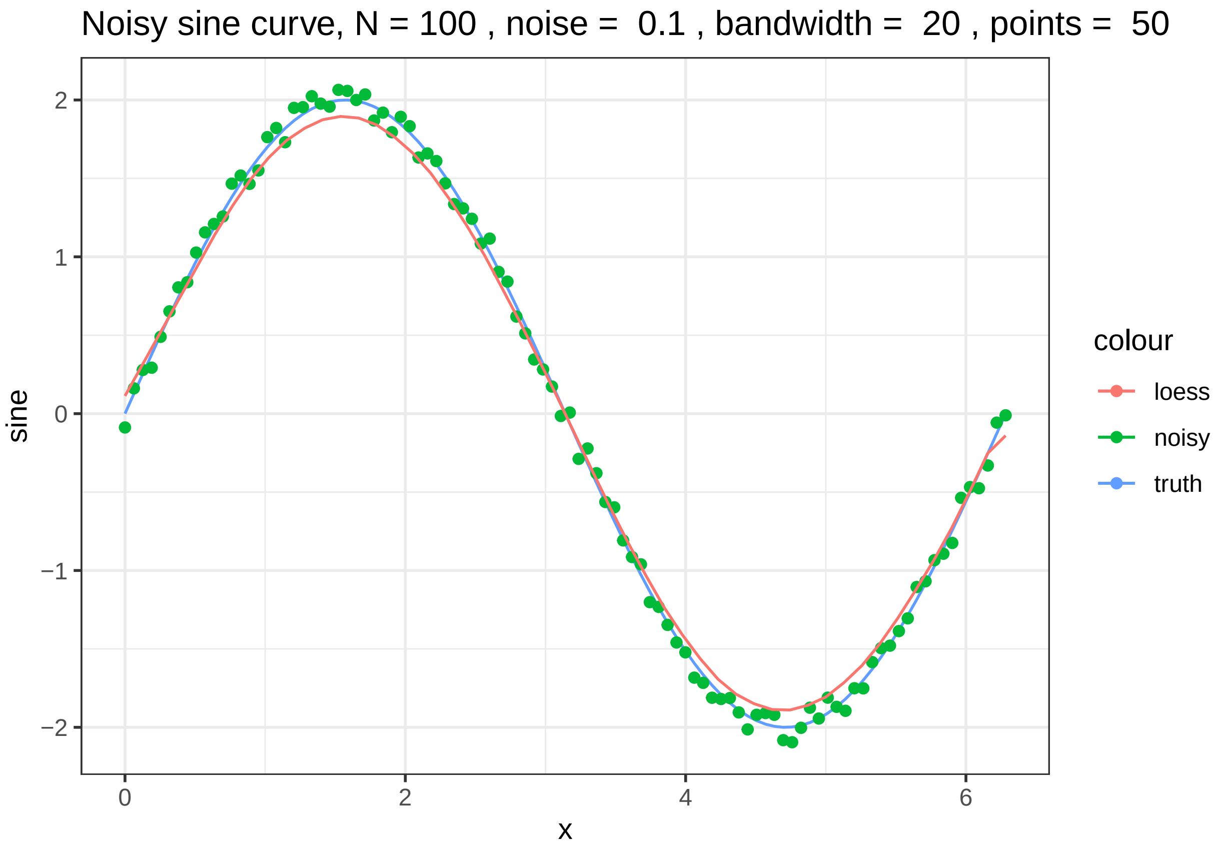


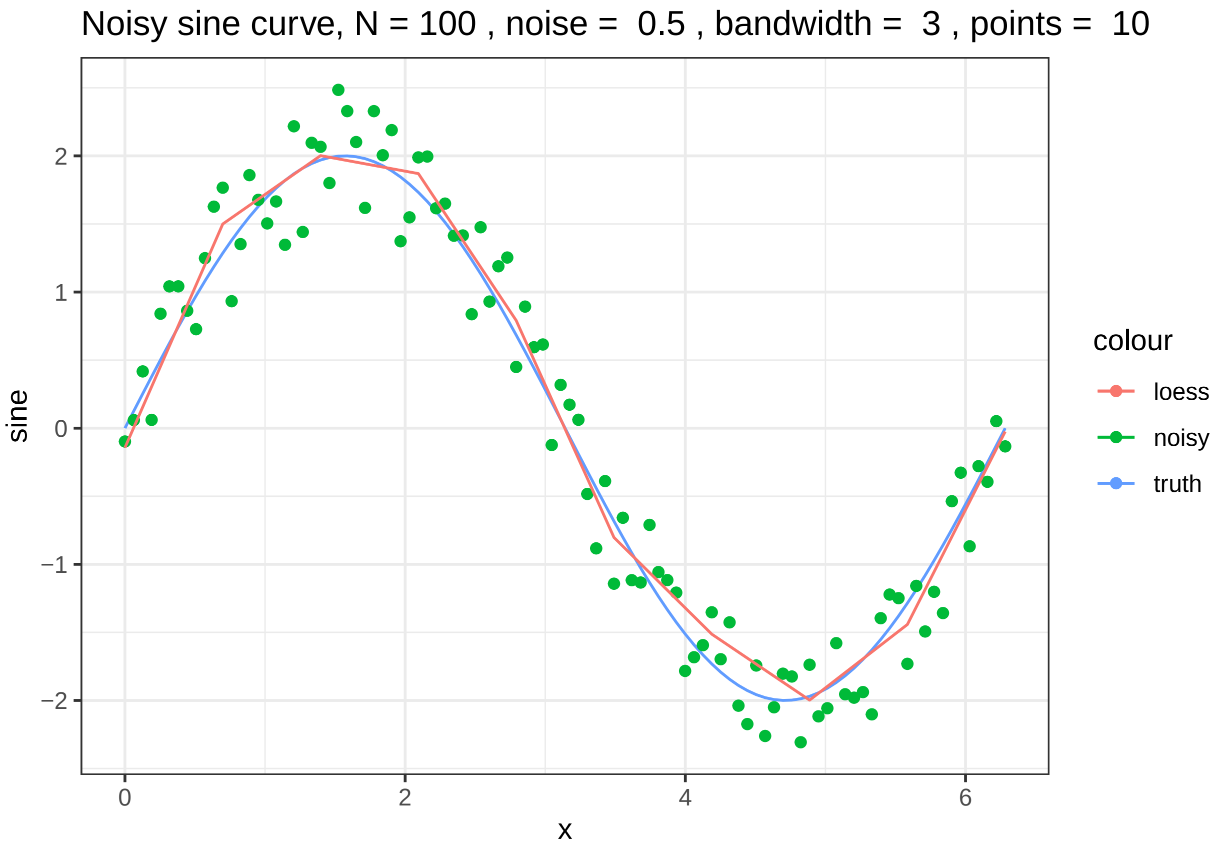
Noisy sine curve, N = 100 , noise = 0.1 , bandwidth = 10 , points = 25
sine
x
colour
loess
noisy
truth


Noisy sine curve, N = 100 , noise = 0.1 , bandwidth = 10 , points = 50
sine
x
2
1
0
−1
−2
0
2
4
6
colour
loess
noisy
truth

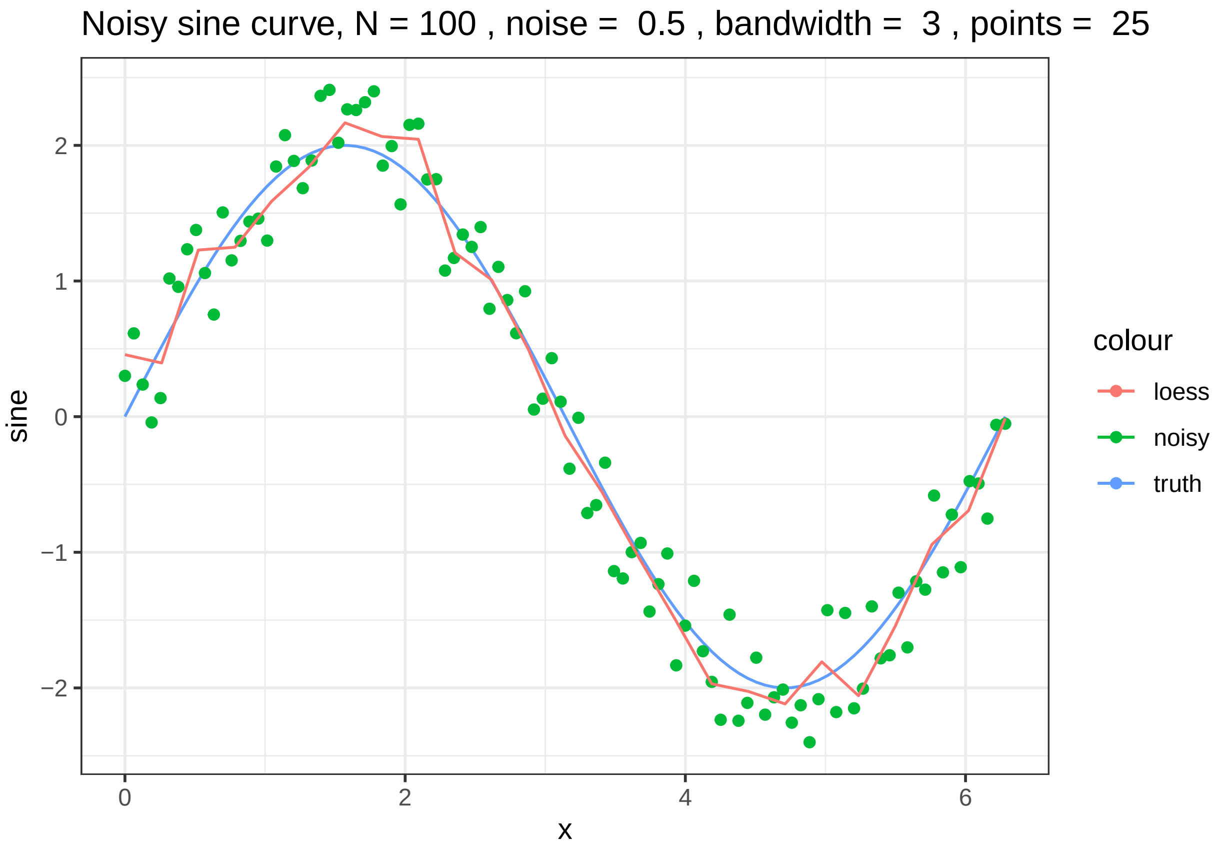

Noisy sine curve, N = 100 , noise = 0.1 , bandwidth = 20 , points = 10
sine
x
colour
loess
noisy
truth
2
1
0
−1
−2
0
2
4
6

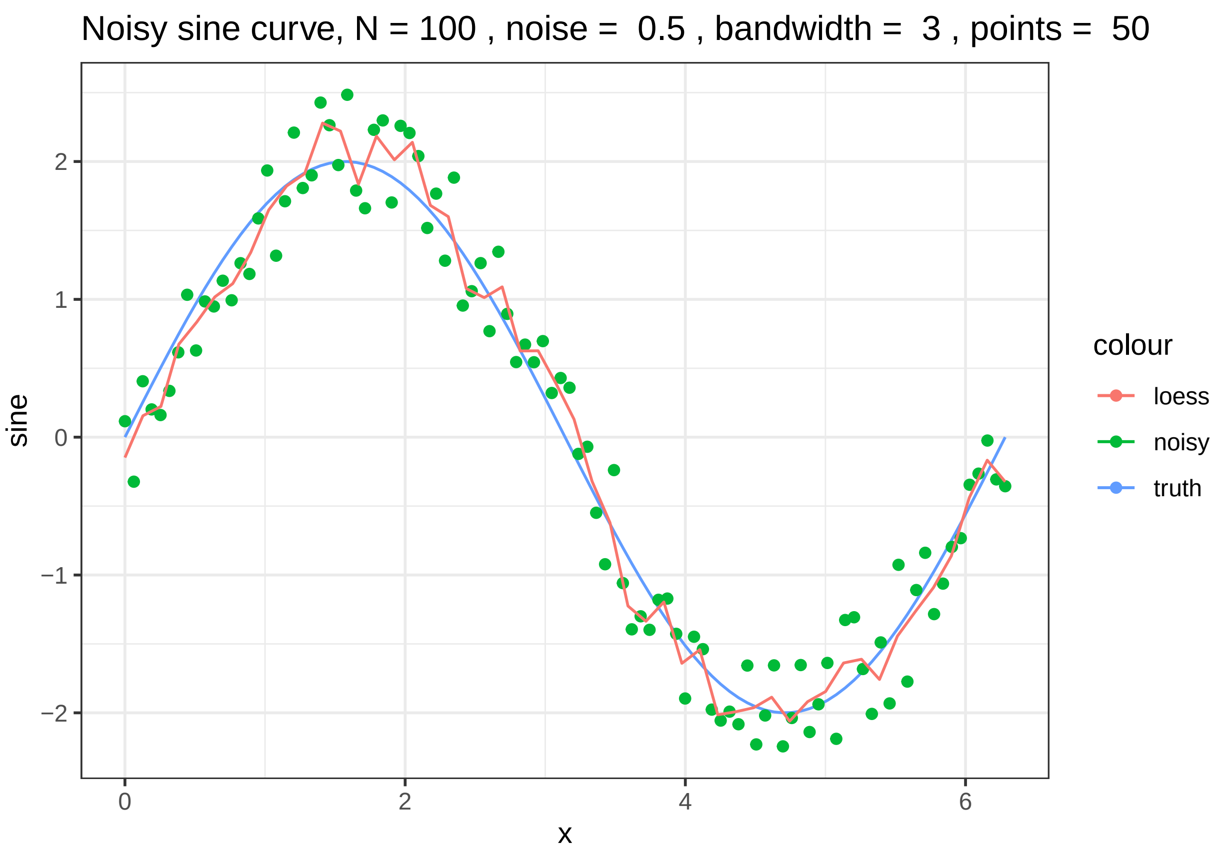

Noisy sine curve, N = 100 , noise = 0.1 , bandwidth = 20 , points = 25
sine
x
colour
loess
noisy
truth
2
1
0
−1
−2
0
2
4
6


Noisy sine curve, N = 100 , noise = 0.1 , bandwidth = 20 , points = 50
sine
x
2
1
0
−1
−2
0
2
4
6
colour
loess
noisy
truth

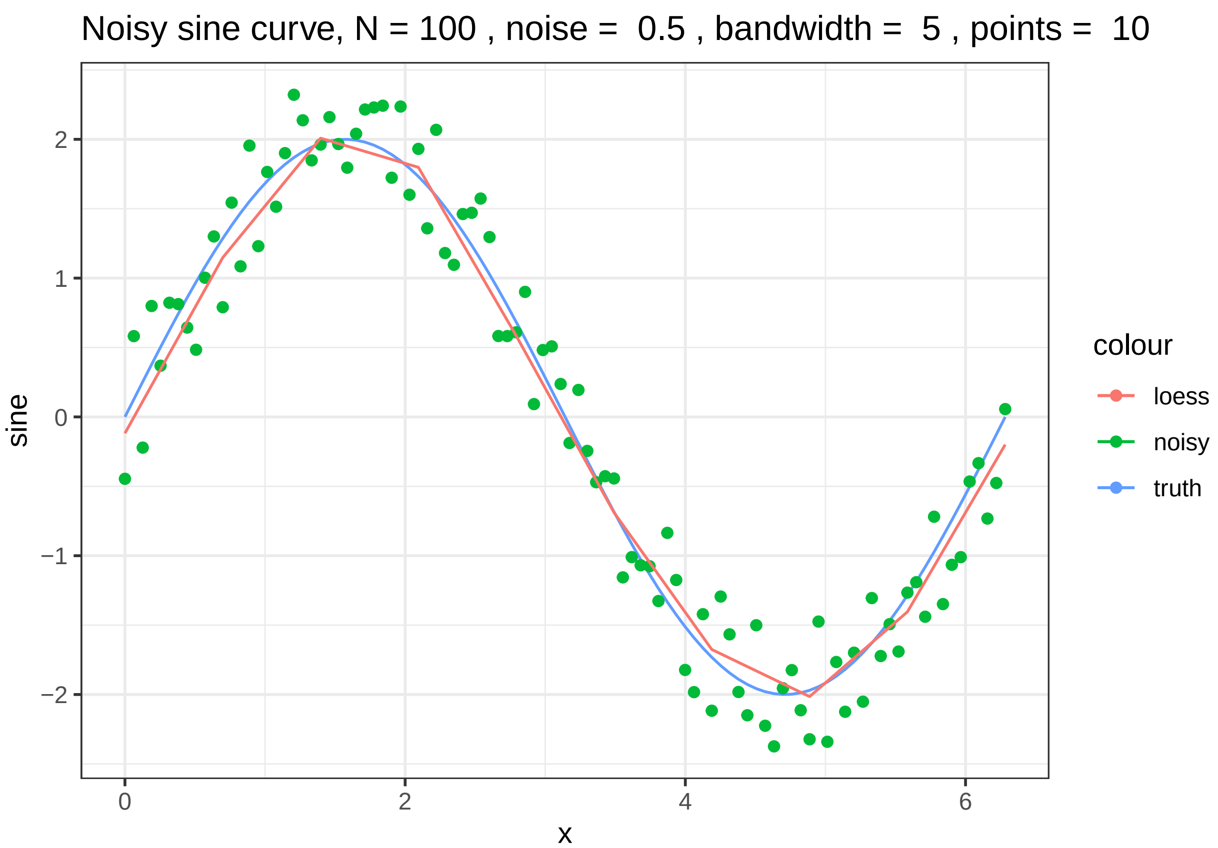


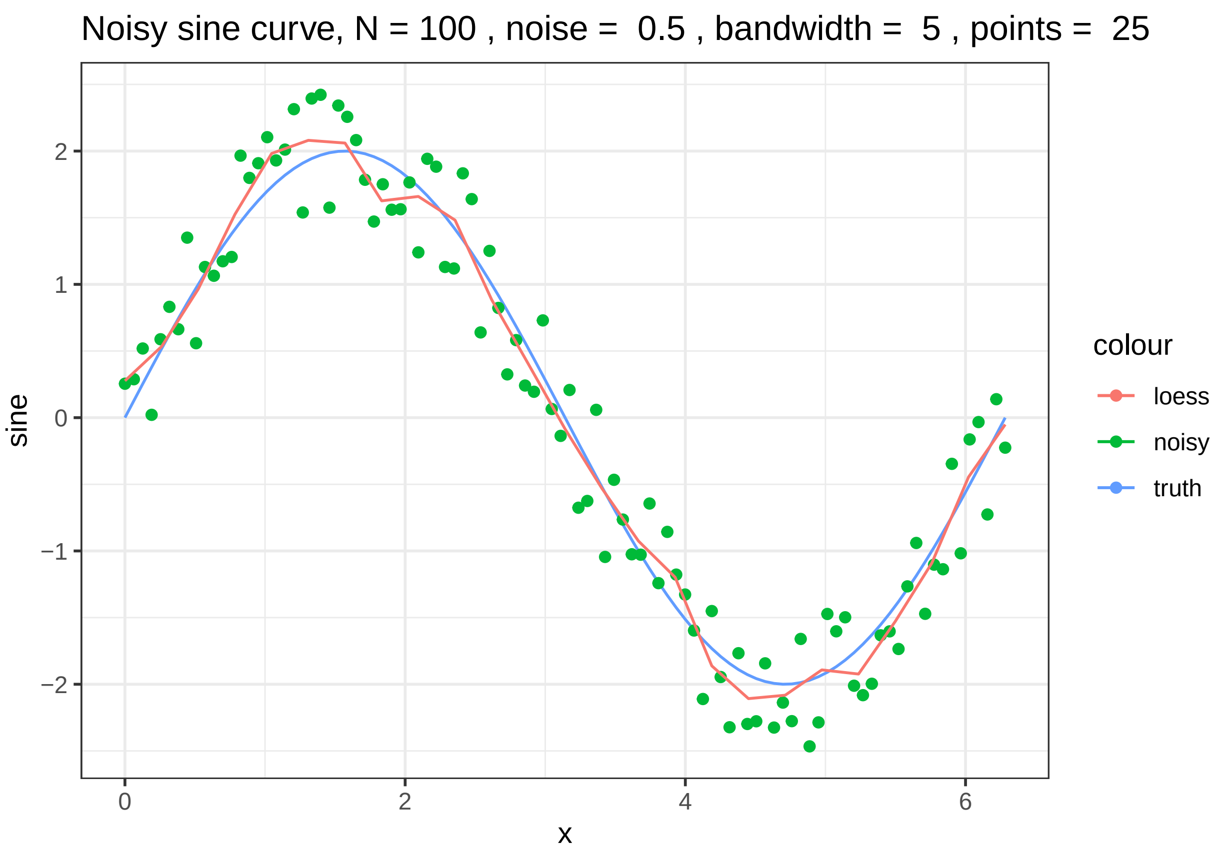
Noisy sine curve, N = 100 , noise = 0.5 , bandwidth = 3 , points = 10
sine
x
colour
loess
noisy
truth

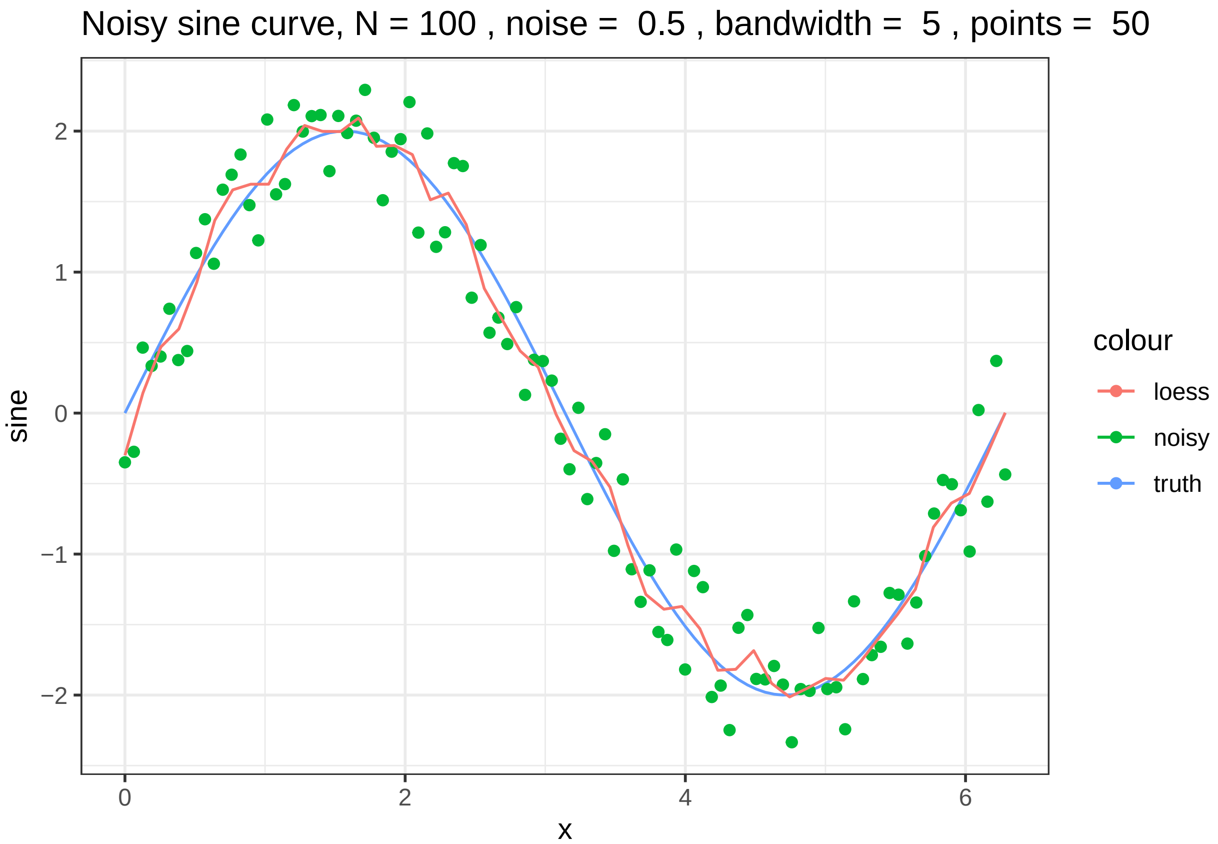

Noisy sine curve, N = 100 , noise = 0.5 , bandwidth = 3 , points = 25
sine
x
colour
loess
noisy
truth

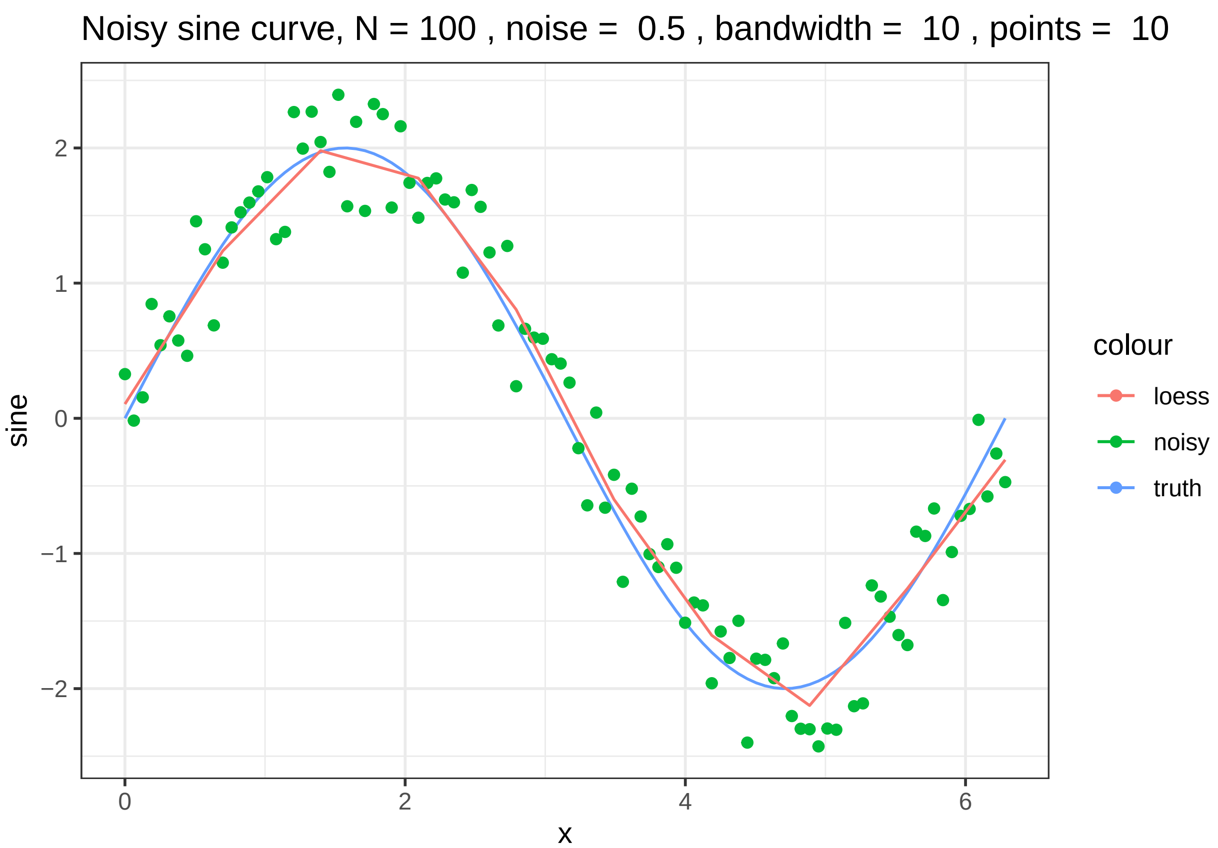


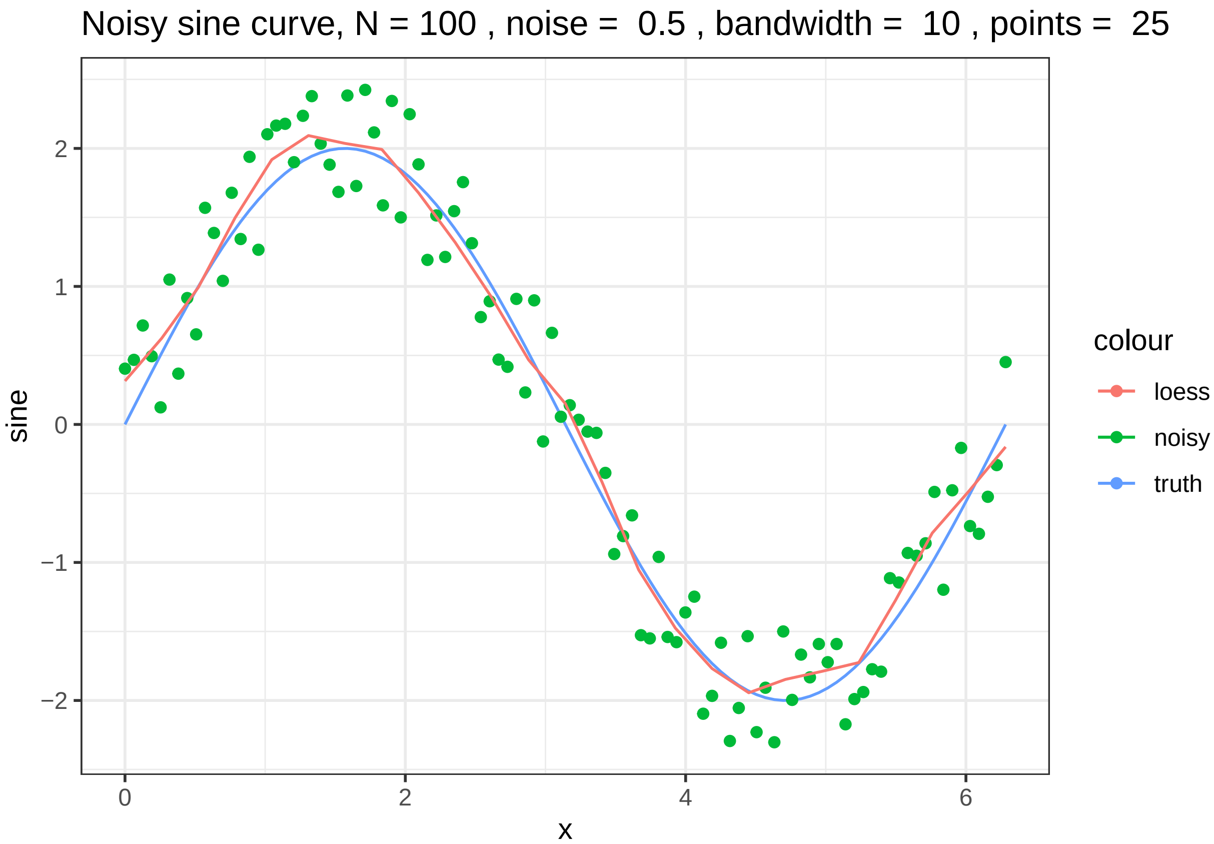
Noisy sine curve, N = 100 , noise = 0.5 , bandwidth = 3 , points = 50
sine
x
2
1
0
−1
−2
0
2
4
6
colour
loess
noisy
truth

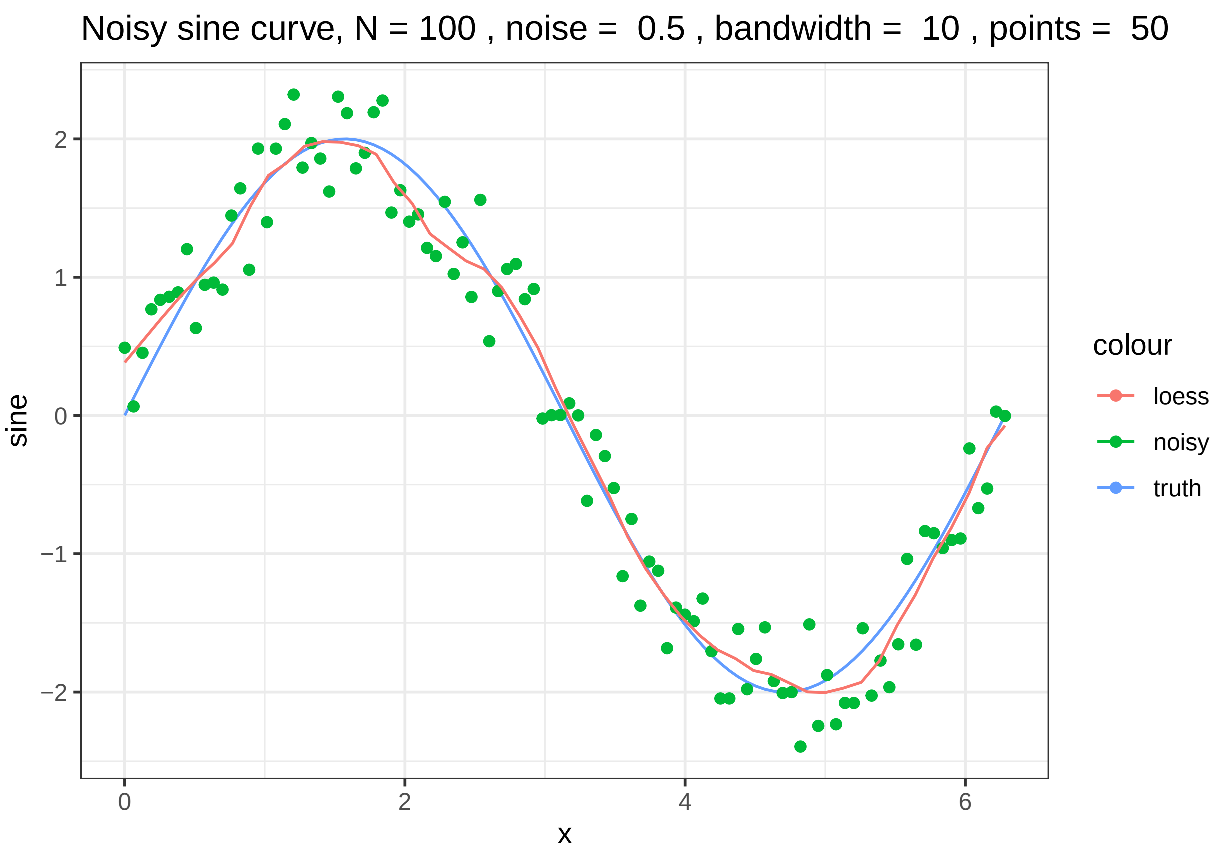

Noisy sine curve, N = 100 , noise = 0.5 , bandwidth = 5 , points = 10
sine
x
2
1
0
−1
−2
0
2
4
6
colour
loess
noisy
truth

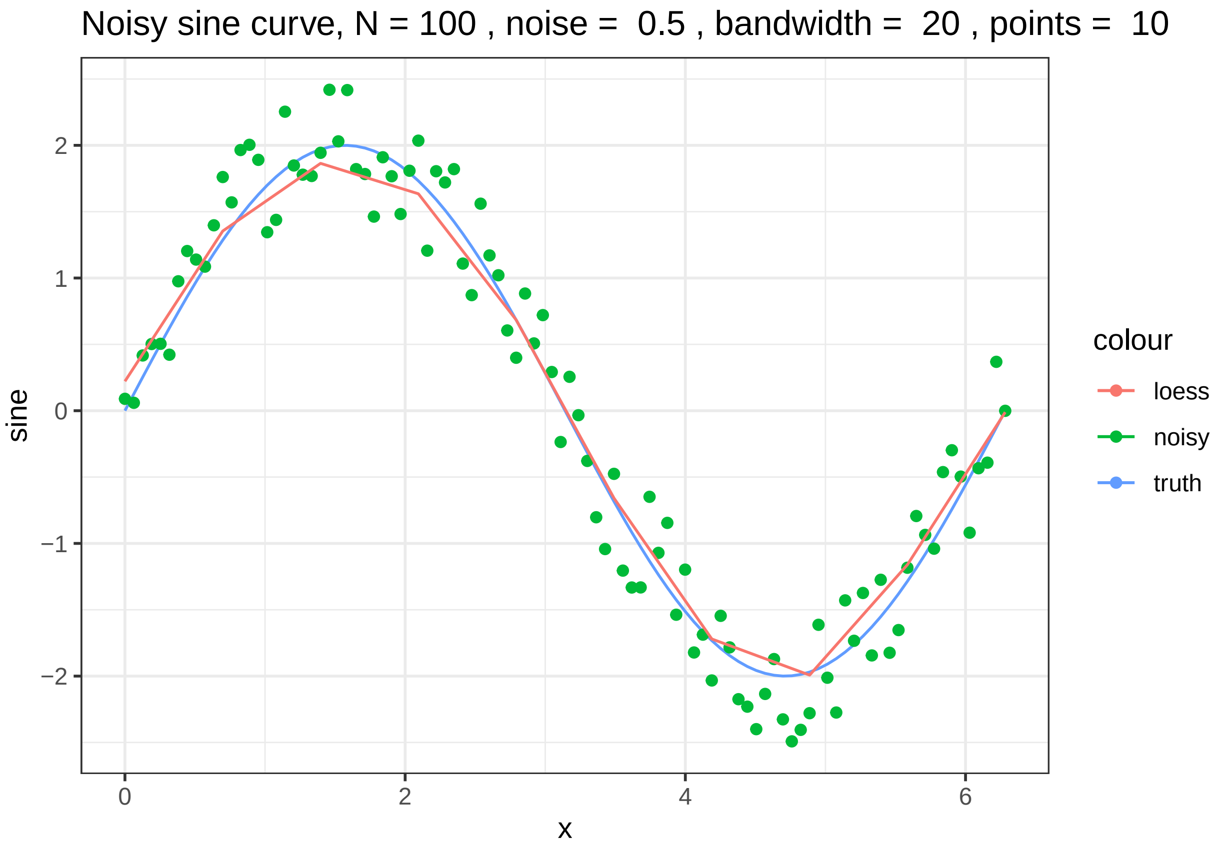

Noisy sine curve, N = 100 , noise = 0.5 , bandwidth = 5 , points = 25
sine
x
2
1
0
−1
−2
0
2
4
6
colour
loess
noisy
truth

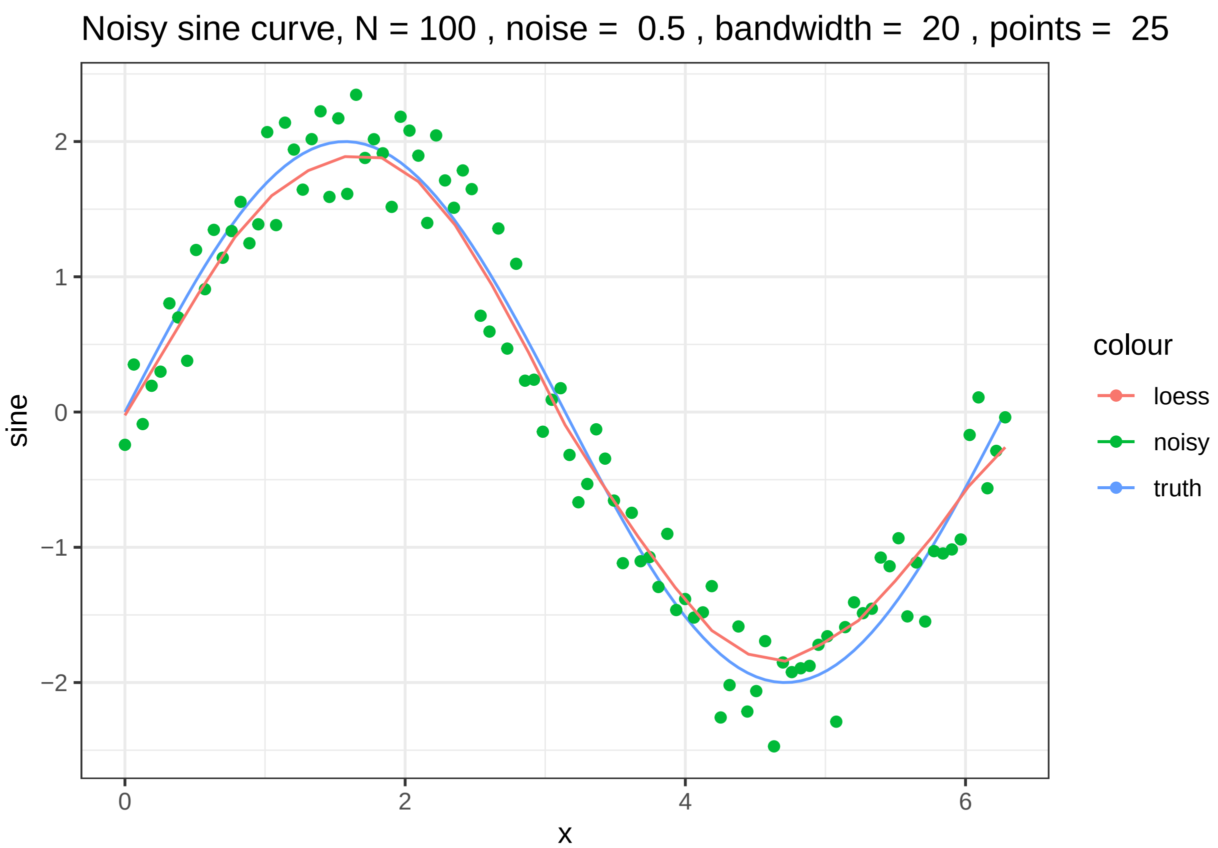

Noisy sine curve, N = 100 , noise = 0.5 , bandwidth = 5 , points = 50
sine
2
1
0
−1
−2
0
2
4
6
x
colour
loess
noisy
truth

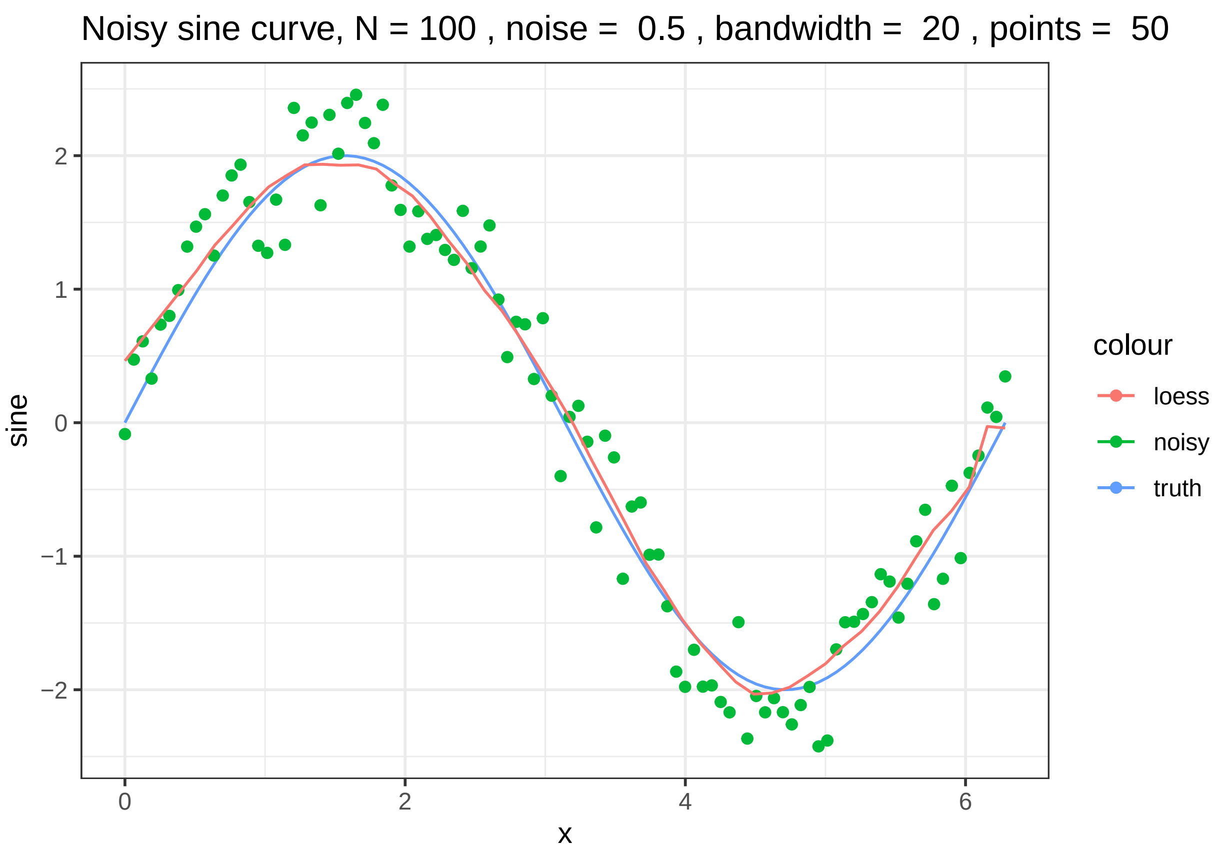

Noisy sine curve, N = 100 , noise = 0.5 , bandwidth = 10 , points = 10
sine
x
2
1
0
−1
−2
0
2
4
6
colour
loess
noisy
truth


Noisy sine curve, N = 100 , noise = 0.5 , bandwidth = 10 , points = 25
sine
x
2
1
0
−1
−2
0
2
4
6
colour
loess
noisy
truth

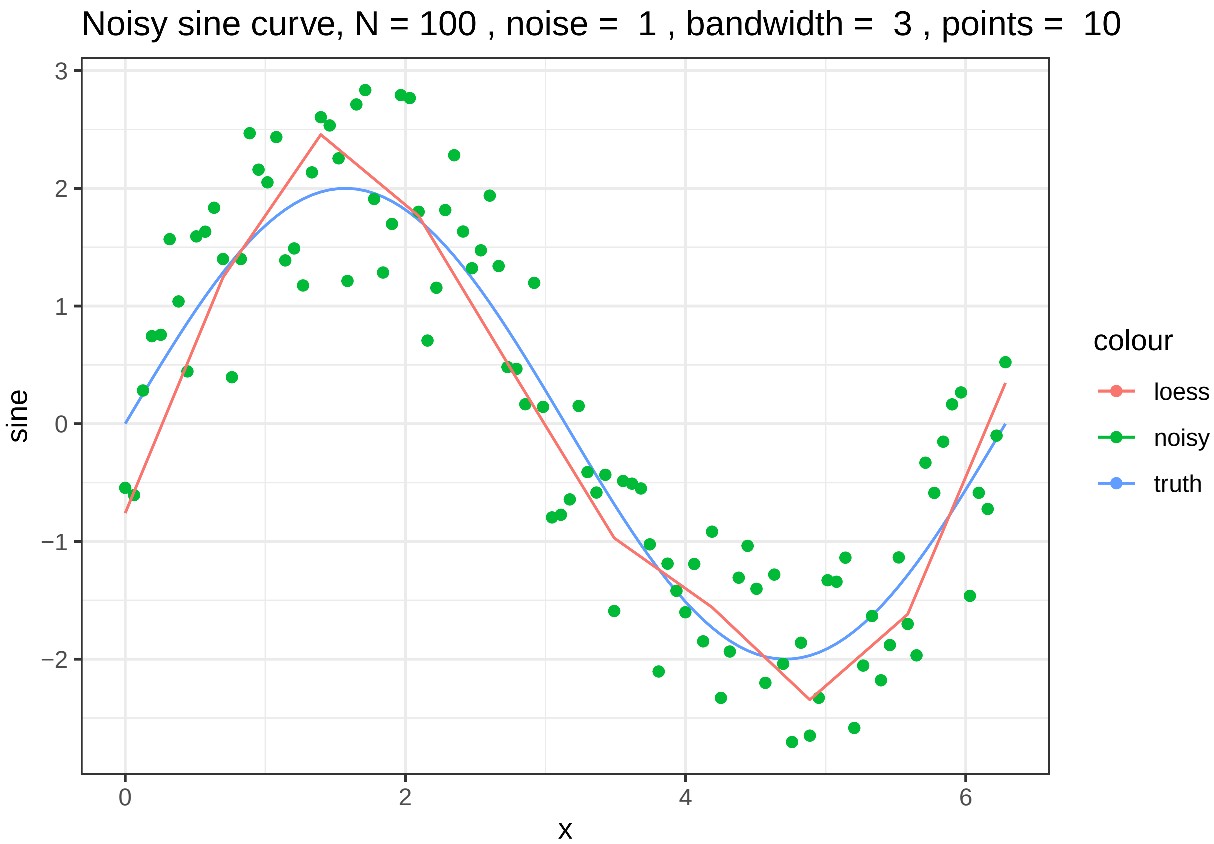

Noisy sine curve, N = 100 , noise = 0.5 , bandwidth = 10 , points = 50
sine
x
2
1
0
−1
−2
0
2
4
6
colour
loess
noisy
truth

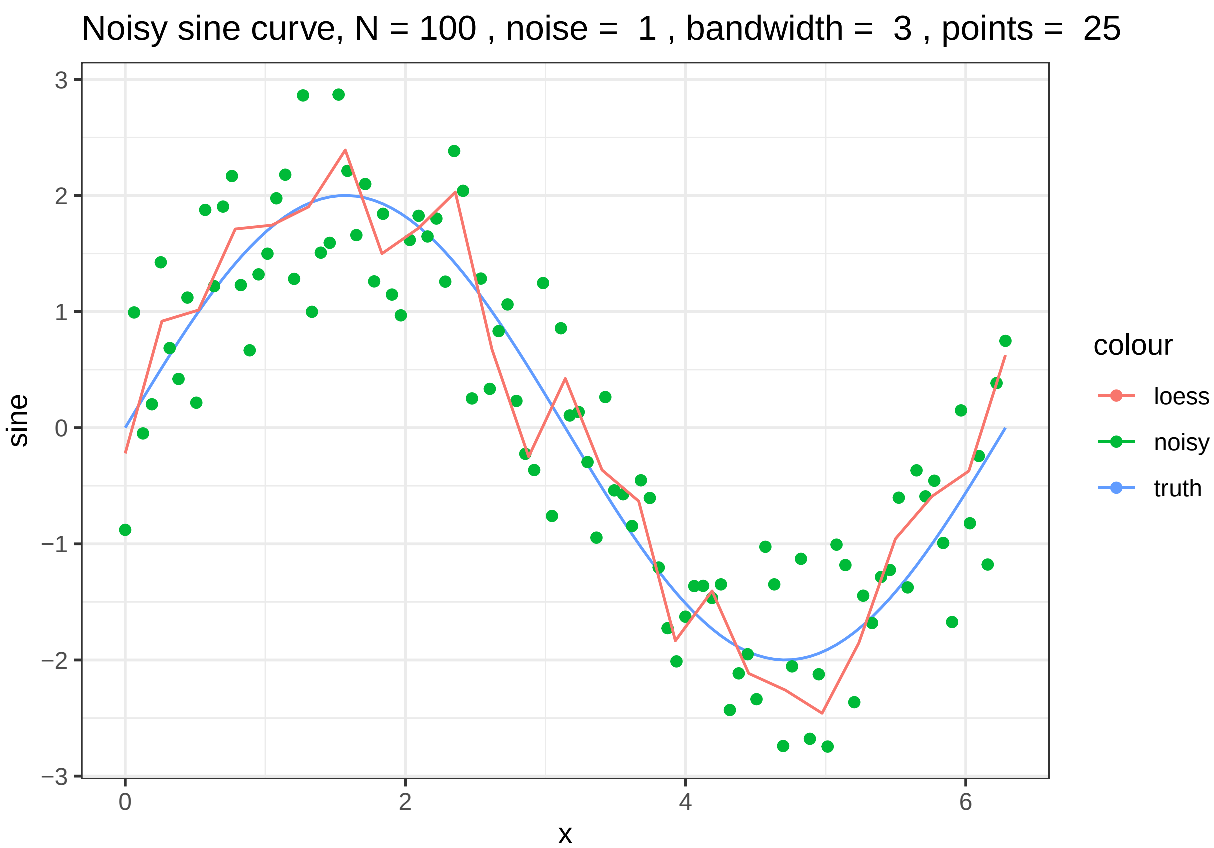

Noisy sine curve, N = 100 , noise = 0.5 , bandwidth = 20 , points = 10
sine
x
2
1
0
−1
−2
0
2
4
6
colour
loess
noisy
truth

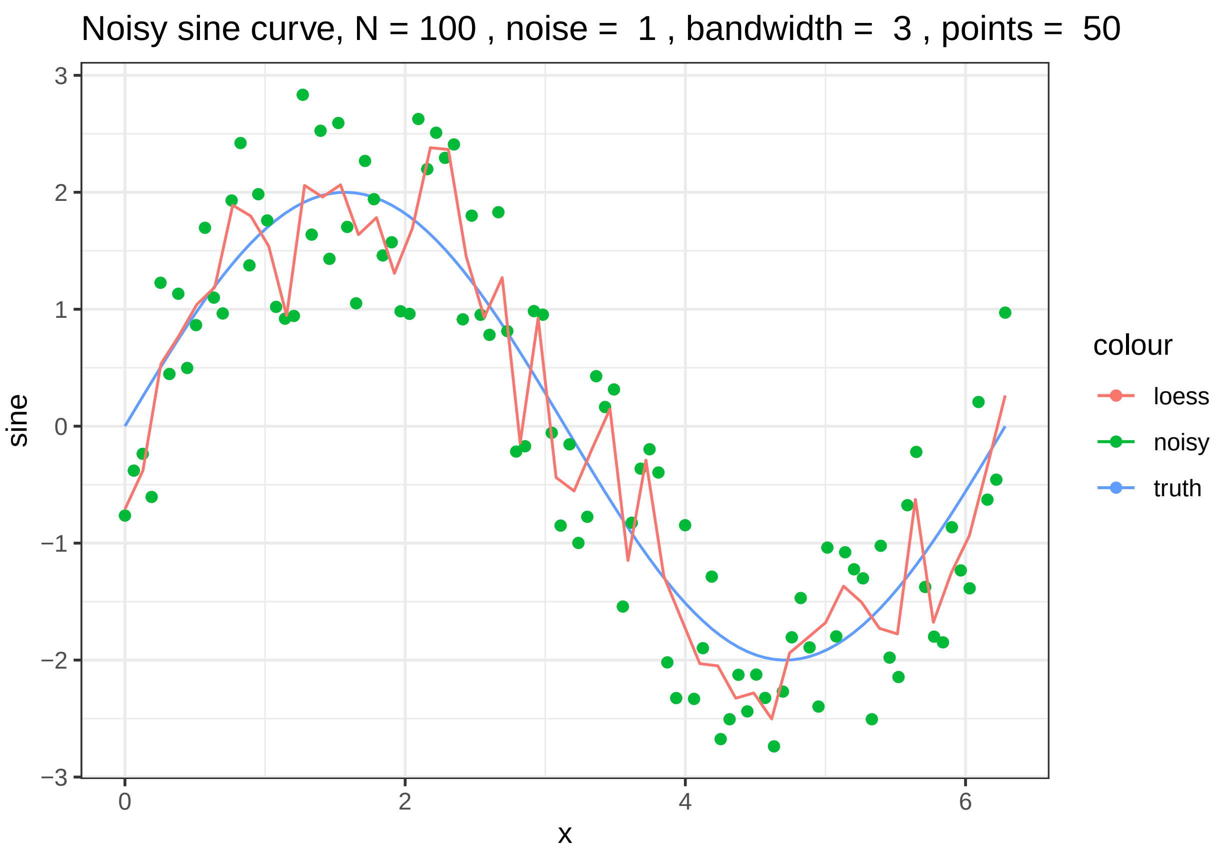

Noisy sine curve, N = 100 , noise = 0.5 , bandwidth = 20 , points = 25
sine
2
1
0
−1
−2
0
2
4
6
x
colour
loess
noisy
truth

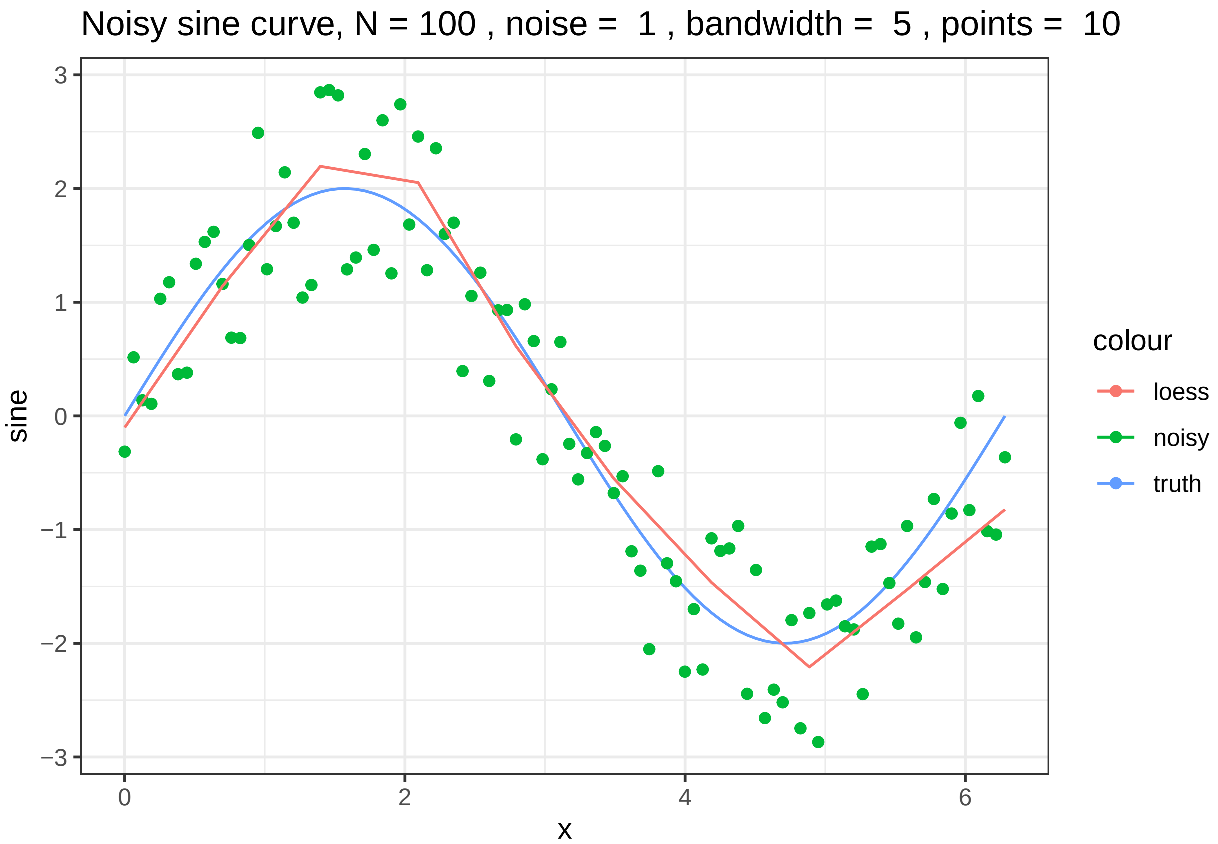


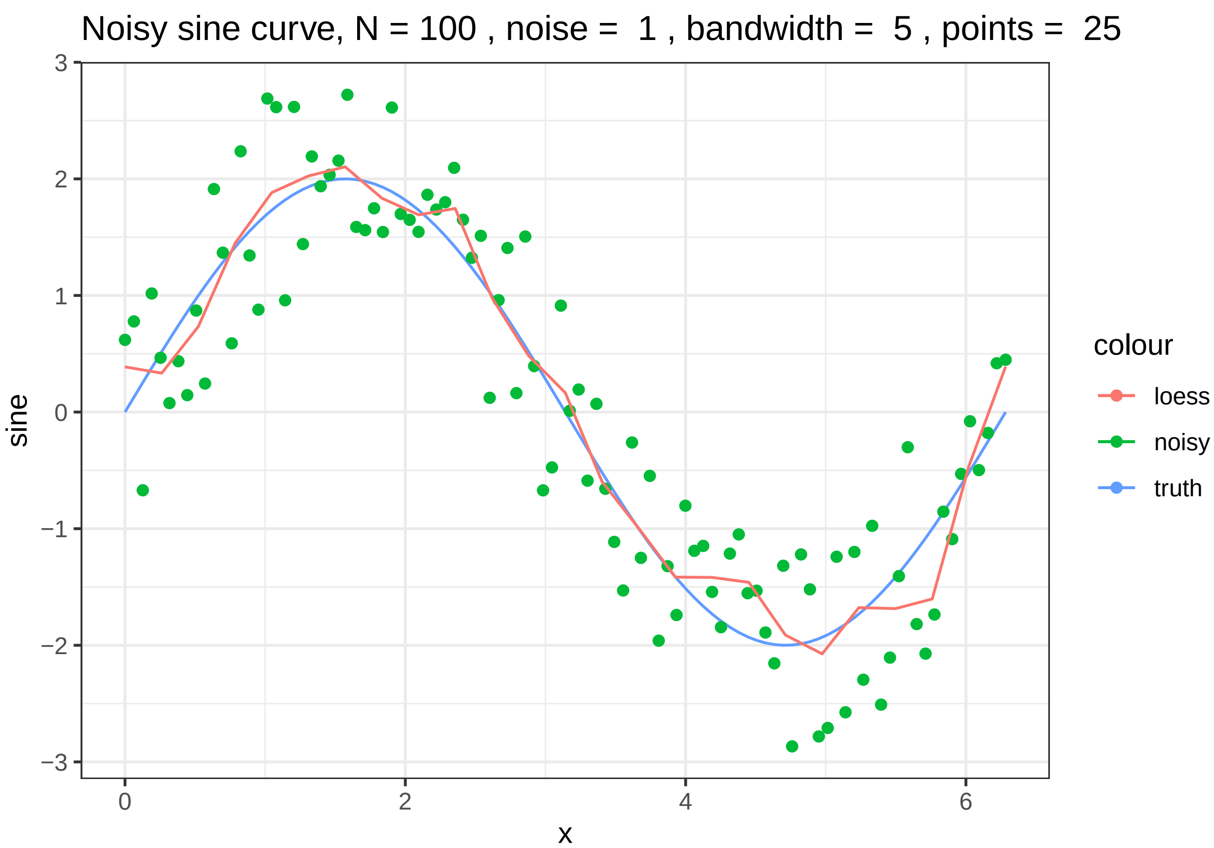
Noisy sine curve, N = 100 , noise = 0.5 , bandwidth = 20 , points = 50
sine
x
colour
loess
noisy
truth
2
1
0
−1
−2
0
2
4
6

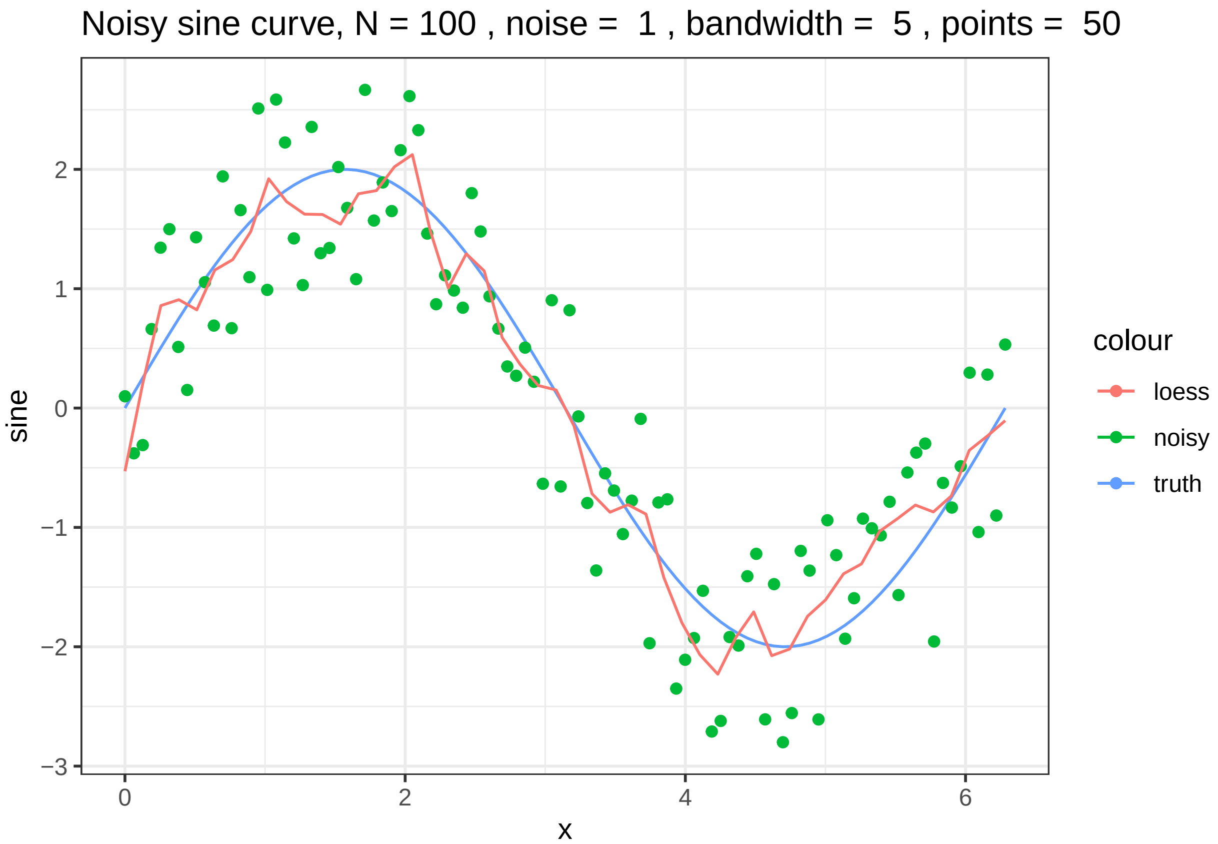

Noisy sine curve, N = 100 , noise = 1 , bandwidth = 3 , points = 10
sine
x
colour
loess
noisy
truth

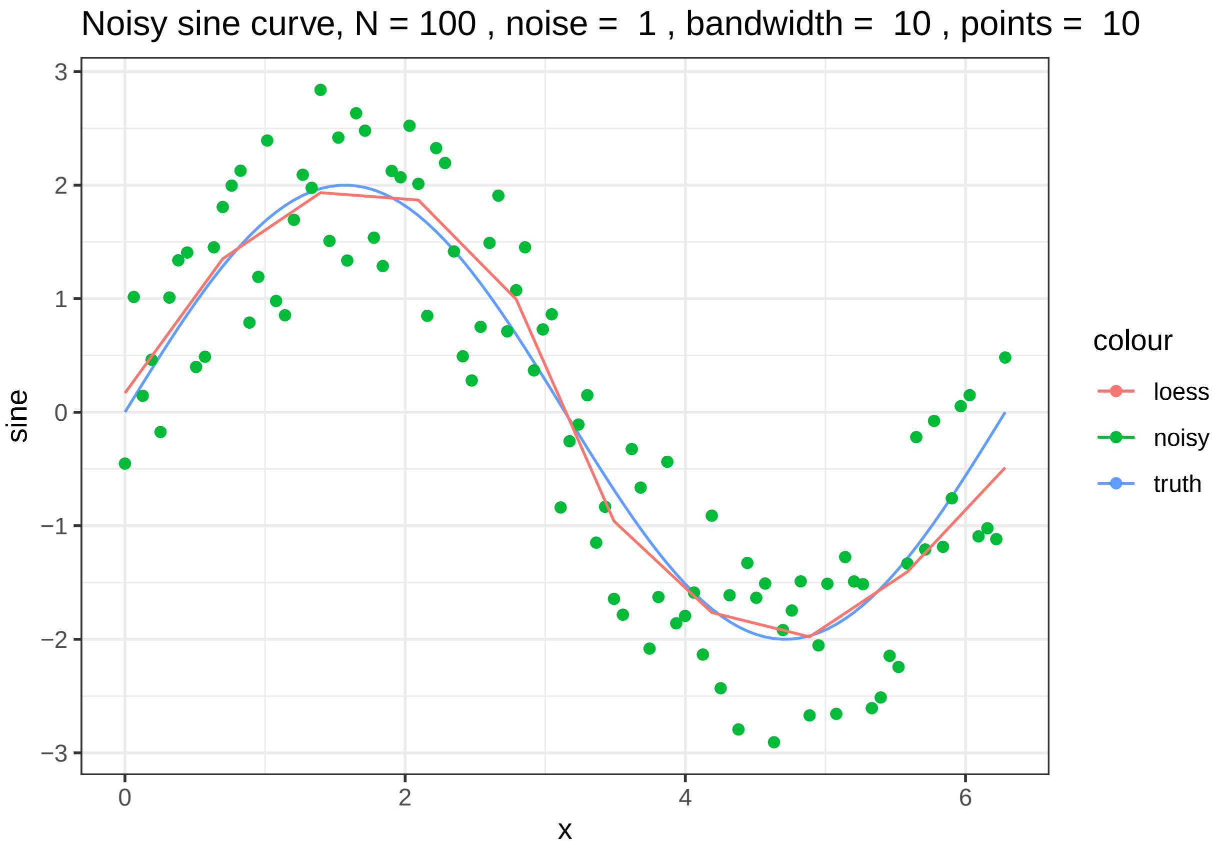

Noisy sine curve, N = 100 , noise = 1 , bandwidth = 3 , points = 25
sine
x
3
2
1
0
−1
−2
−3
0
2
4
6
colour
loess
noisy
truth


Noisy sine curve, N = 100 , noise = 1 , bandwidth = 3 , points = 50
sine
x
colour
loess
noisy
truth

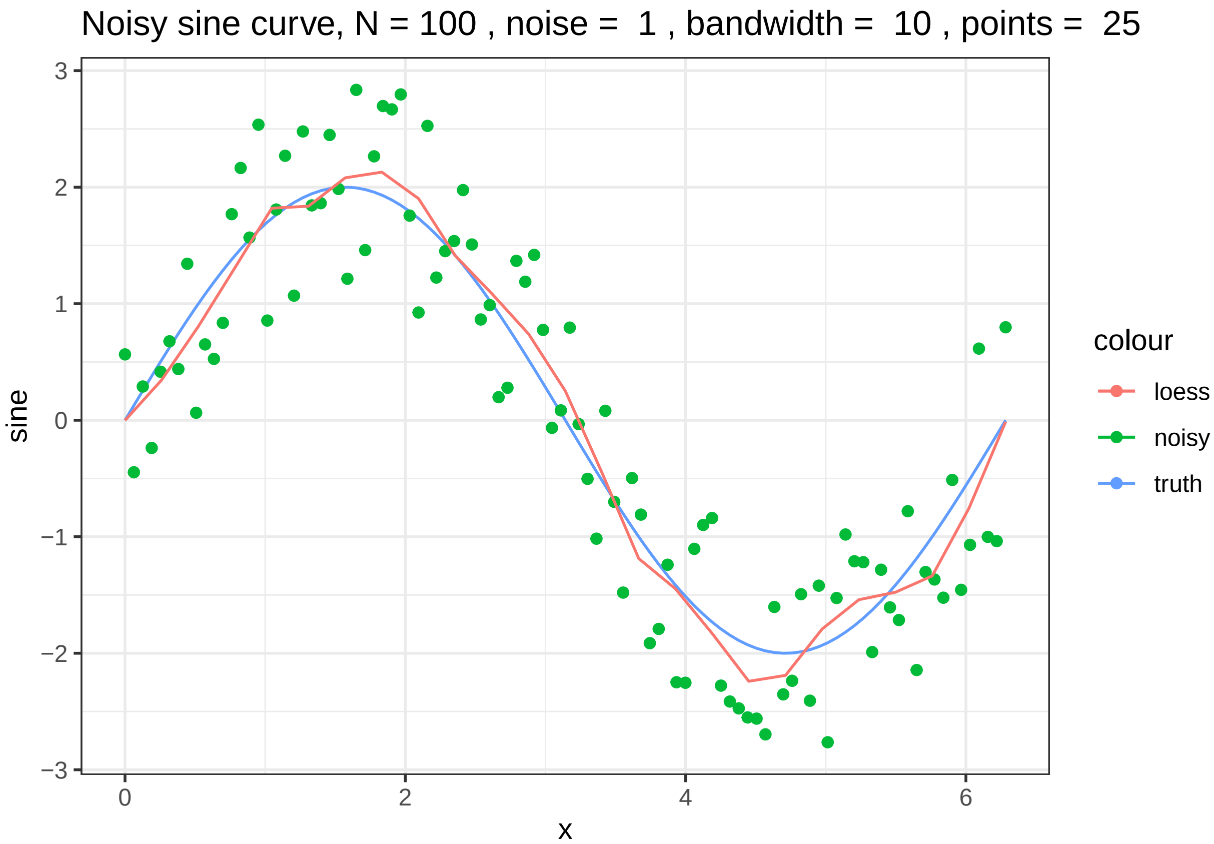

Noisy sine curve, N = 100 , noise = 1 , bandwidth = 5 , points = 10
sine
3
2
1
0
−1
−2
−3
0
2
4
6
x
colour
loess
noisy
truth

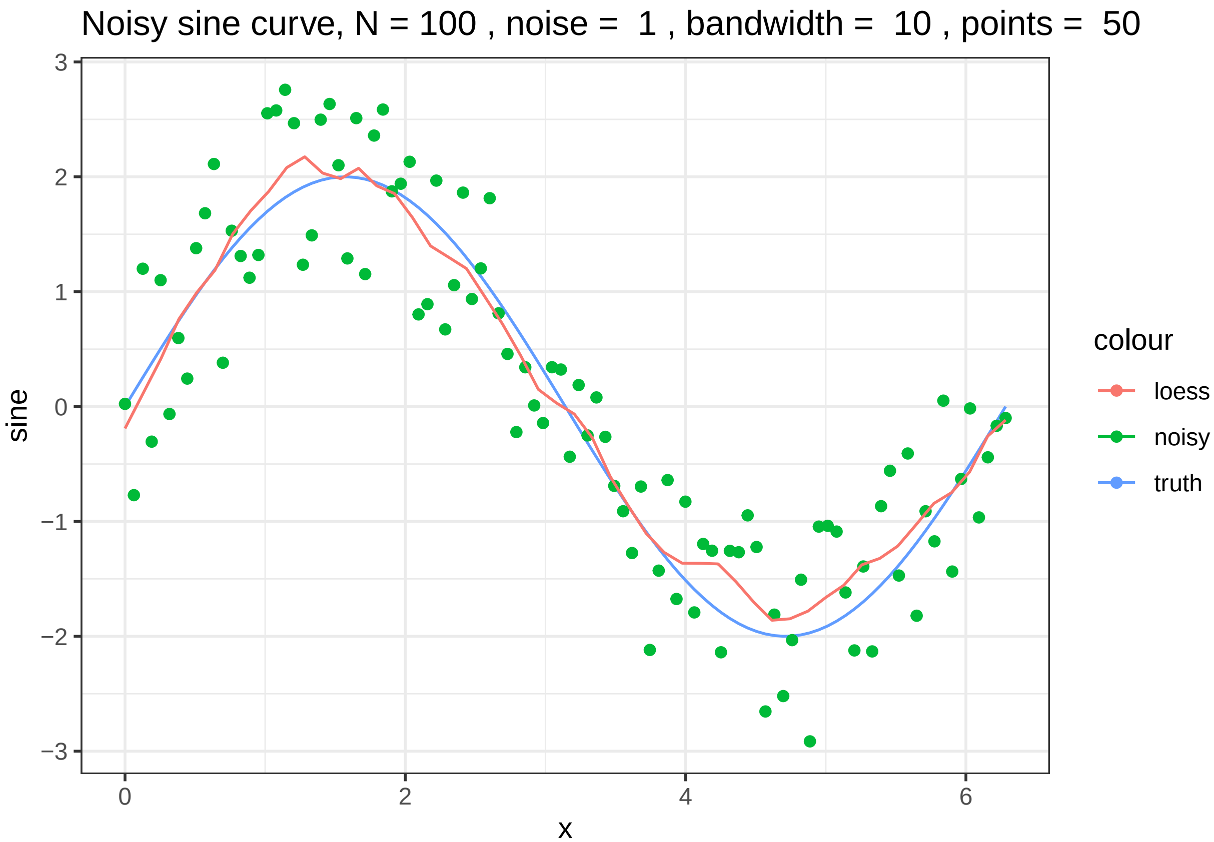

Noisy sine curve, N = 100 , noise = 1 , bandwidth = 5 , points = 25
sine
x
3
2
1
0
−1
−2
−3
0
2
4
6
colour
loess
noisy
truth

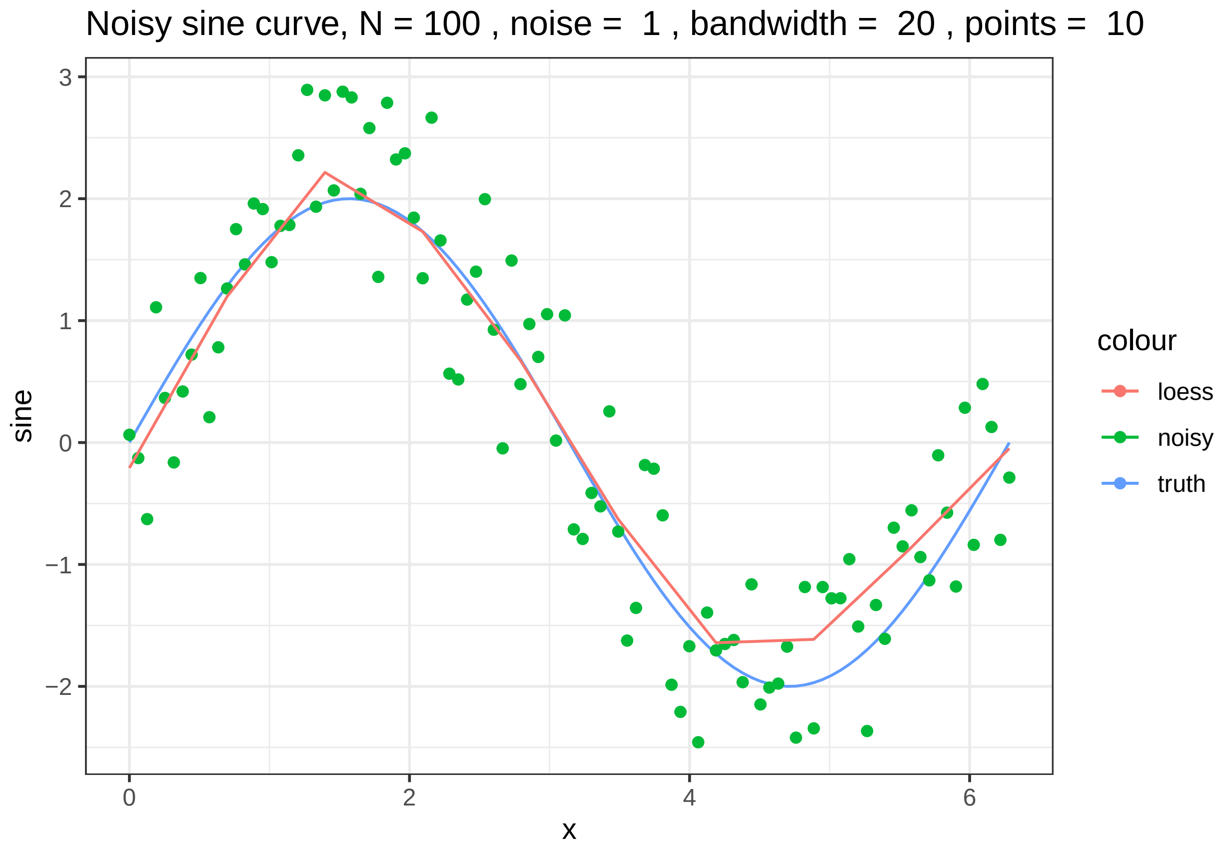

Noisy sine curve, N = 100 , noise = 1 , bandwidth = 5 , points = 50
sine
x
colour
loess
noisy
truth
2
1
0
−1
−2
−3
0
2
4
6

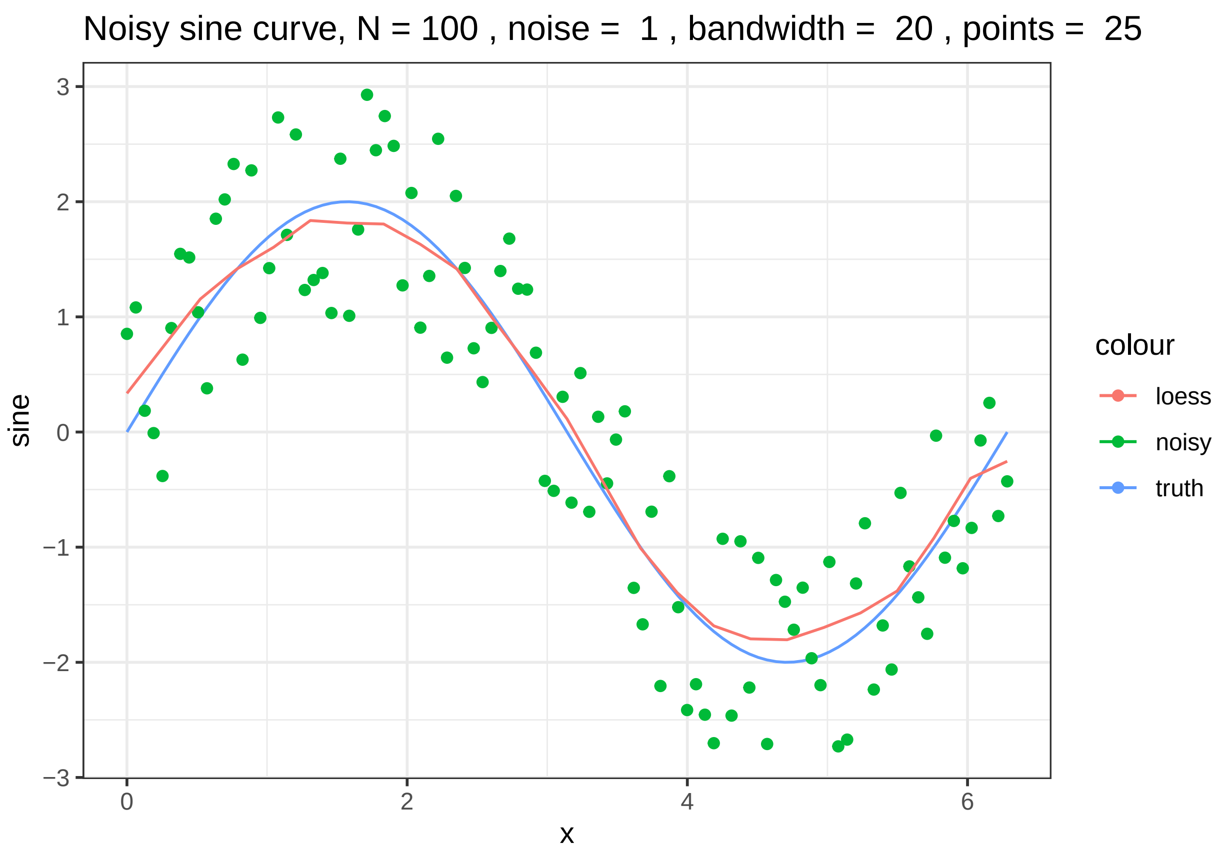


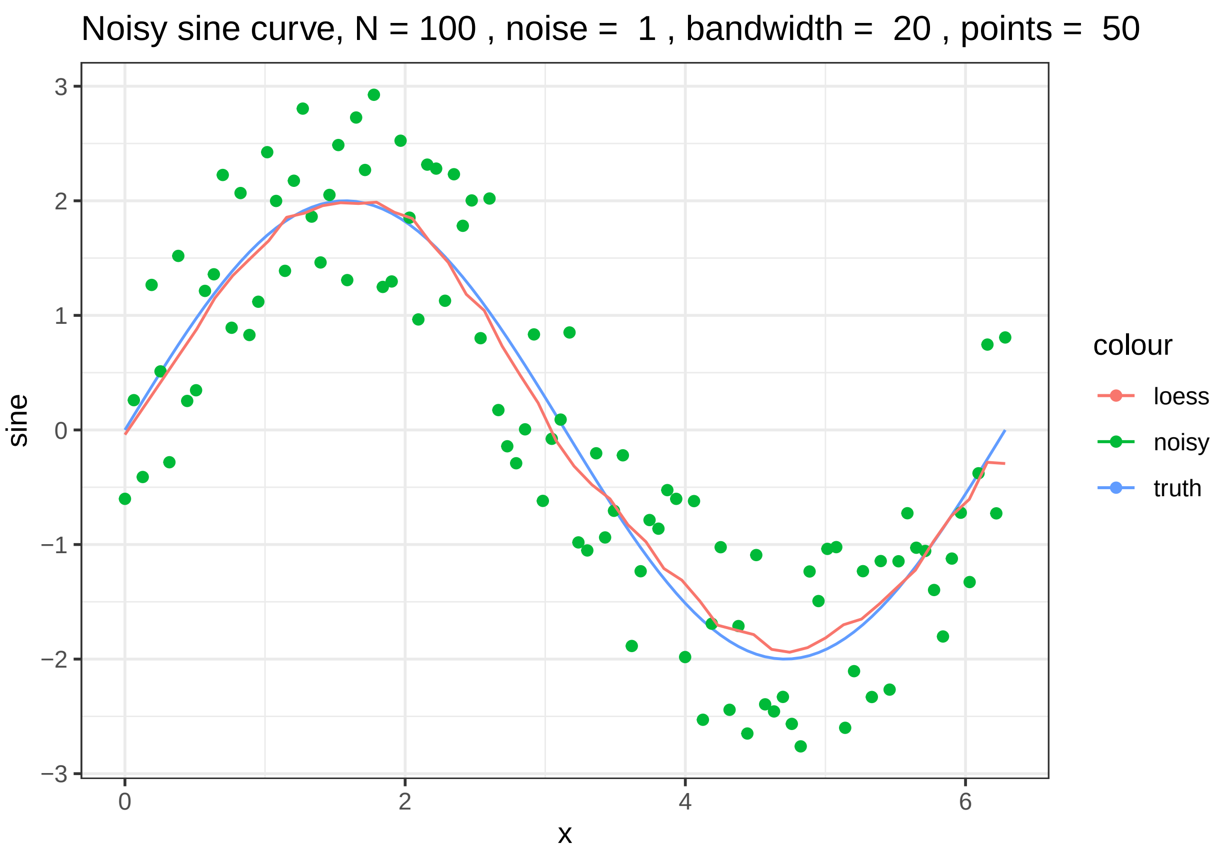
Noisy sine curve, N = 100 , noise = 1 , bandwidth = 10 , points = 10
sine
x
3
2
1
0
−1
−2
−3
0
2
4
6
colour
loess
noisy
truth


Noisy sine curve, N = 100 , noise = 1 , bandwidth = 10 , points = 25
sine
x
3
2
1
0
−1
−2
−3
0
2
4
6
colour
loess
noisy
truth


Noisy sine curve, N = 100 , noise = 1 , bandwidth = 10 , points = 50
sine
x
3
2
1
0
−1
−2
−3
0
2
4
6
colour
loess
noisy
truth


Noisy sine curve, N = 100 , noise = 1 , bandwidth = 20 , points = 10
sine
x
3
2
1
0
−1
−2
0
2
4
6
colour
loess
noisy
truth


Noisy sine curve, N = 100 , noise = 1 , bandwidth = 20 , points = 25
sine
x
3
2
1
0
−1
−2
−3
0
2
4
6
colour
loess
noisy
truth



As in Python, overfitting occurs with high point count and low bandwidth. Performance again improves for higher bandwidths, with a point count of approximately half the length of the original dataset seems generally to perform better than alternatives.

2) Kernel density estimation

Here we define kernel density estimation as a function of an input vector y, a bandwidth bw, and a point count n.

```
kde = function(y, bw, n=500) {
  if (bw == 'SJ') bw = 'SJ-ste'
  model = density(y, bw=bw, n=n)
  model
}
```

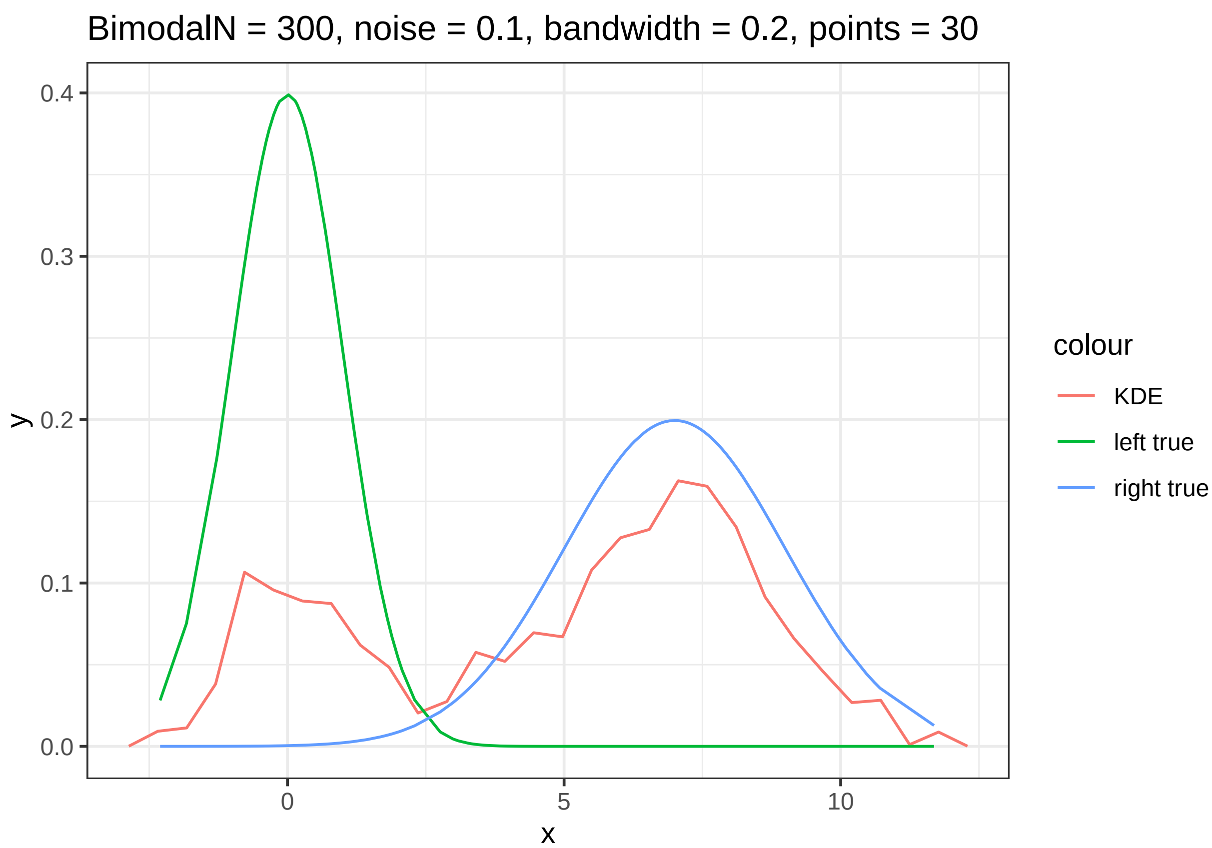
Define a function to generate data from a bimodal distribution.

```
generate_bimodal = function(mu1, mu2, sd1, sd2, n, weight=0.5) {
  spl = as.integer(n / 2)
  x1 = rnorm(as.integer(n * weight), mean=mu1, sd=sd1)
  x2 = rnorm(n - spl, mean=mu2, sd=sd2)
  c(c(x1), (x2))
}
```

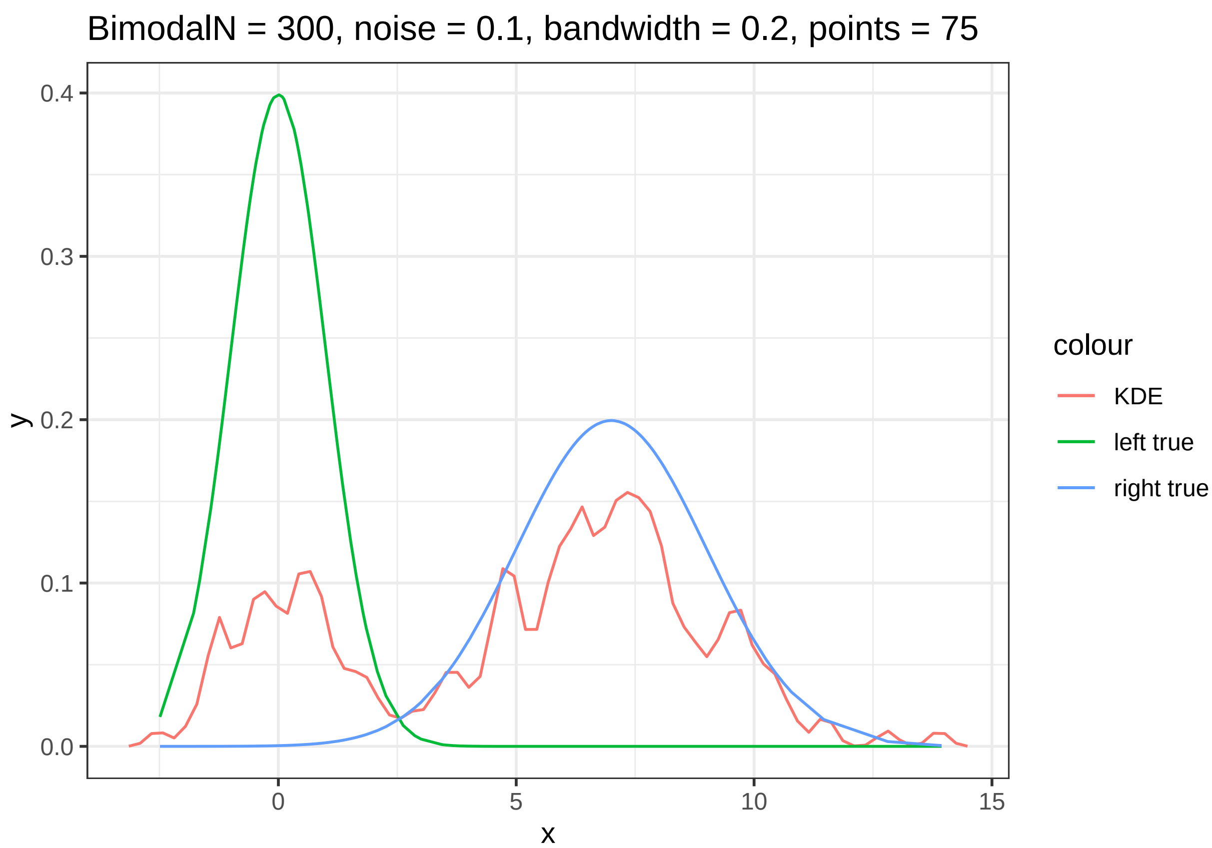
Generate datasets from the bimodal distribution and test the KDE function for different bandwidth values and amounts of noise.

```
# set parameters
mu1 = 0
mu2 = 7
sd1 = 1
sd2 = 2
n = 300
w = 0.2

noises = c(.1, .5, 1)                                                   # amounts of noise
lambdas = c(0.2, 0.5, 1, 3)                                             # manual bandwidths to try
lambda_selection_methods = c('nrd0', 'nrd', 'SJ')                      # bandwidth selection methods
points = c(as.integer(n / 10), as.integer(n / 4), as.integer(n / 2))  # point counts to try

for (noise in noises){
    for (l in lambdas) {
        for (pts in points) {
            # generate samples from bimodal distribution
            y = generate_bimodal(mu1, mu2, sd1, sd2, n, w)

            # apply noise
            noisy_y = sapply(y, function(i) jitter(i, amount=noise))

            # summarize data
            y_min = min(y)
            y_max = max(y)
            y_std = sd(y)
            y_lo = y_min - y_std
            y_hi = y_max + y_std
            y_dom = seq(y_lo, y_hi, length.out=n)

            # fit kernel density model on noisy data
            kde_model = kde(noisy_y, l, pts)
            kde_result = data.frame(x=kde_model$x, y=kde_model$y)

            # create plot
            p = ggplot(kde_result) +
              geom_line(aes(x=x, y=y, color='KDE')) +
              geom_line(
                data=data.frame(x=y, y=dnorm(y, mu1, sd1)),
                aes(x=x, y=y, color='left true')) +
              geom_line(
                data=data.frame(x=y, y=dnorm(y, mu2, sd2)),
                aes(x=x, y=y, color='right true')) +
              theme_bw() +
              labs(title=paste0("Bimodal",
                                 "N = ", n,
                                 ", noise = ", noise,
                                 ", bandwidth = ", l,
                                 ", points = ", pts))
            print(p)
        }
```

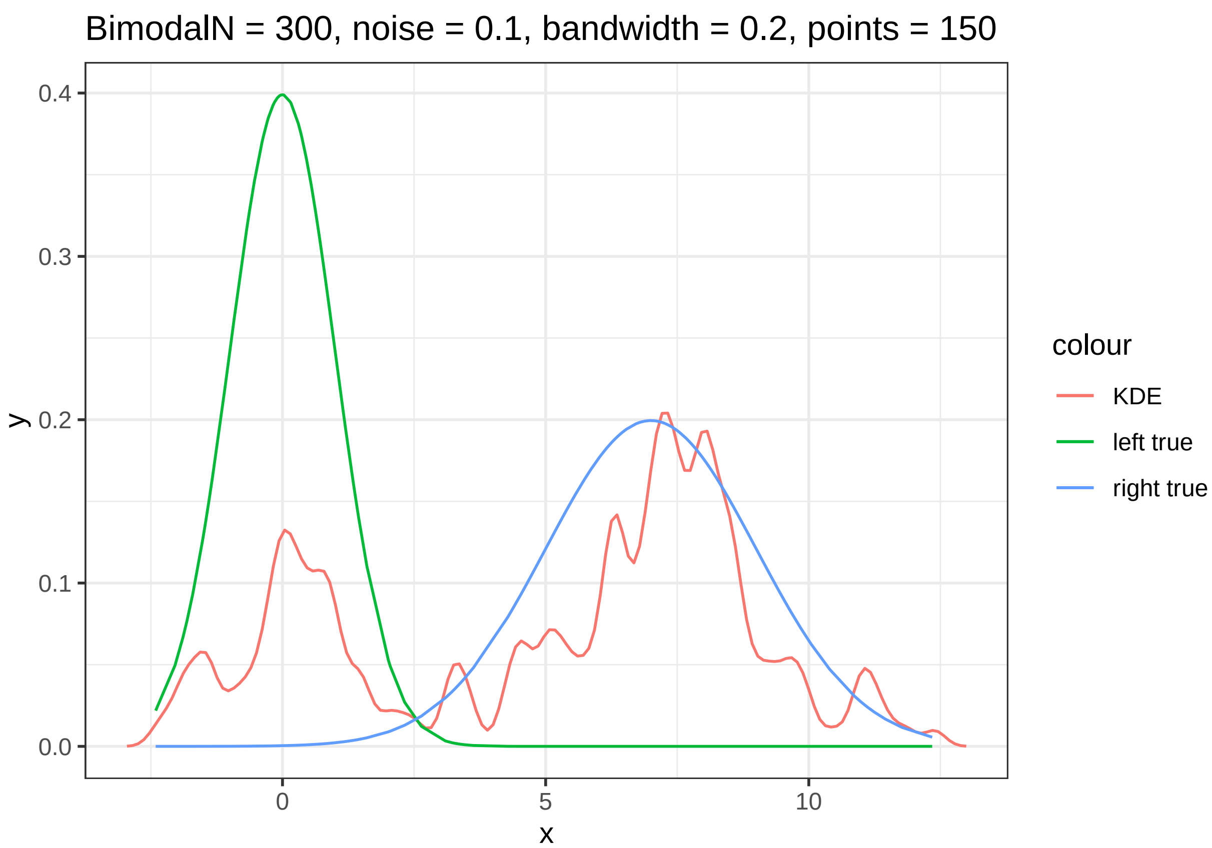
```
    }

  for (l in lambda_selection_methods) {
        for (pts in points) {
            # generate samples from bimodal distribution
            y = generate_bimodal(mu1, mu2, sd1, sd2, n, w)

            # apply noise
            noisy_y = sapply(y, function(i) jitter(i, amount=noise))

            # summarize data
            y_min = min(y)
            y_max = max(y)
            y_std = sd(y)
            y_lo = y_min - y_std
            y_hi = y_max + y_std
            y_dom = seq(y_lo, y_hi, length.out=n)

            # fit kernel density model on noisy data
            kde_model = kde(noisy_y, l, pts)
            kde_result = data.frame(x=kde_model$x, y=kde_model$y)

            # create plot
            p = ggplot(kde_result) +
              geom_line(aes(x=x, y=y, color='KDE')) +
              geom_line(
                data=data.frame(x=y, y=dnorm(y, mu1, sd1)),
                aes(x=x, y=y, color='left true')) +
              geom_line(
                data=data.frame(x=y, y=dnorm(y, mu2, sd2)),
                aes(x=x, y=y, color='right true')) +
              theme_bw() +
              labs(title=paste0("Bimodal",
                                  "N = ", n,
                                  ", noise = ", noise,
                                  ", bandwidth = ", l,
                                  ", points = ", pts))
            print(p)
        }
    }
}
```

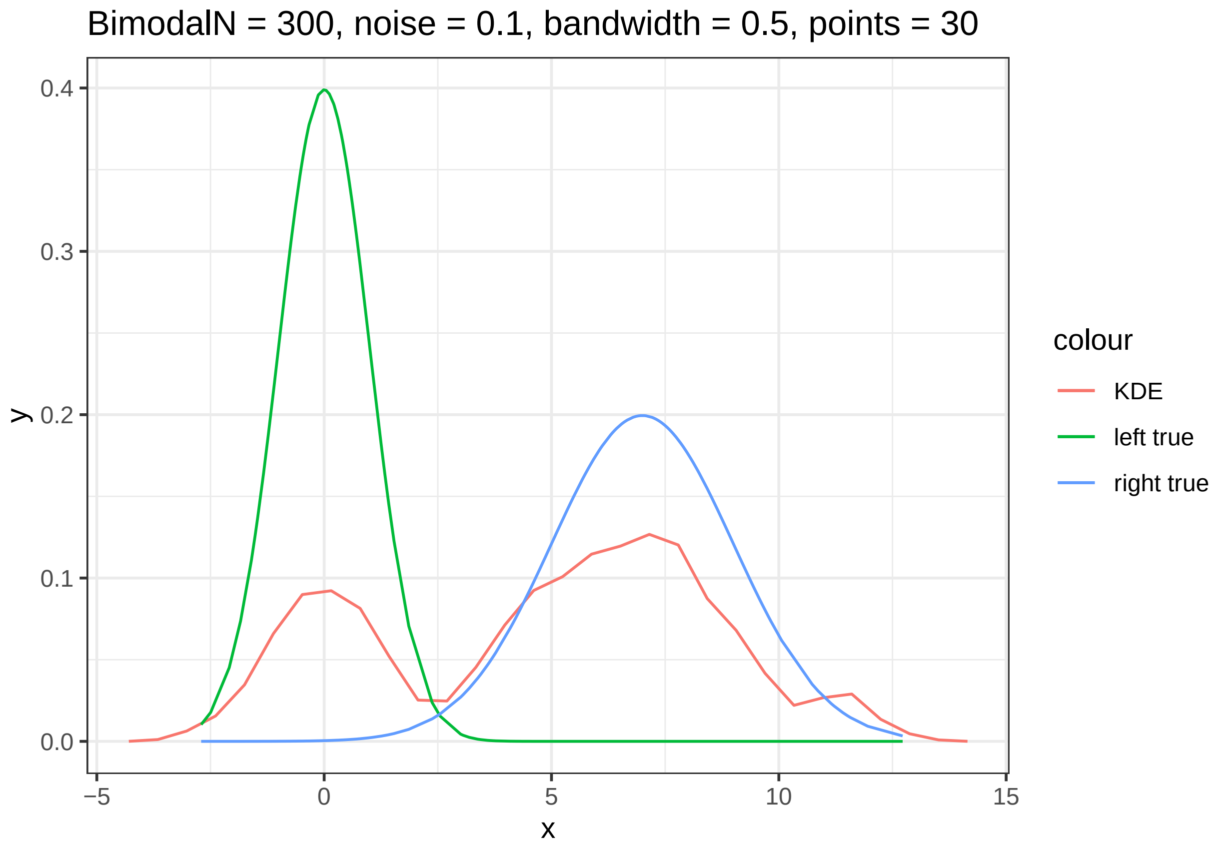


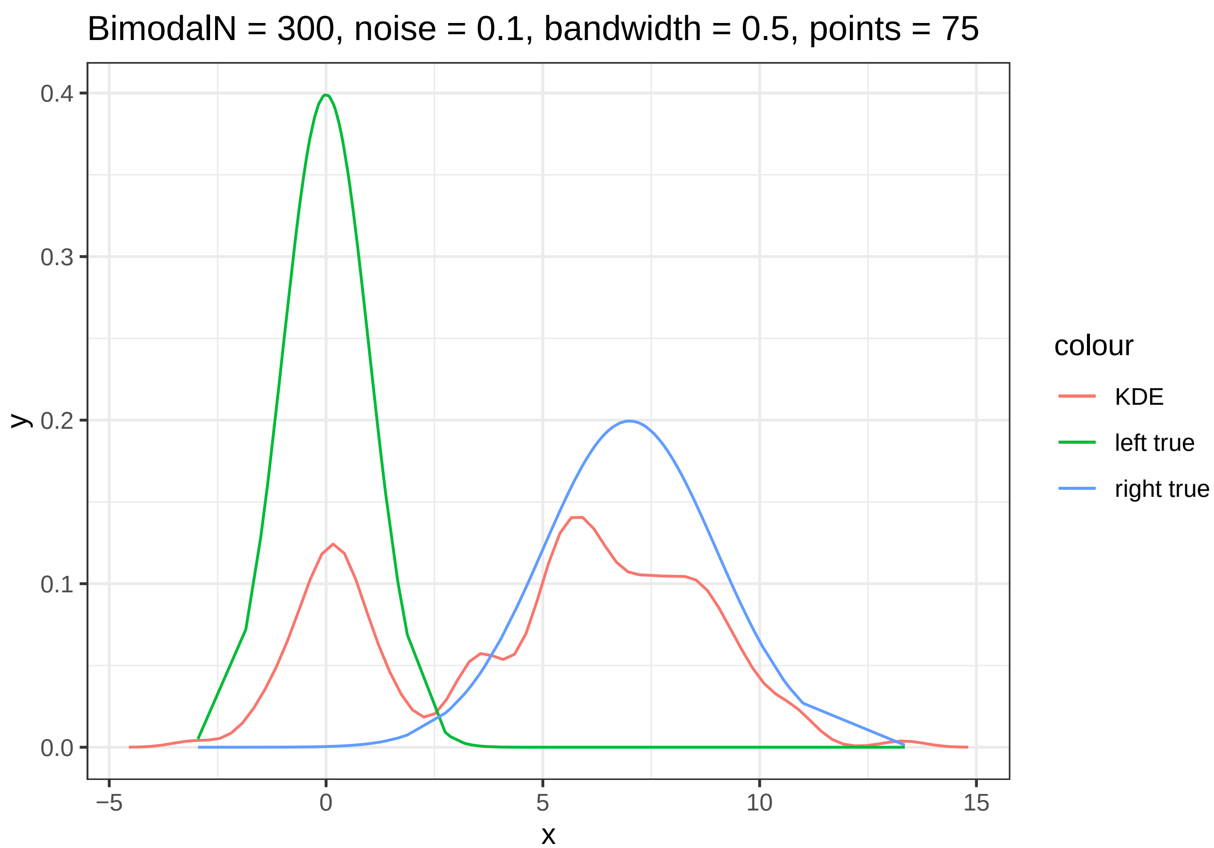
BimodalN = 300, noise = 0.1, bandwidth = 0.2, points = 30
0.4
0.3
0.2
0.1
0.0
y
0
5
10
x
colour
KDE
left true
right true



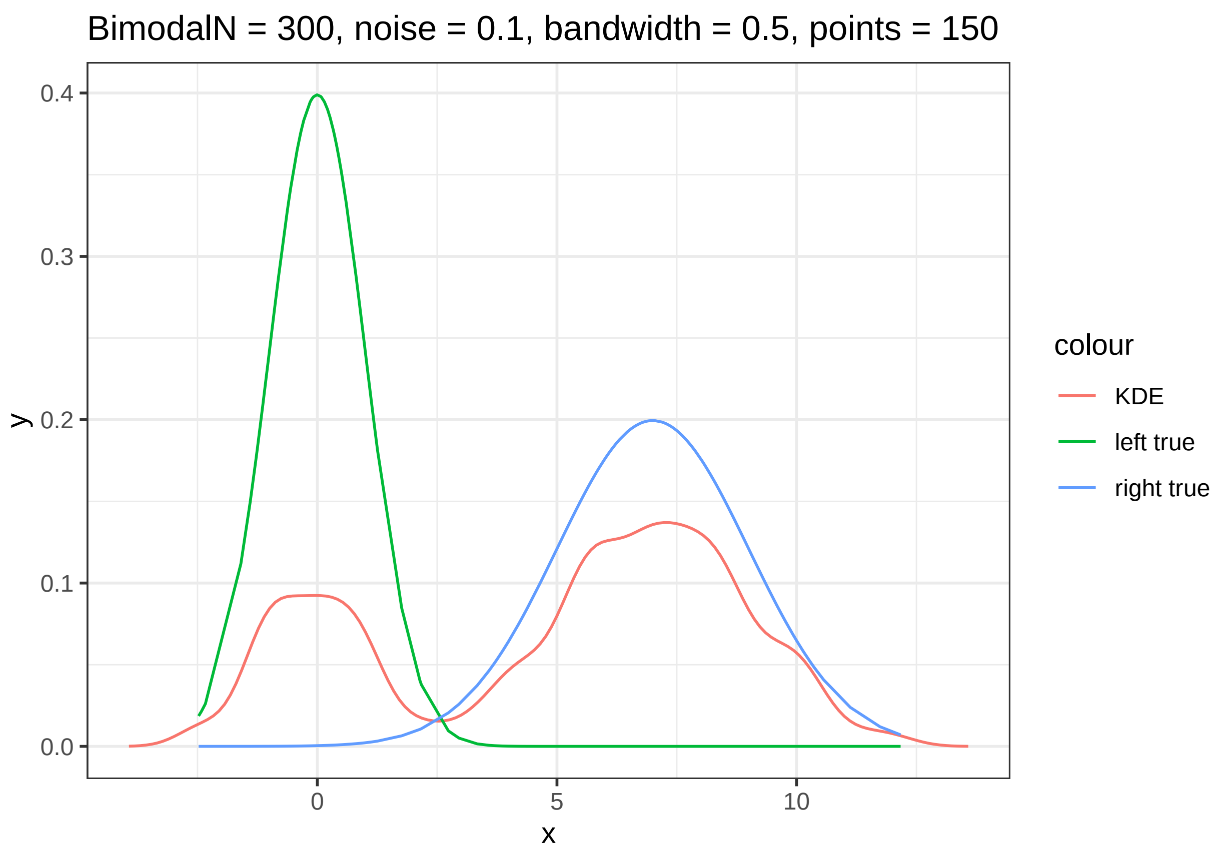
BimodalN = 300, noise = 0.1, bandwidth = 0.2, points = 75
y
0.4
0.3
0.2
0.1
0.0
0
5
10
15
x
colour
KDE
left true
right true

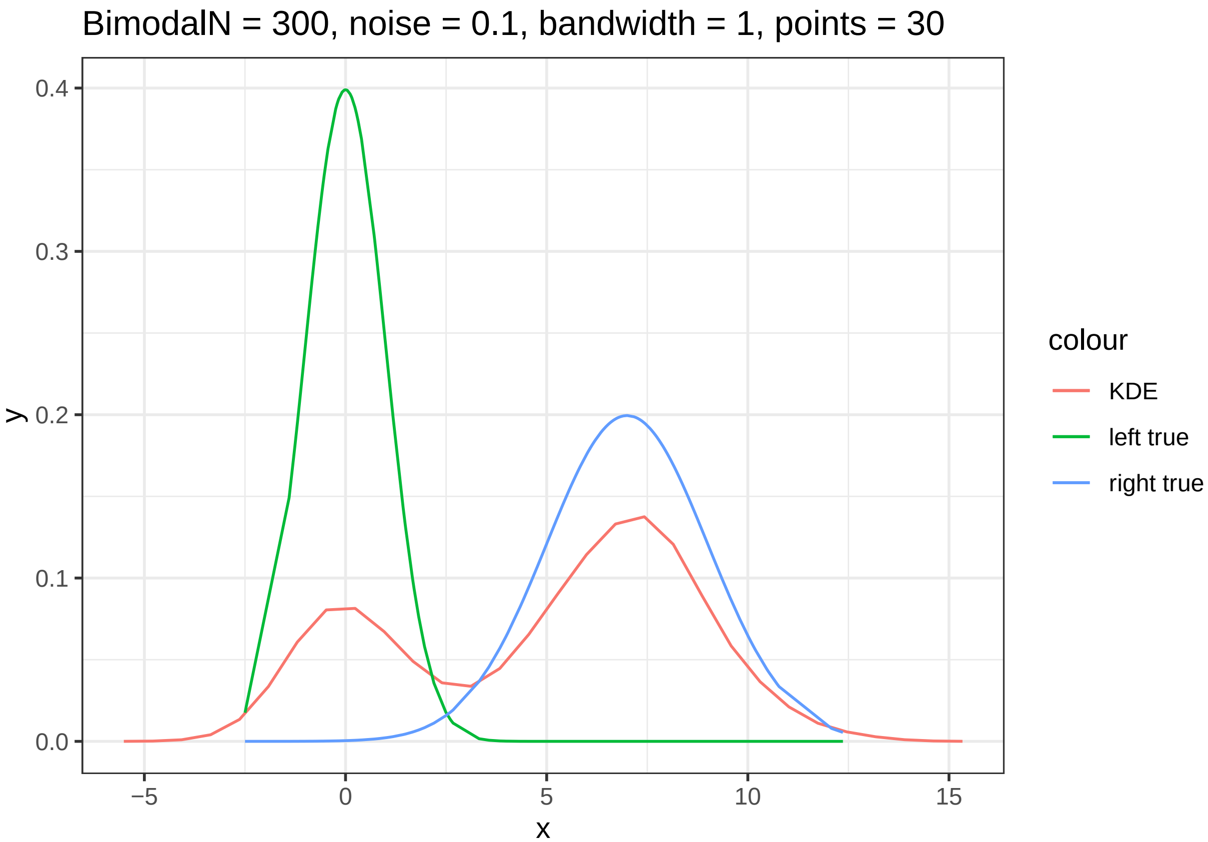

BimodalN = 300, noise = 0.1, bandwidth = 0.2, points = 150
y
0.0
0.1
0.2
0.3
0.4
x
0
5
10
colour
KDE
left true
right true

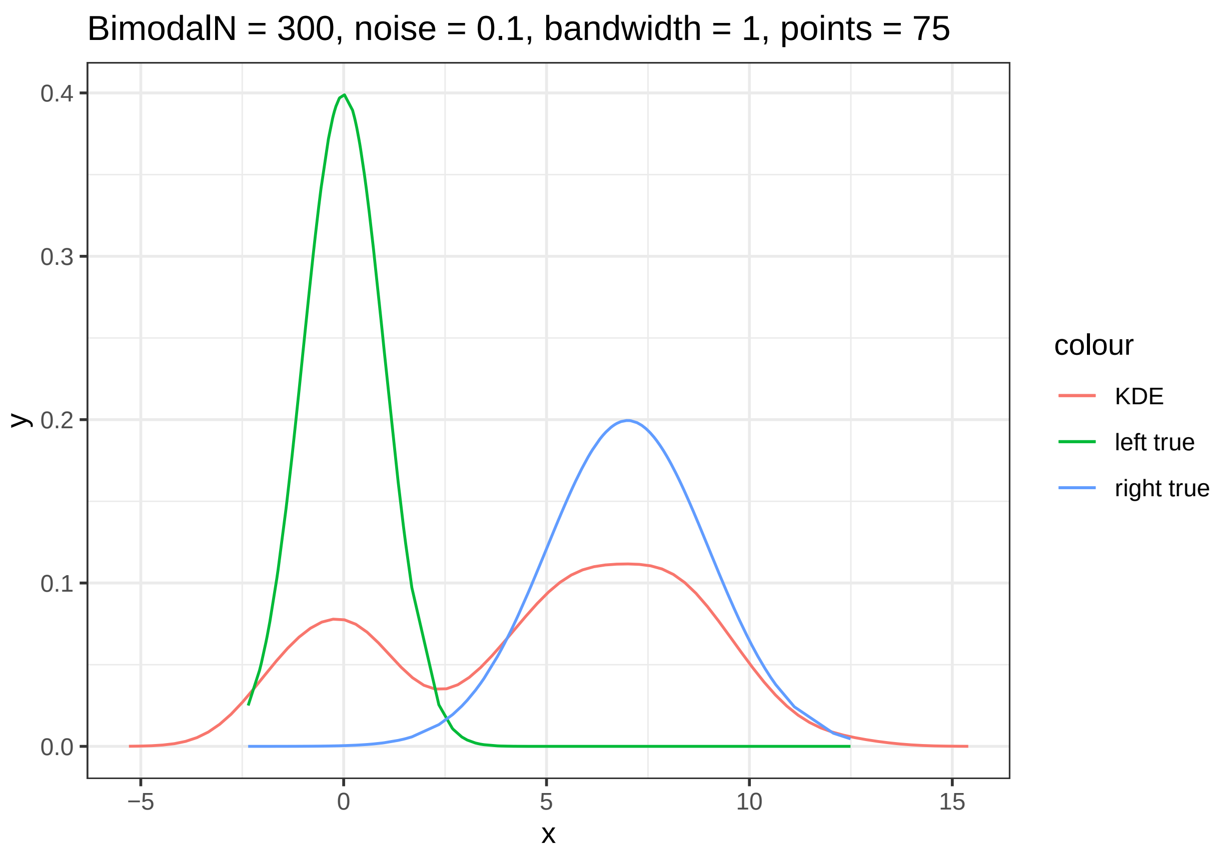

BimodalN = 300, noise = 0.1, bandwidth = 0.5, points = 30
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true

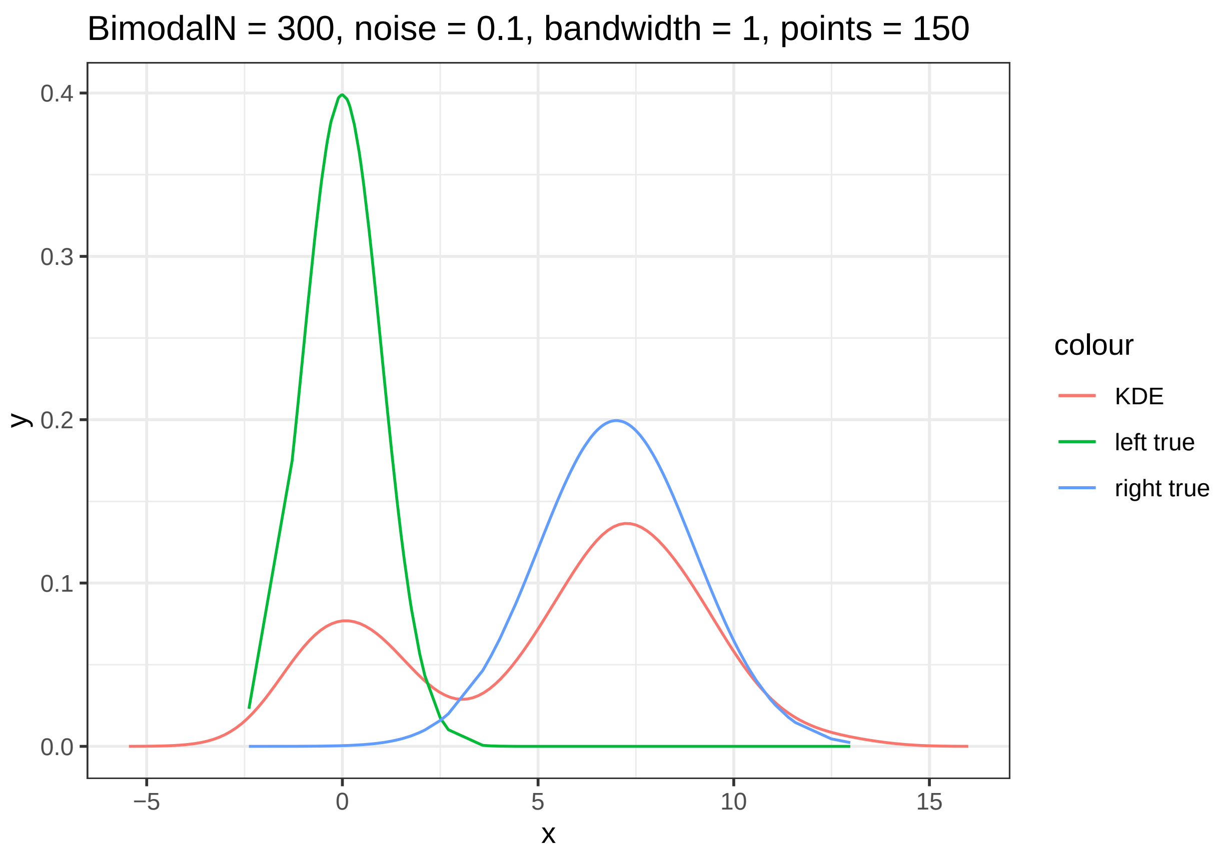


BimodalN = 300, noise = 0.1, bandwidth = 0.5, points = 75
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true

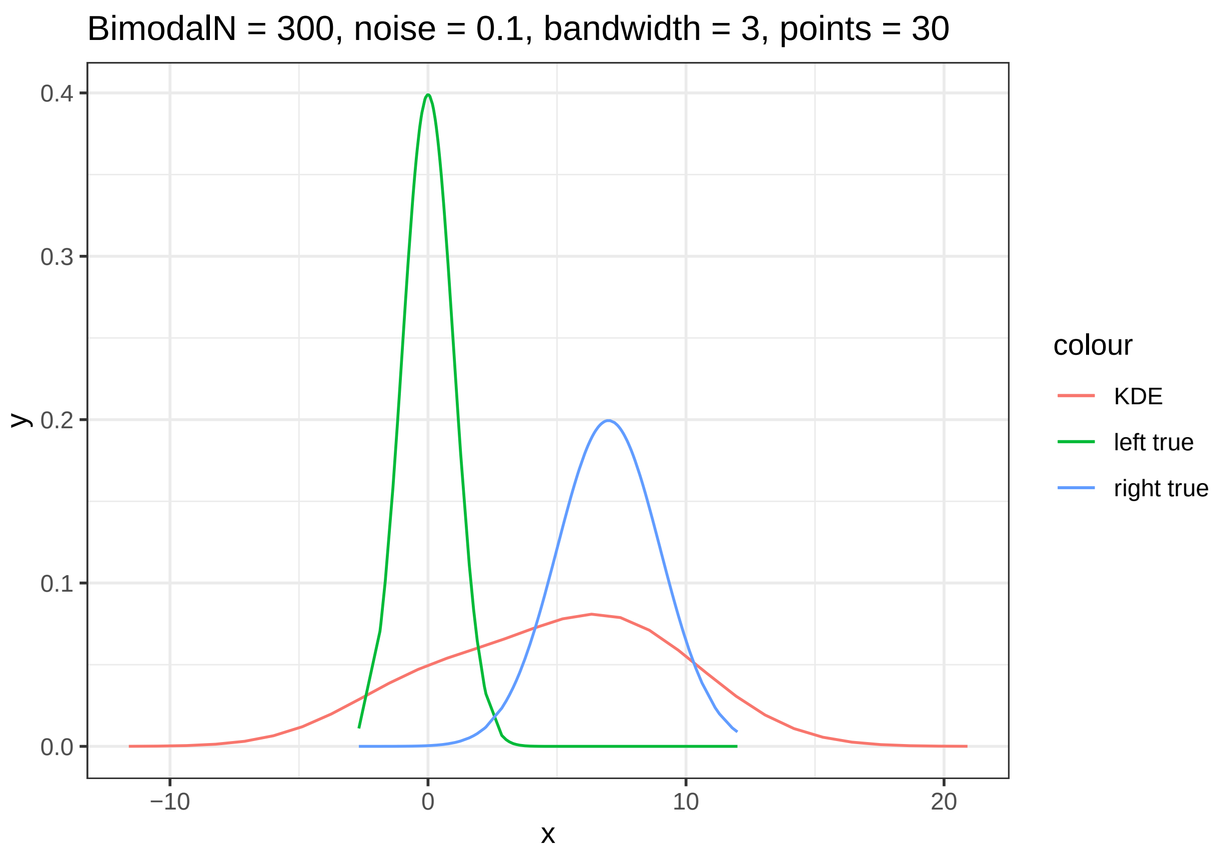

BimodalN = 300, noise = 0.1, bandwidth = 0.5, points = 150
0.4
0.3
0.2
0.1
0.0
y
0
5
10
x
colour
KDE
left true
right true

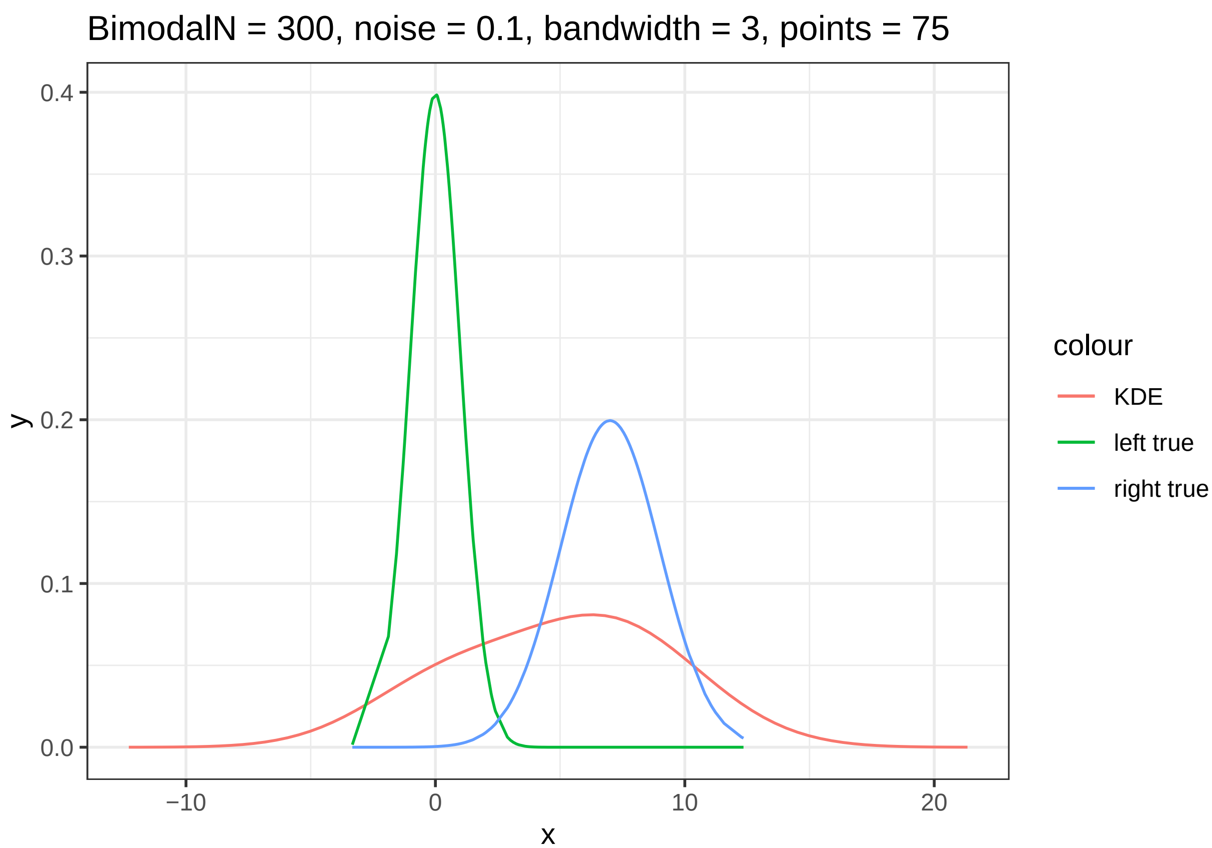

BimodalN = 300, noise = 0.1, bandwidth = 1, points = 30
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true

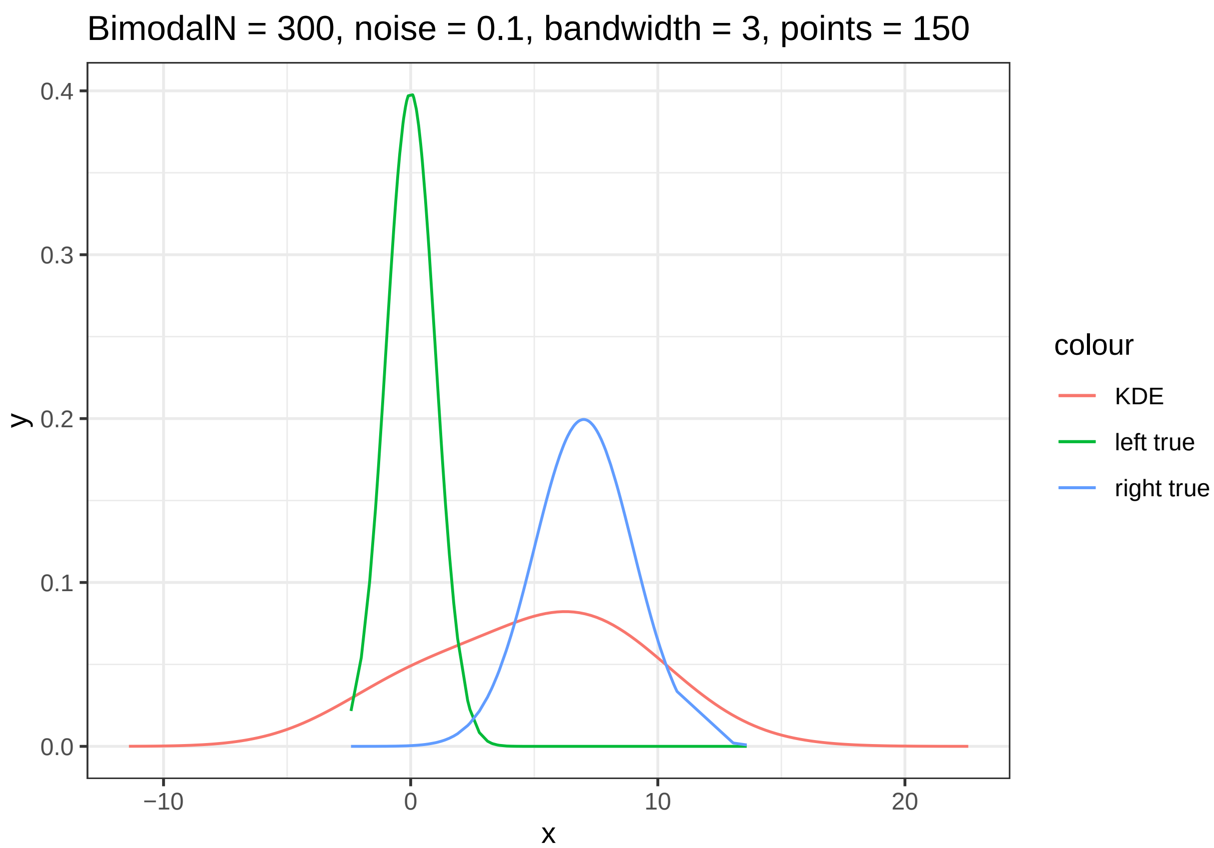

BimodalN = 300, noise = 0.1, bandwidth = 1, points = 75
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true

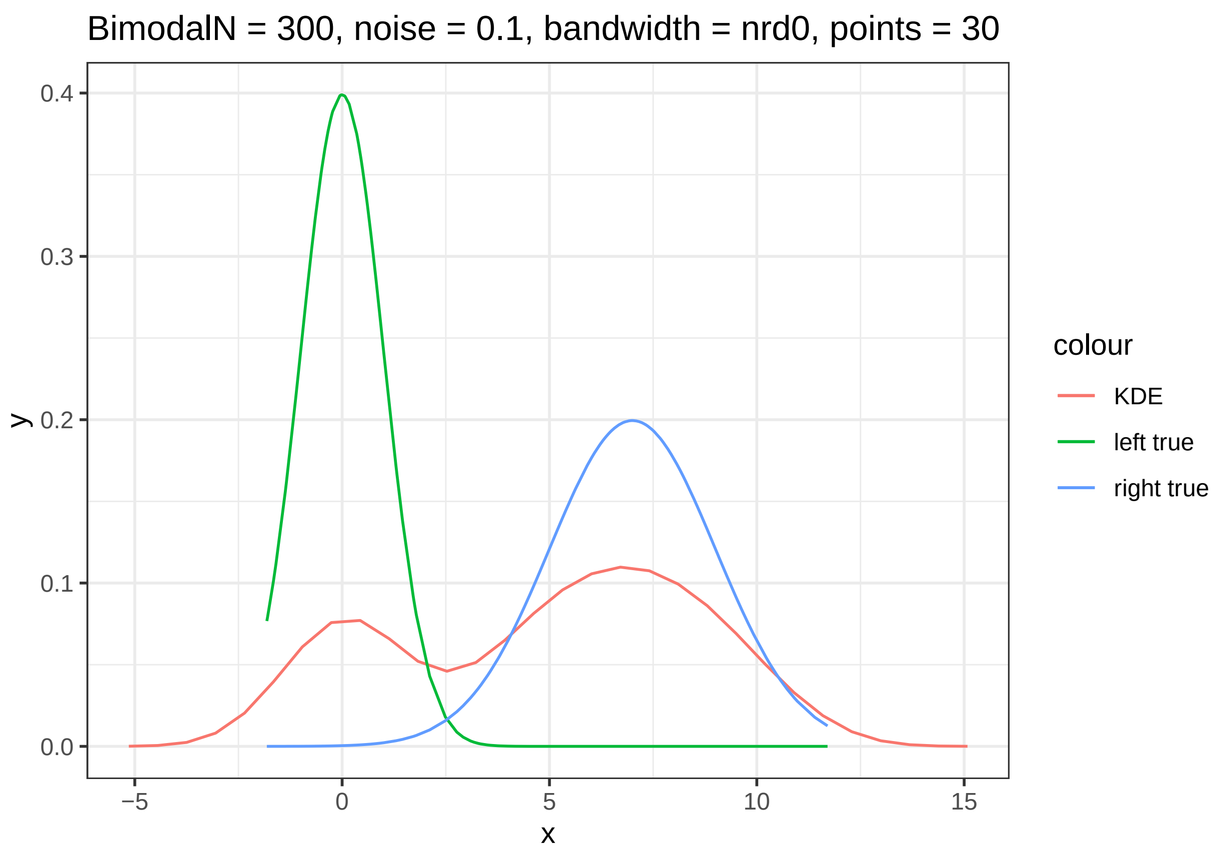


BimodalN = 300, noise = 0.1, bandwidth = 1, points = 150
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true

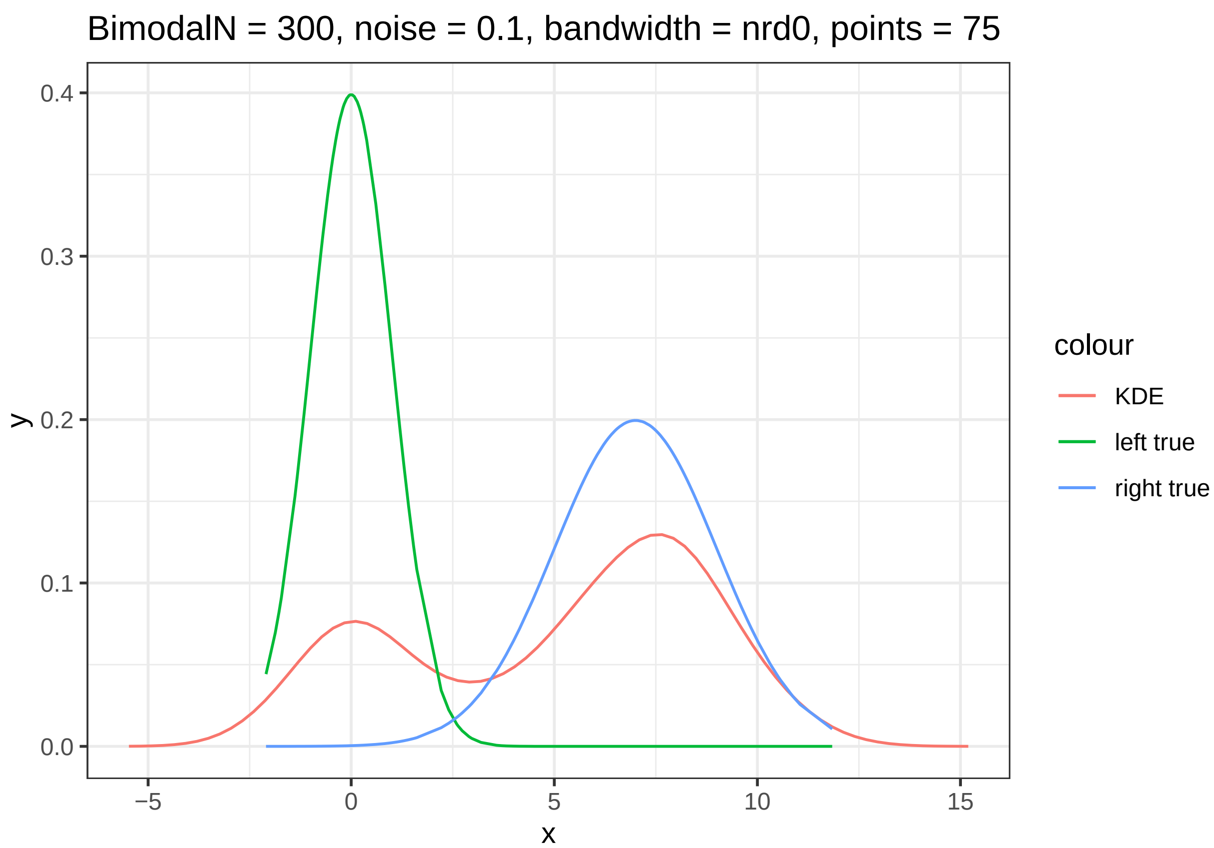

BimodalN = 300, noise = 0.1, bandwidth = 3, points = 30
0.4
0.3
0.2
0.1
0.0
y
−10
0
10
20
x
colour
KDE
left true
right true

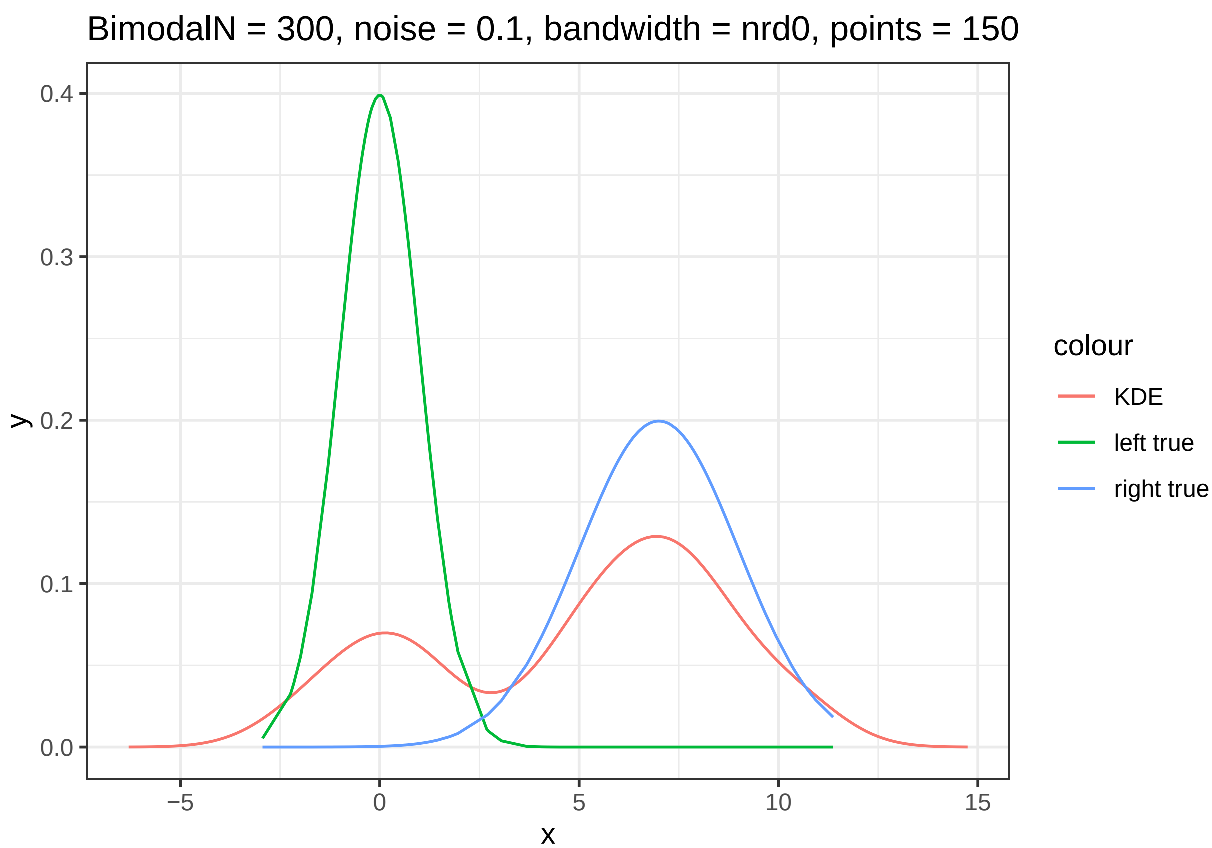

BimodalN = 300, noise = 0.1, bandwidth = 3, points = 75
0.4
0.3
0.2
0.1
0.0
y
−10
0
10
20
x
colour
KDE
left true
right true

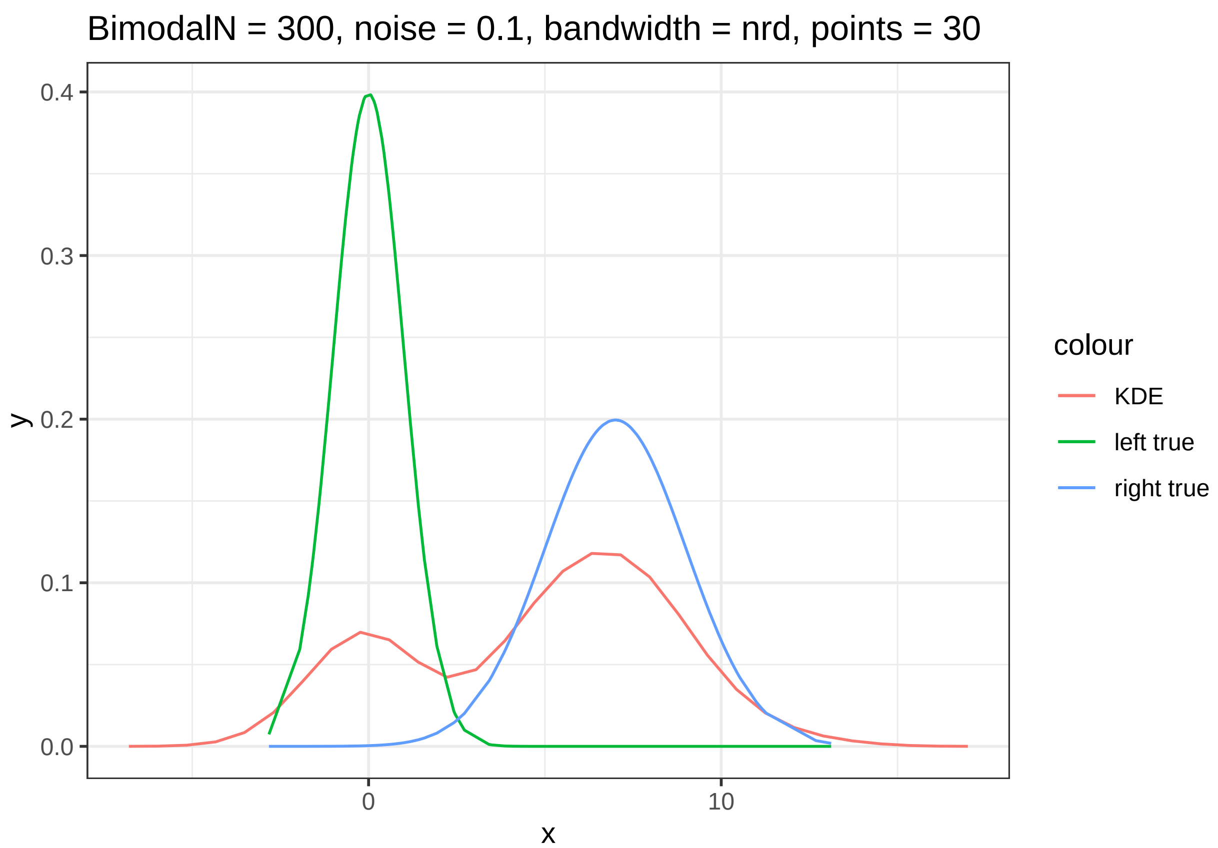

BimodalN = 300, noise = 0.1, bandwidth = 3, points = 150
y
0.0
0.1
0.2
0.3
0.4
−10
0
10
20
x
colour
KDE
left true
right true

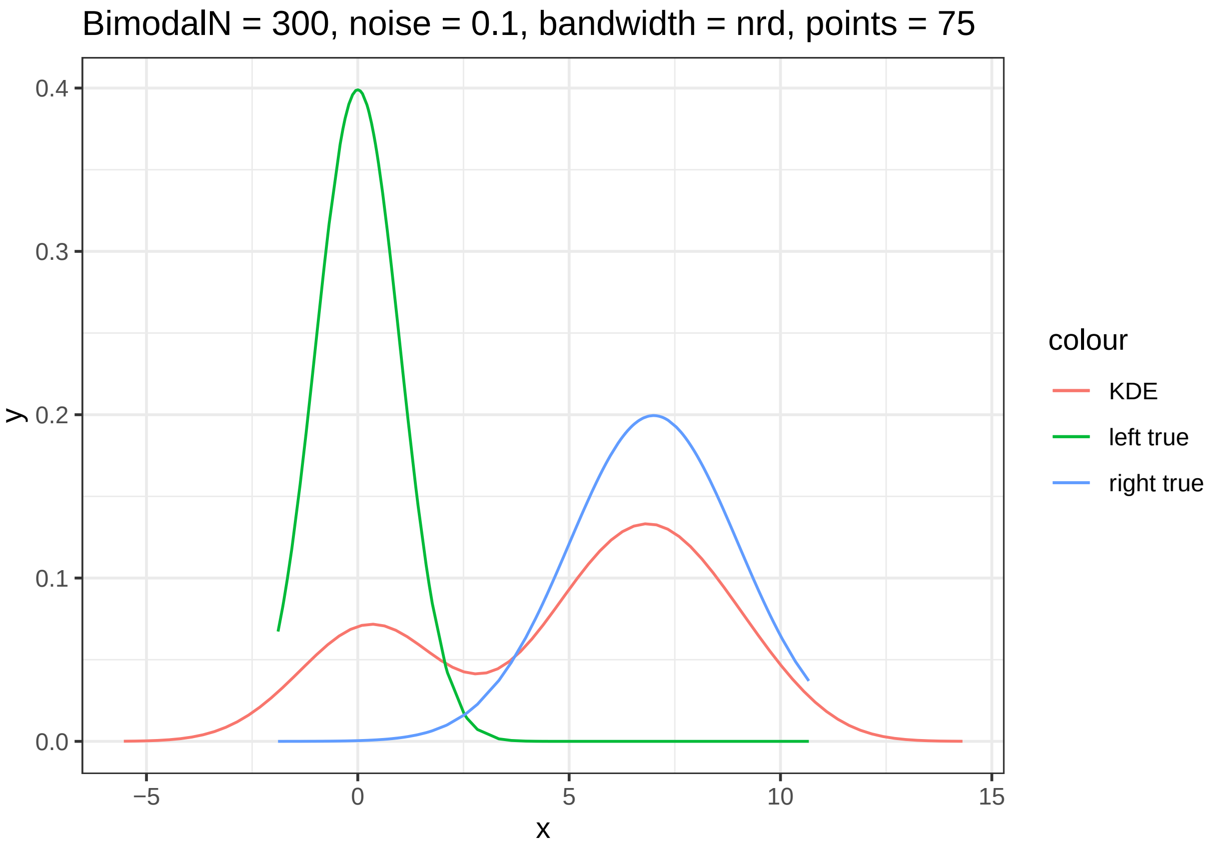


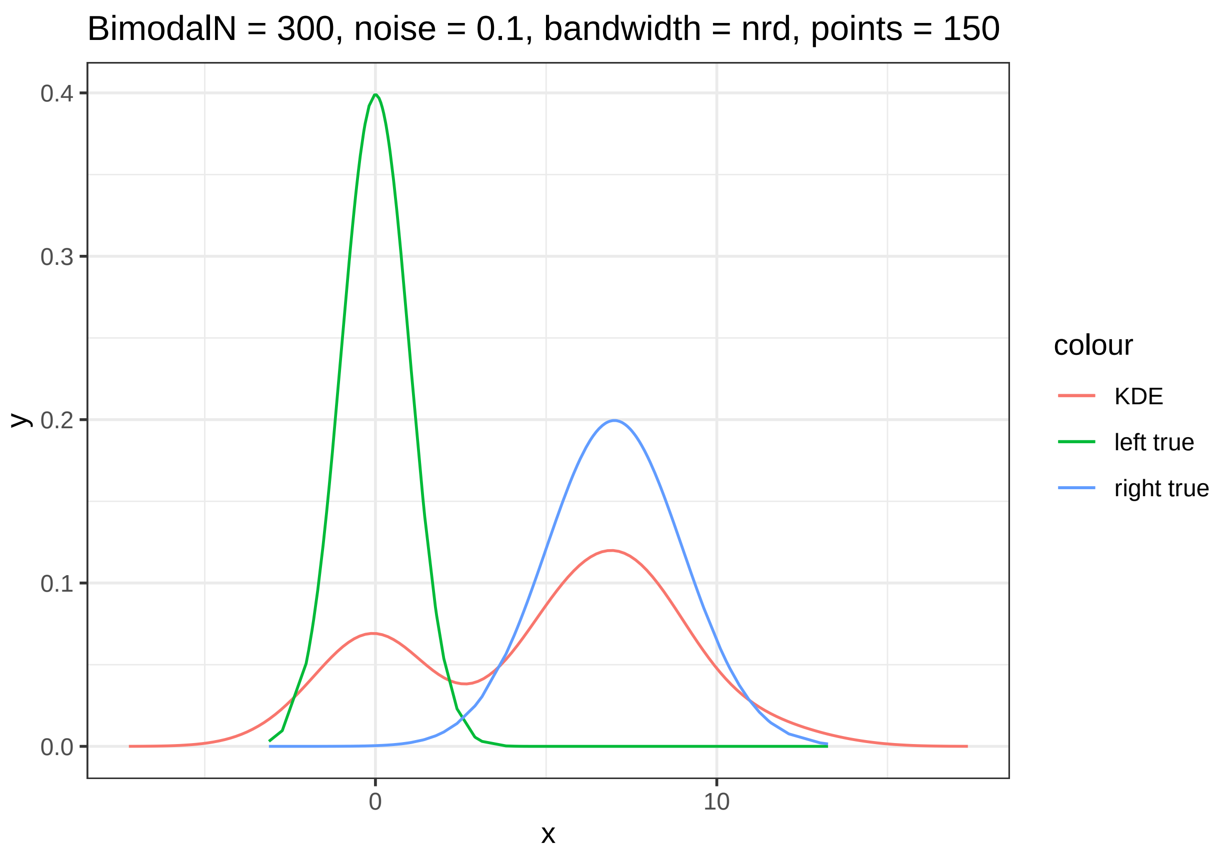
BimodalN = 300, noise = 0.1, bandwidth = nrd0, points = 30
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true


BimodalN = 300, noise = 0.1, bandwidth = nrd0, points = 75
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true

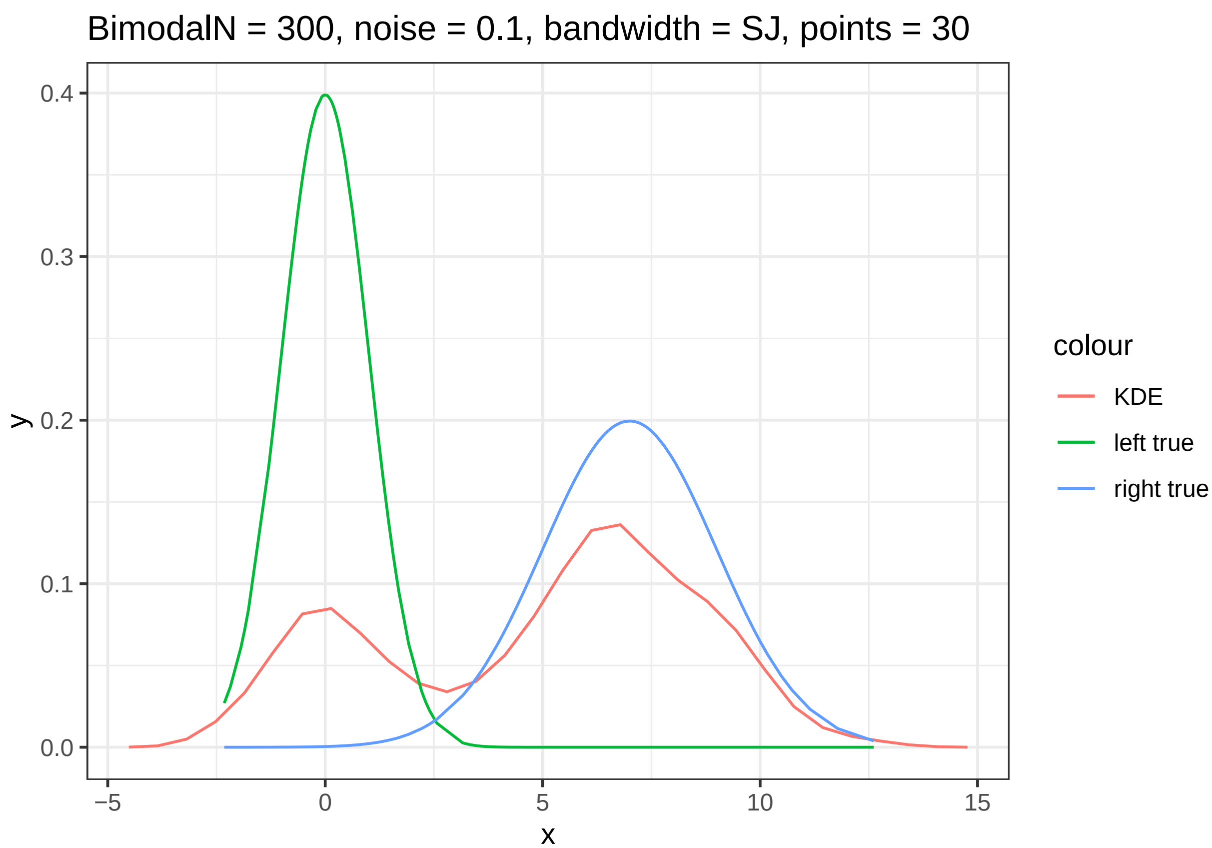

BimodalN = 300, noise = 0.1, bandwidth = nrd0, points = 150
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true


BimodalN = 300, noise = 0.1, bandwidth = nrd, points = 30
0.4
0.3
0.2
0.1
0.0
y
0
10
x
colour
KDE
left true
right true

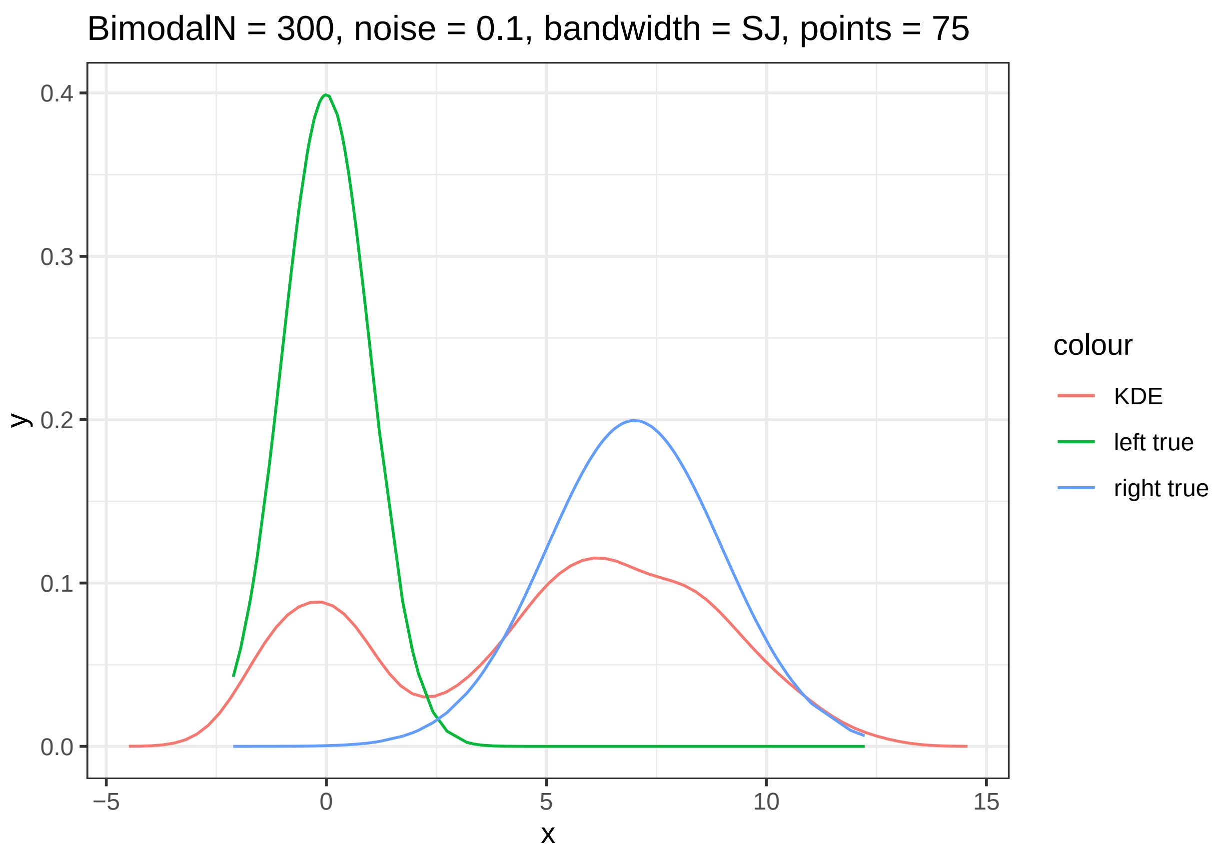

BimodalN = 300, noise = 0.1, bandwidth = nrd, points = 75
y
0.4
0.3
0.2
0.1
0.0
−5
0
5
10
15
x
colour
KDE
left true
right true



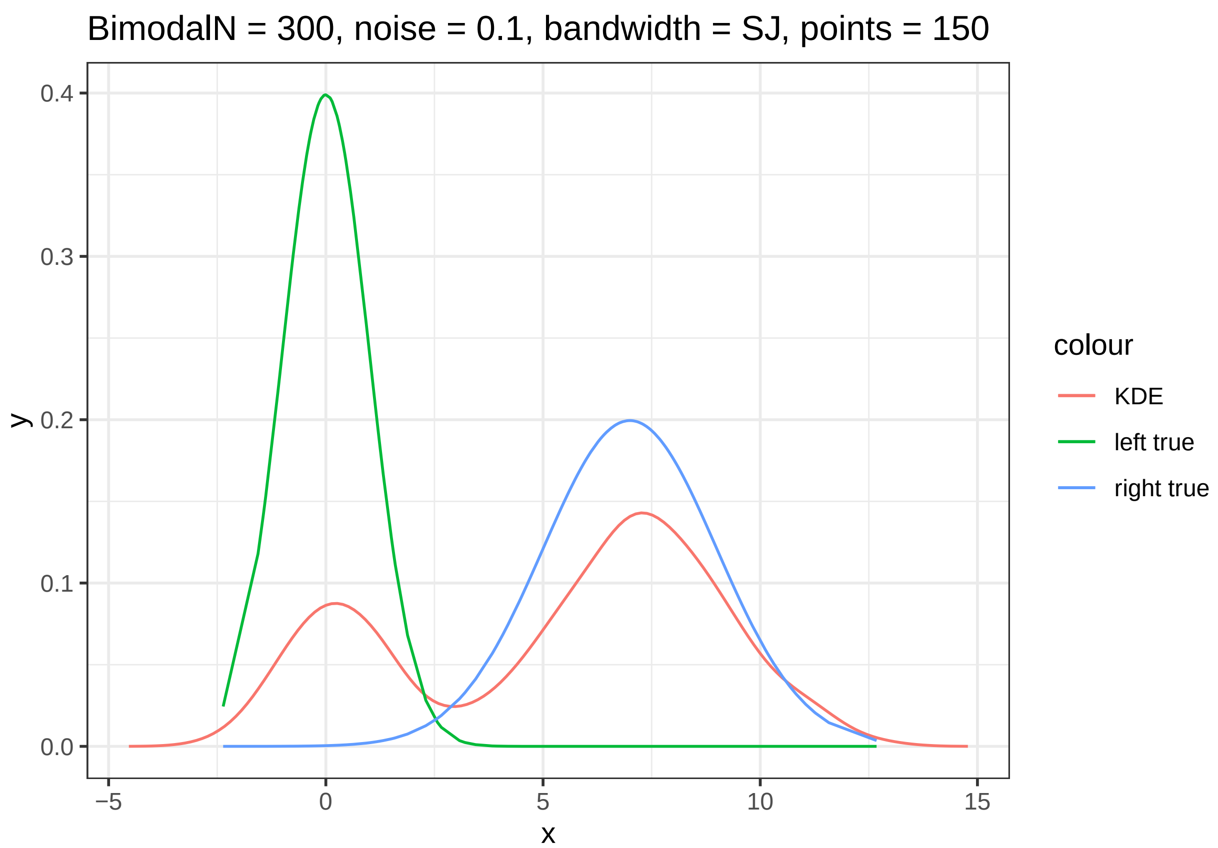
BimodalN = 300, noise = 0.1, bandwidth = nrd, points = 150
0.4
0.3
0.2
0.1
0.0
y
0
10
x
colour
KDE
left true
right true

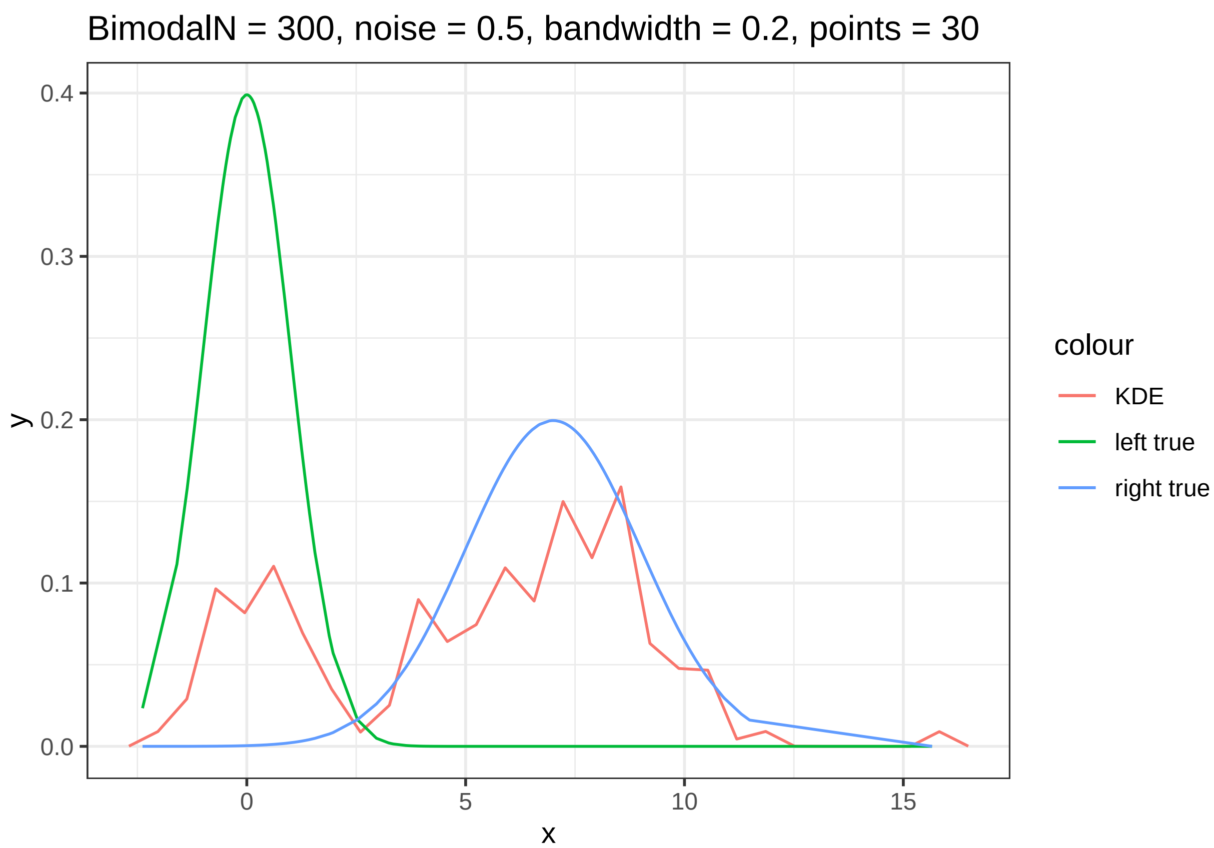


BimodalN = 300, noise = 0.1, bandwidth = SJ, points = 30
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true

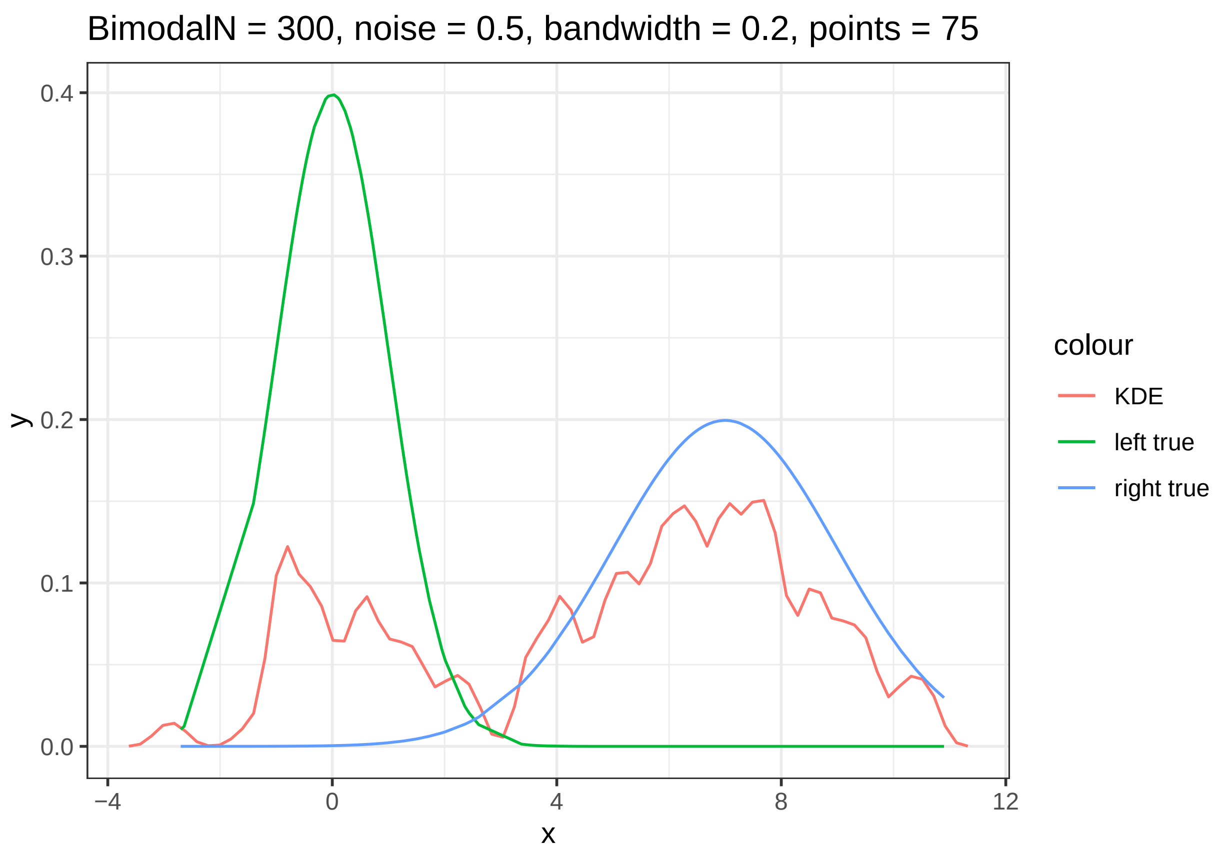


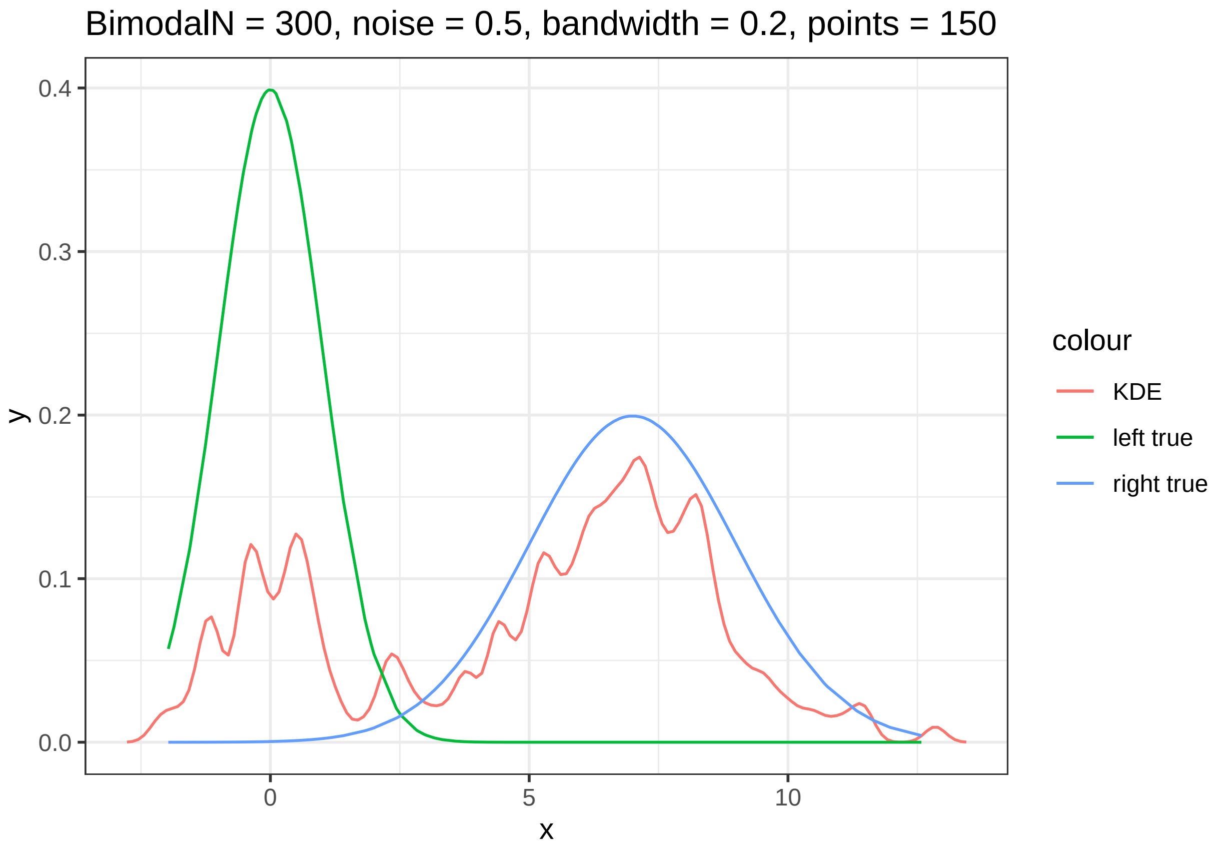
BimodalN = 300, noise = 0.1, bandwidth = SJ, points = 75
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true

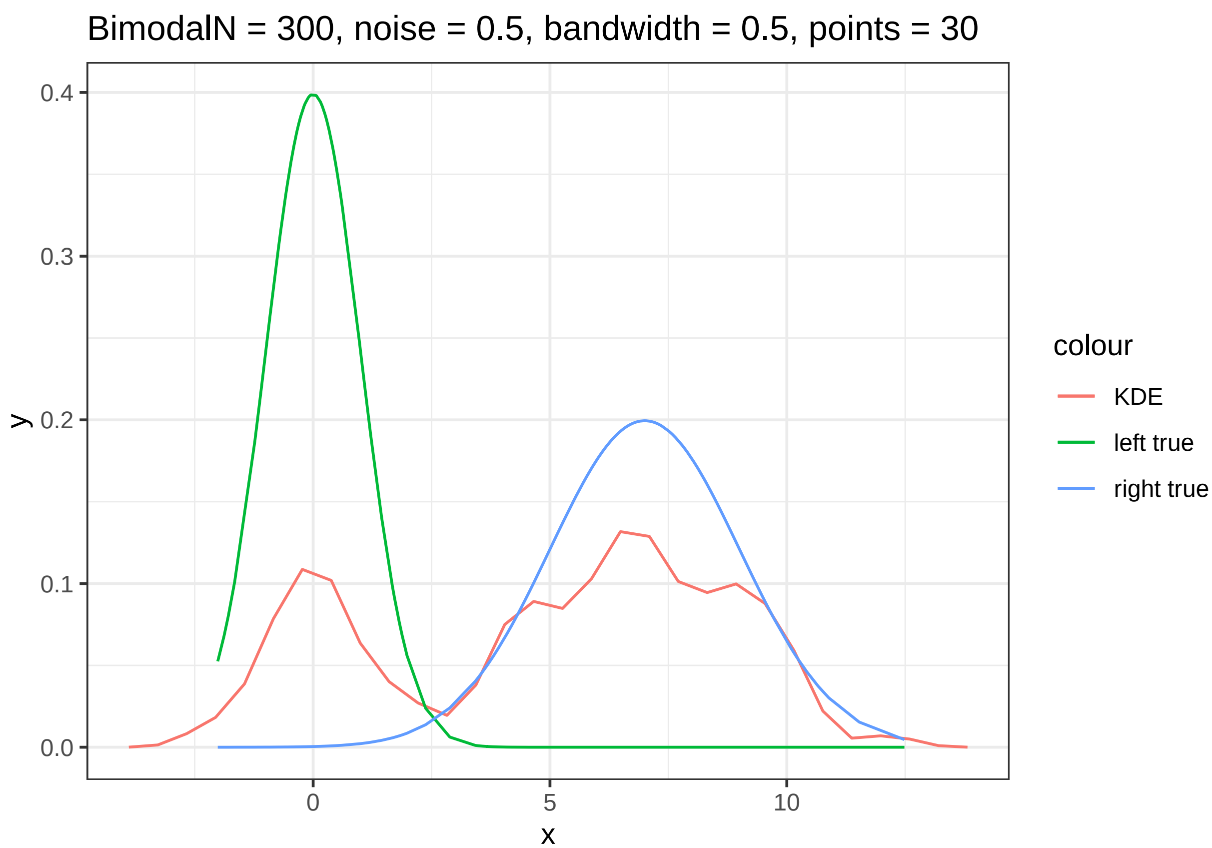

BimodalN = 300, noise = 0.1, bandwidth = SJ, points = 150
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true

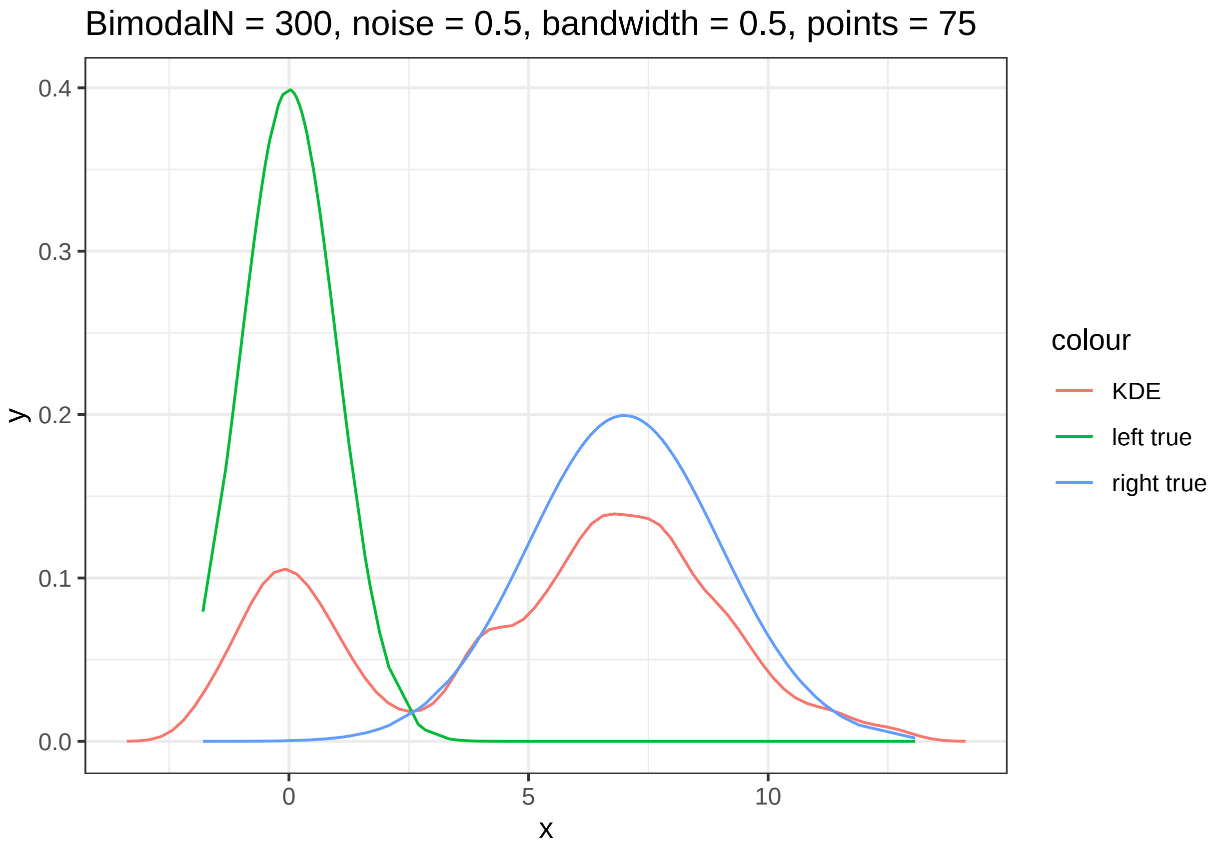


BimodalN = 300, noise = 0.5, bandwidth = 0.2, points = 30
0.4
0.3
0.2
0.1
0.0
y
0
5
10
15
x
colour
KDE
left true
right true

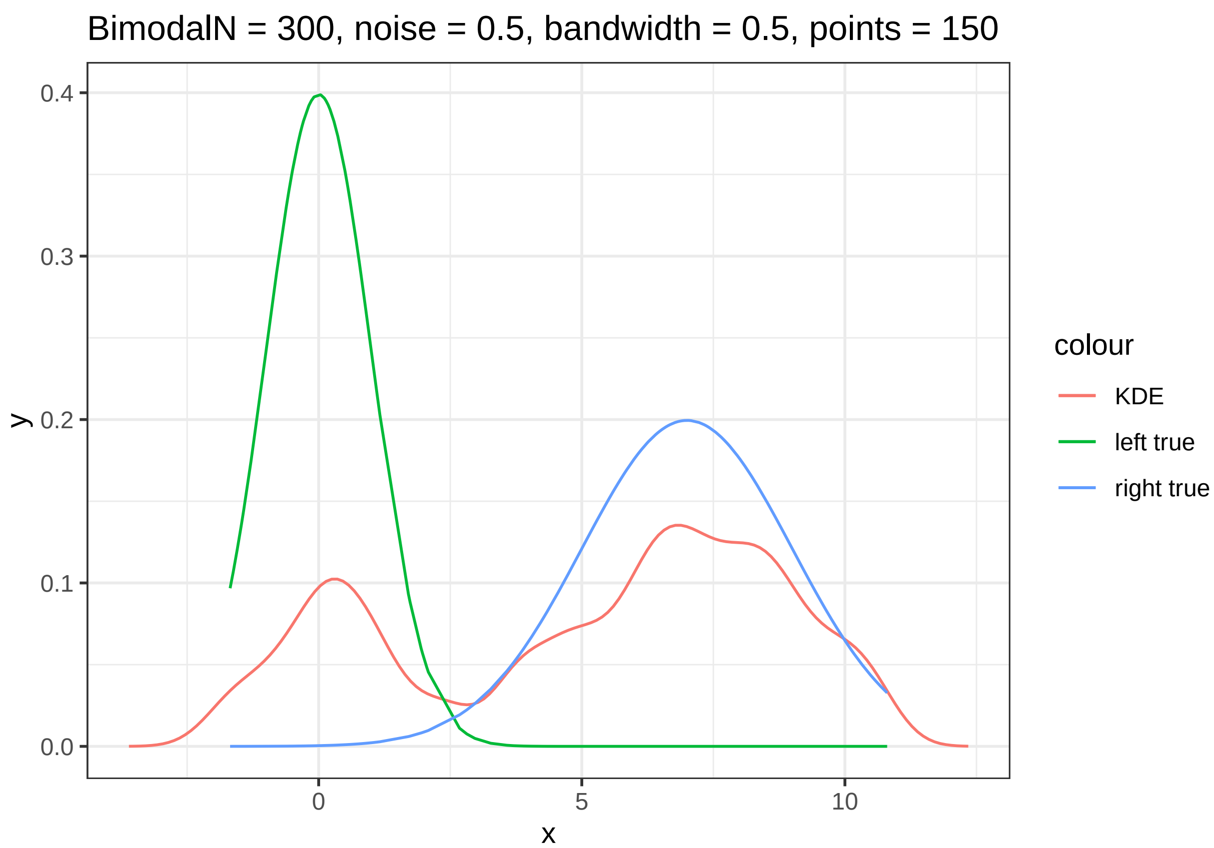

BimodalN = 300, noise = 0.5, bandwidth = 0.2, points = 75
y
0.0
0.1
0.2
0.3
0.4
x
−4
0
4
8
12
colour
KDE
left true
right true

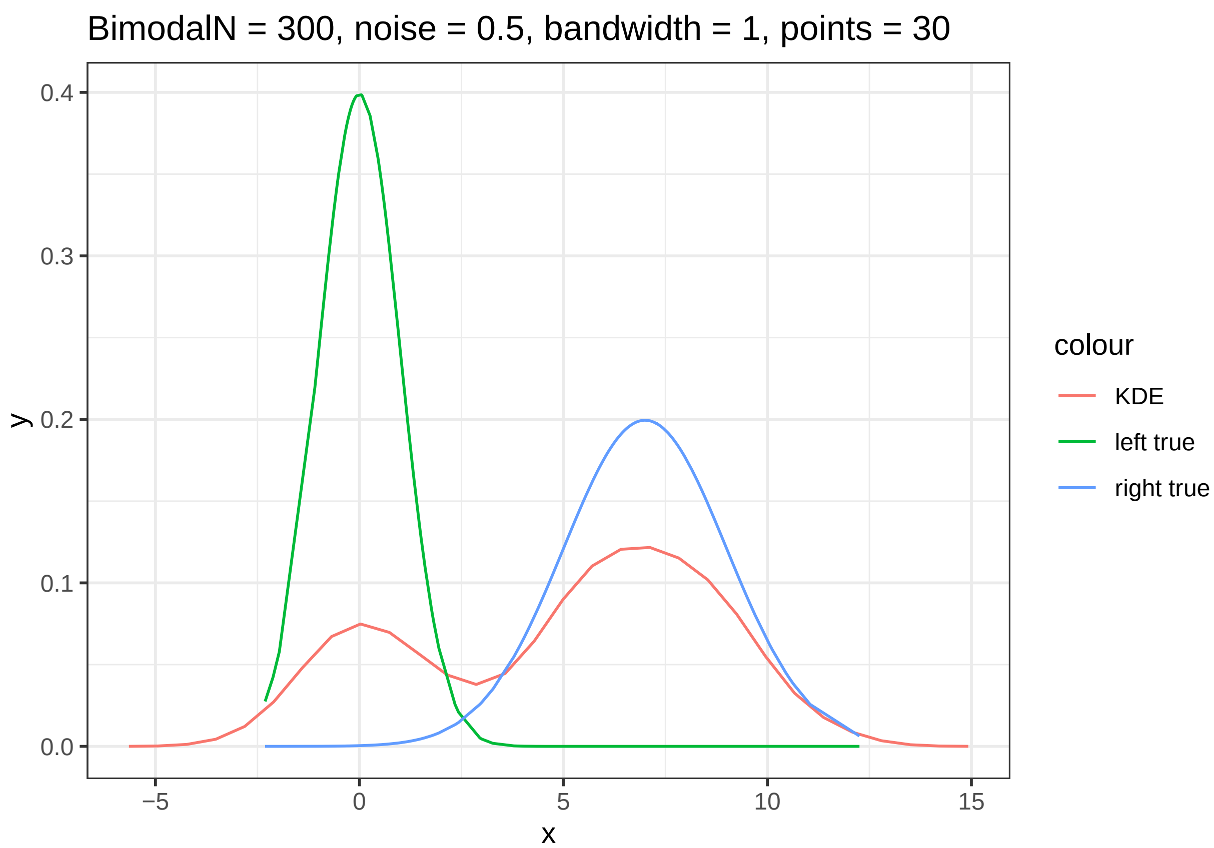

BimodalN = 300, noise = 0.5, bandwidth = 0.2, points = 150
0.4
0.3
0.2
0.1
0.0
y
0
5
10
x
colour
KDE
left true
right true

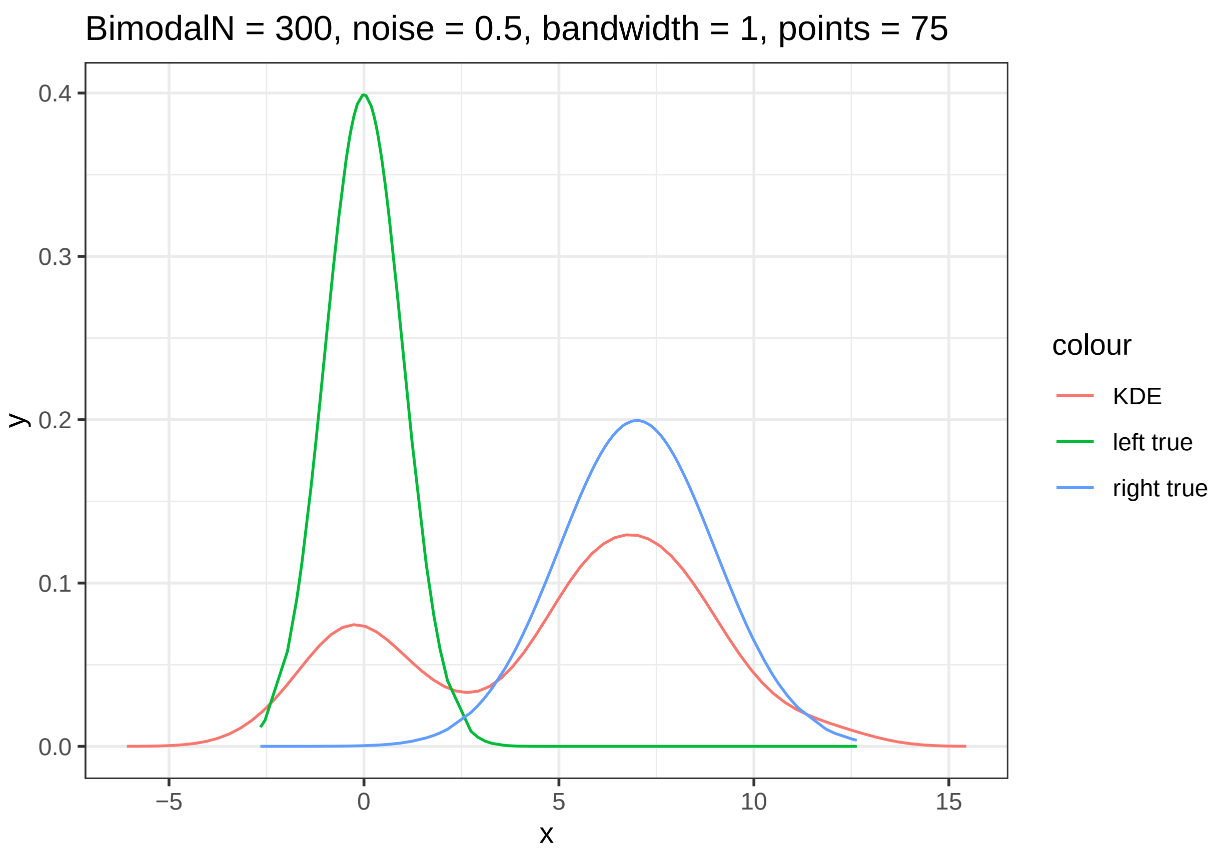

BimodalN = 300, noise = 0.5, bandwidth = 0.5, points = 30
0.4
0.3
0.2
0.1
0.0
y
0
5
10
x
colour
KDE
left true
right true



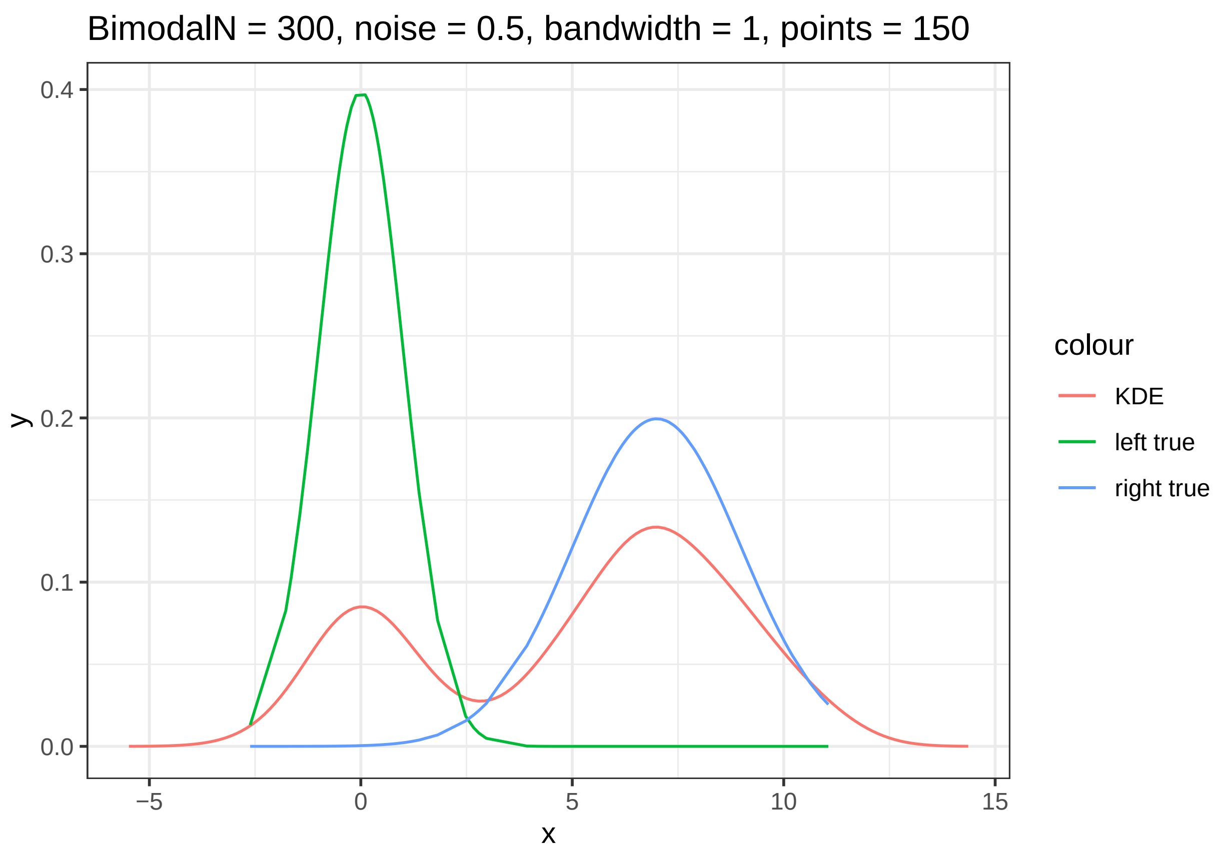
BimodalN = 300, noise = 0.5, bandwidth = 0.5, points = 75
0.4
0.3
0.2
0.1
0.0
y
0
5
10
x
colour
KDE
left true
right true


BimodalN = 300, noise = 0.5, bandwidth = 0.5, points = 150
0.4
0.3
0.2
0.1
0.0
y
0
5
10
x
colour
KDE
left true
right true

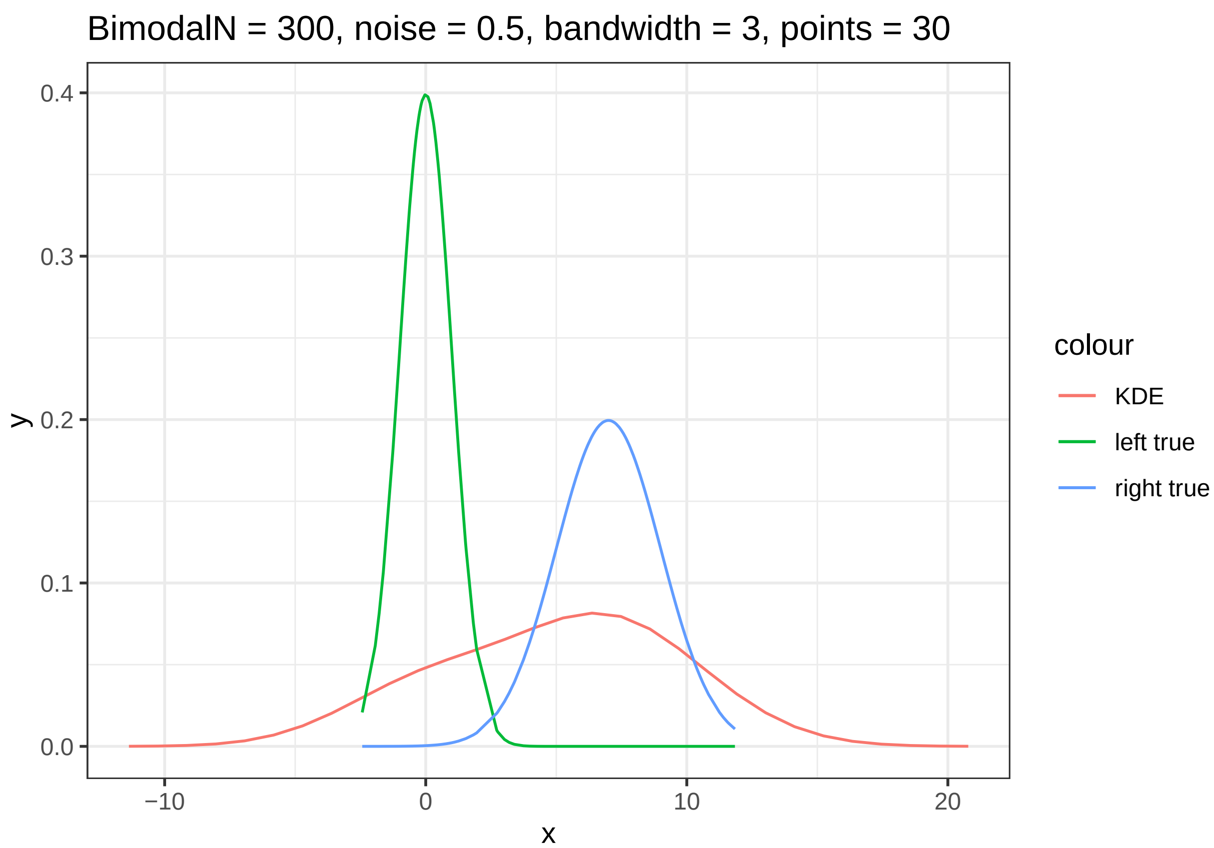

BimodalN = 300, noise = 0.5, bandwidth = 1, points = 30
y
0.0
0.1
0.2
0.3
0.4
−5
0
5
10
15
x
colour
KDE
left true
right true



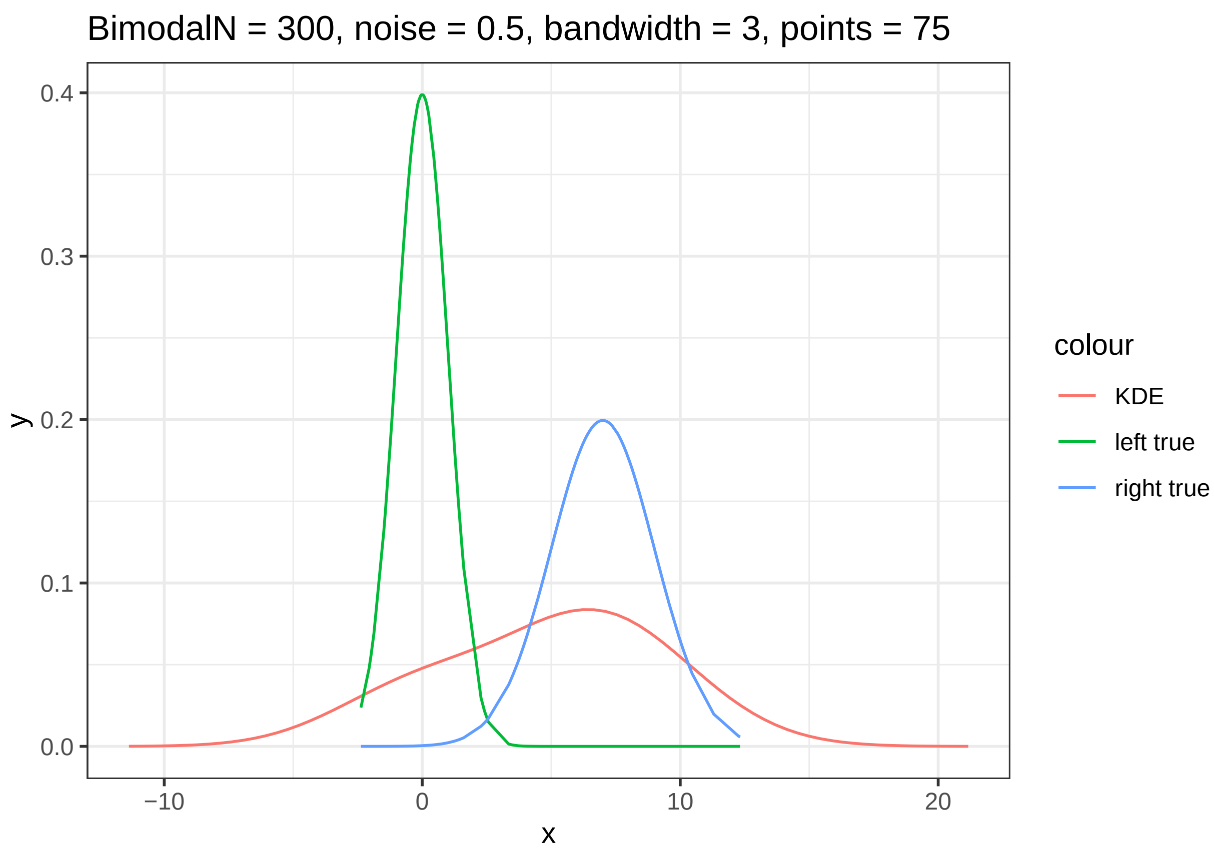
BimodalN = 300, noise = 0.5, bandwidth = 1, points = 75
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true

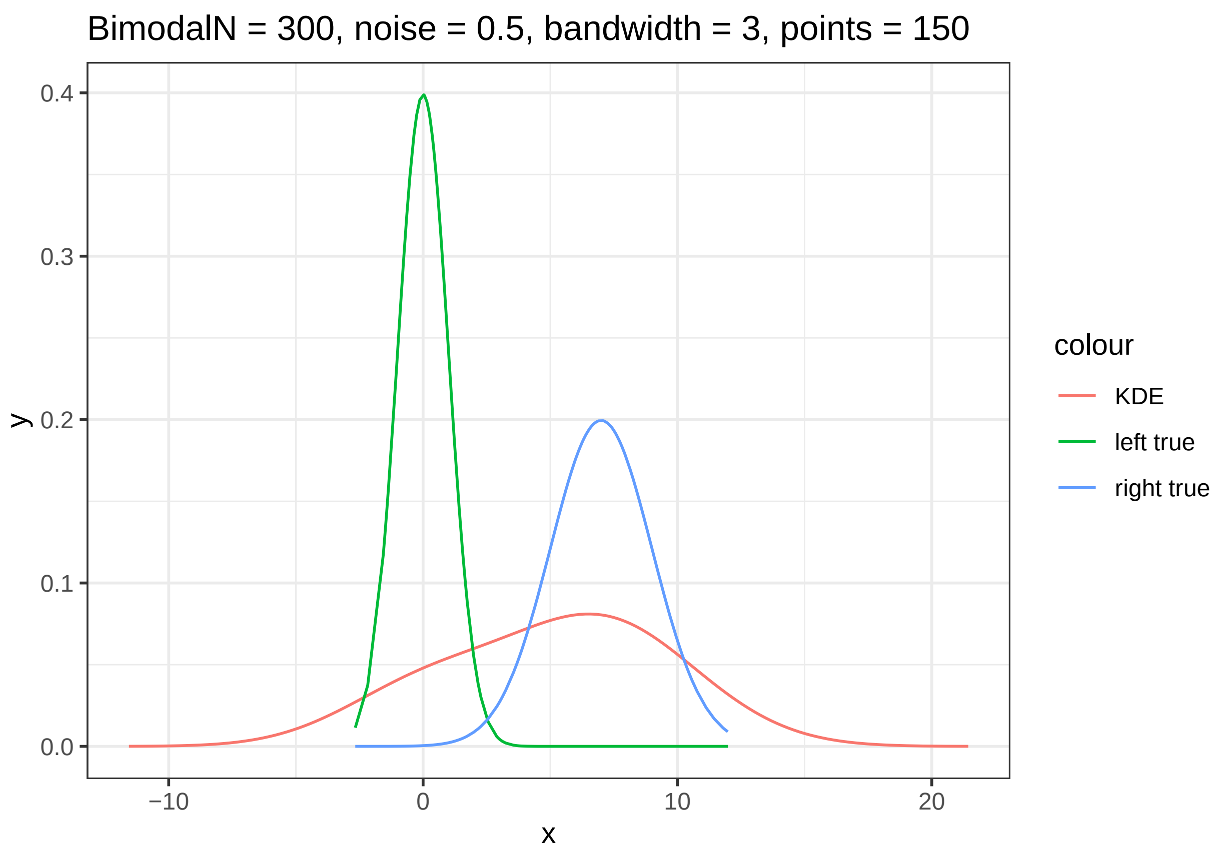


BimodalN = 300, noise = 0.5, bandwidth = 1, points = 150
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true

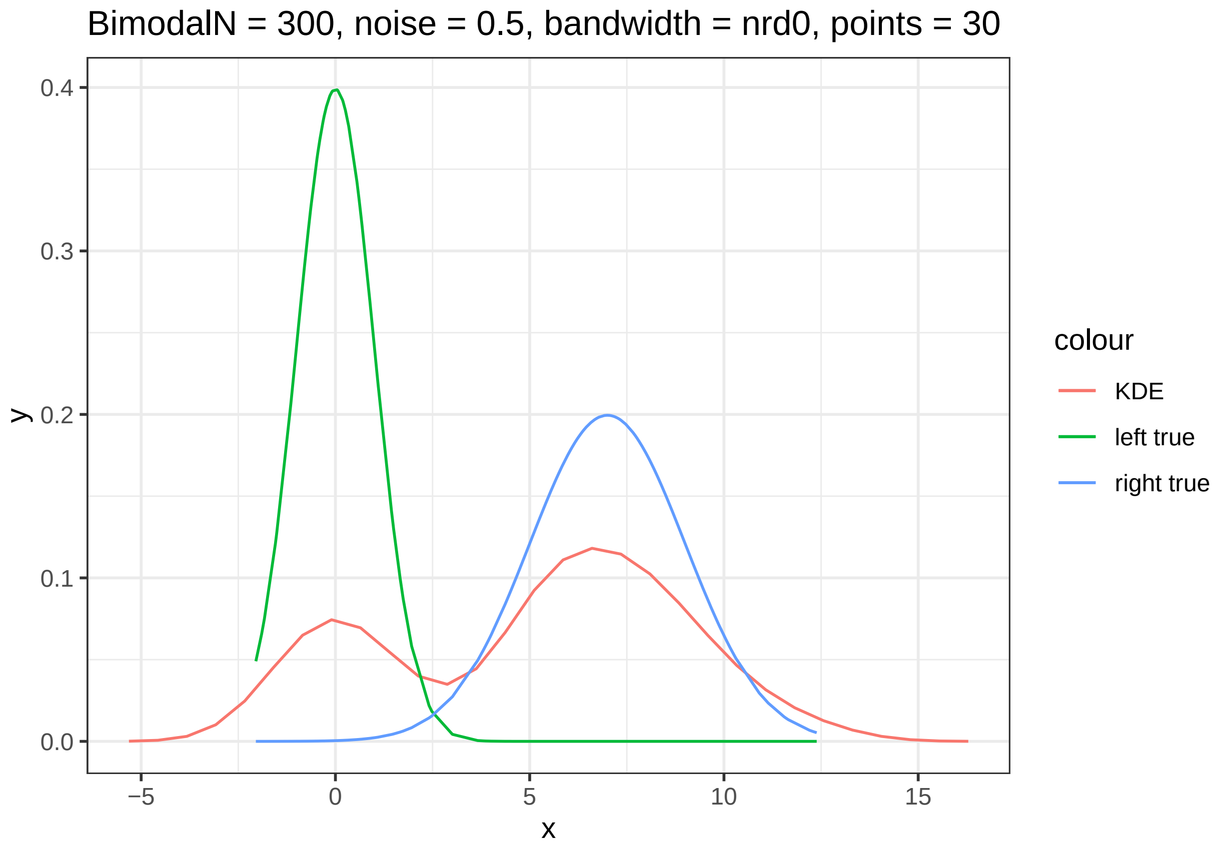


BimodalN = 300, noise = 0.5, bandwidth = 3, points = 30
0.4
0.3
0.2
0.1
0.0
y
−10
0
10
20
x
colour
KDE
left true
right true

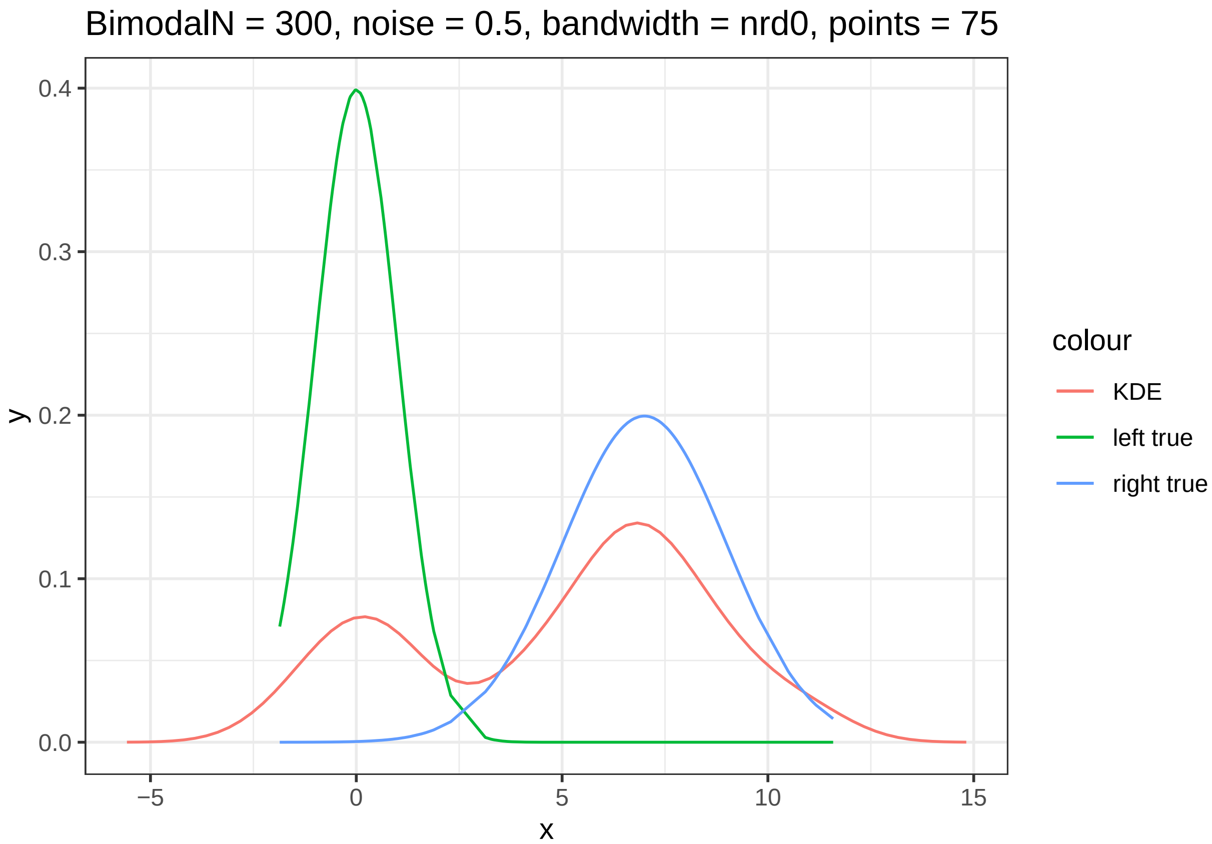

BimodalN = 300, noise = 0.5, bandwidth = 3, points = 75
y
0.4
0.3
0.2
0.1
0.0
−10
0
10
20
x
colour
KDE
left true
right true

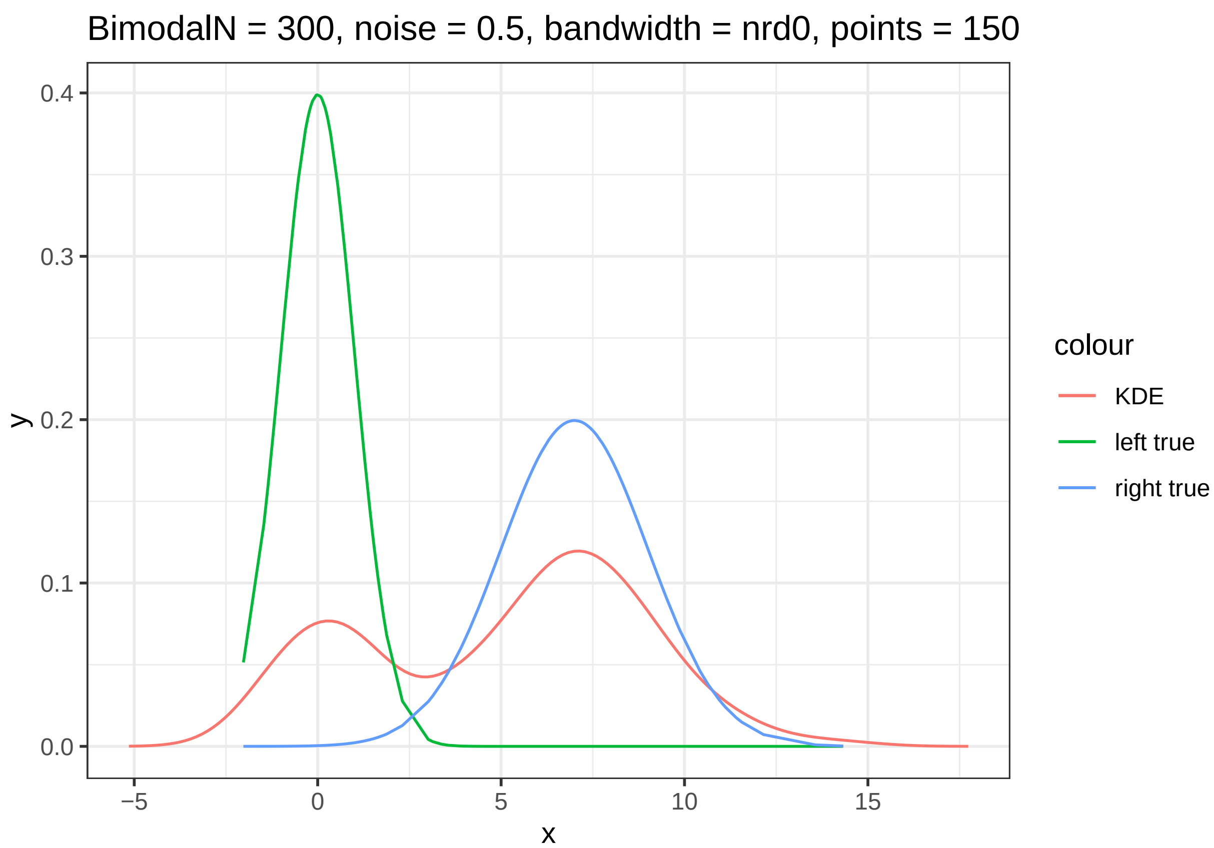


BimodalN = 300, noise = 0.5, bandwidth = 3, points = 150
0.4
0.3
0.2
0.1
0.0
y
−10
0
10
20
x
colour
KDE
left true
right true

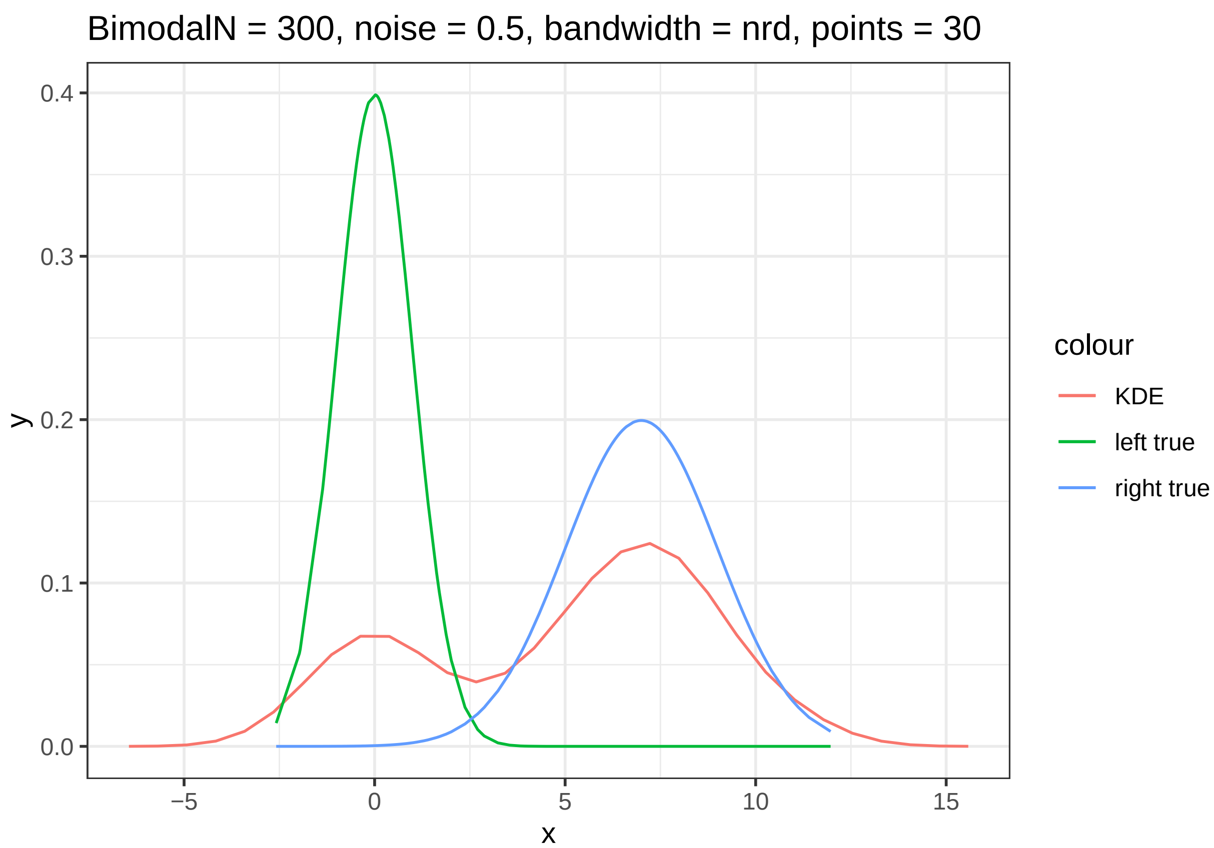


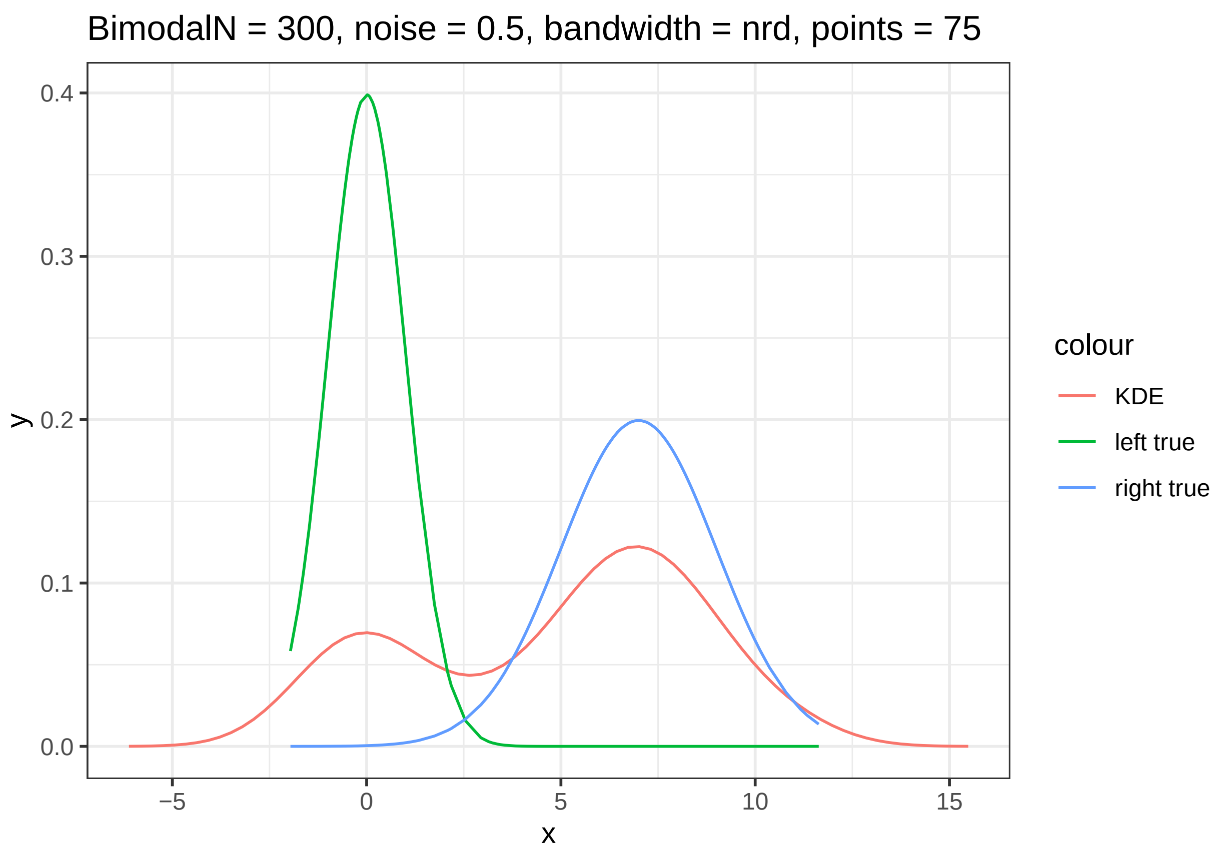
BimodalN = 300, noise = 0.5, bandwidth = nrd0, points = 30
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true

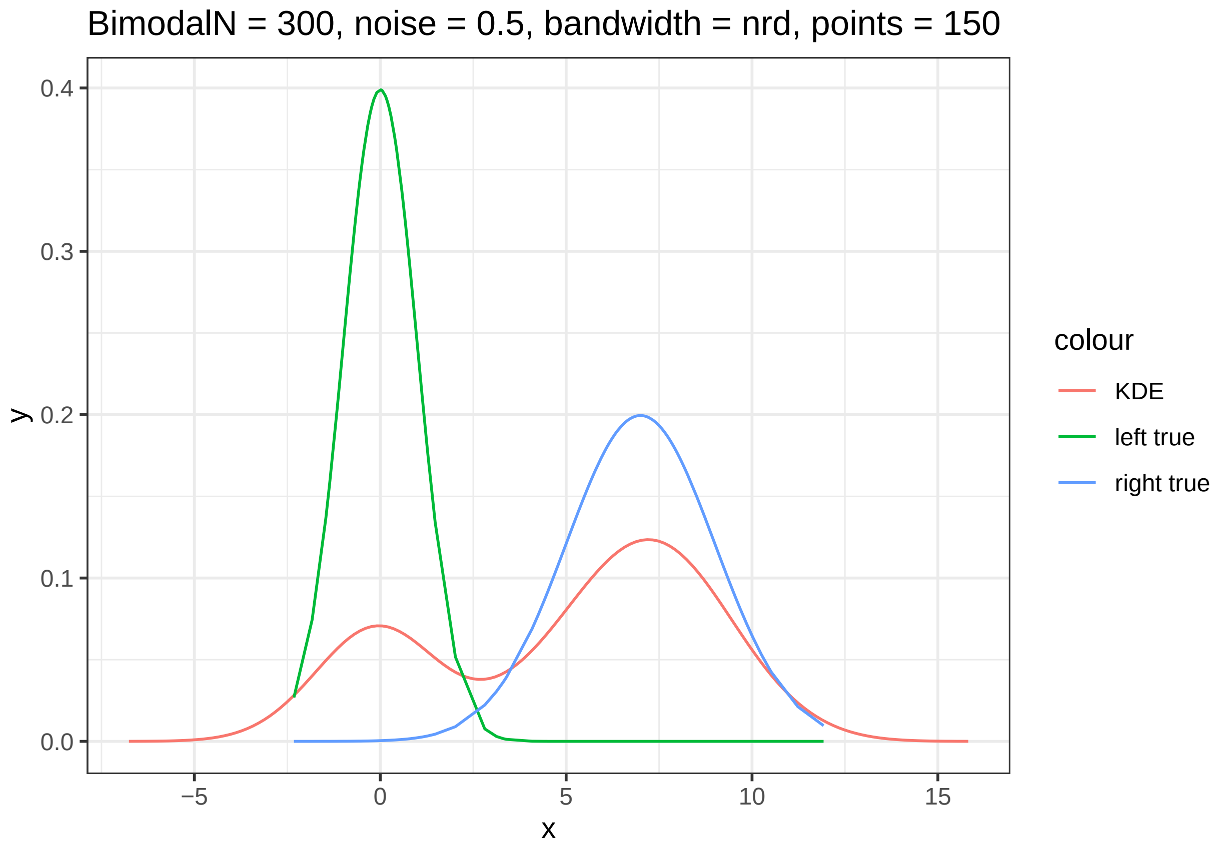

BimodalN = 300, noise = 0.5, bandwidth = nrd0, points = 75
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true

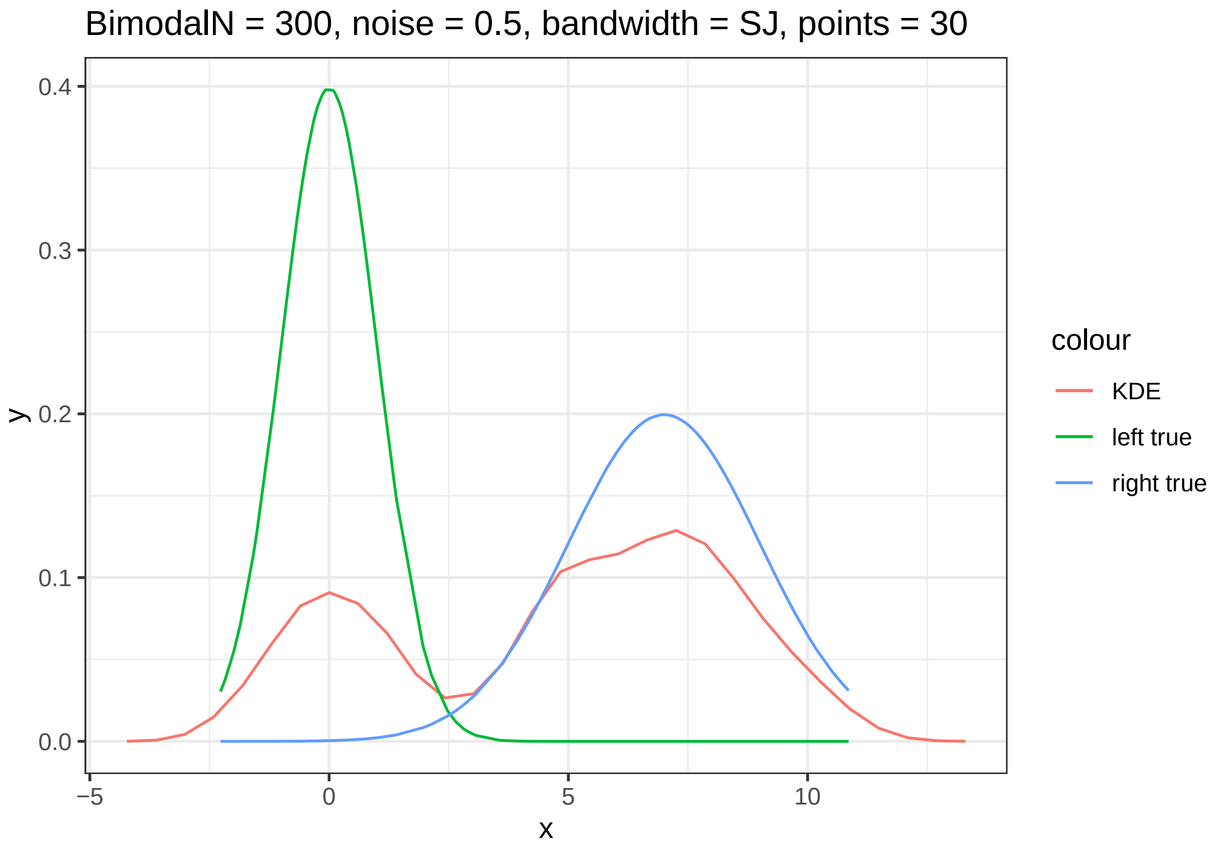


BimodalN = 300, noise = 0.5, bandwidth = nrd0, points = 150
y
0.0
0.1
0.2
0.3
0.4
x
−5
0
5
10
15
colour
KDE
left true
right true

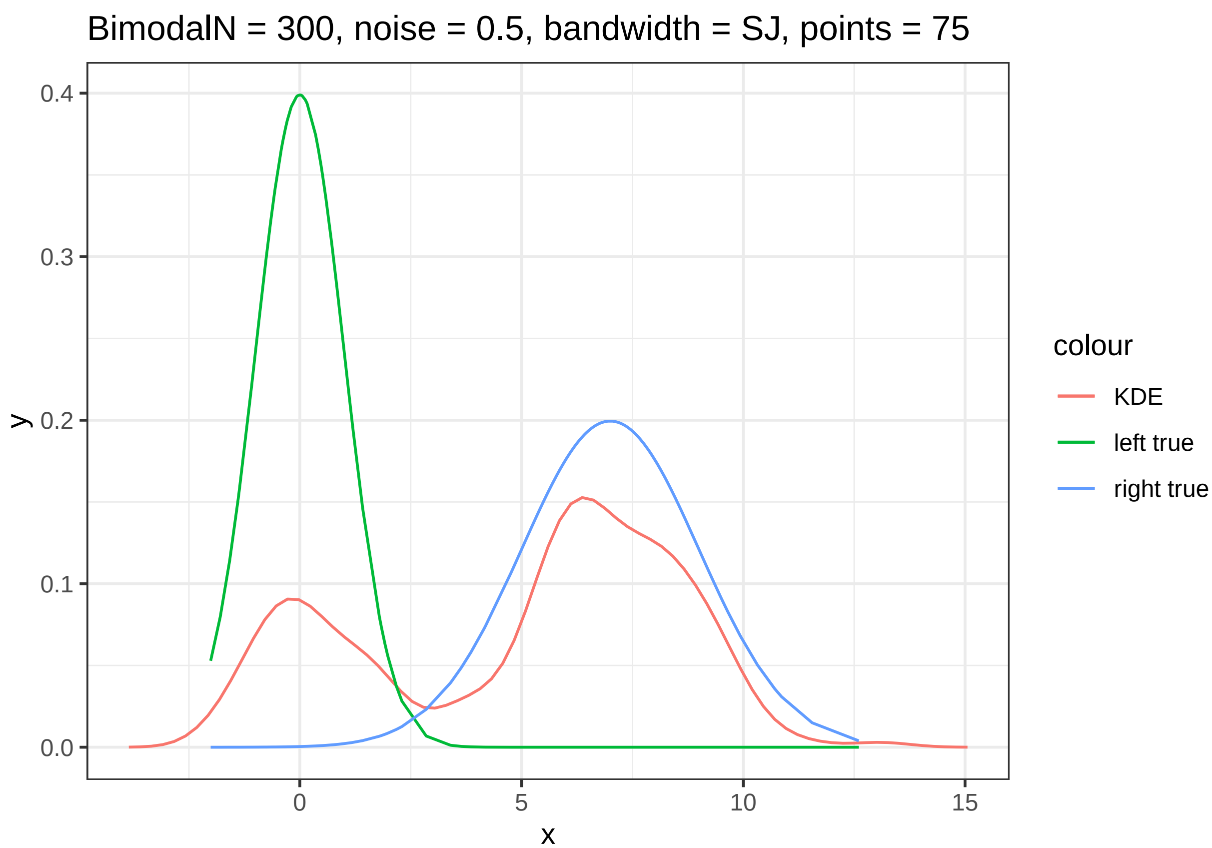

BimodalN = 300, noise = 0.5, bandwidth = nrd, points = 30
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true

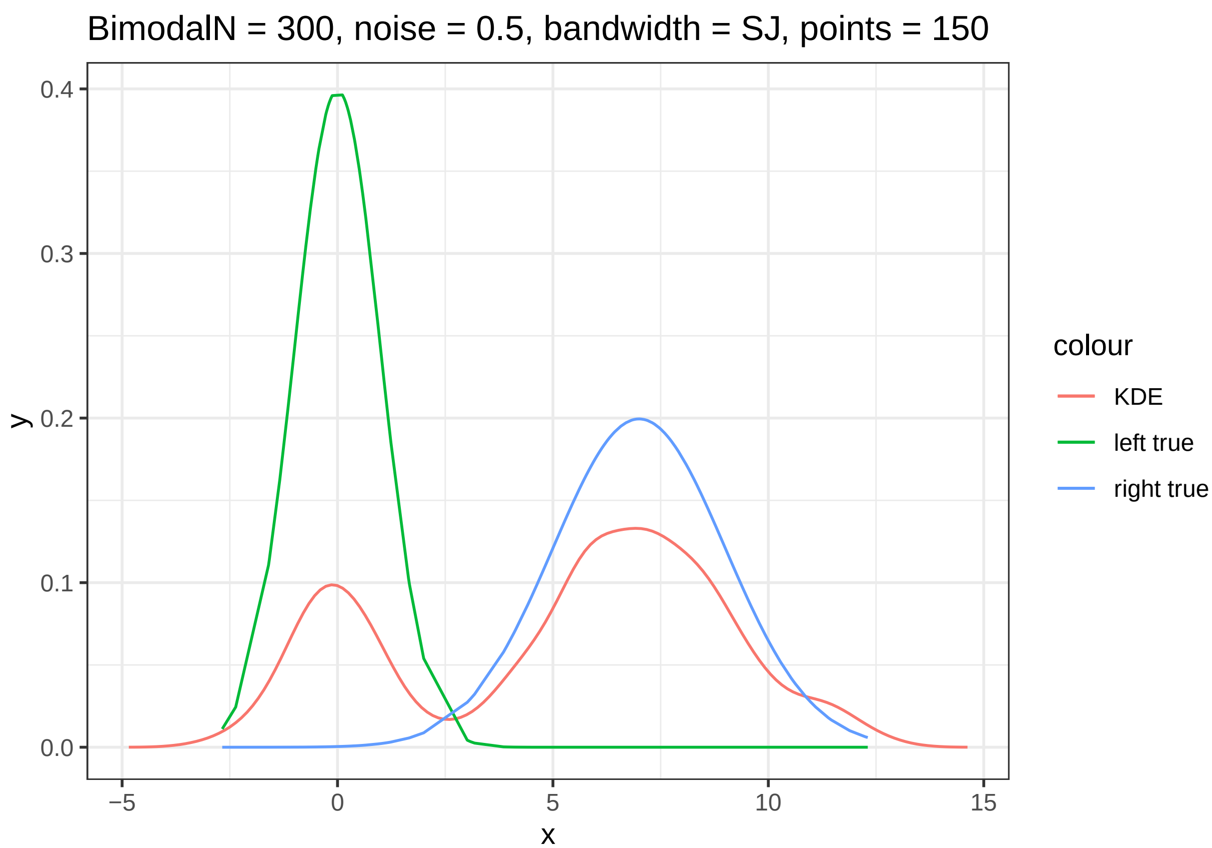

BimodalN = 300, noise = 0.5, bandwidth = nrd, points = 75
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true

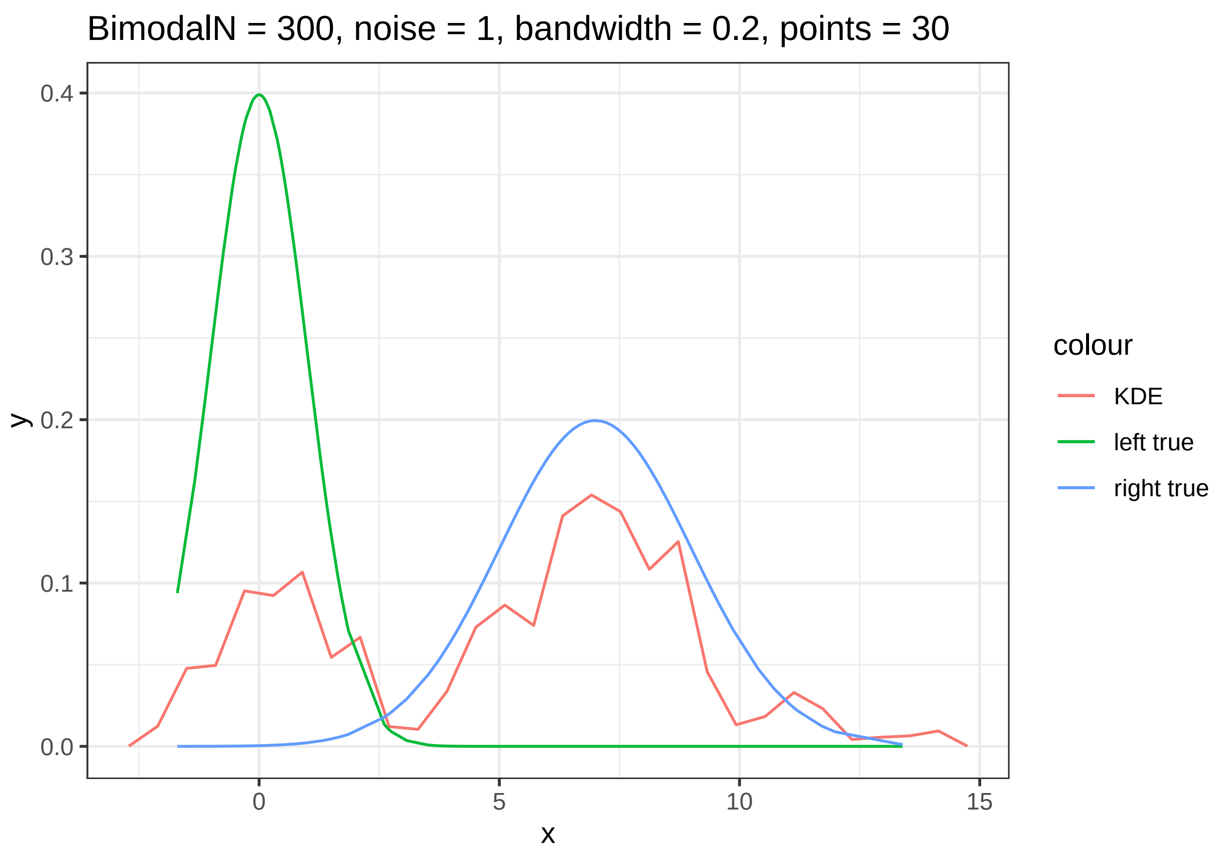

BimodalN = 300, noise = 0.5, bandwidth = nrd, points = 150
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true

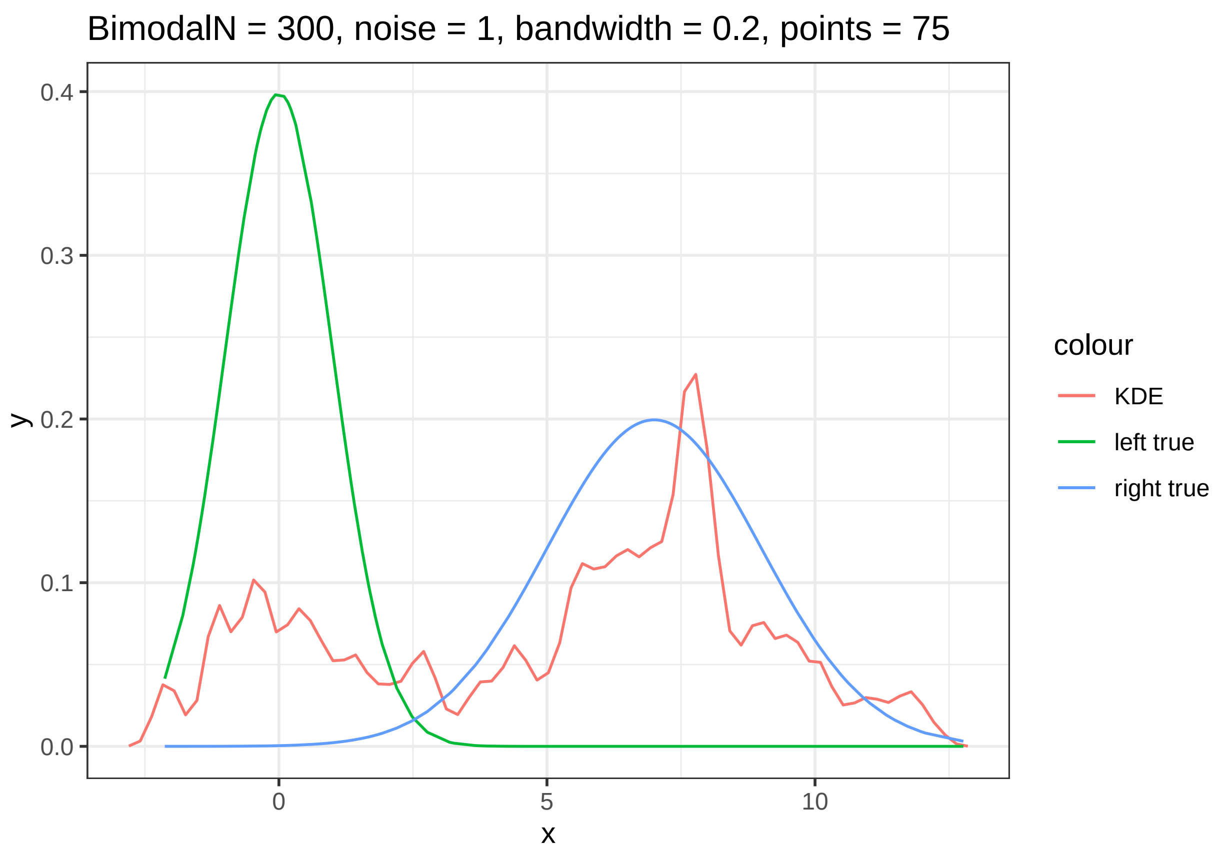


BimodalN = 300, noise = 0.5, bandwidth = SJ, points = 30
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
x
colour
KDE
left true
right true

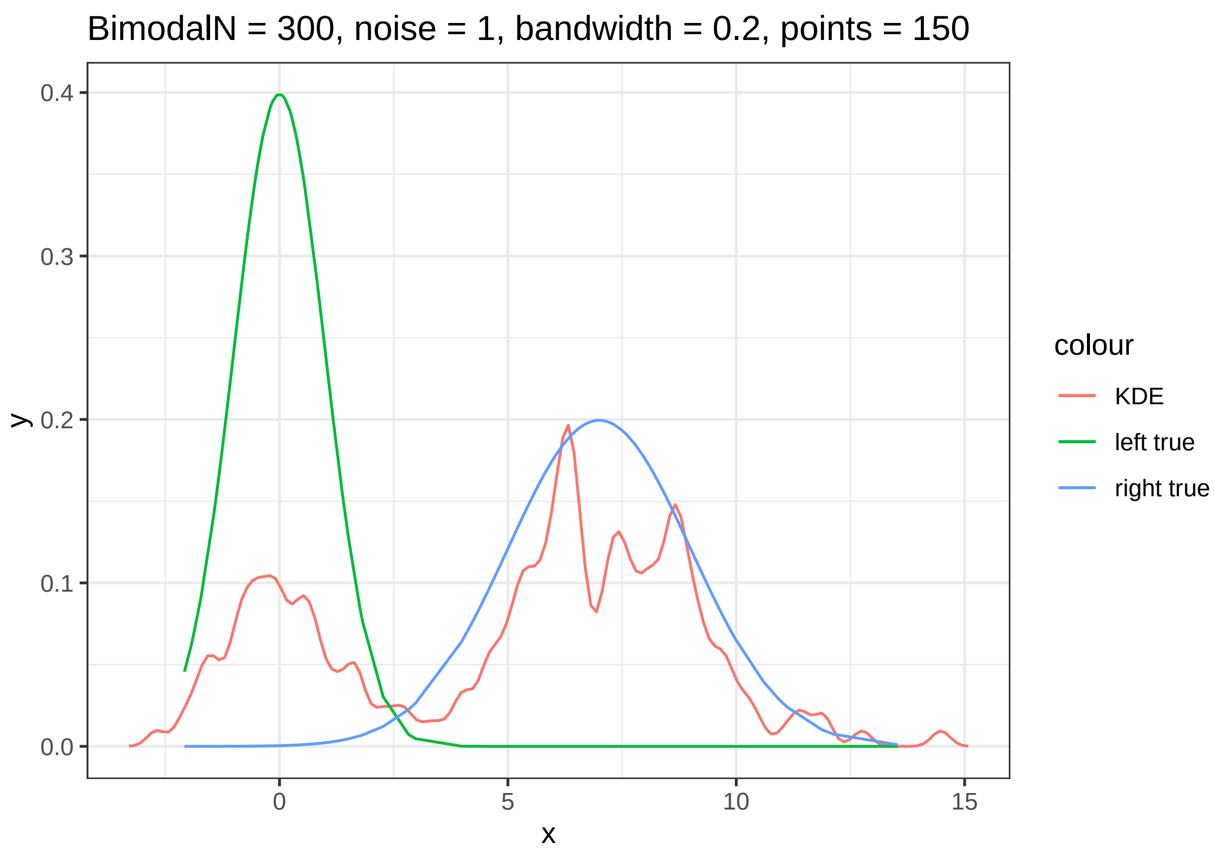

BimodalN = 300, noise = 0.5, bandwidth = SJ, points = 75
0.4
0.3
0.2
0.1
0.0
y
0
5
10
15
x
colour
KDE
left true
right true

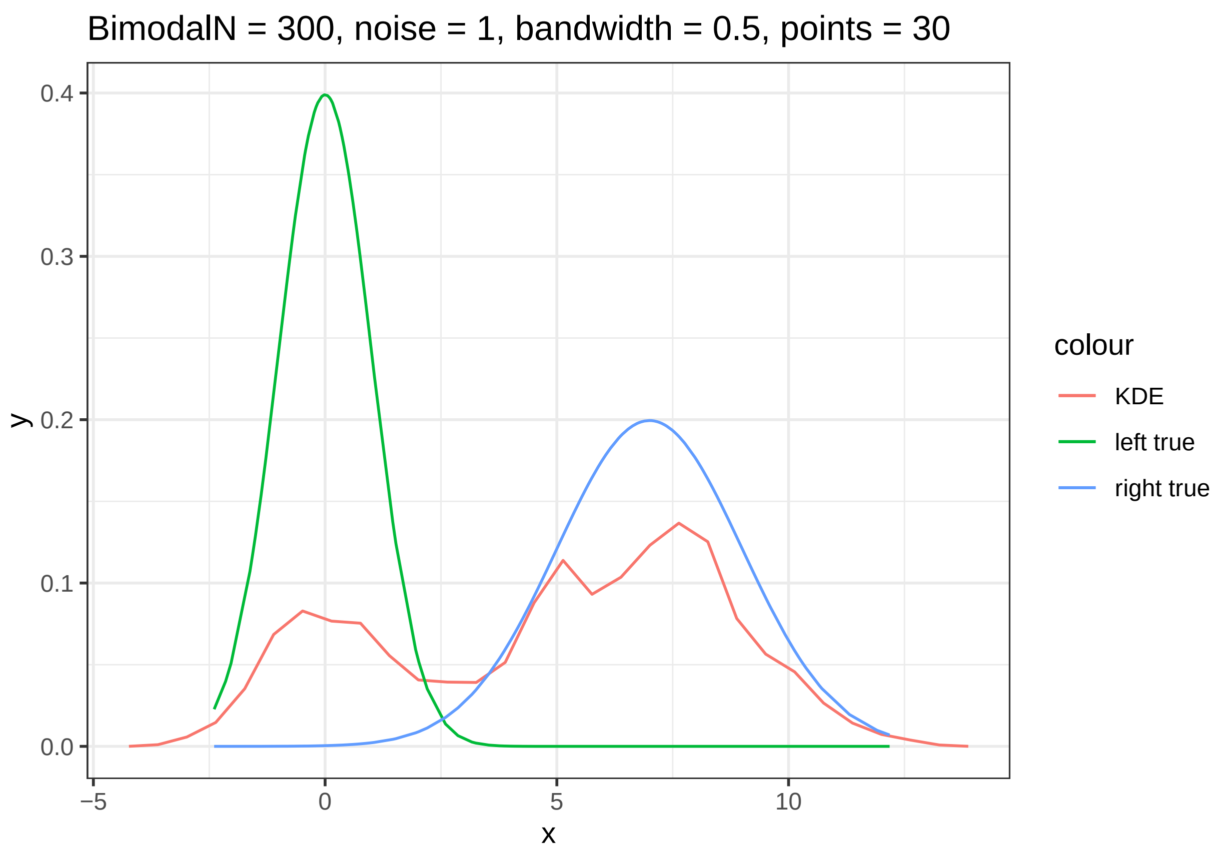

BimodalN = 300, noise = 0.5, bandwidth = SJ, points = 150
y
0.4
0.3
0.2
0.1
0.0
−5
0
5
10
15
x
colour
KDE
left true
right true

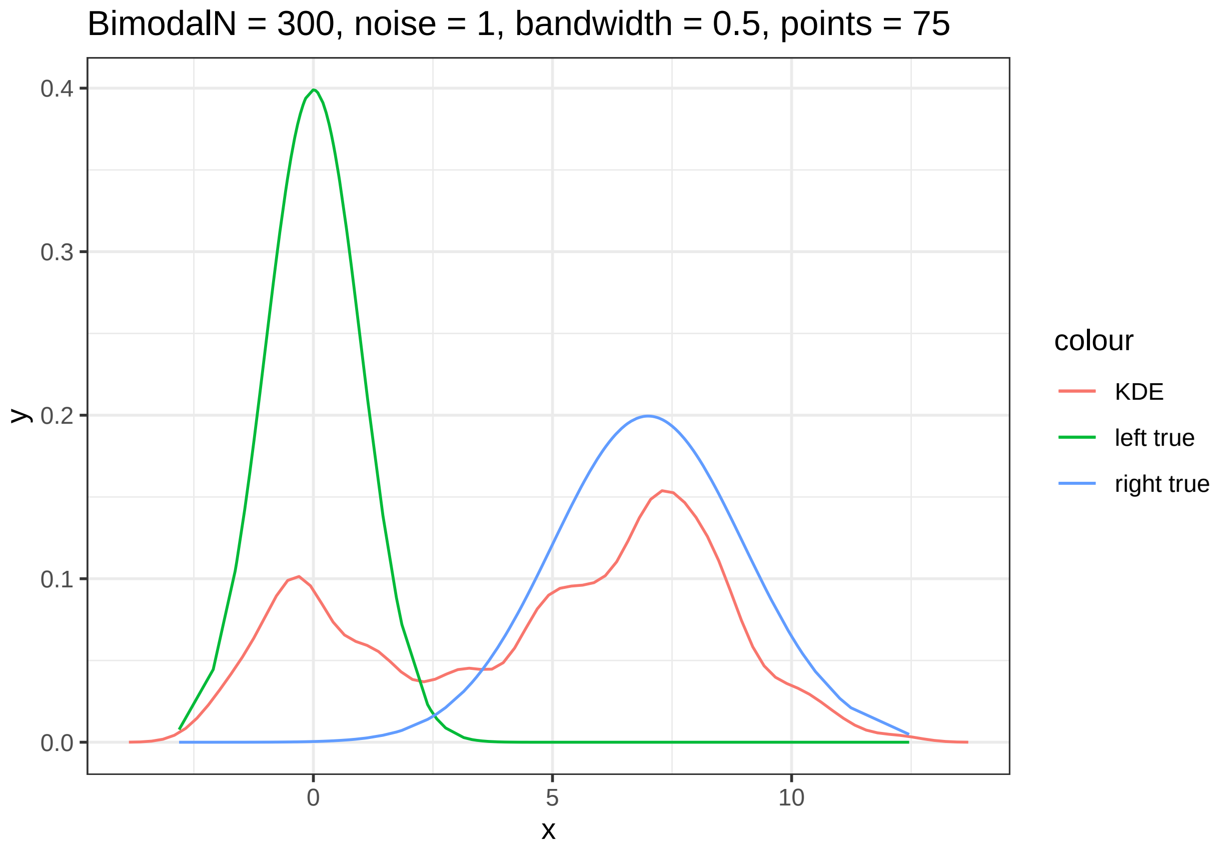

BimodalN = 300, noise = 1, bandwidth = 0.2, points = 30
0.4
0.3
0.2
0.1
0.0
y
0
5
10
15
x
colour
KDE
left true
right true

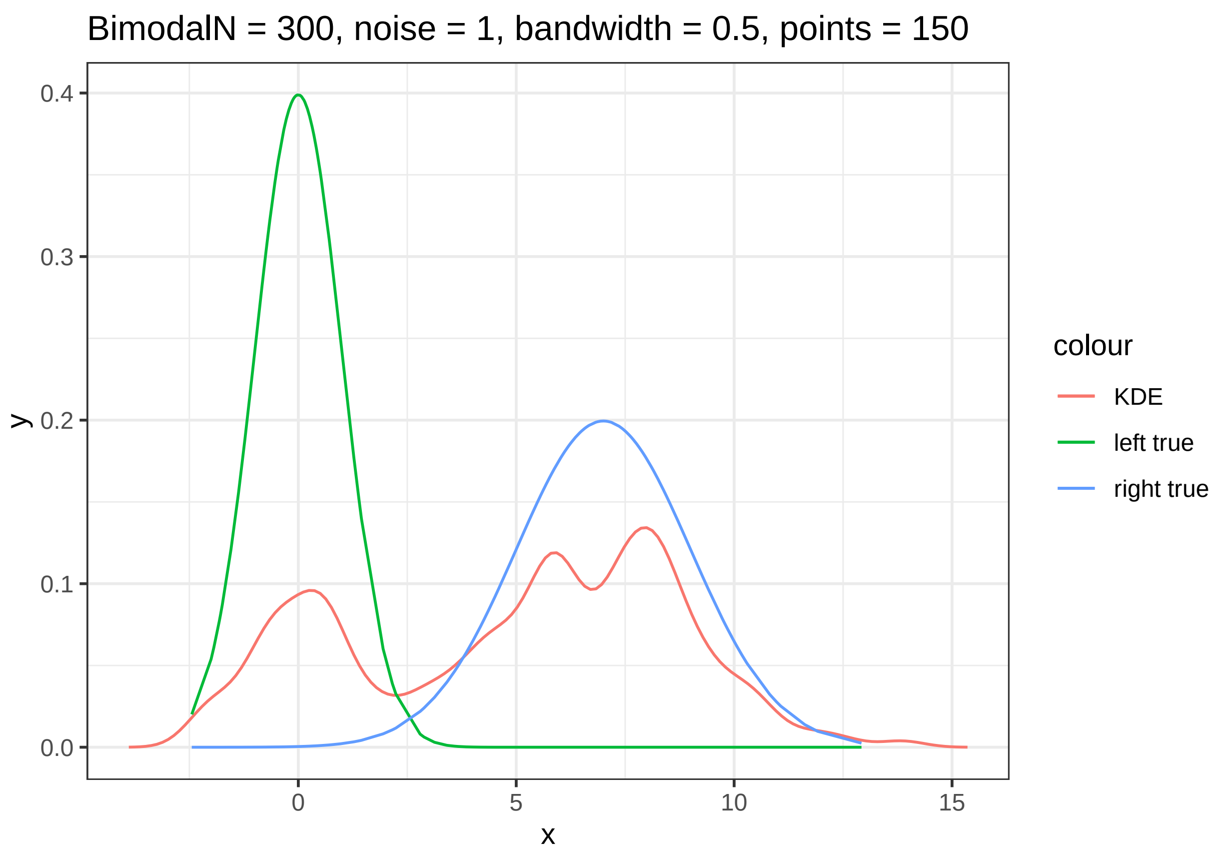


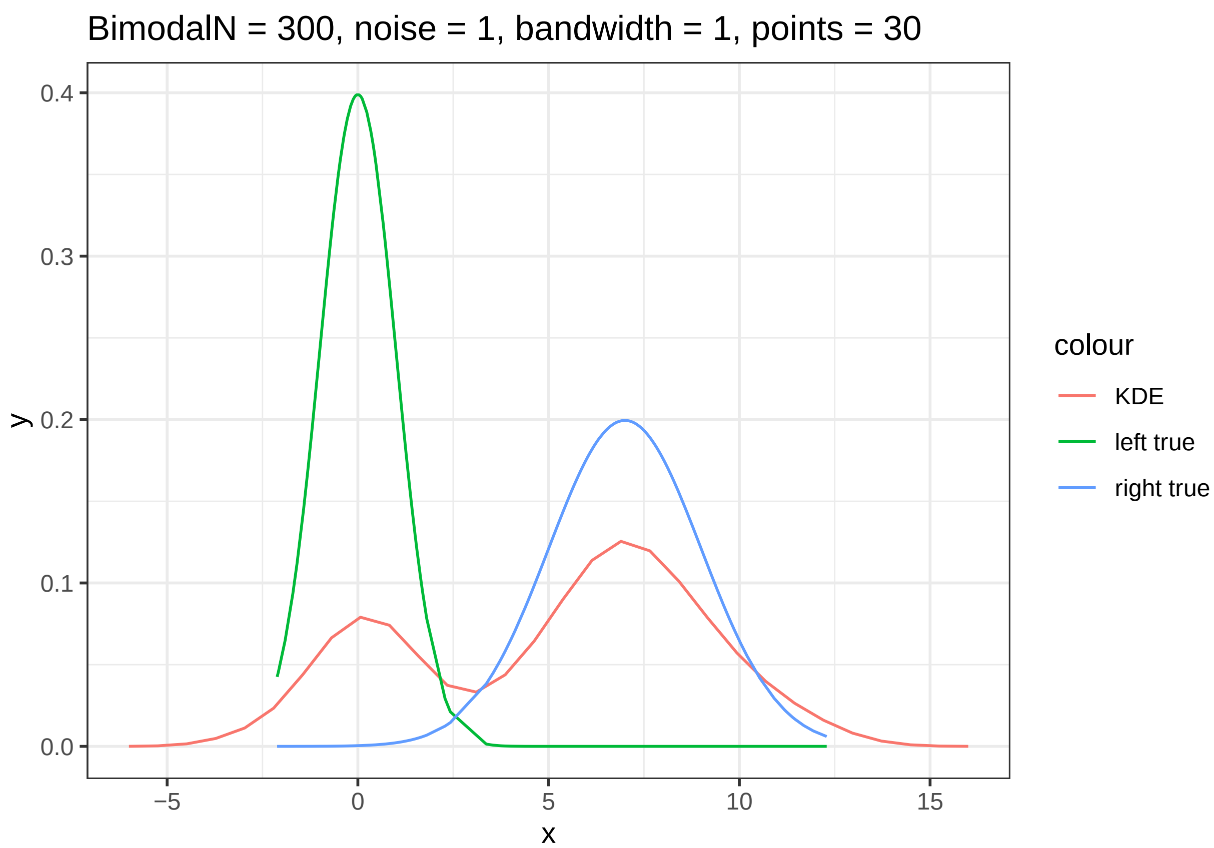
BimodalN = 300, noise = 1, bandwidth = 0.2, points = 75
0.4
0.3
0.2
0.1
0.0
y
0
5
10
x
colour
KDE
left true
right true

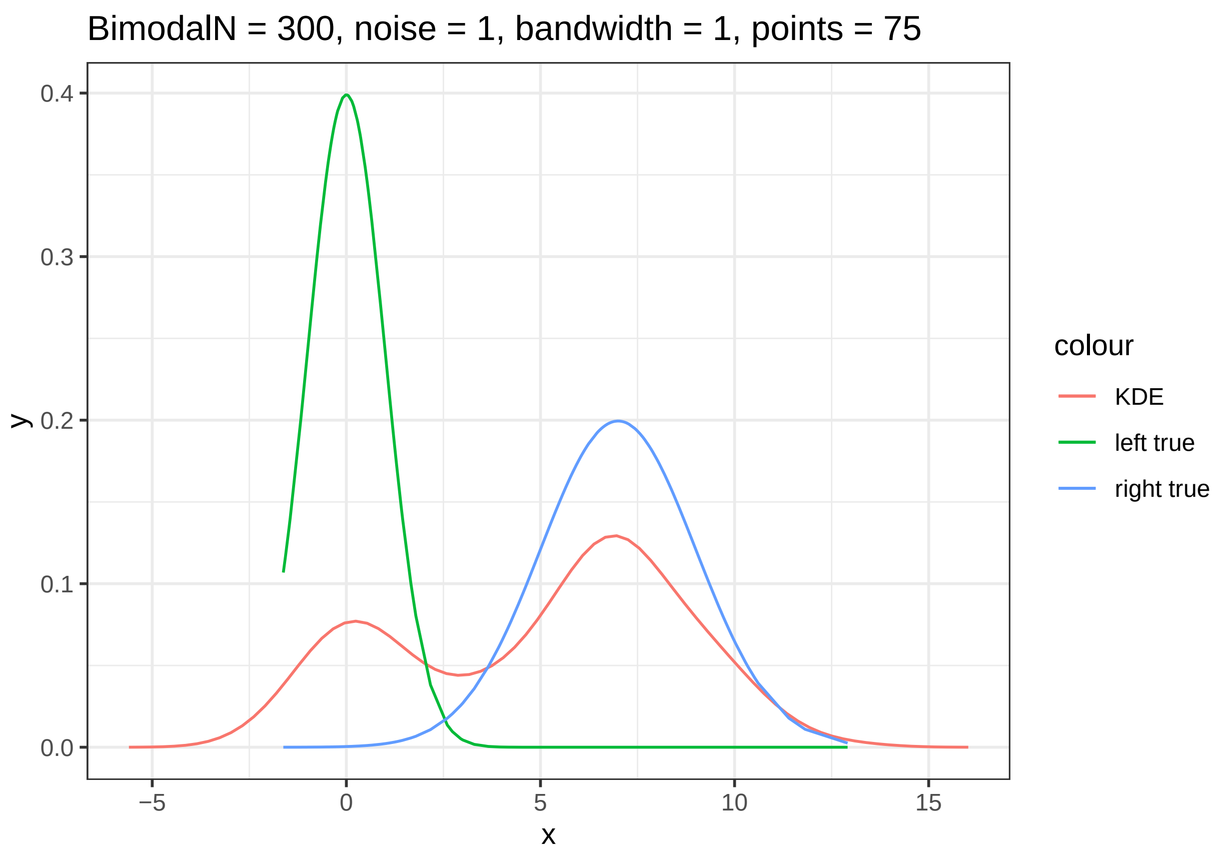

BimodalN = 300, noise = 1, bandwidth = 0.2, points = 150
y
0.0
0.1
0.2
0.3
0.4
0
5
10
15
x
colour
KDE
left true
right true

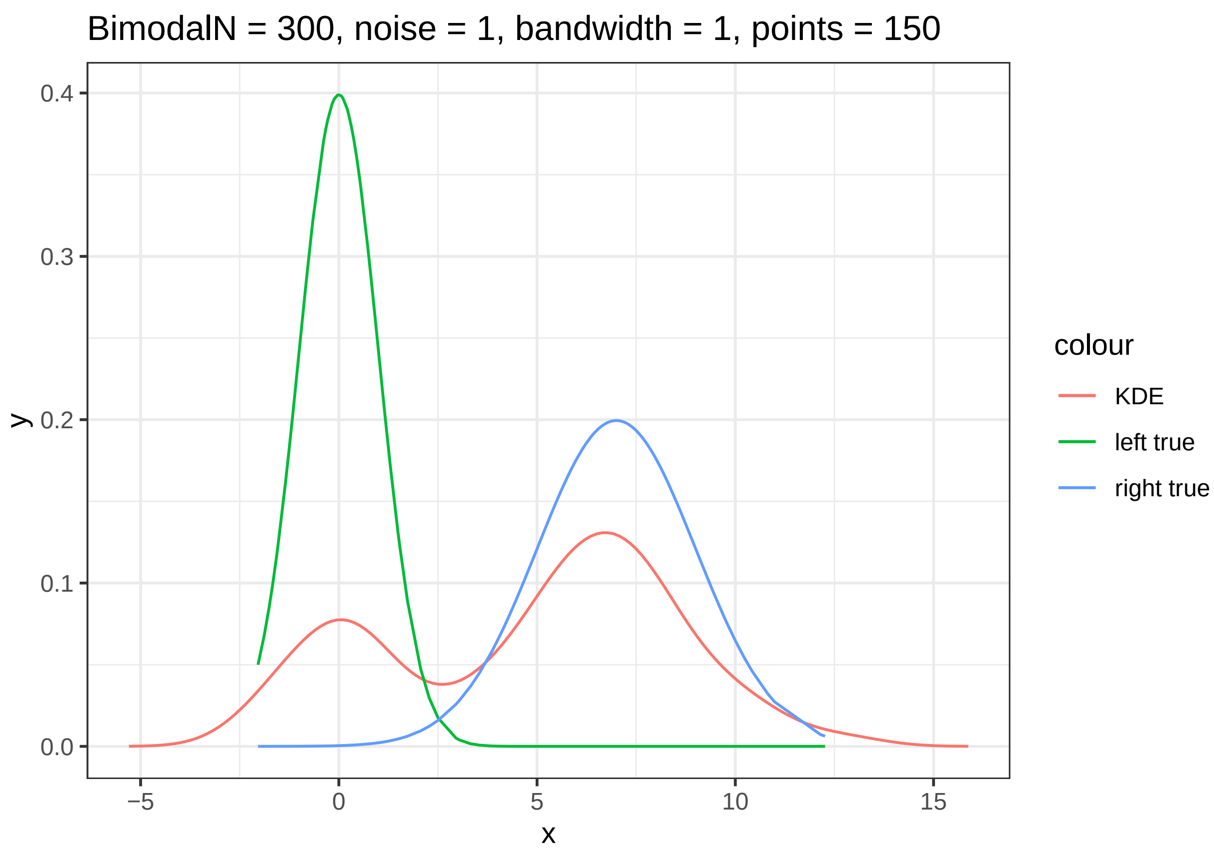

BimodalN = 300, noise = 1, bandwidth = 0.5, points = 30
y
0.0
0.1
0.2
0.3
0.4
x
−5
0
5
10
colour
KDE
left true
right true


BimodalN = 300, noise = 1, bandwidth = 0.5, points = 75
0.4
0.3
0.2
0.1
0.0
y
0
5
10
x
colour
KDE
left true
right true

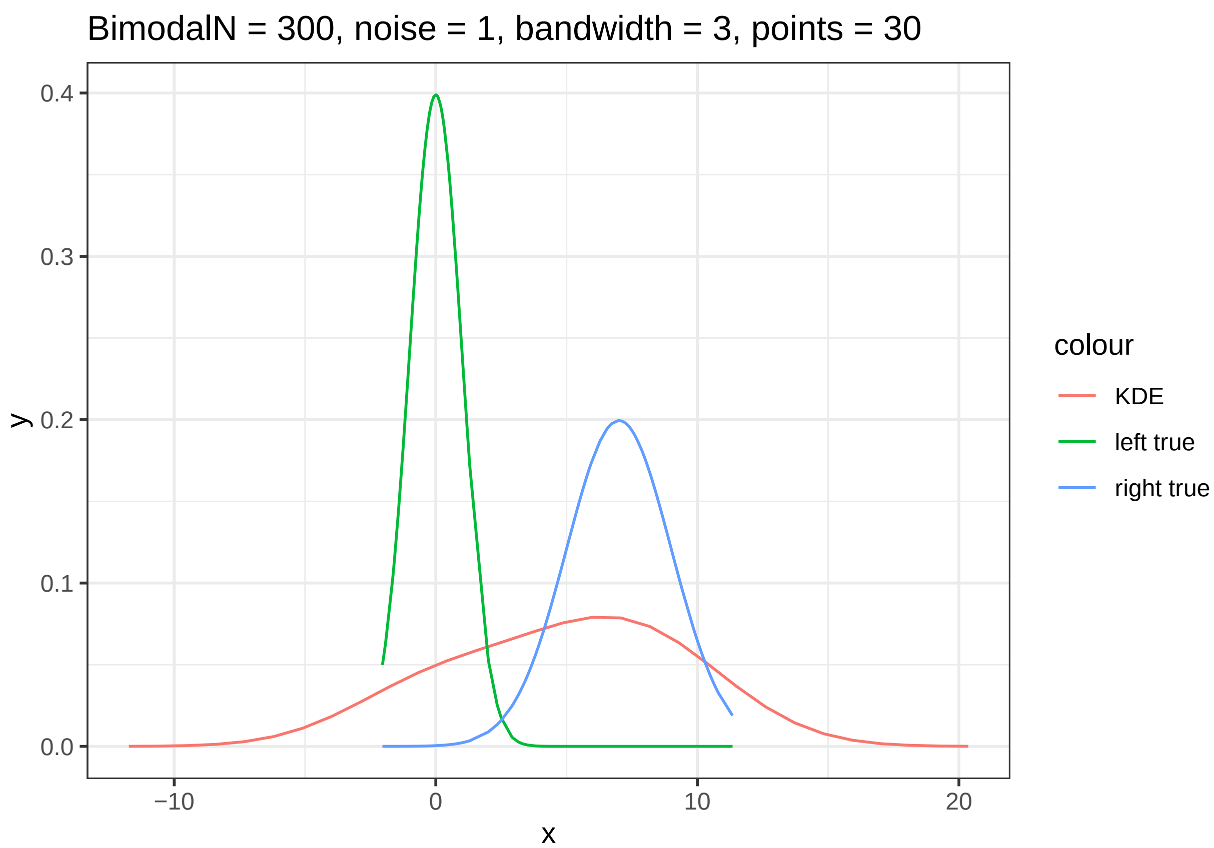

BimodalN = 300, noise = 1, bandwidth = 0.5, points = 150
0.4
0.3
0.2
0.1
0.0
y
0
5
10
15
x
colour
KDE
left true
right true


BimodalN = 300, noise = 1, bandwidth = 1, points = 30
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true

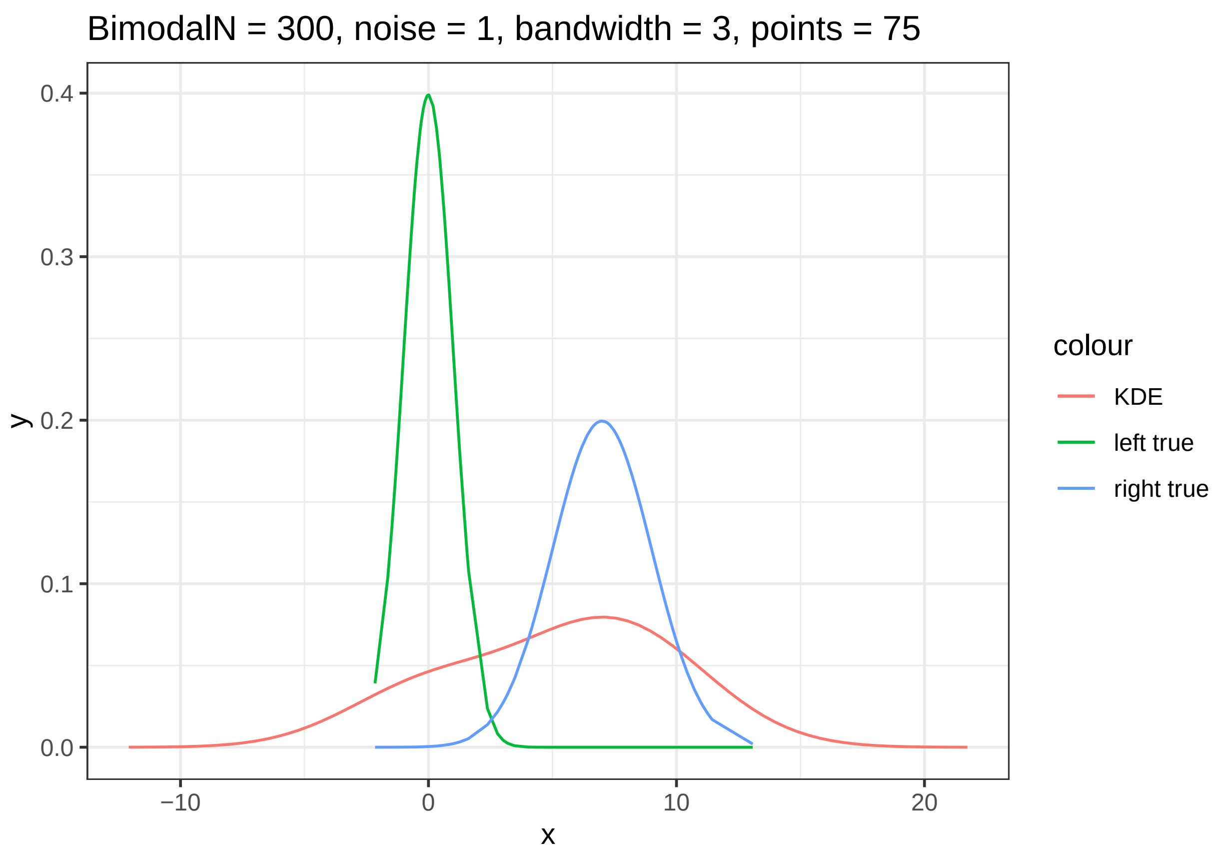

BimodalN = 300, noise = 1, bandwidth = 1, points = 75
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true

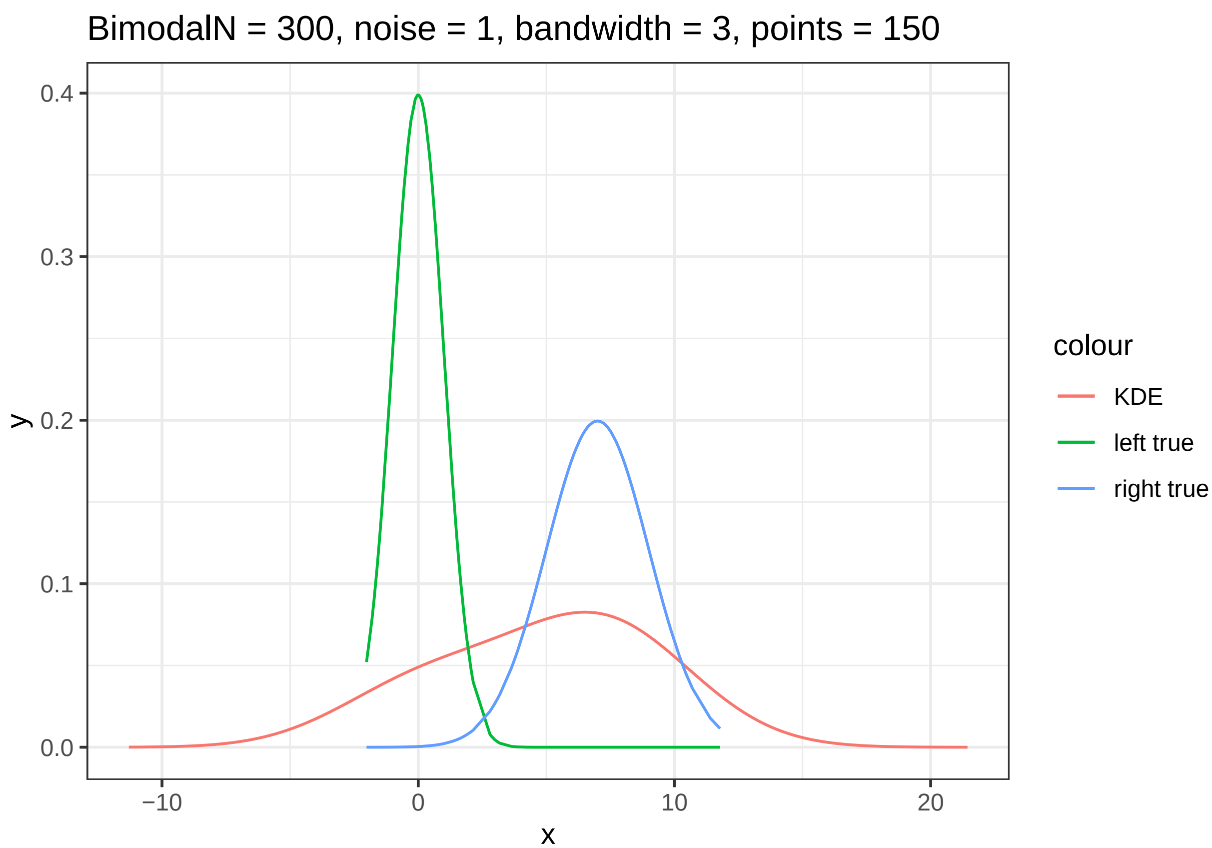


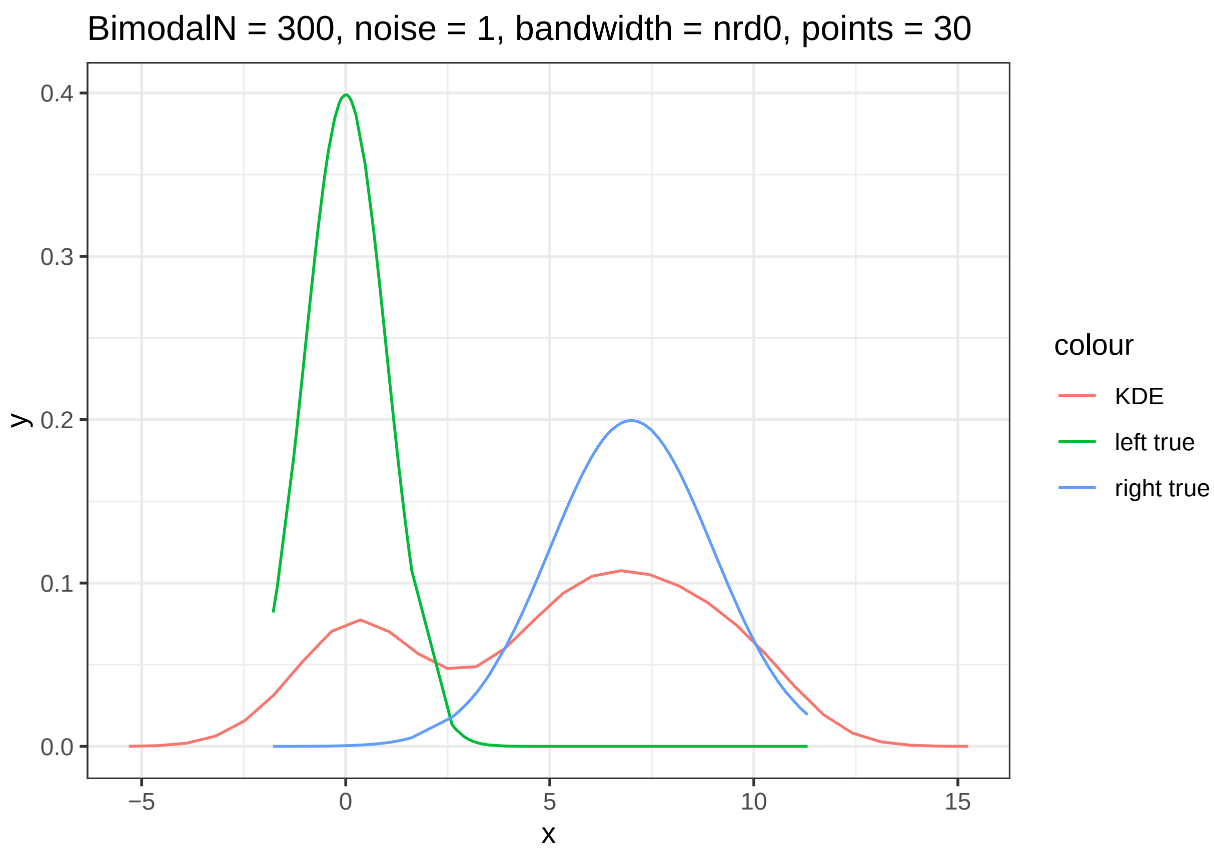
BimodalN = 300, noise = 1, bandwidth = 1, points = 150
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true

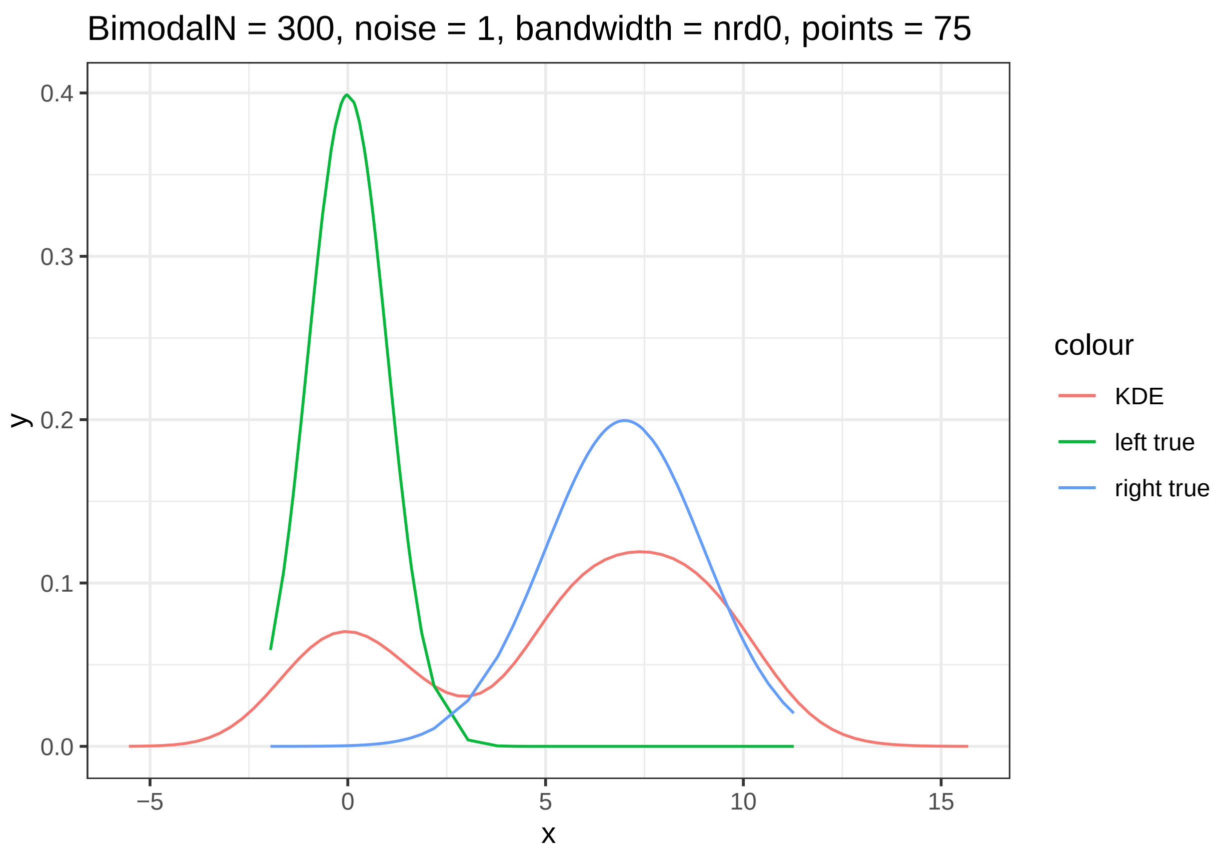


BimodalN = 300, noise = 1, bandwidth = 3, points = 30
0.4
0.3
0.2
0.1
0.0
y
−10
0
10
20
x
colour
KDE
left true
right true

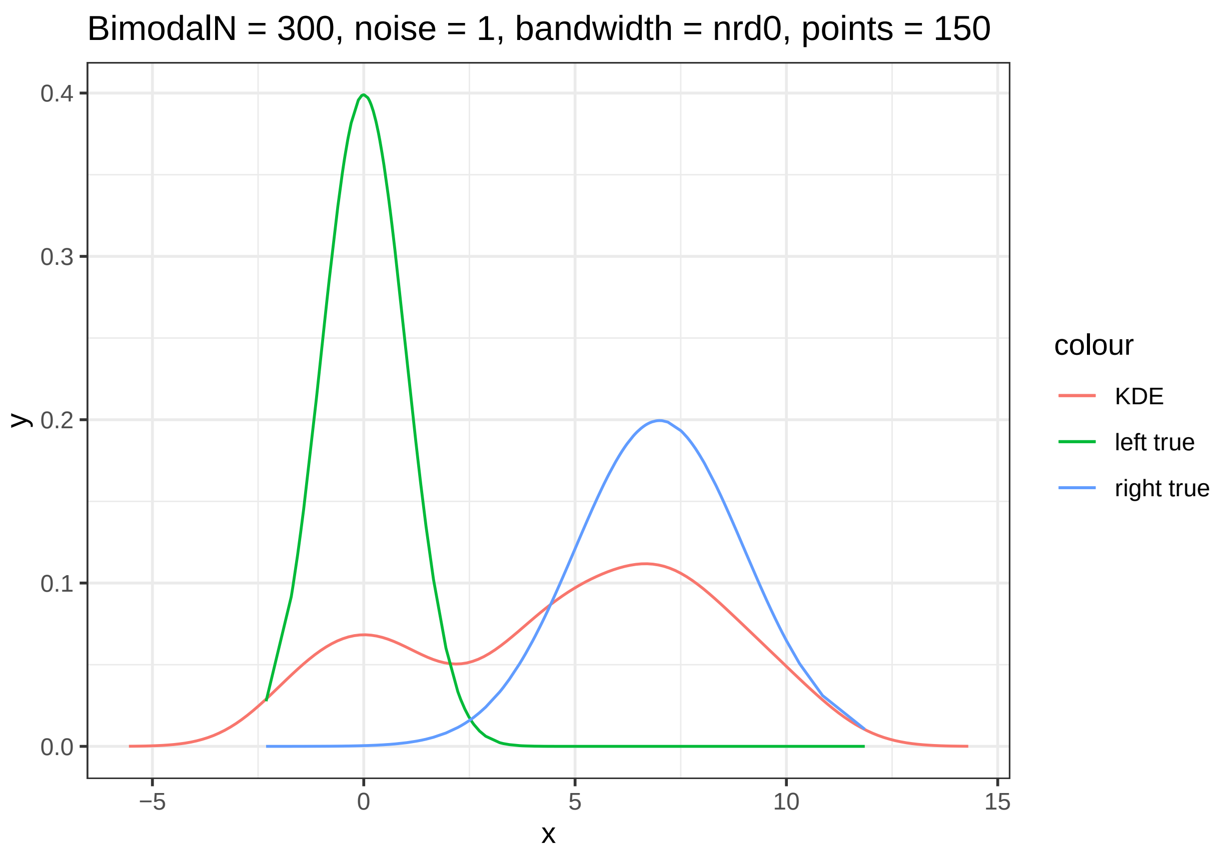

BimodalN = 300, noise = 1, bandwidth = 3, points = 75
y
0.4
0.3
0.2
0.1
0.0
−10
0
10
20
x
colour
KDE
left true
right true

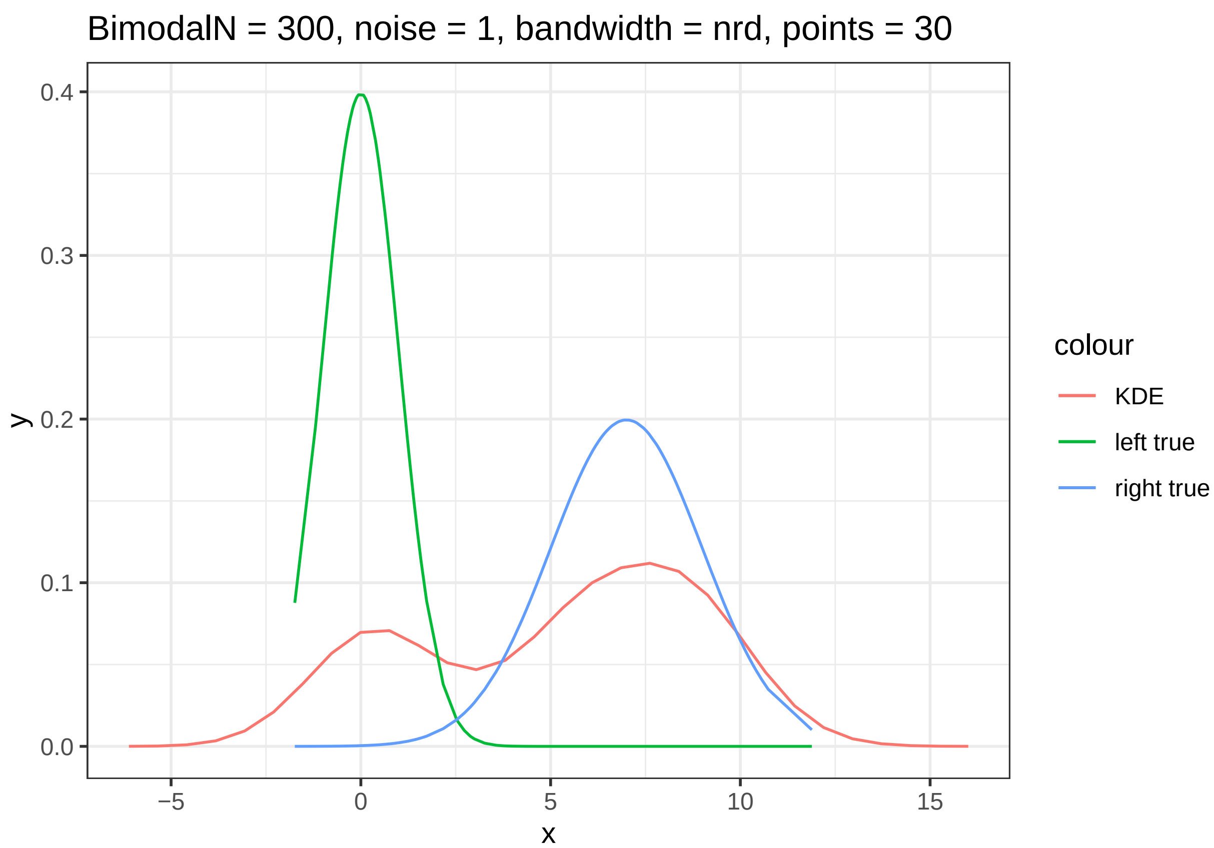

BimodalN = 300, noise = 1, bandwidth = 3, points = 150
0.4
0.3
0.2
0.1
0.0
y
−10
0
10
20
x
colour
KDE
left true
right true


BimodalN = 300, noise = 1, bandwidth = nrd0, points = 30
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true

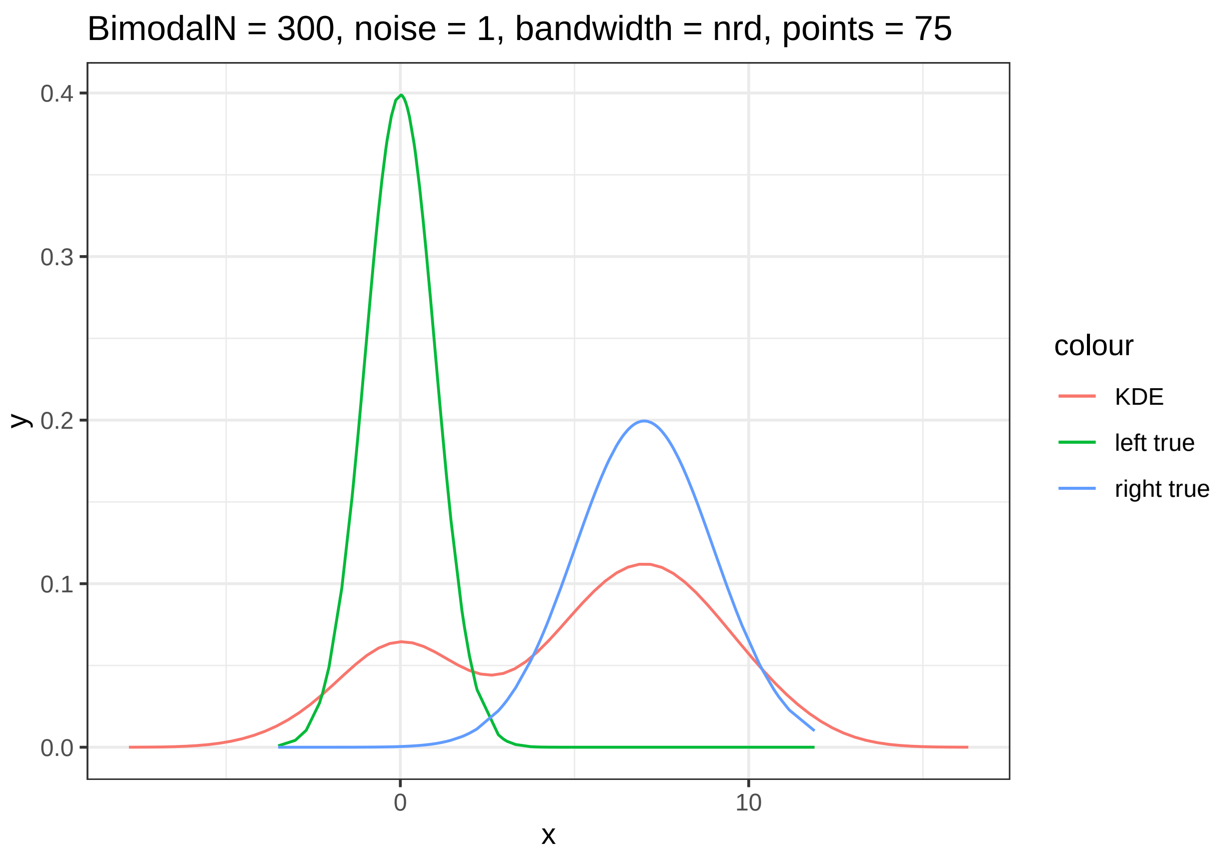


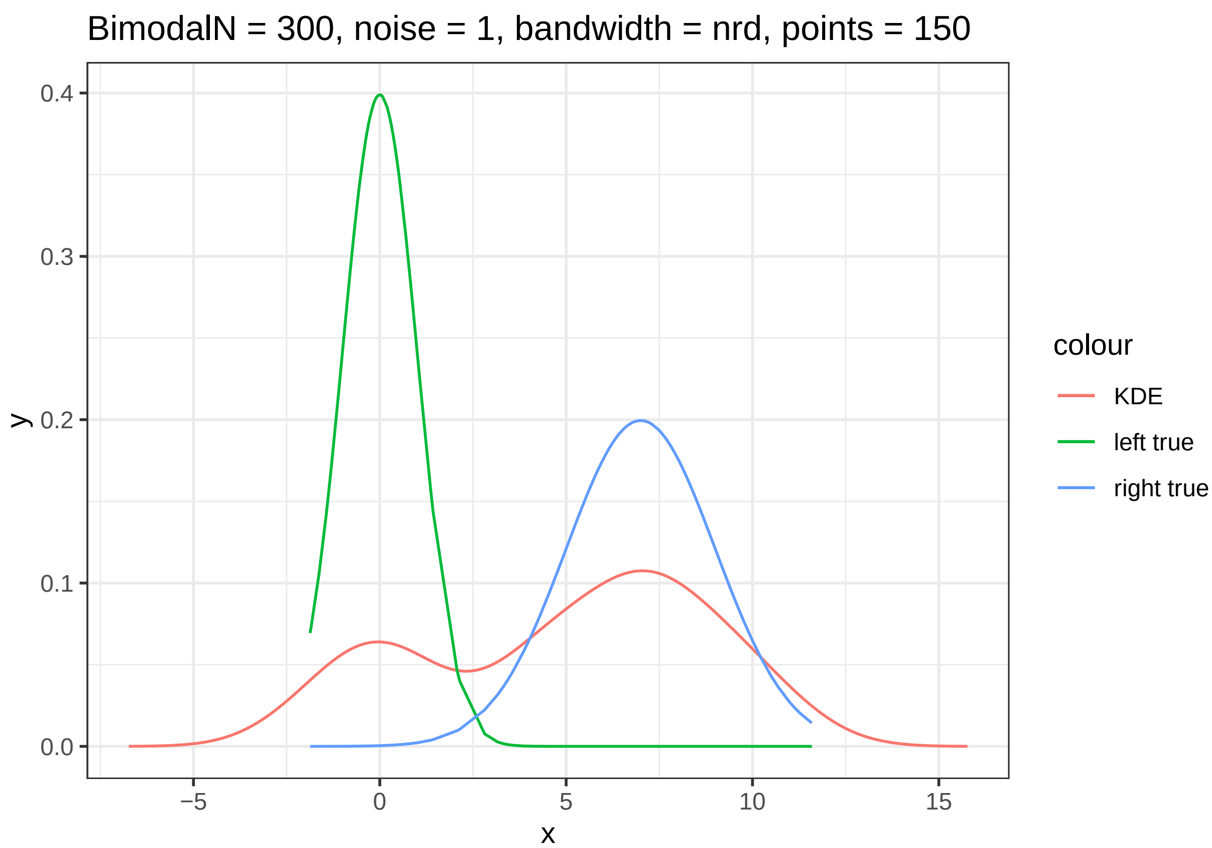
BimodalN = 300, noise = 1, bandwidth = nrd0, points = 75
y
0.0
0.1
0.2
0.3
0.4
x
−5
0
5
10
15
colour
KDE
left true
right true


BimodalN = 300, noise = 1, bandwidth = nrd0, points = 150
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true

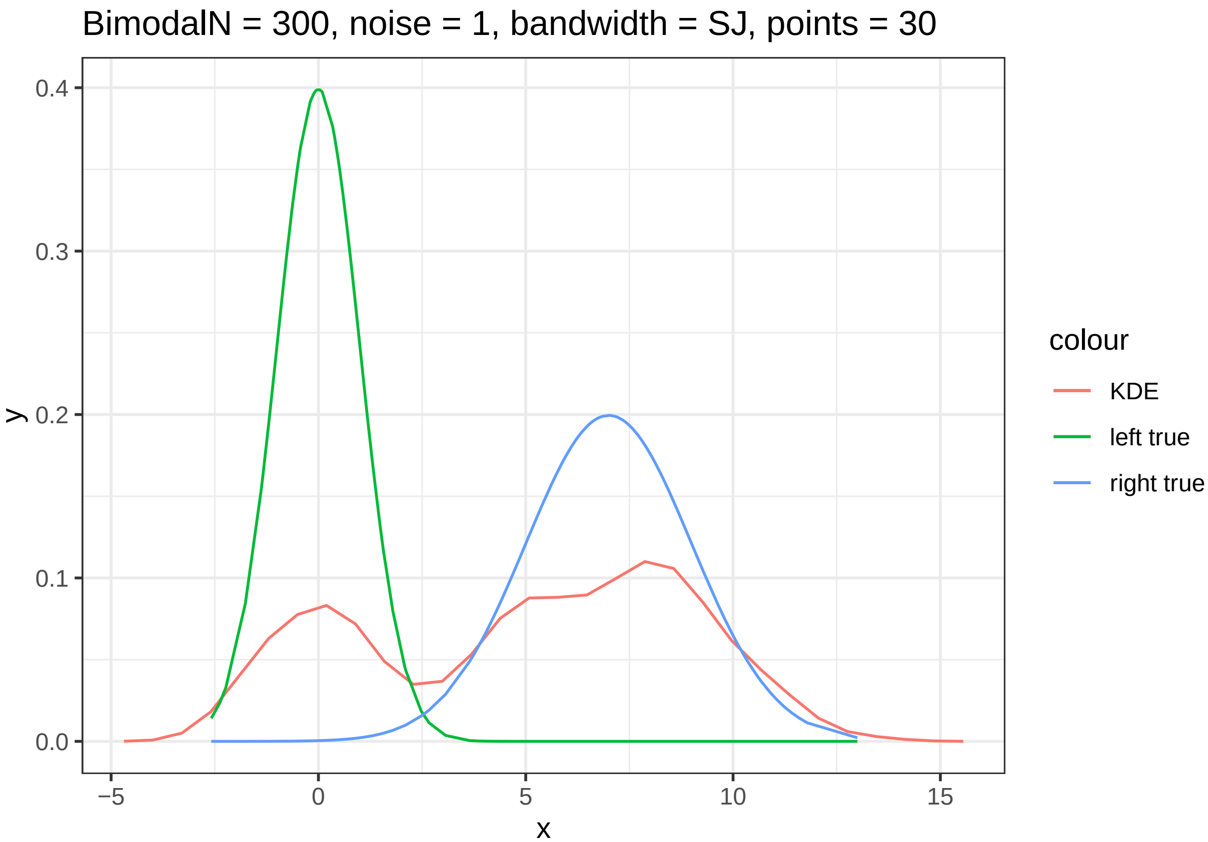


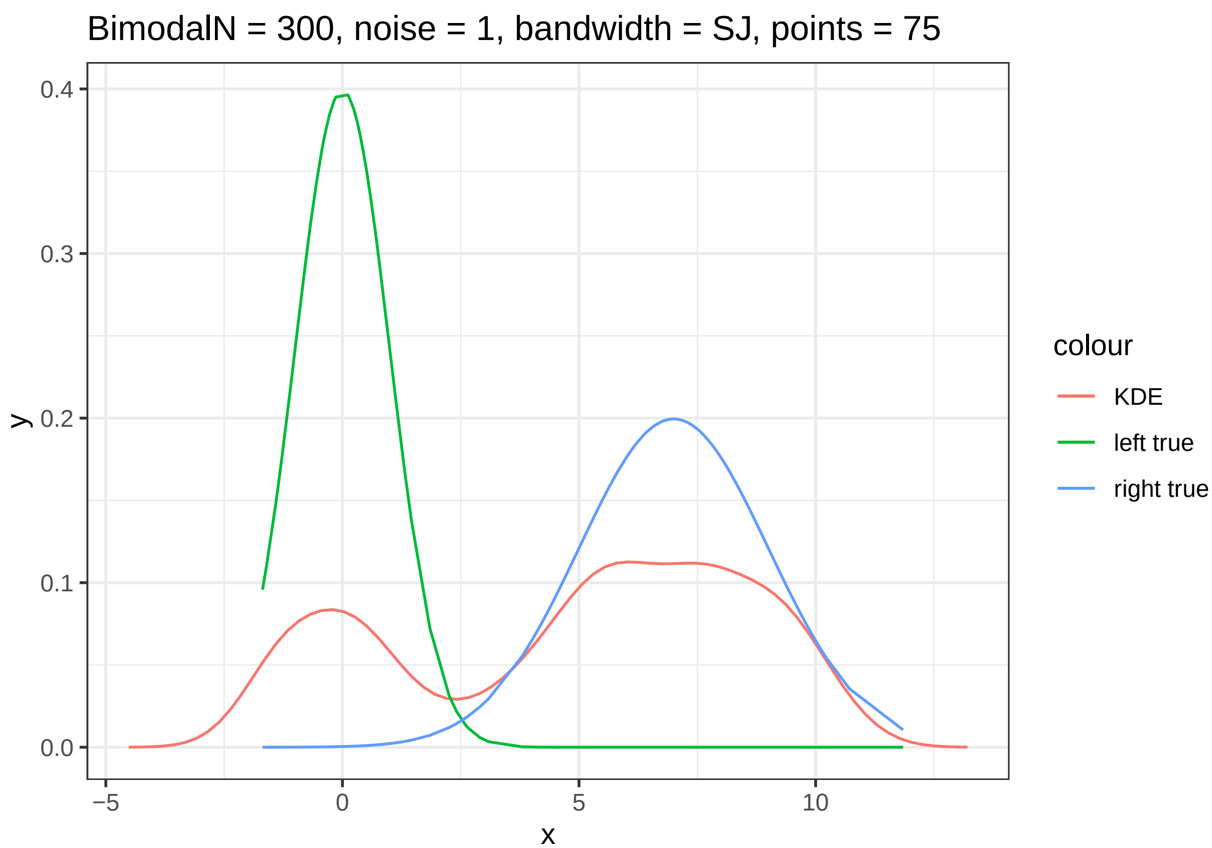
BimodalN = 300, noise = 1, bandwidth = nrd, points = 30
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true

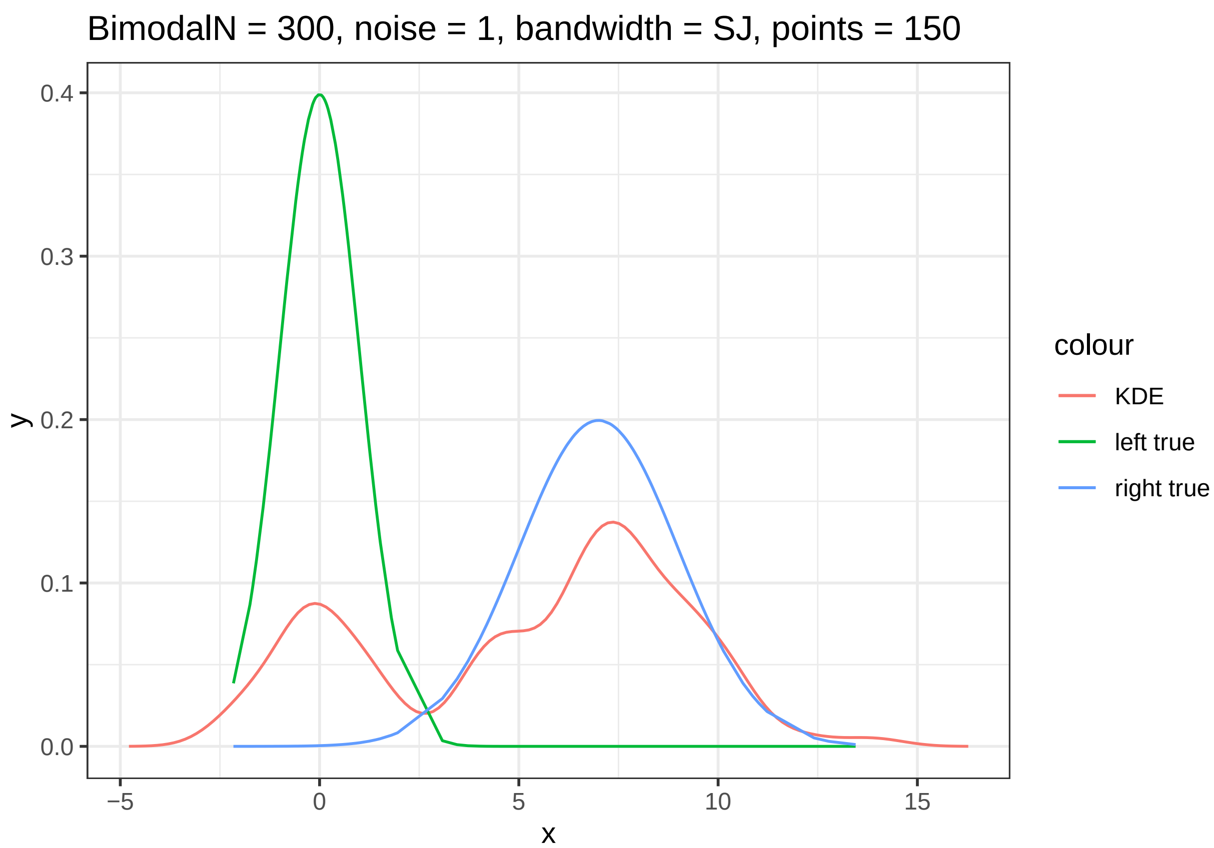

BimodalN = 300, noise = 1, bandwidth = nrd, points = 75
0.4
0.3
0.2
0.1
0.0
y
0
10
x
colour
KDE
left true
right true



BimodalN = 300, noise = 1, bandwidth = nrd, points = 150
y
0.0
0.1
0.2
0.3
0.4
x
−5
0
5
10
15
colour
KDE
left true
right true


BimodalN = 300, noise = 1, bandwidth = SJ, points = 30
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true


BimodalN = 300, noise = 1, bandwidth = SJ, points = 75
y
0.0
0.1
0.2
0.3
0.4
x
−5
0
5
10
colour
KDE
left true
right true



BimodalN = 300, noise = 1, bandwidth = SJ, points = 150
0.4
0.3
0.2
0.1
0.0
y
−5
0
5
10
15
x
colour
KDE
left true
right true