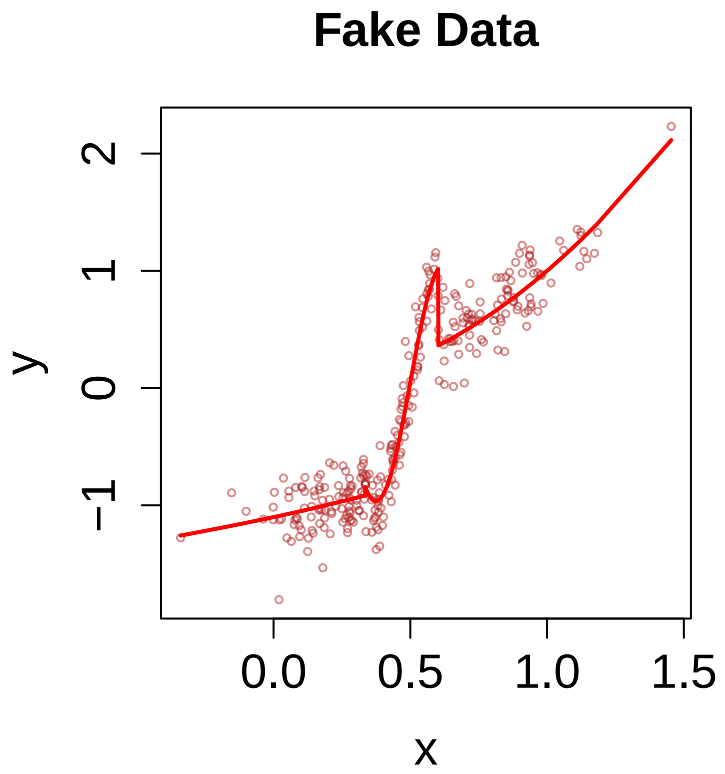
Contents

```
do_scatter_plot <- function(x, y, main)
{
    plot(x, y,
         col=adjustcolor("firebrick", 0.5),
         pch=1, cex=0.5,
         main=main,
         cex.axis=1.5, cex.main=1.5, cex.lab=1.5
        )
}
```

Fake Data Example.

```
set.seed(444)
N <- 150
x <- c(runif(N, 0,1), rnorm(N, 0.5, 0.3))
mu <- function(x)
{
    xrange <- range(x)
    vals <- vector("numeric", length=length(x))
    breaks <- unname(quantile(x, probs=c(1/3,2/3)))
    first <- x <= breaks[1]
    second <- (x > breaks[1]) & (x <= breaks[2])
    third  <- x > breaks[2]
    vals[first]  <- -(1 - x[first])^0.5 -0.1
    vals[second] <- sin(x[second] * 4 * pi) + x[second]/10
    vals[third]  <- x[third]^2
    vals
}
y <- mu(x) + rnorm(2*N, 0, .2)
```

```
do_scatter_plot(x, y, main="Fake Data")
xordered <- sort(x)
lines(xordered, mu(xordered), col="red", lwd=2)
```


Fake Data
y
2
1
0
−1
0.0
0.5
1.0
1.5
x

Piecewise Models

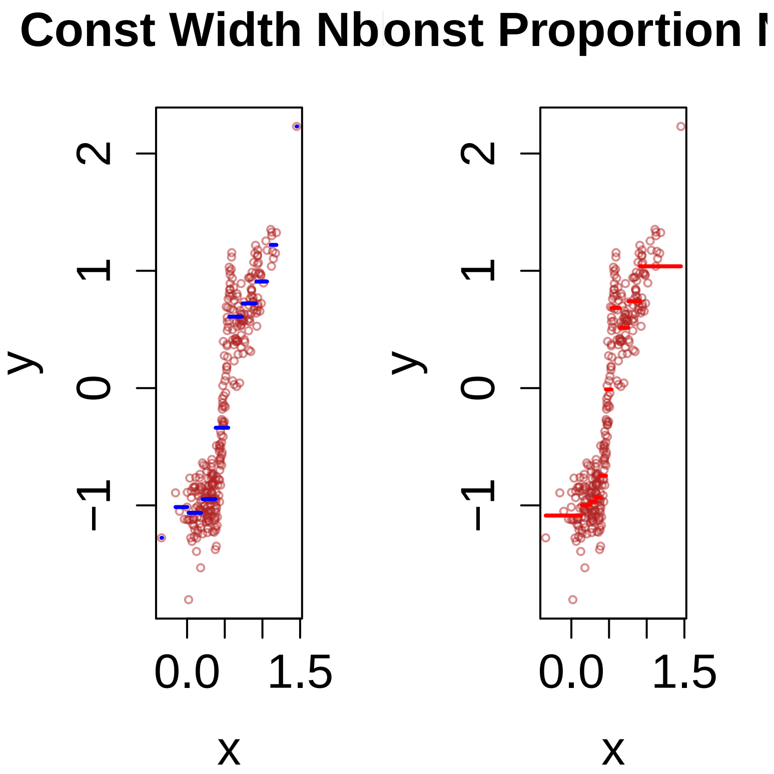
```
plot_fitted_vals <- function(locals, nbhds, x, mu, ...)
{
    for (i in 1:length(locals))
    {
        index <- nbhds == locals[i]
        newx <- x[index]
        newmu <- mu[index]
        Xorder <- order(newx)
        if (length(newx)==1)
        {
            lines(rep(newx[Xorder],2),
                rep(newmu[Xorder],2),
                ...)
        }
        else
        {
            lines(newx[Xorder], newmu[Xorder], ...)
        }
    }
}
```

Piecewise Constant.

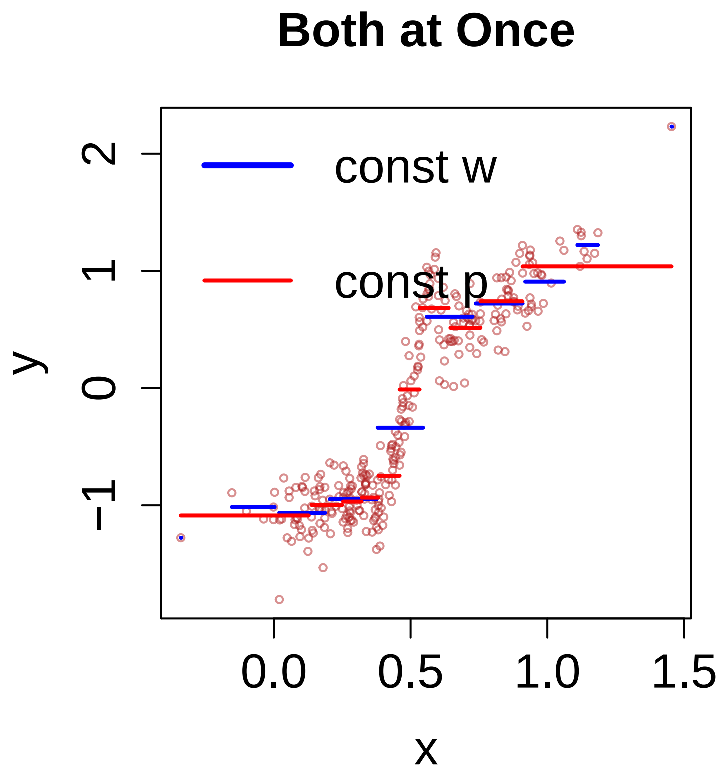
```
get_fitted_vals.constant <- function(locals, nbhds)
{
    fitted_vals <- numeric(length(x))
    for (i in 1:length(locals))
    {
        index <- nbhds==locals[i]
        fitted_vals[index] <- mean(y[index])
    }
    return(fitted_vals)
}
```

```
# Constant Width Neighborhoods
breaks_v <- seq(min(x), max(x), length.out=11)
nbhd_v <- cut(x, breaks=breaks_v, include.lowest=TRUE)
local_v <- levels(nbhd_v)
mu_v <- get_fitted_vals.constant(local_v, nbhd_v)
# Constant Proportion Neighborhoods
breaks_p <- unname(quantile(x, seq(0, 1, 0.1)))
nbhd_p <- cut(x, breaks=breaks_p, include.lowest=TRUE)
local_p <- levels(nbhd_p)
mu_p <- get_fitted_vals.constant(local_p, nbhd_p)
```

```
par(mfrow=c(1, 2))
do_scatter_plot(x, y, "Const Width Nbhd")
plot_fitted_vals(local_v, nbhd_v, x, mu_v, col="blue", lwd=2)
do_scatter_plot(x, y, "Const Proportion Nbhd")
plot_fitted_vals(local_p, nbhd_p, x, mu_p, col="red", lwd=2)
```



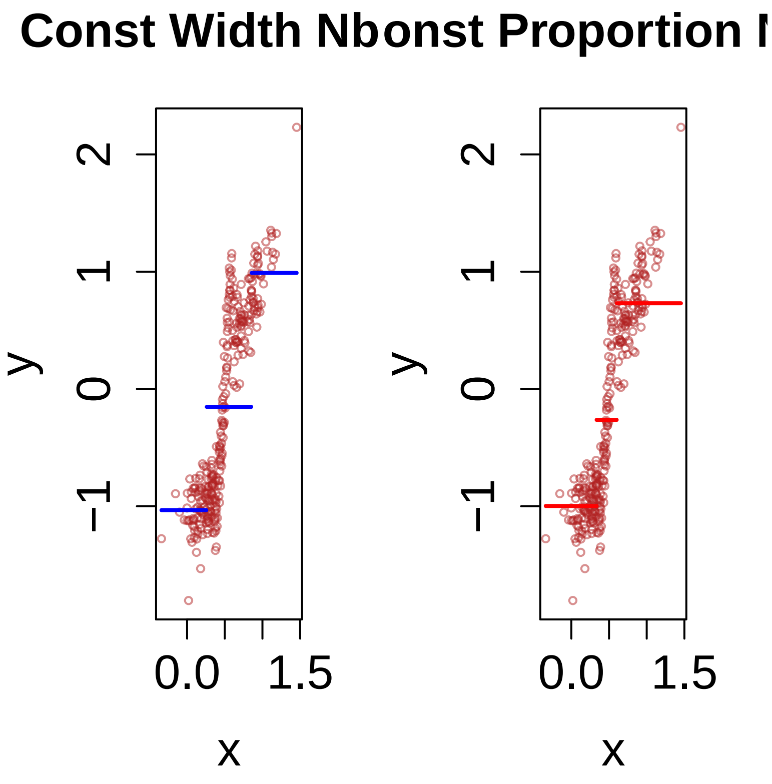
```
par(mfrow=c(1, 1))
do_scatter_plot(x, y, "Both at Once")
plot_fitted_vals(local_v, nbhd_v, x, mu_v, col="blue", lwd=2)
plot_fitted_vals(local_p, nbhd_p, x, mu_p, col="red", lwd=2)
legend("topleft", bty="n", cex=1.5,
       legend=c("const w", "", "const p"),
       col=c("blue","","red"), lty=c(1,0,1), lwd=c(3,0,2),
       text.width=2
      )
```


Both at Once
const w
const p
y
2
1
0
−1
0.0
0.5
1.0
1.5
x

Larger Neighbourhoods.

```
# Constant Width Neighborhoods
breaks_v <- seq(min(x),max(x), length.out=4)
nbhd_v <- cut(x, breaks=breaks_v, include.lowest=TRUE)
local_v <- levels(nbhd_v)
mu_v <- get_fitted_vals.constant(local_v, nbhd_v)
# Constant Proportion Neighborhoods
breaks_p <- unname(quantile(x, seq(0, 1, 1/3)))
nbhd_p <- cut(x, breaks=breaks_p, include.lowest=TRUE)
local_p <- levels(nbhd_p)
mu_p <- get_fitted_vals.constant(local_p, nbhd_p)
```

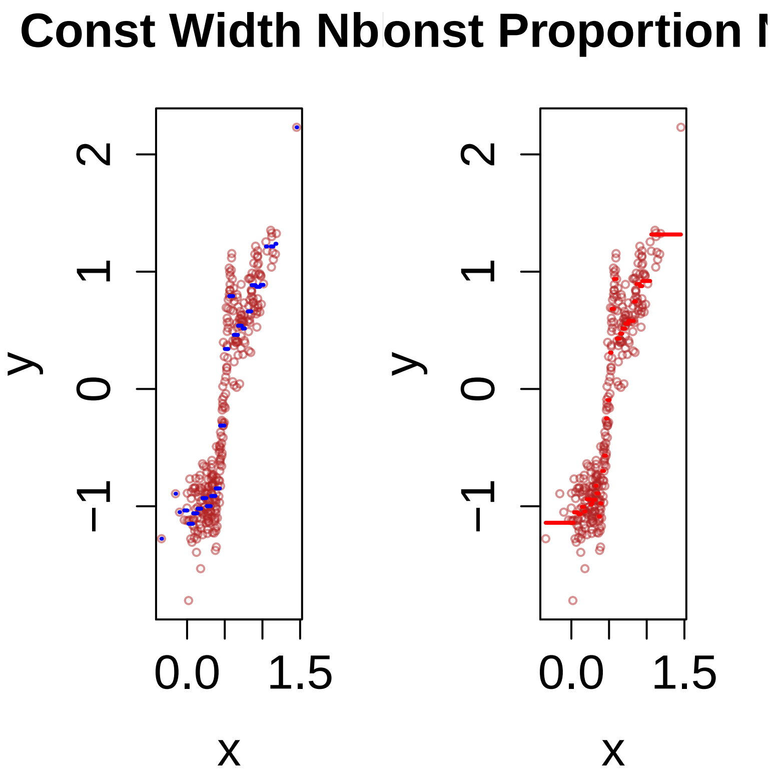
```
par(mfrow=c(1, 2))
do_scatter_plot(x, y, "Const Width Nbhd")
plot_fitted_vals(local_v, nbhd_v, x, mu_v, col="blue", lwd=2)
do_scatter_plot(x, y, "Const Proportion Nbhd")
plot_fitted_vals(local_p, nbhd_p, x, mu_p, col="red", lwd=2)
```



Smaller Neighbourhoods.

```
# Constant Width Neighborhoods
breaks_v <- seq(min(x),max(x), length.out=31)
nbhd_v <- cut(x, breaks=breaks_v, include.lowest=TRUE)
local_v <- levels(nbhd_v)
mu_v <- get_fitted_vals.constant(local_v, nbhd_v)
# Constant Proportion Neighborhoods
breaks_p <- unname(quantile(x, seq(0, 1, 1/30)))
nbhd_p <- cut(x, breaks=breaks_p, include.lowest=TRUE)
local_p <- levels(nbhd_p)
mu_p <- get_fitted_vals.constant(local_p, nbhd_p)
```

```
par(mfrow=c(1, 2))
do_scatter_plot(x, y, "Const Width Nbhd")
plot_fitted_vals(local_v, nbhd_v, x, mu_v, col="blue", lwd=2)
do_scatter_plot(x, y, "Const Proportion Nbhd")
plot_fitted_vals(local_p, nbhd_p, x, mu_p, col="red", lwd=2)
```



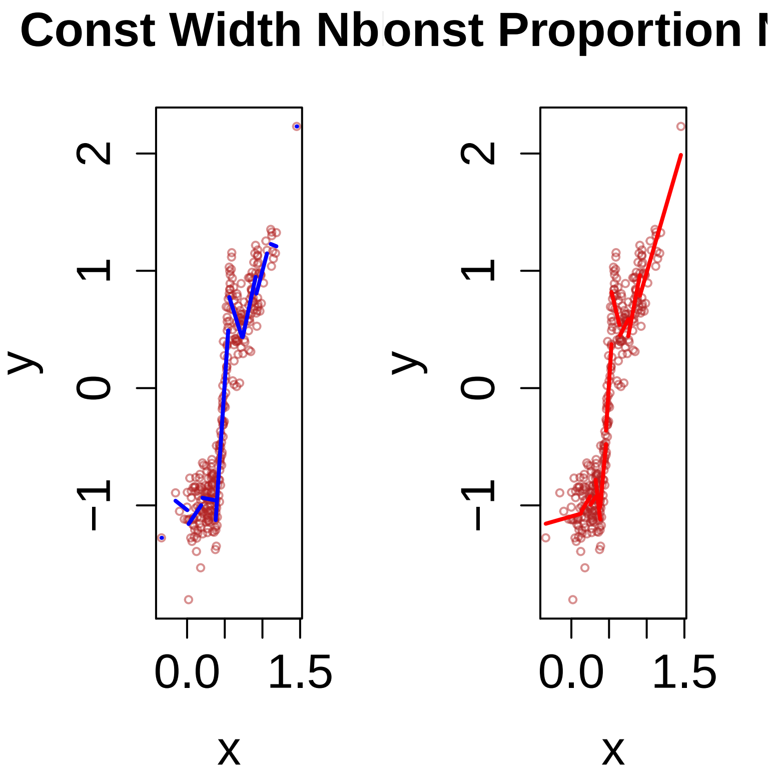
Piecewise Linear Model.

Without Continuity Constraint.

```
get_fitted_vals.linear <- function(locals, nbhds)
{
    mu <- vector(mode="numeric", length=length(x))
    for (i in 1:length(locals))
    {
        index <- nbhds == locals[i]
        local_x <- x[index]
        local_y <- y[index]
        local_data <- data.frame(x=local_x, y=local_y)
        local_fit <- lm(y~x, data=local_data)
        mu[index] <- predict(local_fit)
    }
    return(mu)
}
```

```
# Constant Width Neighborhoods
breaks_v <- seq(min(x),max(x), length.out=11)
nbhd_v <- cut(x, breaks=breaks_v, include.lowest=TRUE)
local_v <- levels(nbhd_v)
mu_v <- get_fitted_vals.linear(local_v, nbhd_v)
# Constant Proportion Neighborhoods
breaks_p <- unname(quantile(x, seq(0, 1, length.out=11)))
nbhd_p <- cut(x, breaks=breaks_p, include.lowest=TRUE)
local_p <- levels(nbhd_p)
mu_p <- get_fitted_vals.linear(local_p, nbhd_p)
```

```
par(mfrow=c(1, 2))
do_scatter_plot(x, y, "Const Width Nbhd")
plot_fitted_vals(local_v, nbhd_v, x, mu_v, col="blue", lwd=2)
do_scatter_plot(x, y, "Const Proportion Nbhd")
plot_fitted_vals(local_p, nbhd_p, x, mu_p, col="red", lwd=2)
```


Const Width Nb
onst Proportion N
y
2
1
0
−1
0.0
1.5
x
y
2
1
0
−1
0.0
1.5
x

With Continuity Constraint.

```
library(segmented)
piecewise.lm <- lm(y ~ x)
```

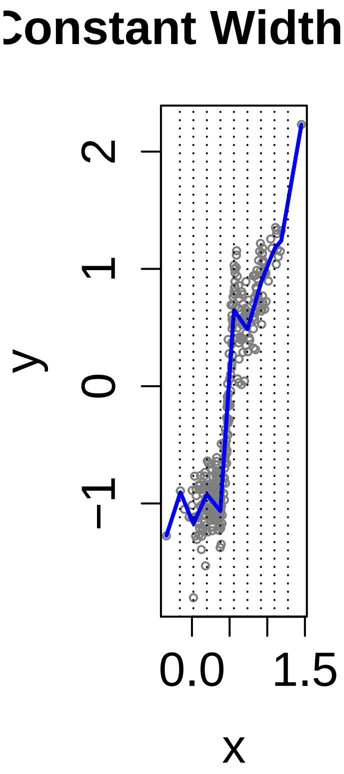
```
par(mfrow=c(1, 2))
# have to provide estimates for breakpoints.
# after looking a the data,
segments = breaks_v[-c(1, length(breaks_v))]
my.seg <- segmented(piecewise.lm,
                    seg.Z = ~ x,
                    psi = segments,
                    control=seg.control(it.max=0)
                   )
summary(my.seg)
```

```
##
## Call:
## lm(formula = y ~ x + U1.x + U2.x + U3.x + U4.x + U5.x + U6.x +
##     U7.x + U8.x + U9.x, data = mfExt)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.63204 -0.14588  0.00549  0.14438  0.53739
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.5515     0.3838  -1.437  0.15179
## x             2.1327     1.5357   1.389  0.16599
## U1.x         -3.6884     2.3626  -1.561  0.11959
## U2.x          2.9589     1.3435   2.202  0.02843 *
## U3.x         -2.2051     0.7114  -3.100  0.00213 **
## U4.x         10.3682     0.5714  18.144  < 2e-16 ***
## U5.x        -10.5071     0.6084 -17.270  < 2e-16 ***
## U6.x          3.1508     0.6959   4.527 8.74e-06 ***
## U7.x         -0.5558     0.9785  -0.568  0.57043
## U8.x         -0.9446     2.9476  -0.320  0.74884
## U9.x          4.4626     4.4508   1.003  0.31687
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2119 on 289 degrees of freedom
## Multiple R-squared:  0.9396, Adjusted R-squared:  0.9375
## F-statistic: 449.9 on 10 and 289 DF,  p-value: < 2.2e-16
```

```
my.fitted <- fitted(my.seg)
FakeData2 <- data.frame(x,y,y.hat=my.fitted)
plot(x, y,
     col="grey50", pch=1, cex=0.5,
     main="Constant Width Nbhd",
     cex.axis=1.5, cex.main=1.5, cex.lab=1.5
    )
lines(y.hat[order(x)]~sort(x), col="blue", data=FakeData2, lwd=2)
abline(v=segments,lty=3,col="black")
```

```
print("Degreed of Freedom is: ", my.seg$rank)
```

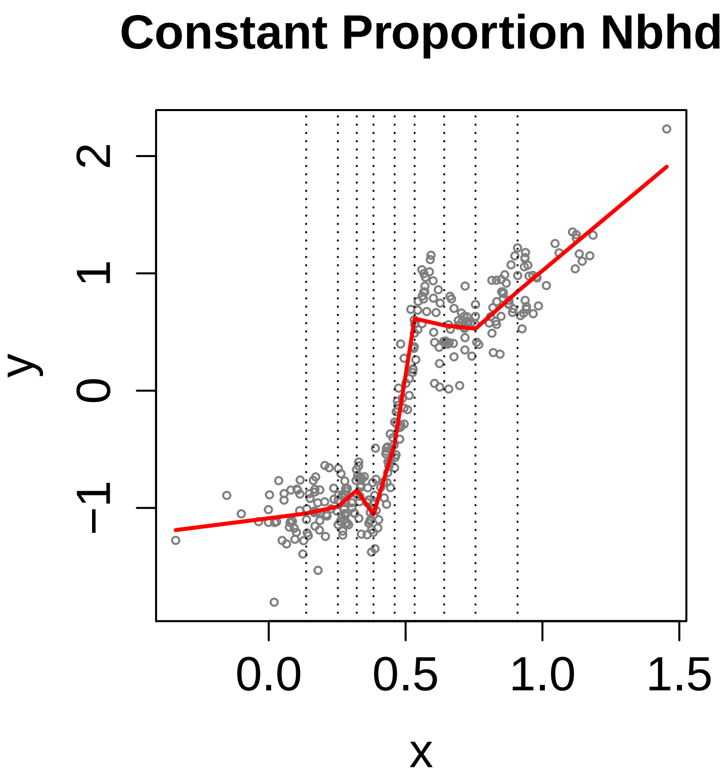
```
## [1] "Degreed of Freedom is: "
```



```
segments = breaks_p[-c(1, length(breaks_p))]
my.seg <- segmented(piecewise.lm,
                    #seg.Z = ~ x,
                    psi = segments,
                    control=seg.control(it.max=0)
                   )
summary(my.seg)
```

```
##
## Call:
## lm(formula = y ~ x + U1.x + U2.x + U3.x + U4.x + U5.x + U6.x +
##     U7.x + U8.x + U9.x, data = mfExt)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.72177 -0.15138 -0.00711  0.15437  0.57222
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.08772    0.04052 -26.846  < 2e-16 ***
## x            0.29726    0.35761   0.831 0.406526
## U1.x         0.20636    0.84579   0.244 0.807414
## U2.x          1.51154    1.64342   0.920 0.358466
## U3.x         -5.39759    2.21394  -2.438 0.015371 *
## U4.x         11.17963    2.04468   5.468 9.87e-08 ***
## U5.x          6.80578    1.78725   3.808 0.000171 ***
## U6.x        -15.12726    1.60347  -9.434  < 2e-16 ***
## U7.x          0.26693    1.37433   0.194 0.846136
## U8.x          2.32155    1.17820   1.970 0.049743 *
## U9.x         -0.11198    0.69087  -0.162 0.871348
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2109 on 289 degrees of freedom
## Multiple R-squared:  0.9402, Adjusted R-squared:  0.9381
## F-statistic: 454.4 on 10 and 289 DF,  p-value: < 2.2e-16
```

```
my.fitted <- fitted(my.seg)
FakeData2 <- data.frame(x,y,y.hat=my.fitted)
plot(x, y,
     col="grey50", pch=1, cex=0.5,
     main="Constant Proportion Nbhd",
     cex.axis=1.5, cex.main=1.5, cex.lab=1.5
    )
lines(y.hat[order(x)]~sort(x),col="red",data=FakeData2, lwd=2)
abline(v=segments,lty=3,col="black")
```



```
print("Degreed of Freedom is: ", my.seg$rank)
```

```
## [1] "Degreed of Freedom is: "
```

Piecewise Cubic Model.

```r
get_fitted_vals.cubic <- function(locals, nbhds, degree=3)
{
    mu <- vector(mode="numeric", length=length(x))
    for (i in 1:length(locals))
    {
        index <- nbhds == locals[i]
        local_x <- x[index]
        local_y <- y[index]
        local_data <- data.frame(x=local_x, y=local_y)
        local_fit <- lm(y~poly(x,degree=degree), data=local_data)
        mu[index] <- predict(local_fit)
    }
    return(mu)
}
```

```r
breaks_v <- seq(min(x), max(x), length.out=11)
nbhd_v <- cut(x, breaks= breaks_v, include.lowest=TRUE)
local_v <- levels(nbhd_v)

count <- vector("numeric", length = length(breaks_v)-1)
for (i in 1:length(count)) { count[i] <- sum(nbhd_v==local_v[i]) }
print(count)
```

```
##  [1]  1  6 43 67 68 51 35 20  8  1
```

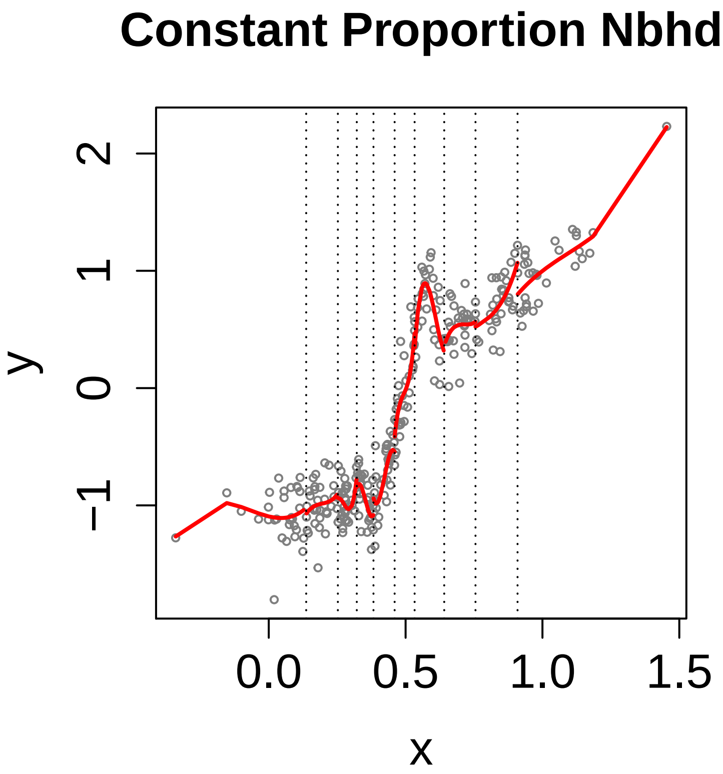
```r
breaks_p <- unname(quantile(x, seq(0, 1, 0.1)))
nbhd_p <- cut(x, breaks=breaks_p, include.lowest=TRUE)
local_p <- levels(nbhd_p)

count <- vector("numeric", length = length(breaks_v)-1)
for (i in 1:length(count)) { count[i] <- sum(nbhd_p==local_p[i]) }
print(count)
```

```
##  [1] 30 30 30 30 30 30 30 30 30 30
```

```r
# Now the fixed width will get us in trouble by
# not having enough points, so just the fixed proportion
mu_p <- get_fitted_vals.cubic(local_p, nbhd_p, degree=3)
```

```r
plot(x, y,
     col="grey50", pch=1, cex=0.5,
     main="Constant Proportion Nbhd",
     cex.axis=1.5, cex.main=1.5, cex.lab=1.5
    )
plot_fitted_vals(local_p, nbhd_p, x, mu_p, col="red", lwd=2)
abline(v=segments,lty=3,col="black")
```


Constant Proportion Nbhd
y
2
1
0
−1
0.0
0.5
1.0
1.5
x