Functional Sensitivity

Eamonn O'Brien

Tuesday, May 26, 2015

Program is provided which fits 11 models to single predictor and uses sum of square of residual as model selection criteria

1 Functional Sensitivity function

```
func.sens <- function (x, y, model, spec, print.plot=1) {

# Define models

  if (model %in% 1 ) {mod="Linear Y=a+bX"}
  if (model %in% 2 ) {mod="Exponential Y=exp(a+bX)"}
  if (model %in% 3 ) {mod="Reciprocal-Y Y=1/(a+bX)"}
  if (model %in% 4 ) {mod="Reciprocal-X Y=a+b/X"}
  if (model %in% 5 ) {mod="Double Reciprocal Y=1/(a+b/X)"}
  if (model %in% 6 ) {mod="Logarithmic-X Y=a+b(log(X))"}
  if (model %in% 7 ) {mod="Multiplicative Y=aX^b"}
  if (model %in% 8 ) {mod="Square Root-X Y=a+b(sqrt(X))"}
  if (model %in% 9 ) {mod="Square Root-Y Y=(a+bX)^2"}
  if (model %in% 10) {mod="S-curve Y=exp(a+b/X)"}
  if (model %in% 11) {mod="Square X and Y Y^2=a+X^2/b"}

# transformation of data for 11 models

    ty1 <- y;       tx1 <- x
    ty2 <- log(y);  tx2 <- x
    ty3 <- 1/y;     tx3 <- x
    ty4 <- y;       tx4 <- 1/x
    ty5 <- 1/y;     tx5 <- 1/x
    ty6 <- y;       tx6 <- log(x)
    ty7 <- log(y);  tx7 <- log(x)
    ty8 <- y;       tx8 <- sqrt(x)
    ty9 <- sqrt(y); tx9 <- x
    ty10 <- log(y); tx10 <- 1/x
    ty11 <- y^2;    tx11 <- x^2

# save the original data

    x1 <- x
    y1 <- y

# transform using the selected model (1 - 11)

    x <- eval(parse(text=(paste("tx", model, sep="")) ))
    y <- eval(parse(text=(paste("ty", model, sep="")) ))
```

```
# summary statistics of the independent variable

    n <- length(x)
    txbar <- mean(x)
    txstd <- sd(x)

# run regression on the transformed data grab slope intercept

    f <- lm(y~x)
    intercept <- coef(f)[1][[1]]
    slope <- coef(f)[2][[1]]

# R2 on the transformed x and y,  an idea but not used

    v1 <- unlist(cor.test(x,y)$estimate^2)[1][[1]]

# obtain the predictions

    p <- predict.lm(f, interval="confidence")

# transform the predicted values & 95%CI back to original scale

    if (model %in% c(2,7,10)) {p <- exp(p)}
    if (model %in% c(3,5)  )  {p <- 1/p}
    if (model %in% c(9)    )  {p <- p^2}
    if (model %in% c(11)   )  {p <- p^.5}

# calculate residual squared
# residuals original y and transformed back predicted values

    r <- (y1-p[,1])^2

# R2 on the original x and transformed back predicted y, an idea but not used

    v2 <- unlist(cor.test(x1,p[,1])$estimate^2)[1][[1]]

# residual sum of squares, this will be used to judge best model

   ssr <- sum(r, na.rm=T)

   # transform the response that we will read back

   tyspec <- spec
   if (model %in% c(2,7,10)) {tyspec <- log(tyspec)}
   if (model %in% c(3,5)  )  {tyspec <- 1/tyspec}
   if (model %in% c(9)    )  {tyspec <- sqrt(tyspec)}
   if (model %in% c(11)   )  {tyspec <- tyspec ^2}

# grab the residual standard deviation

    rsd2<-as.data.frame(anova(f))[2,3]^.5

# read back on transformed scale
```

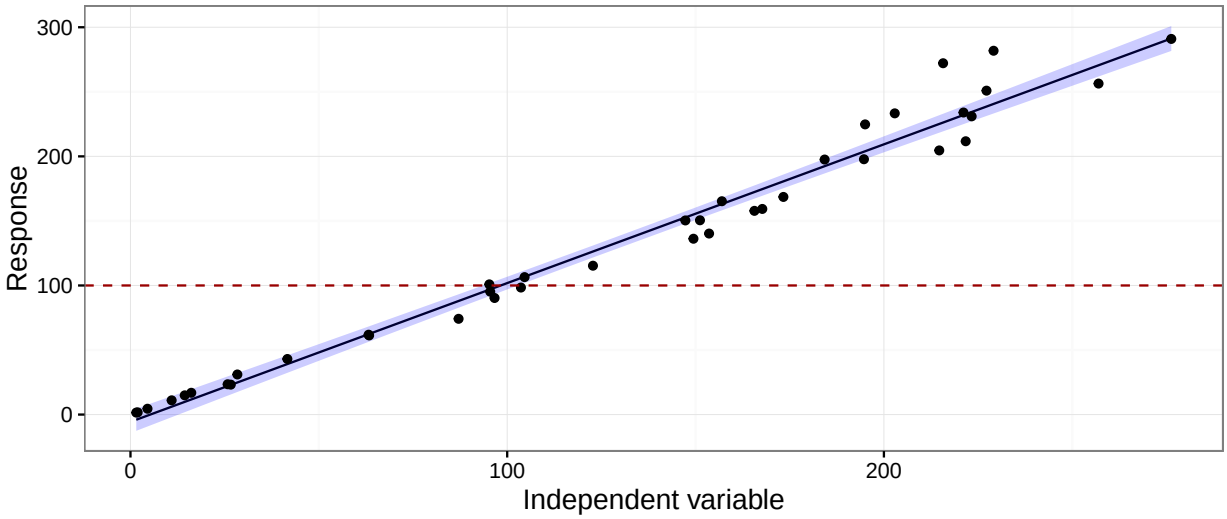
```
    mse <- rsd2^2
    t <- qt(0.975, n-2)
    a <- t^2*mse/((n-1)*txstd^2)-slope^2
    b <- 2*slope*(tyspec-intercept-slope*txbar)
    c <- t^2*mse/n-(tyspec-intercept-slope*txbar) ^2
    txpre <- (tyspec-intercept)/slope
    txup <-  (-b+sqrt(b^2-4*a*c))/(2*a) + txbar
    txlow <- (-b-sqrt(b^2-4*a*c))/(2*a) + txbar

# transform the read back estimates to the original scale

   if (model %in% c(4,5,10)) {txpre <- 1/txpre; txup <- 1/txup; txlow <- 1/txlow}
   if (model %in% c(6,7)  )  {txpre <- exp(txpre); txup <- exp(txup); txlow <- exp(txlow)}
   if (model %in% c(8)    )  {txpre <- txpre^2;  txup <- txup^2;  txlow <- txlow^2}
   if (model %in% c(11)   )  {txpre <- txpre^.5; txup <- txup^.5; txlow <- txlow^.5}

# ensure order of limits is correct

    limits <- sort(c(txlow,txup))
    txlow <- limits[1]
    txup <-  limits[2]

# help with plotting

    ymin <- min(y)
    ymax <- max(y)
    ystep <- (ymax-ymin)/8
    ymin1 <-  ymin-ystep
    ymax1 <-  ymax+ystep

    xmin <- min(x)
    xmax <- max(x)
    xstep <- (xmax-xmin)/8
    xmin1 <-  xmin-xstep
    xmax1 <-  xmax+xstep

# put all pertinent data together, original data and predicted with 95%CI

    foo <- data.frame(cbind(x=x1,obsy=y1, pred= p[,1], p2a=p[,2], p3=p[,3]))
    foo <- foo[order(foo$obsy),]

# plot and present the estimated read back

  p1 <- ggplot(foo, aes(x=x,y=pred)) +
    geom_line( ) +
    geom_ribbon(data=foo , aes(ymin= p2a,ymax= p3),alpha=0.2,fill="blue") +
    geom_point(data=foo, aes(x=x ,y=obsy))
    scale_x_continuous(limits = c(xmin1, xmax1))
    scale_y_continuous(limits = c(xmin1, xmax1))

      p <- p1  + geom_hline(yintercept=spec, colour="#990000", linetype="dashed")

      p <- p + theme(axis.text.x = element_text(angle = 90, hjust = 1, size=13,color="darkred"))
```

```
  p <- p + scale_color_manual(values=c("Red","blue"))
  p <- p + theme_bw()
  p <- p + xlab('Independent variable')
  p <- p + ylab('Dependent variable')

  p <- p + ggtitle(paste("Model for the curve #",model," ",mod,", ","\nExploration of model fitting
# ", R2 transformed x, y ",p2(v1),", R2 x, trans. back pred. y",p2(v2),
 sep=" ")) + theme(plot.title = element_text(lineheight=1, face="bold", color="black", size=11))

  p <- p + labs(x = "Independent variable", y = "Response")
  p <- p + theme(axis.title.y = element_text(size = rel(1.1), angle = 90))
  p <- p + theme(axis.title.x = element_text(size = rel(1.1), angle = 00))

  if (print.plot==1) {print(p)}


  return(ssr)

}
```

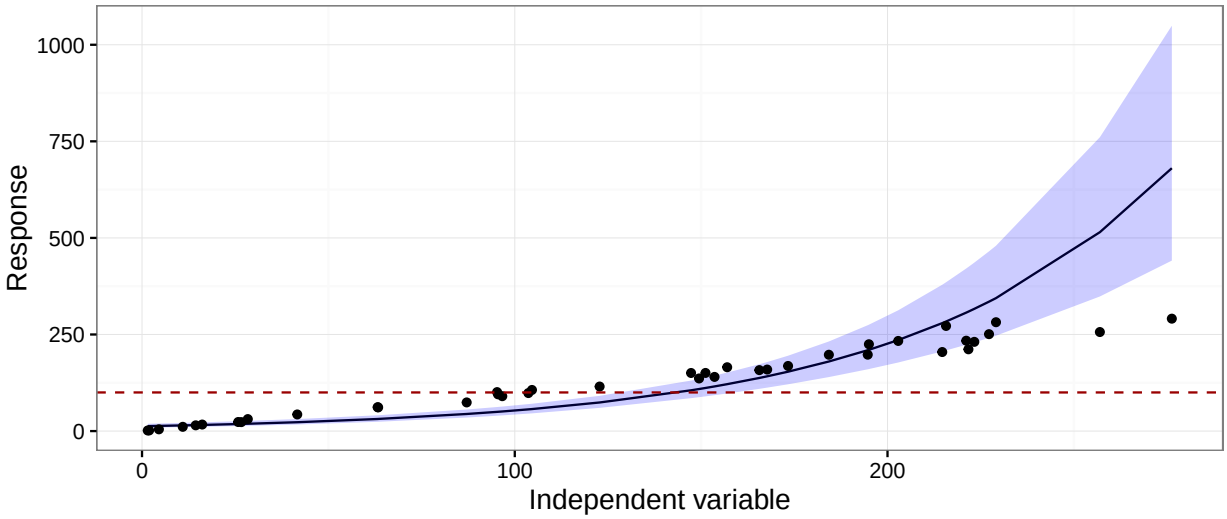
2 Test data 1

```
require(ggplot2)

toread <- "sample  _NAME_  COL1  COL2
A046  conc        122.766   115.304
A048    conc    221.714 211.704
A049    conc    223.287 231.001
A050    conc    276.281 290.975
A051    conc    229.109 281.813
A052    conc    156.985 165.255
A054    conc    147.296 150.360
A058    conc    87.098  74.180
A062    conc    103.648 98.389
A063    conc    173.321 168.583
A069    conc    214.687 204.615
H098    conc    195.010 224.759
H099    conc    202.874 233.319
H100    conc    227.235 250.898
H101    conc    221.122 233.964
H102    conc    256.975 256.384
H105    conc    215.708 272.139
H112    conc    151.172 150.507
H113    conc    167.717 159.274
H114    conc    165.622 157.793
M023    conc    1.537   1.541
M024    conc    194.695 197.779
M025    conc    95.258  100.862
M026    conc    10.932  11.033
M027    conc    14.404  14.902
M030    conc    4.521   4.530
M031    conc    16.144  16.884
M032    conc    96.648  90.294
M036    conc    28.378  31.051
M037    conc    26.629  23.187
M038    conc    63.352  61.304
M044    conc    41.647  43.043
N008    conc    25.798  23.489
N011    conc    1.907   1.705
T079    conc    63.212  61.854
T081    conc    95.499  95.207
T082    conc    104.613 106.458
T085    conc    149.441 136.205
T086    conc    153.592 140.220
T090    conc    184.267 197.540"

survey <- read.table(textConnection(toread), header = TRUE)

for (j in 1:11) {
 func.sens(x=survey$COL1, y=survey$COL2, model=j, spec=100)
}
```

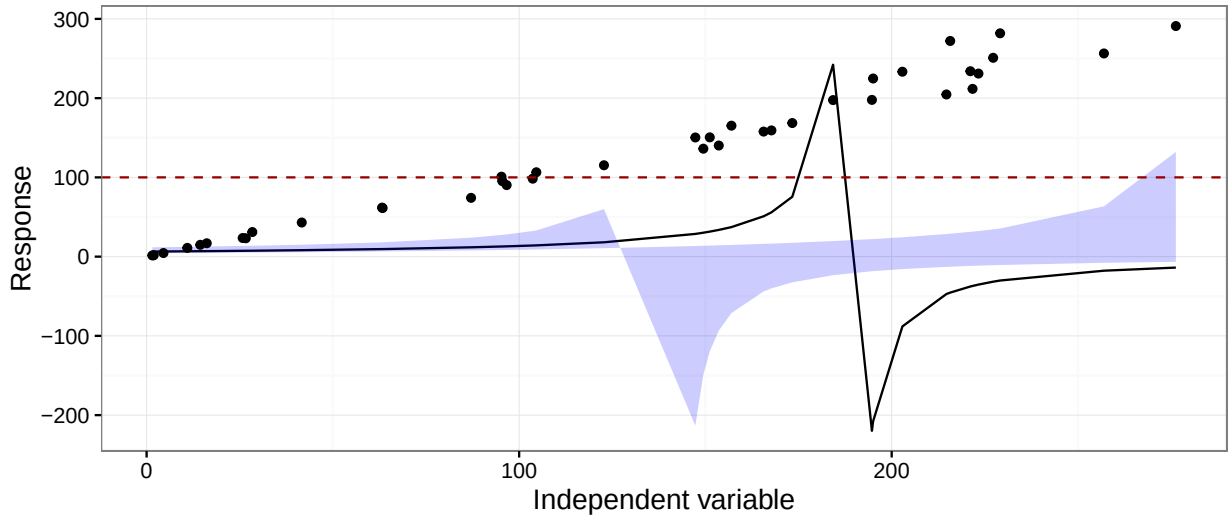
Model for the curve # 1 Linear Y=a+bX ,
Exploration of model fitting at response of 100 , 98.2904 and 95% CI: (93.5925 , 102.8185)
Residual sum of squares 8000.1736 , Residual standard deviation 14.5097



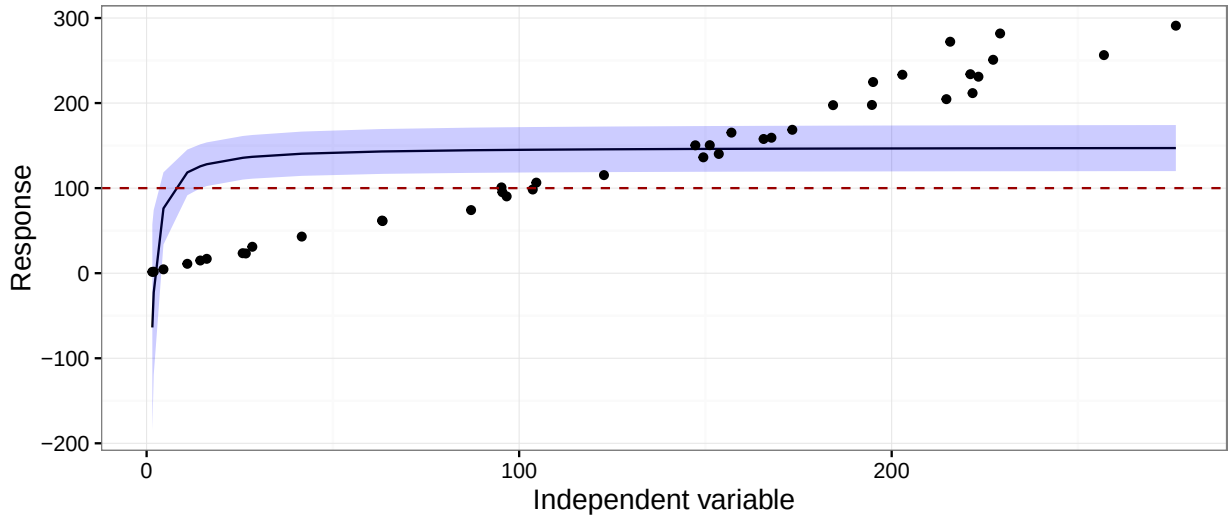
Model for the curve # 2 Exponential Y=exp(a+bX) ,
Exploration of model fitting at response of 100 , 143.5647 and 95% CI: (129.1227 , 158.9999)
Residual sum of squares 281528.4841 , Residual standard deviation 0.6519



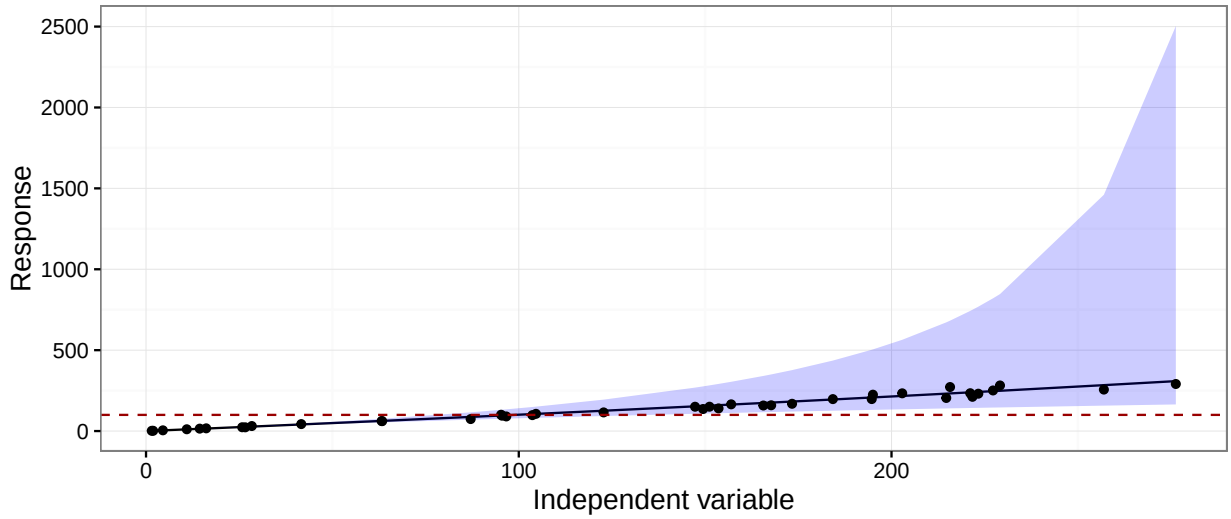
Model for the curve # 3 Reciprocal−Y Y=1/(a+bX) ,
Exploration of model fitting at response of 100 , 177.2108 and 95% CI: (130.8675 , 270.4737)
Residual sum of squares 1338842.6841 , Residual standard deviation 0.1204



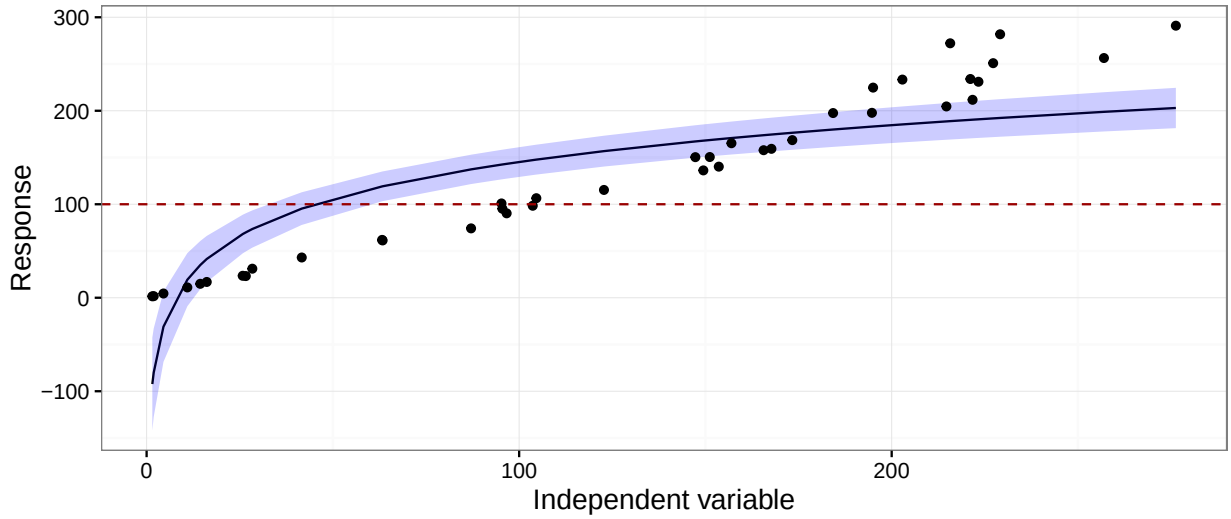
Model for the curve # 4 Reciprocal−X Y=a+b/X ,
Exploration of model fitting at response of 100 , 6.7509 and 95% CI: (2.9322 , 14.5270)
Residual sum of squares 242215.6512 , Residual standard deviation 79.8379



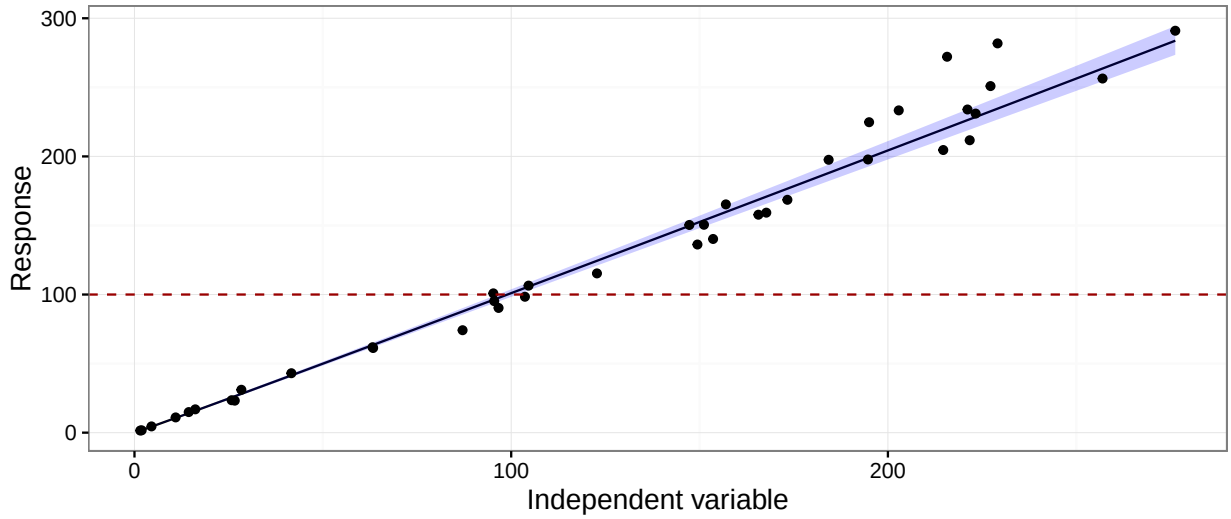
Model for the curve # 5 Double Reciprocal Y=1/(a+b/X) ,
Exploration of model fitting at response of 100 , 98.9373 and 95% CI: (78.2508 , 135.0492)
Residual sum of squares 7991.2452 , Residual standard deviation 0.0084



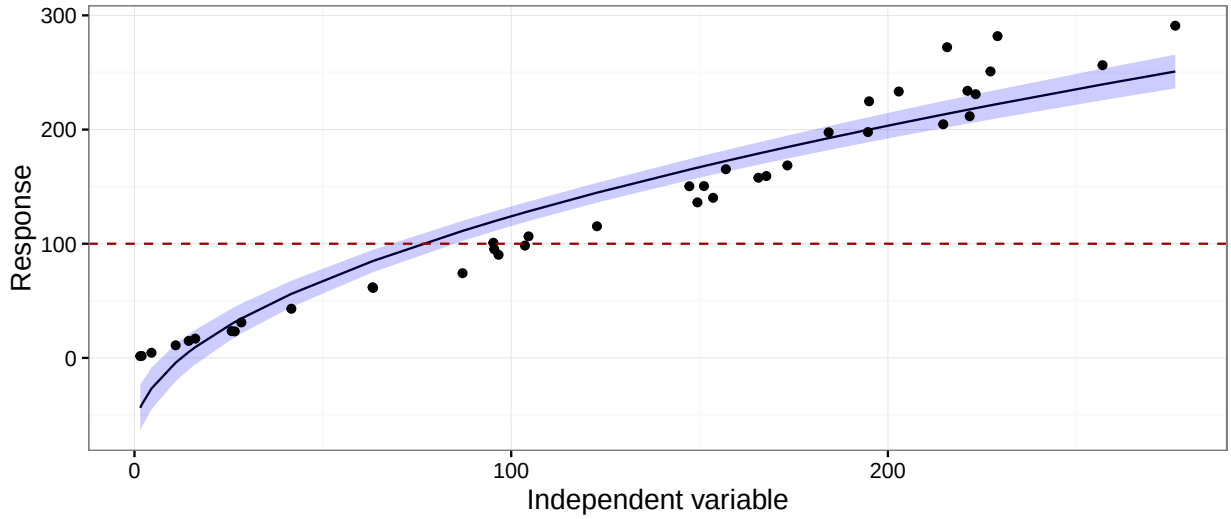
Model for the curve # 6 Logarithmic−X Y=a+b(log(X)) ,
Exploration of model fitting at response of 100 , 45.2021 and 95% CI: (32.3587 , 59.9250)
Residual sum of squares 91062.5663 , Residual standard deviation 48.9529



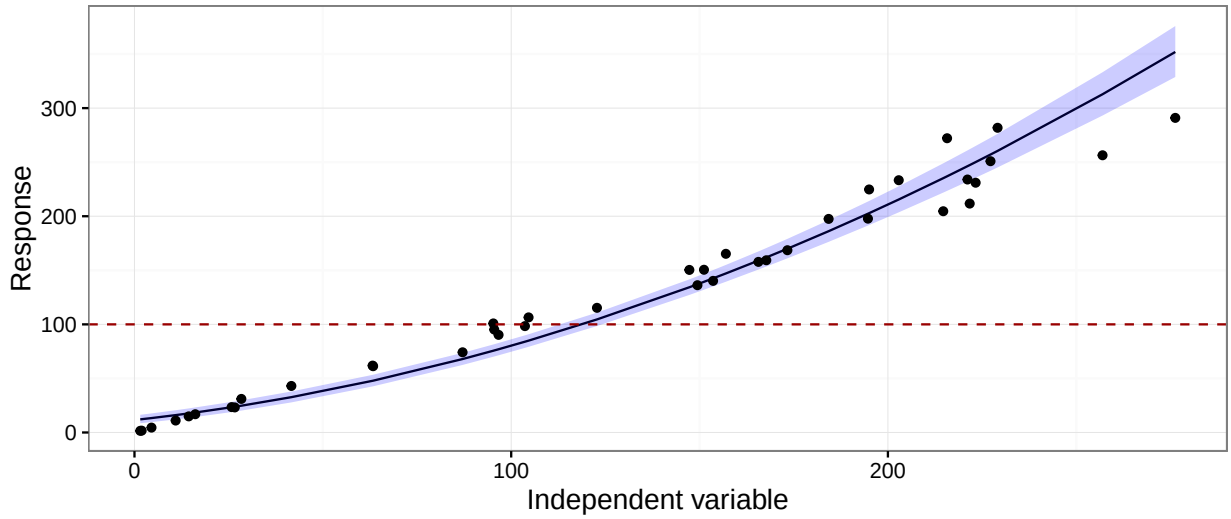
Model for the curve # 7 Multiplicative Y=aX^b ,
Exploration of model fitting at response of 100 , 99.0181 and 95% CI: (96.4279 , 101.6956)
Residual sum of squares 8546.8865 , Residual standard deviation 0.0833



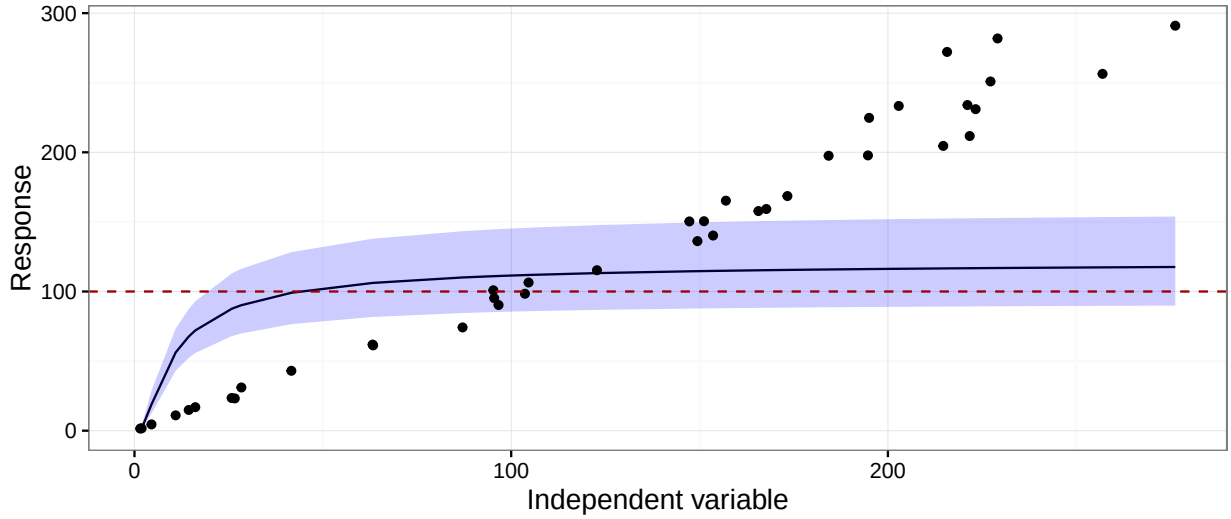
Model for the curve # 8 Square Root−X Y=a+b(sqrt(X)) ,
Exploration of model fitting at response of 100 , 76.4871 and 95% CI: (68.0042 , 84.8241)
Residual sum of squares 27263.3368 , Residual standard deviation 26.7854

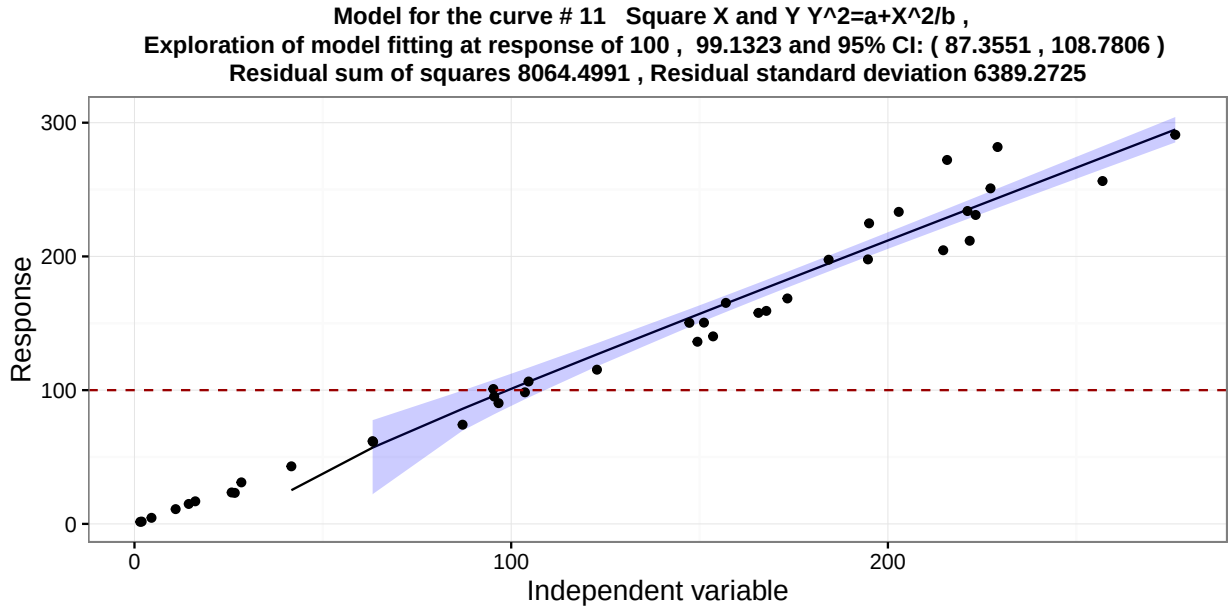


Model for the curve # 9 Square Root−Y Y=(a+bX)^2 ,
Exploration of model fitting at response of 100 , 118.7073 and 95% CI: (113.1718 , 124.1565)
Residual sum of squares 15909.0752 , Residual standard deviation 0.9452



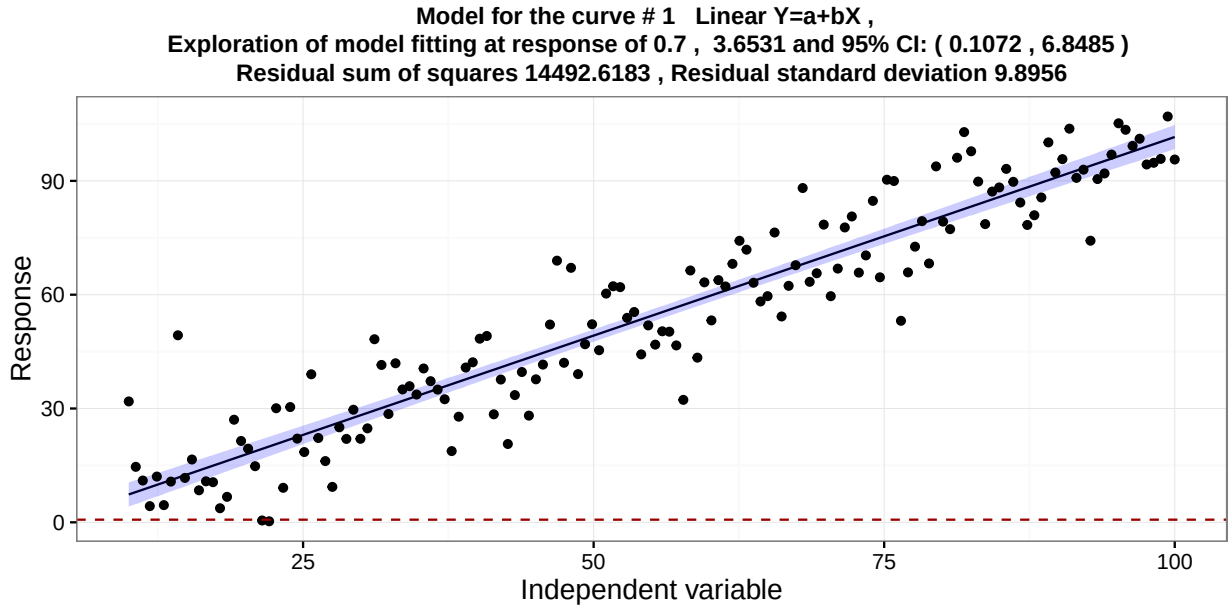
Model for the curve # 10 S−curve Y=exp(a+b/X) ,
Exploration of model fitting at response of 100 , 43.6212 and 95% CI: (−97.5223 , 18.8301)
Residual sum of squares 243260.6736 , Residual standard deviation 0.7919




Model for the curve # 11 Square X and Y Y^2=a+X^2/b ,
Exploration of model fitting at response of 100 , 99.1323 and 95% CI: (87.3551 , 108.7806)
Residual sum of squares 8064.4991 , Residual standard deviation 6389.2725
Response
300
200
100
0
0
100
200
Independent variable

3 Test data 2

```
n <- 150
x99 <- seq( from=10.001 , to=100, length.out=n)

#y99 <- abs((x99)+rnorm(n,0,4 ))^.5    # spec^2
#y99 <- abs((x99)+rnorm(n,0,4 ))^2     # spec^.5
#y99 <- abs((x99)+rnorm(n,0,4 ))^-1    # spec^-1
#y99 <- abs((x99)+rnorm(n,0,4 ))^-0.1  # spec^-(1/.1) == spec^-10 [model 7]
y99 <- abs((x99)+rnorm(n,0,10))       # linear

ssr <- rep(NA,11)

for (j in 1:11) {

  res <- func.sens(x=x99, y=y99, model=j, spec=.7, print.plot=0) # don't print
  ssr[j] <- res

}

func.sens(x=x99, y=y99, model=which(ssr==min(ssr)), spec=.7, print.plot = 1)
```



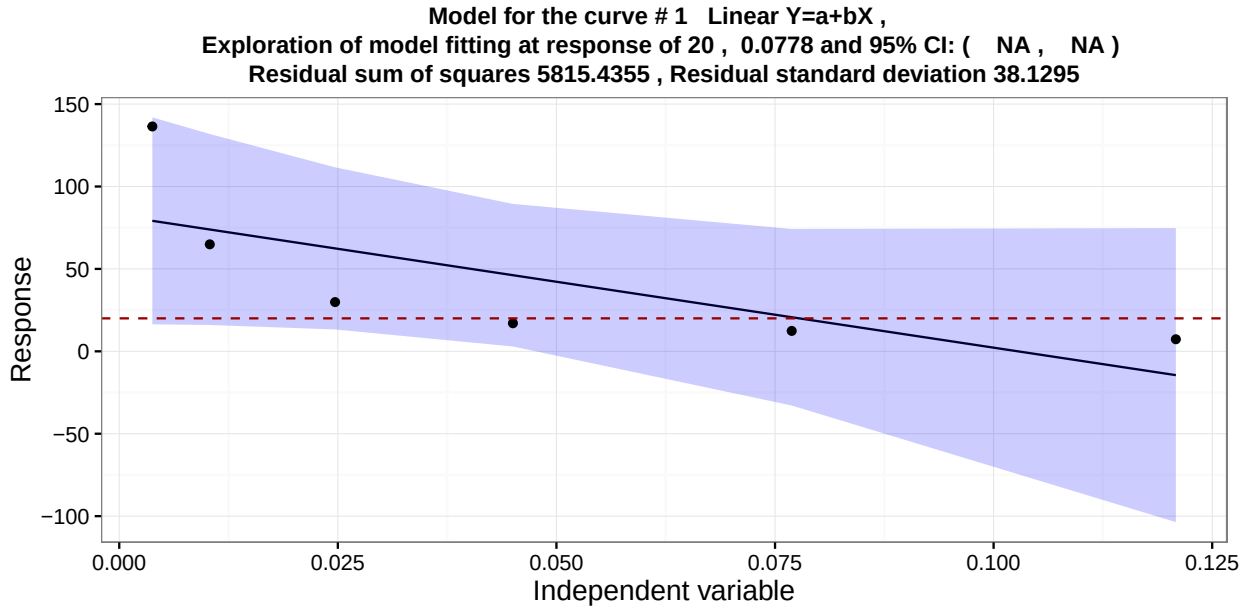

```
[1] 14492.62
```

```
cat(paste0("The best fitting model is #", which(ssr==min(ssr))))
```

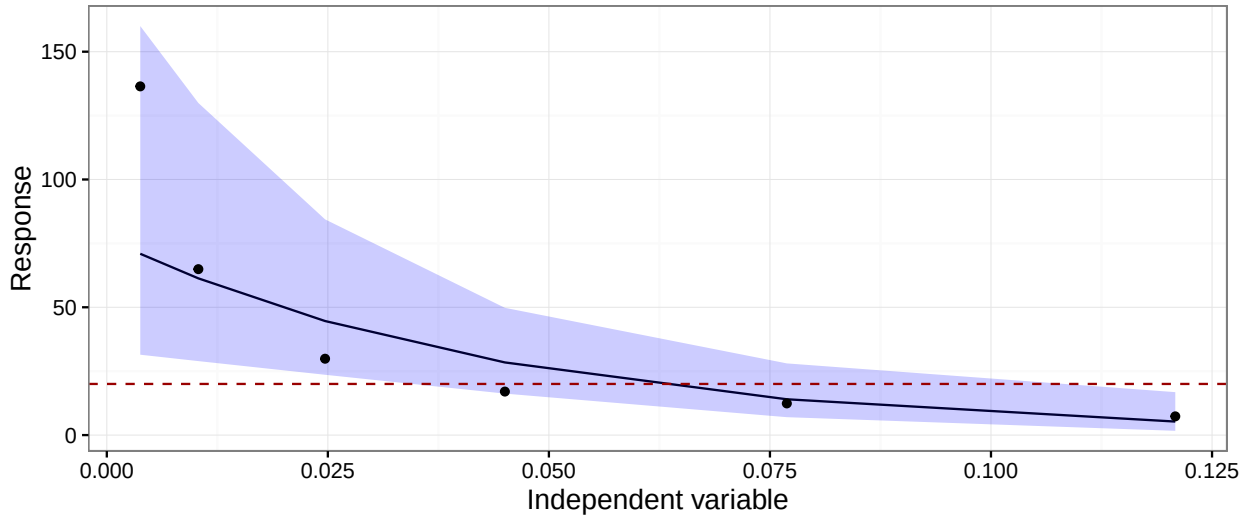
```
The best fitting model is #1
```

4 Test data 3

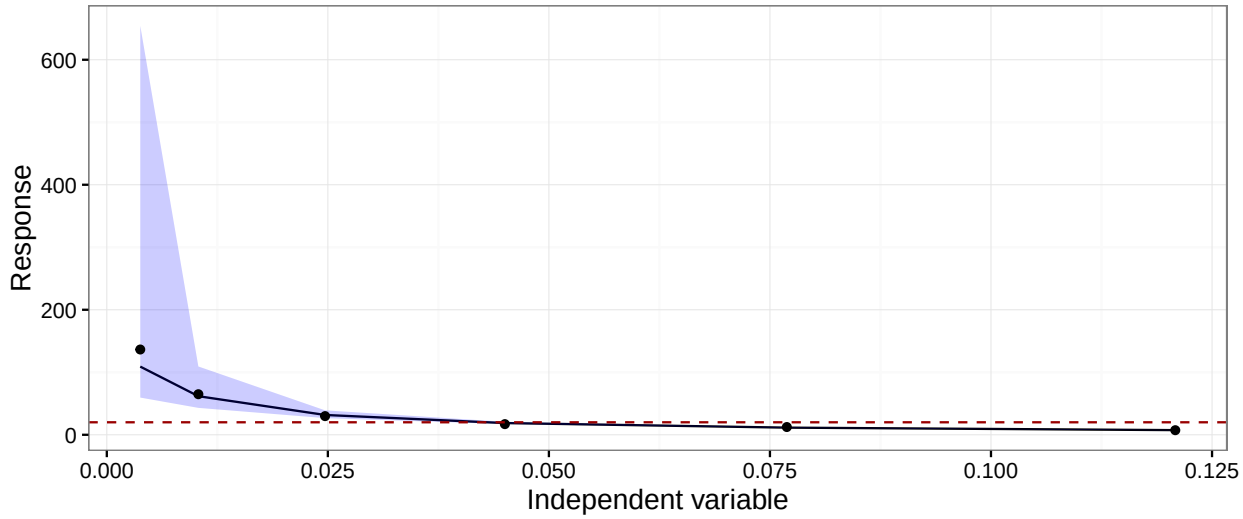
```
mu<-c(
  0.1208444444,
  0.0769101124,
  0.0450227273,
  0.0246853933,
  0.0103555556,
  0.0037840909)

cv<-c(
  7.3282535623,
  12.384142008,
  17.022255039,
  29.872271843,
  64.921152057,
  136.43305684)

for (j in 1:11) {
  func.sens(x=mu, y=cv, model=j, spec=20)
}
```



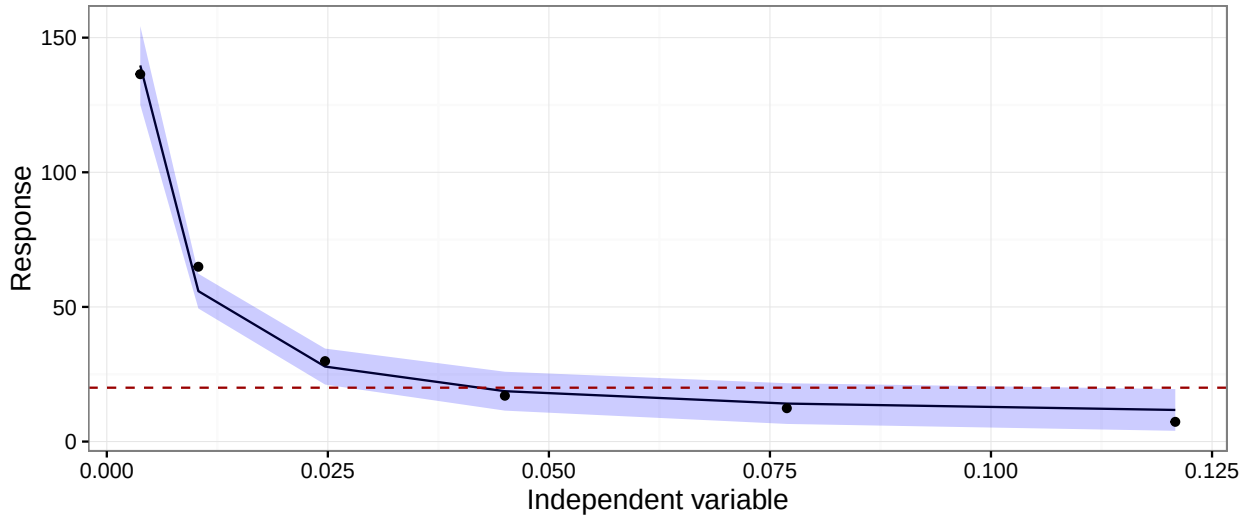
Model for the curve # 2 Exponential Y=exp(a+bX) ,
Exploration of model fitting at response of 20 , 0.0609 and 95% CI: (0.0345 , 0.1044)
Residual sum of squares 4657.3113 , Residual standard deviation 0.4941



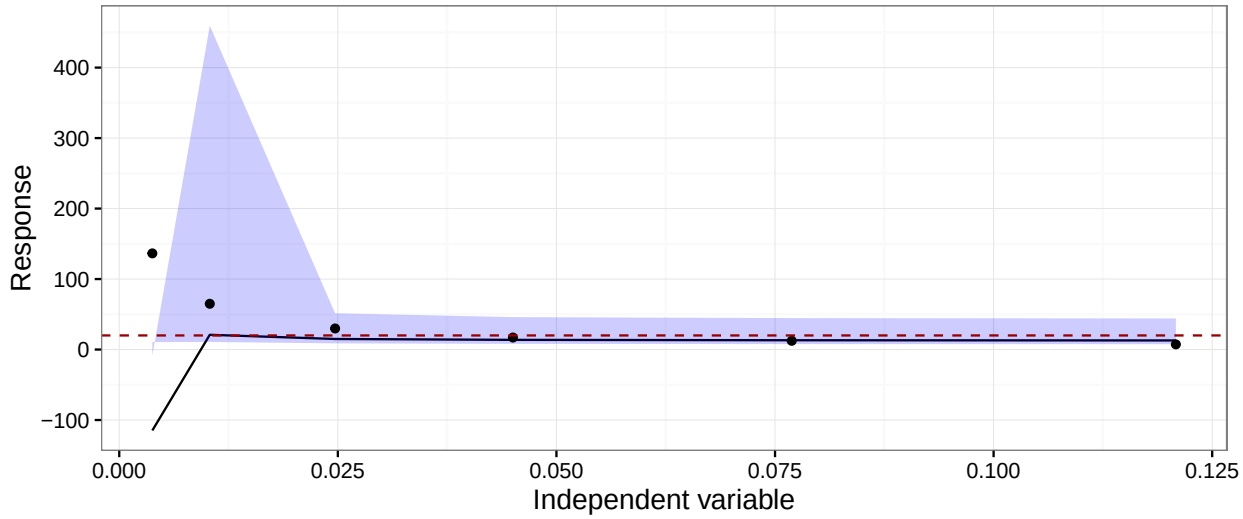
Model for the curve # 3 Reciprocal−Y Y=1/(a+bX) ,
Exploration of model fitting at response of 20 , 0.0419 and 95% CI: (0.0369 , 0.0468)
Residual sum of squares 765.1815 , Residual standard deviation 0.0046



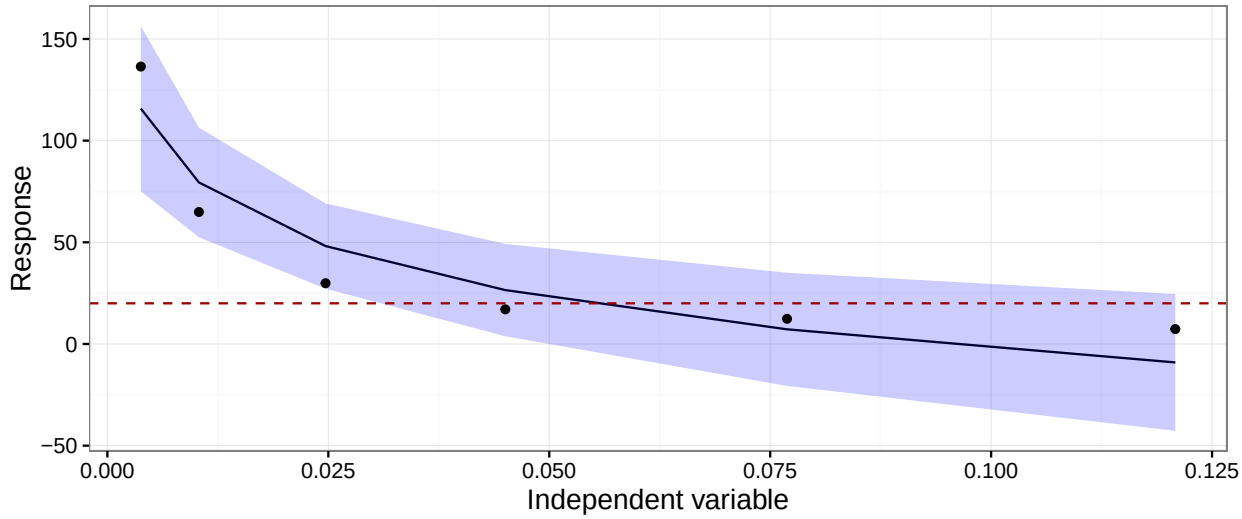
Model for the curve # 4 Reciprocal−X Y=a+b/X ,
Exploration of model fitting at response of 20 , 0.0403 and 95% CI: (0.0261 , 0.1067)
Residual sum of squares 121.8580 , Residual standard deviation 5.5195



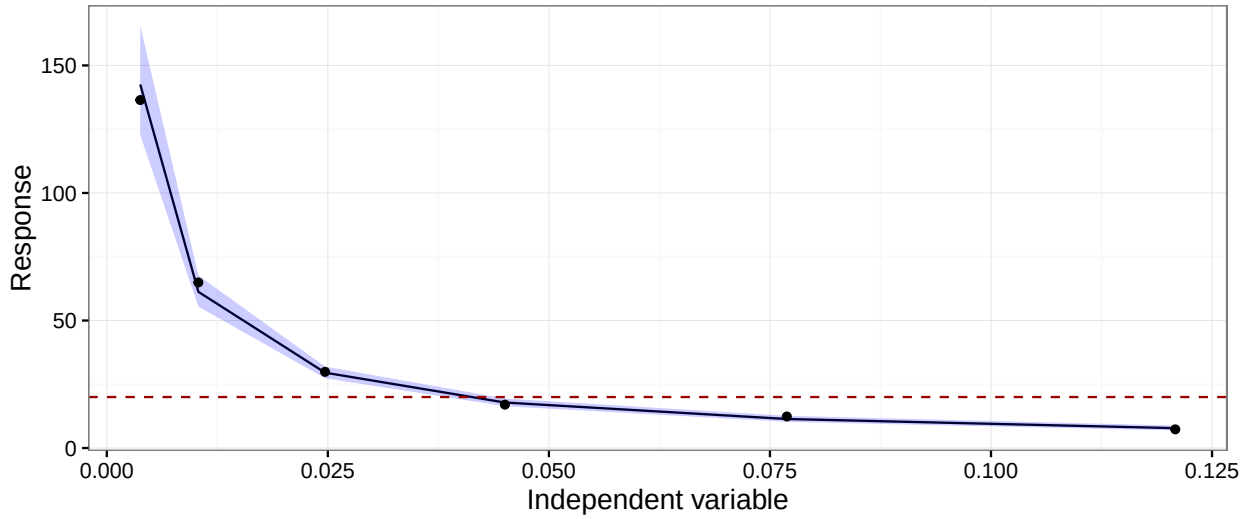
Model for the curve # 5 Double Reciprocal Y=1/(a+b/X) ,
Exploration of model fitting at response of 20 , 0.0111 and 95% CI: (NA , NA)
Residual sum of squares 65339.8134 , Residual standard deviation 0.0391



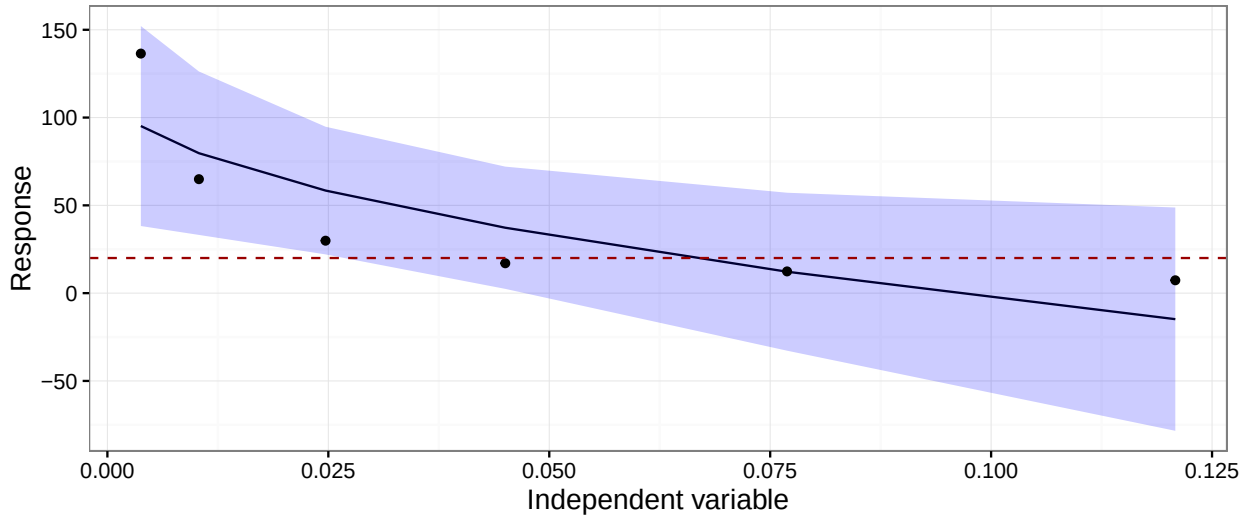
Model for the curve # 6 Logarithmic−X Y=a+b(log(X)) ,
Exploration of model fitting at response of 20 , 0.0539 and 95% CI: (0.0301 , 0.1490)
Residual sum of squares 1359.8382 , Residual standard deviation 18.4380



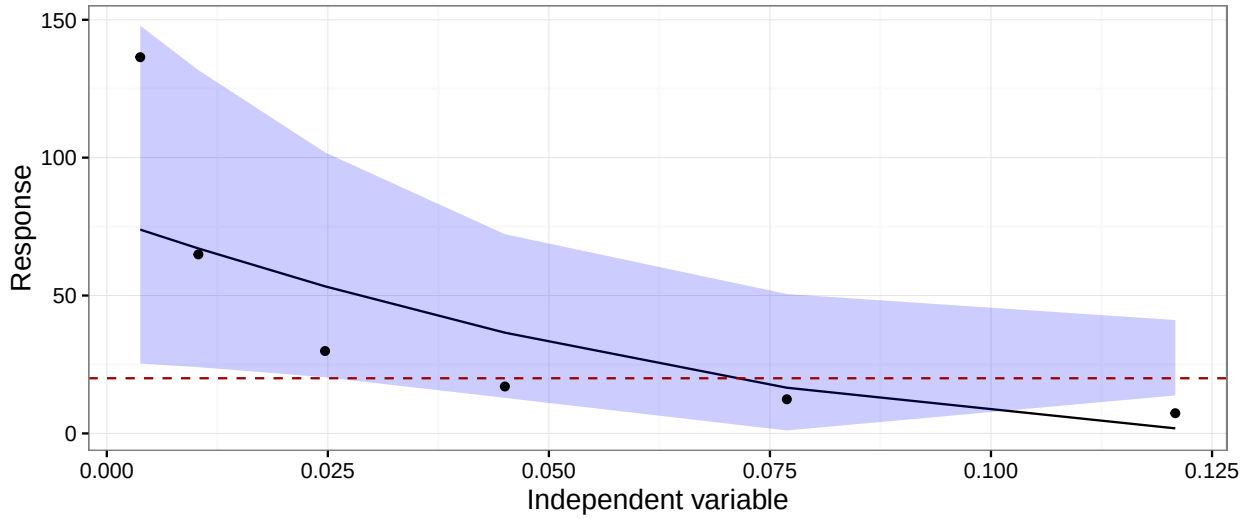
Model for the curve # 7 Multiplicative Y=aX^b ,
Exploration of model fitting at response of 20 , 0.0393 and 95% CI: (0.0358 , 0.0434)
Residual sum of squares 52.6646 , Residual standard deviation 0.0680



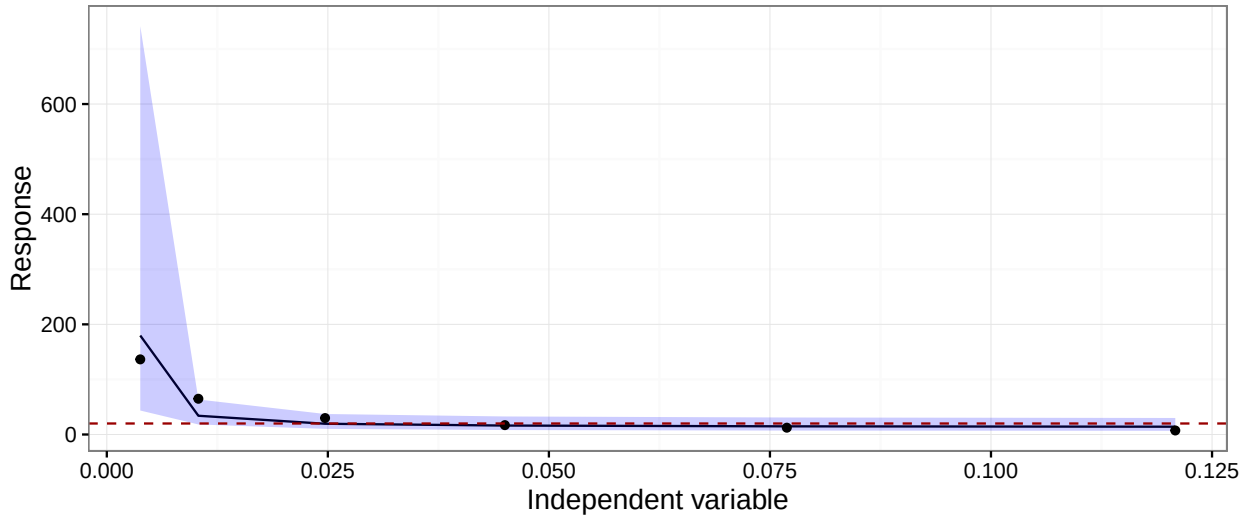
Model for the curve # 8 Square Root−X Y=a+b(sqrt(X)) ,
Exploration of model fitting at response of 20 , 0.0661 and 95% CI: (0.0271 , 0.8248)
Residual sum of squares 3635.4993 , Residual standard deviation 30.1476



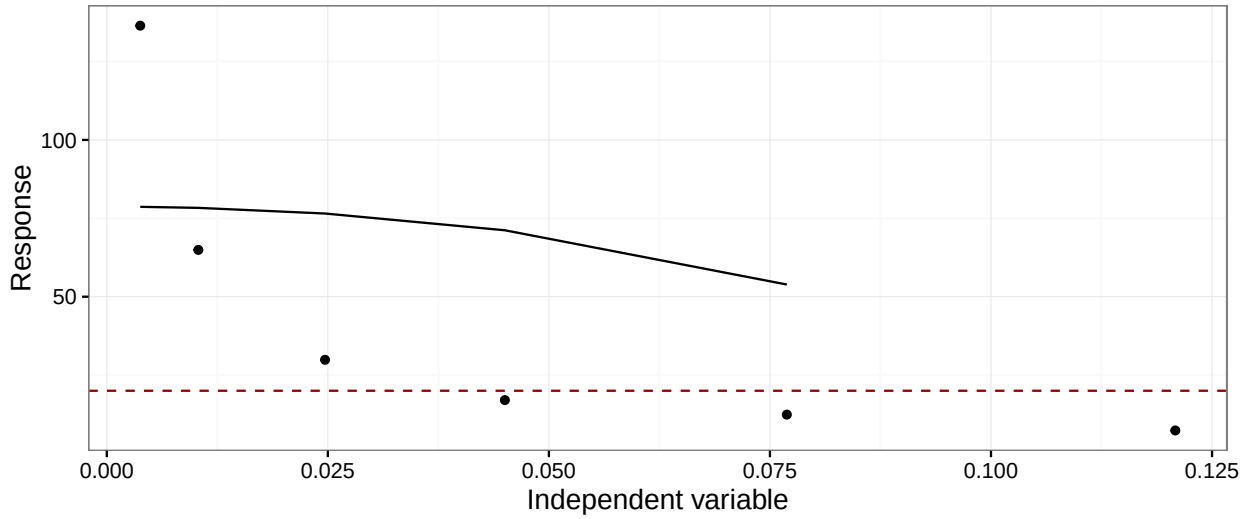
Model for the curve # 9 Square Root−Y Y=(a+bX)^2 ,
Exploration of model fitting at response of 20 , 0.0704 and 95% CI: (0.0259 , 0.8071)
Residual sum of squares 4893.6728 , Residual standard deviation 2.1637



Model for the curve # 10 S−curve Y=exp(a+b/X) ,
Exploration of model fitting at response of 20 , 0.0233 and 95% CI: (−0.0132 , 0.0092)
Residual sum of squares 2987.2487 , Residual standard deviation 0.5349



Model for the curve # 11 Square X and Y Y^2=a+X^2/b ,
Exploration of model fitting at response of 20 , 0.1020 and 95% CI: (NA , NA)
Residual sum of squares 10349.7798 , Residual standard deviation 7366.8110



5 Computing Environment

```
R version 3.2.2 (2015-08-14)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 8 x64 (build 9200)

locale:
[1] LC_COLLATE=English_United Kingdom.1252
[2] LC_CTYPE=English_United Kingdom.1252
[3] LC_MONETARY=English_United Kingdom.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United Kingdom.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods
[7] base

other attached packages:
[1] ggplot2_2.1.0 knitr_1.14

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.6      digest_0.6.10    plyr_1.8.4
 [4] grid_3.2.2       gtable_0.2.0     formatR_1.4
 [7] magrittr_1.5     evaluate_0.9     scales_0.4.0
[10] stringi_1.1.1    rmarkdown_1.0    labeling_0.3
[13] tools_3.2.2      stringr_1.1.0    munsell_0.4.3
[16] yaml_2.1.13      colorspace_1.2-6 htmltools_0.3.5

[1] "~/GIT/Functional-sensitivity"
```

This took 6.73 seconds to execute.