ggDiagnose examples

Installing `ggDiagnose`

The follow code is how to install the package. The package "requires" you to start out with few packages. If you don't have the correct ones it will prompt you to load them when you run specific `ggDiagnose` functions.

```r
library(devtools)
devtools::install_github("benjaminleroy/ggDiagnose")
```

```r
library(ggDiagnose)
```

`ggDiagnose`

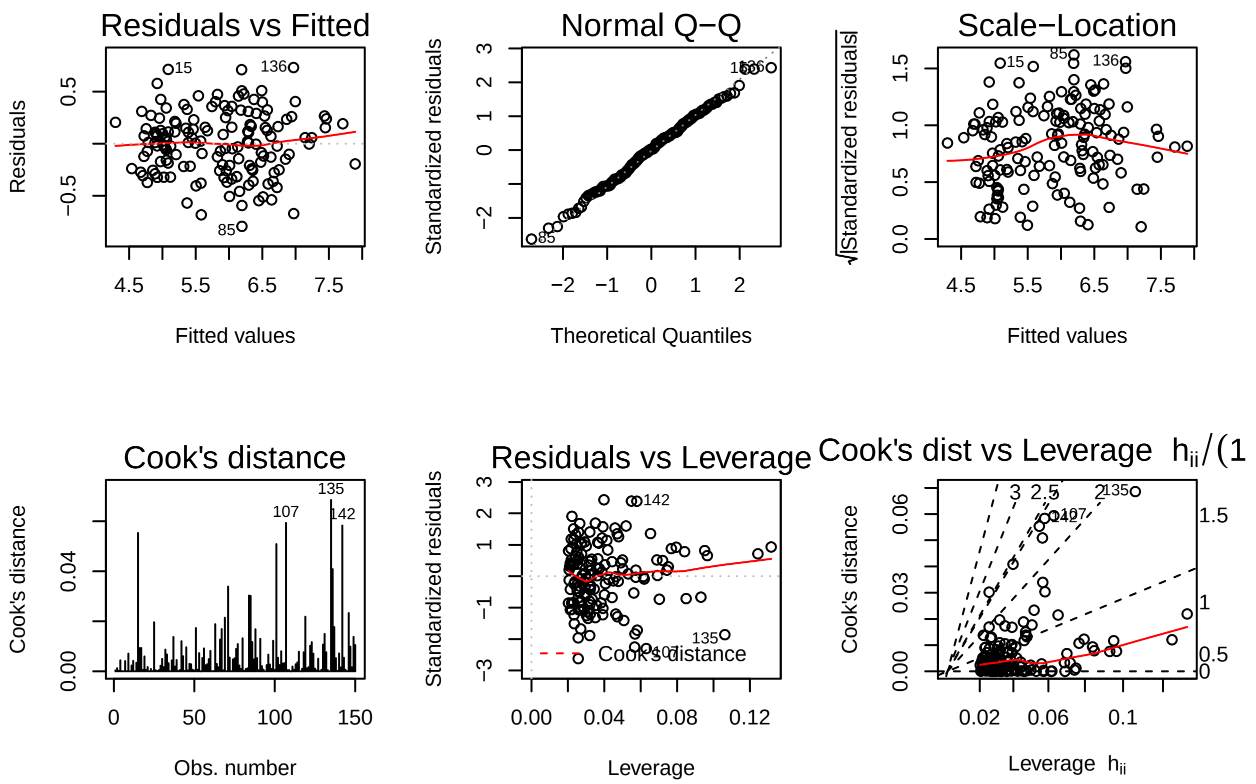
`ggDiagnose.lm`

This example is for an `lm` object, function works for `glm` and `rlm` objects as well.

```r
lm.object <- lm(Sepal.Length ~., data = iris)
```

The original visualization:

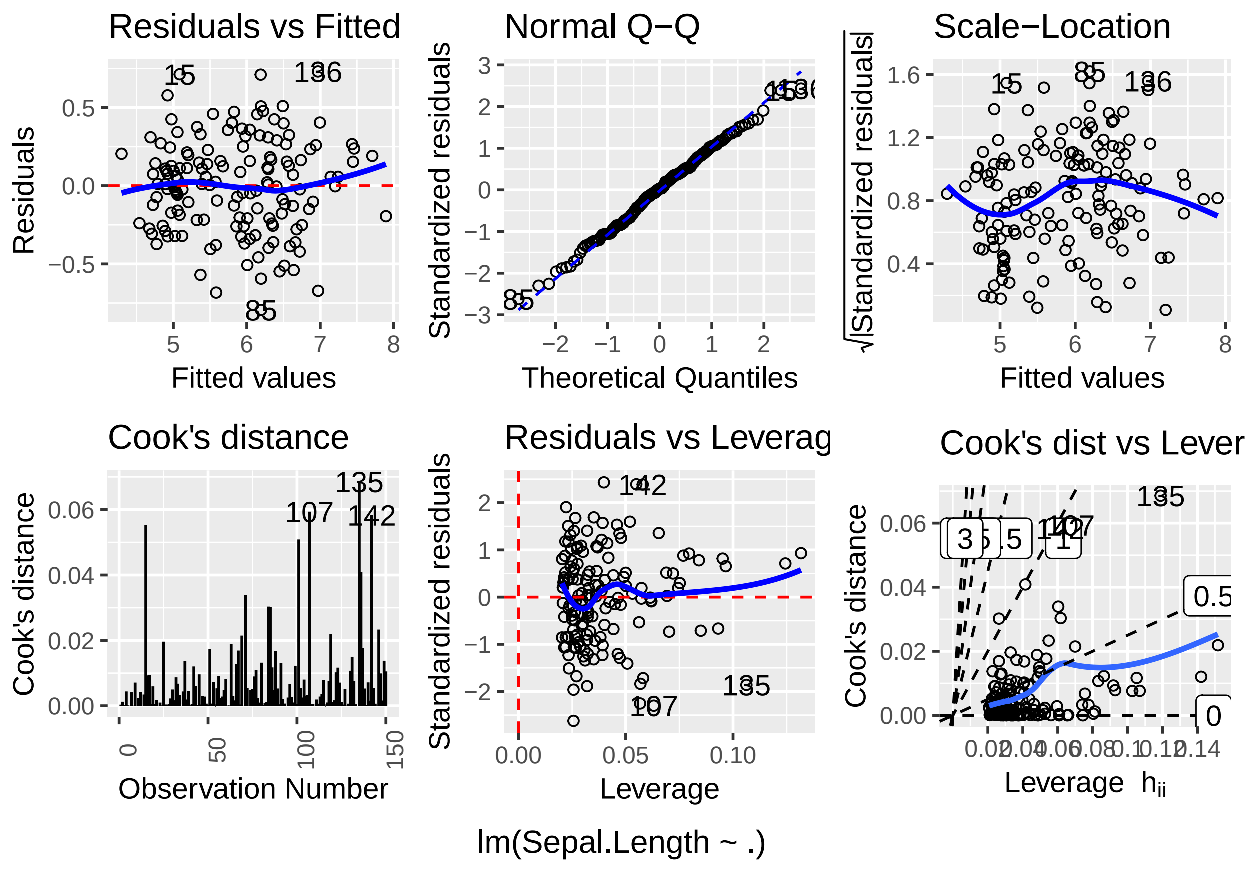
```r
par(mfrow = c(2,3))
plot(lm.object, which = 1:6)
```



The updated visualization (note we supressed warnings):

```
ggDiagnose(lm.object, which = 1:6)
```

```
## Warning: Removed 404 rows containing missing values (geom_path).
```



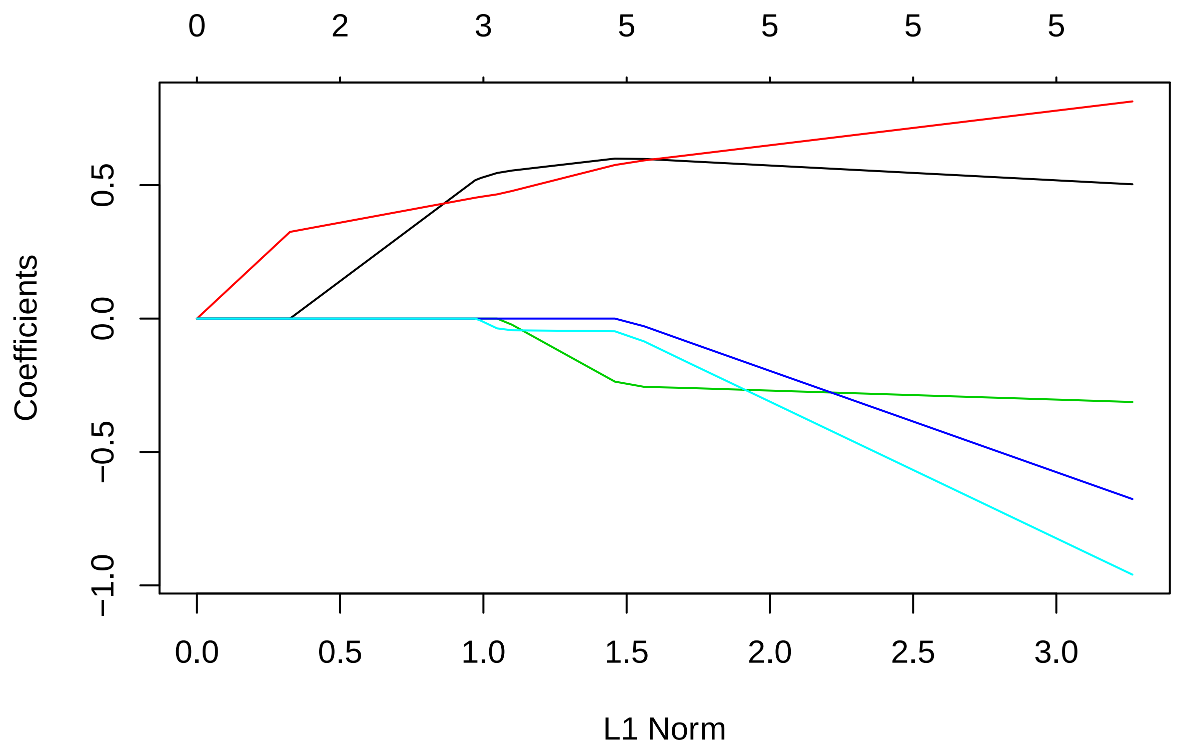
`ggDiagnose.lm` allows for similar parameter inputs as `plot.lm` but also includes additional ones. This may changes as the package evolves.

`ggDiagnose.glmnet`

```
library(glmnet)
glmnet.object <- glmnet(y = iris$Sepal.Length,
                        x = model.matrix(Sepal.Length~., data = iris))
```

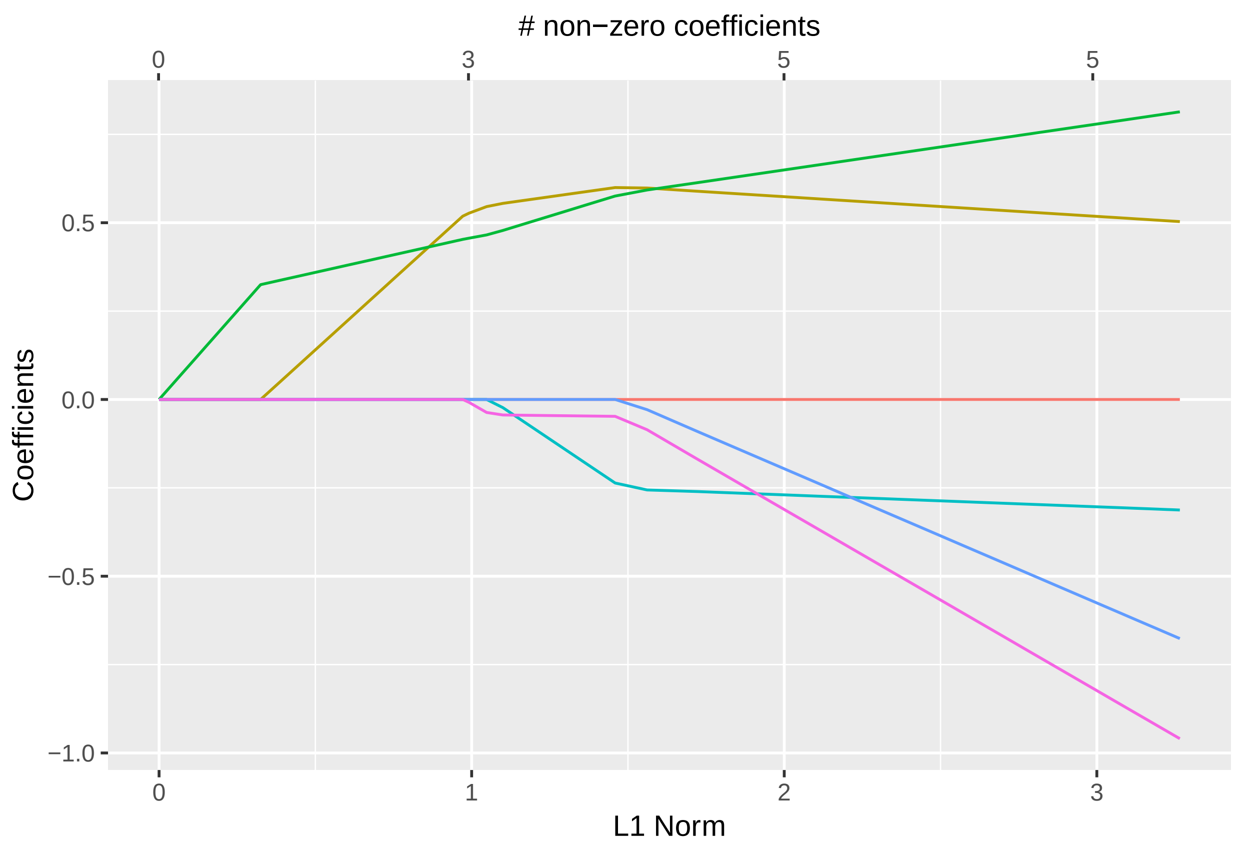
The original visualization:

```
plot(glmnet.object)
```



The updated visualization:

```
ggDiagnose(glmnet.object)
```

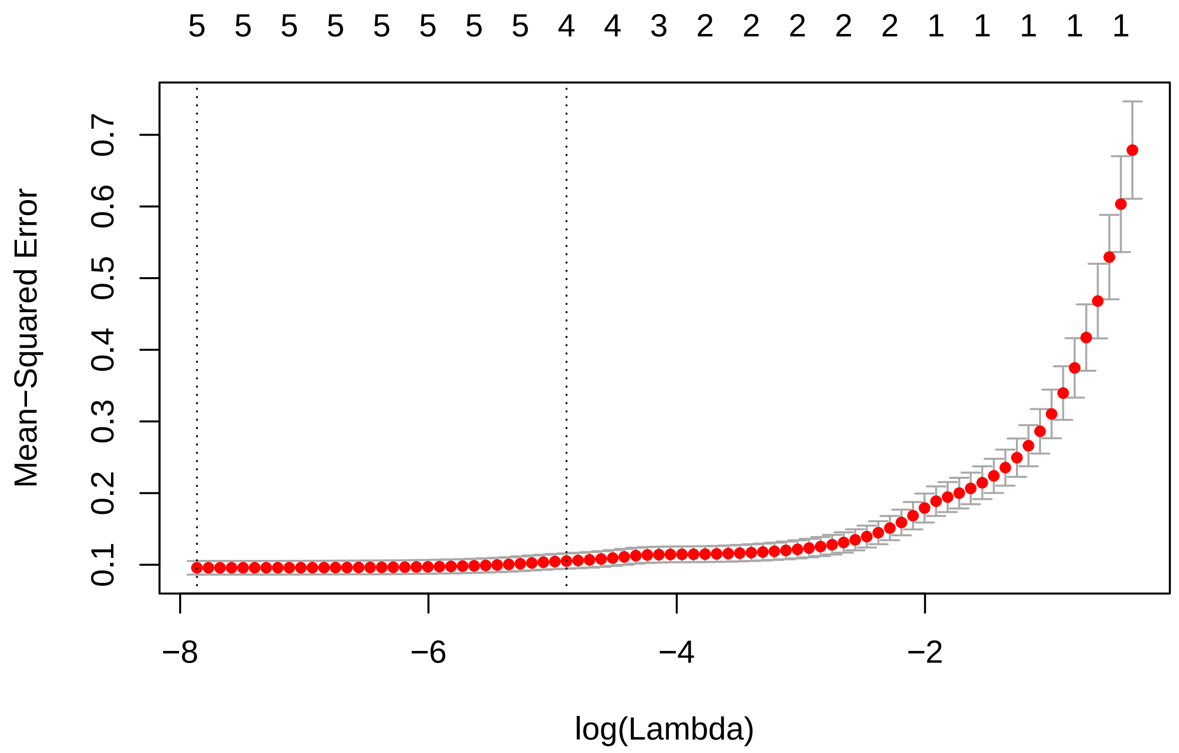


ggDiagnose.cv.glmnet

```
cv.glmnet.object <- cv.glmnet(y = iris$Sepal.Length,
                              x = model.matrix(Sepal.Length~., data = iris))
```

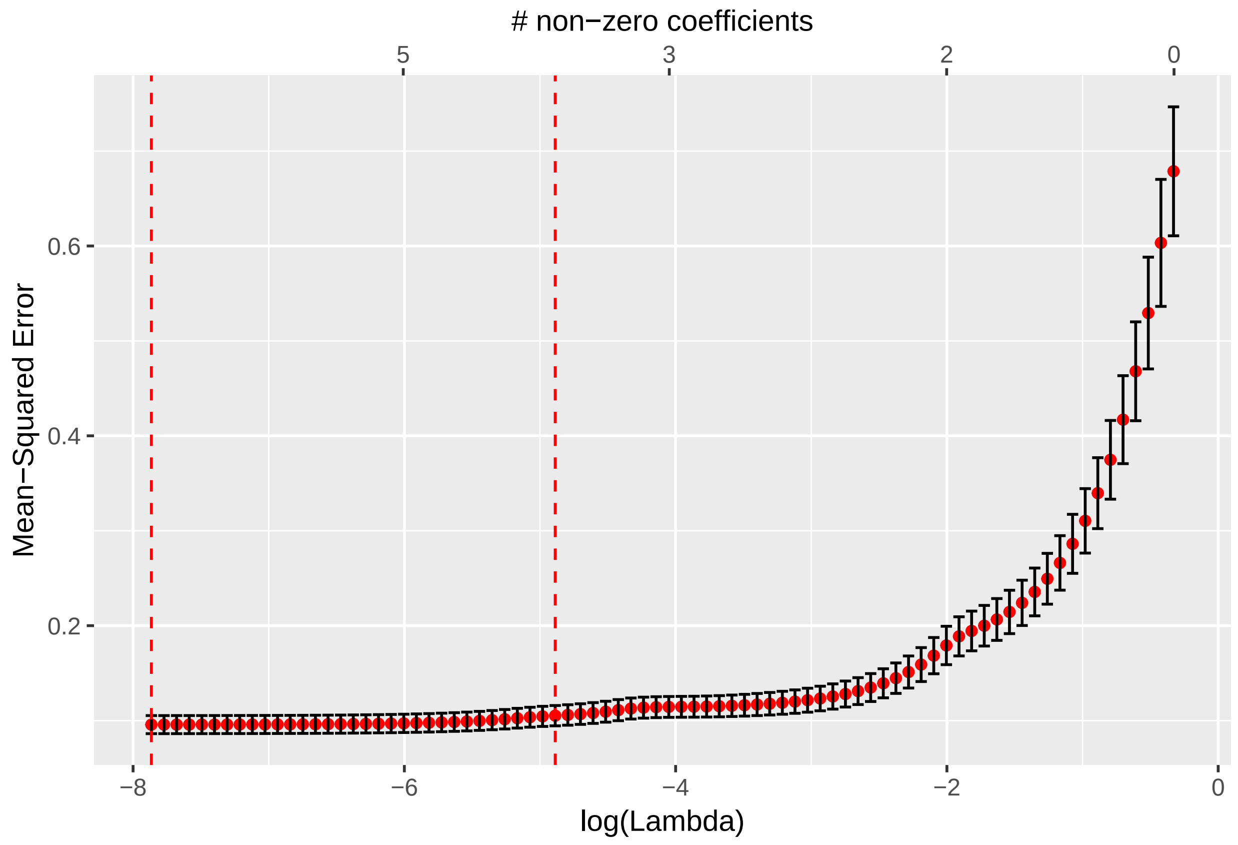
The original visualization:

```
plot(cv.glmnet.object)
```



The updated visualization:

```r
ggDiagnose(cv.glmnet.object)
```



ggDiagnose.Gam

```r
library(gam)
```

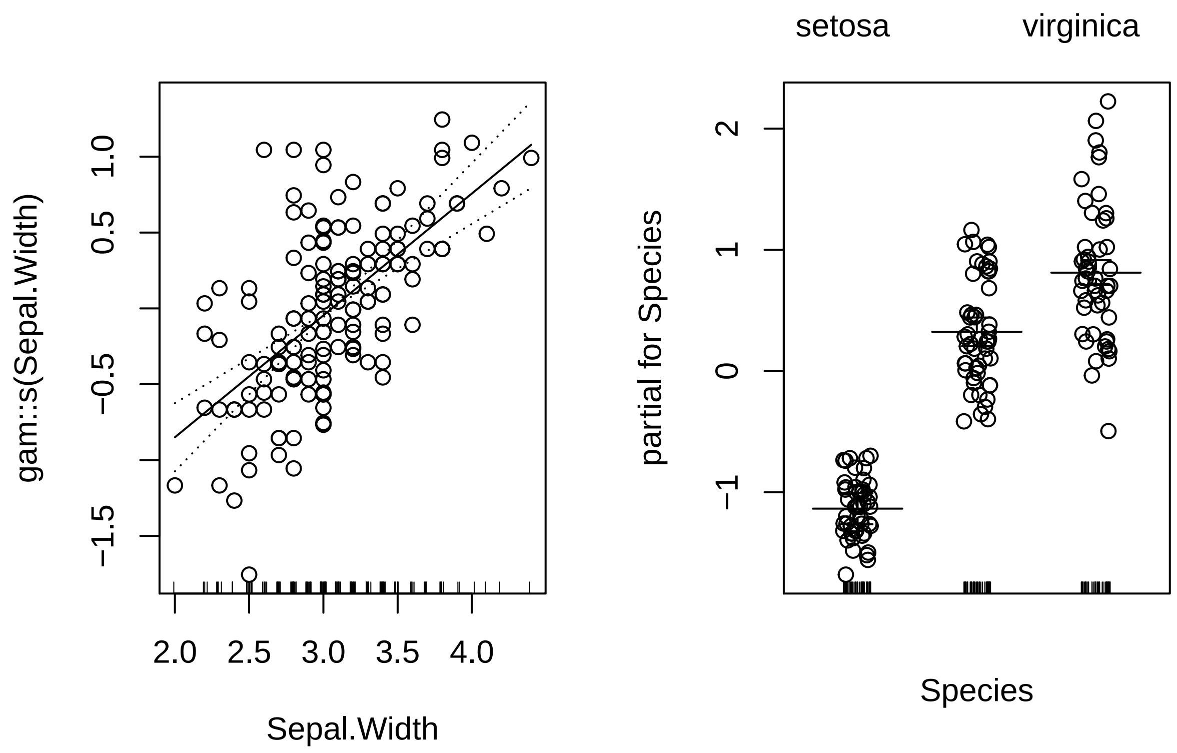
```
## Loading required package: splines
```

```
## Loaded gam 1.16
```

```r
gam.object <- gam::gam(Sepal.Length ~ gam::s(Sepal.Width) + Species,
                       data = iris)
```

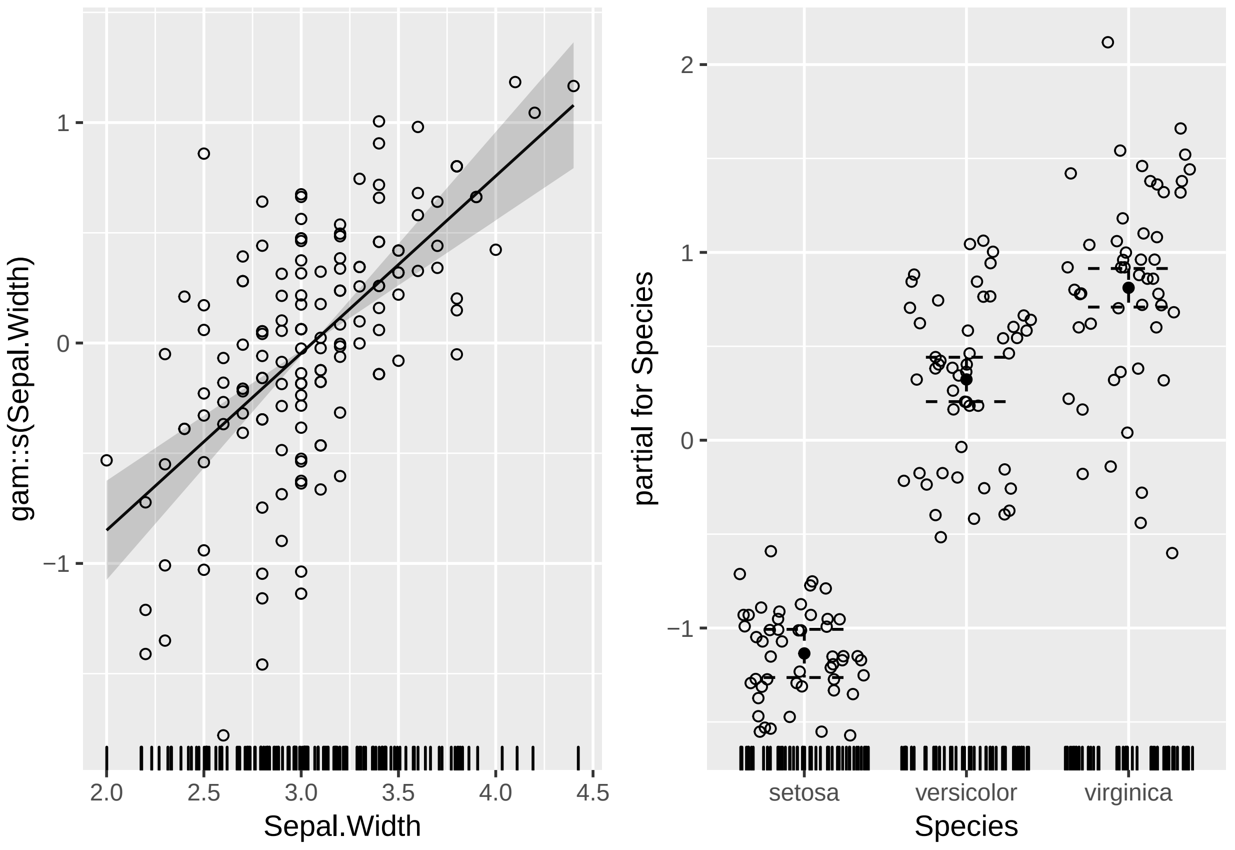
The original visualization:

```r
par(mfrow = c(1,2))
plot(gam.object, se = TRUE, residuals = TRUE)
```



The updated visualization:

```
ggDiagnose(gam.object, residuals = TRUE) # se = TRUE by default
```



ggDiagnose.tree

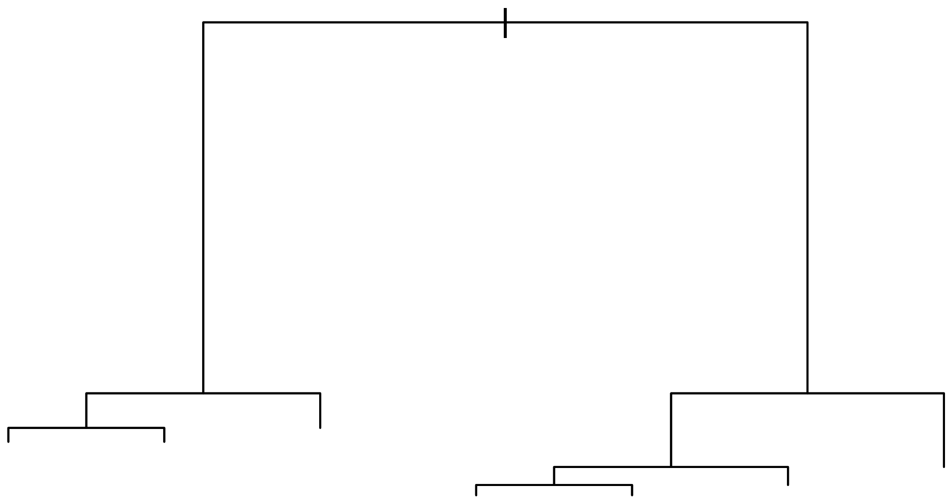
Note, for more perfect replication of the base `plot` function add `+ ggdendro::theme_dendro()` which drops all ggplot background elements.

```
library(tree)

tree.object <- tree(Sepal.Length ~., data = iris)
```

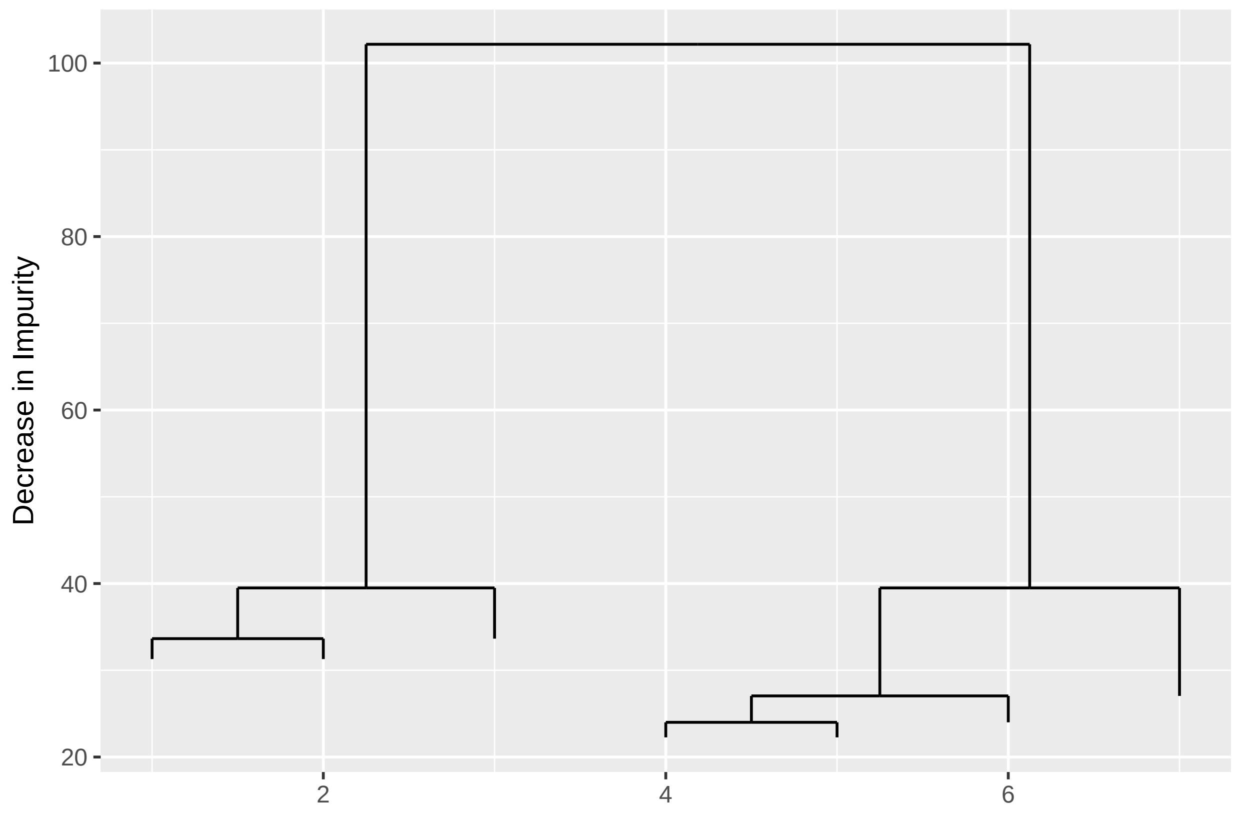
The original visualization:

```
plot(tree.object)
```

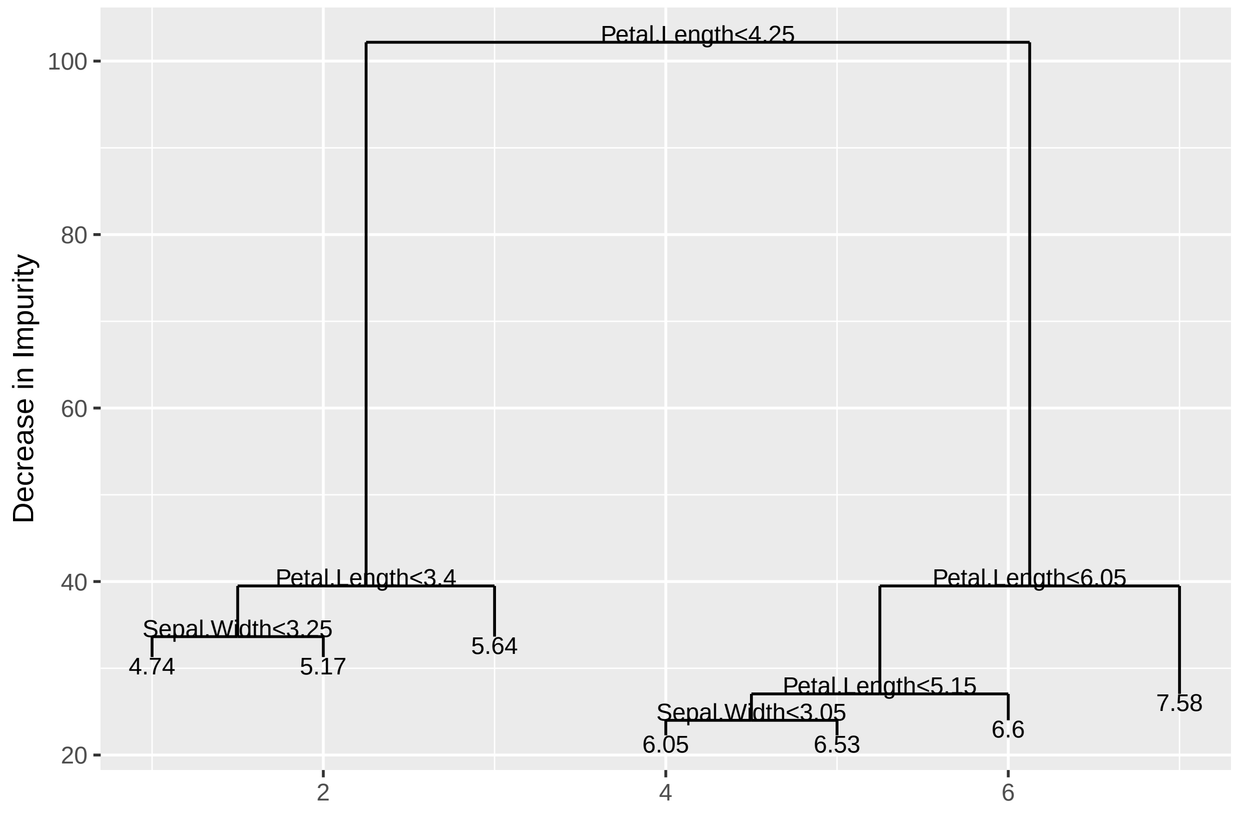


The updated visualization (followed by quick improvement):

```
ggDiagnose(tree.object, split.labels = FALSE)
```



```
ggDiagnose(tree.object, split.labels = TRUE,
           leaf.labels = TRUE)
```



dfCompile

Note this section uses functionality that can be found in the `dplyr` library.

```r
library(dplyr)
```

dfCompile.lm

```r
lm.object <- lm(Sepal.Length ~., data = iris)
dfCompile(lm.object) %>% names
```

```
##  [1] "Sepal.Length"          "Sepal.Width"
##  [3] "Petal.Length"          "Petal.Width"
##  [5] "Species"               ".index"
##  [7] ".labels.id"            ".weights"
##  [9] ".yhat"                 ".resid"
## [11] ".leverage"             ".cooksd"
## [13] ".weighted.resid"       ".std.resid"
## [15] ".sqrt.abs.resid"       ".pearson.resid"
## [17] ".std.pearson.resid"    ".logit.leverage"
## [19] ".ordering.resid"       ".ordering.std.resid"
## [21] ".ordering.cooks"       ".non.extreme.leverage"
```

```
dfCompile(lm.object) %>% head(2) # needs package dplyr for "%>%"
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species .index
## 1          5.1         3.5          1.4         0.2  setosa      1
## 2          4.9         3.0          1.4         0.2  setosa      2
##   .labels.id .weights    .yhat     .resid  .leverage      .cooksd
## 1          1        1 5.004788 0.09521198 0.02131150 0.0003570856
## 2          2        1 4.756844 0.14315645 0.03230694 0.0012517183
##   .weighted.resid .std.resid .sqrt.abs.resid .pearson.resid
## 1      0.09521198  0.3136729       0.5600651     0.09521198
## 2      0.14315645  0.4742964       0.6886918     0.14315645
##   .std.pearson.resid .logit.leverage .ordering.resid .ordering.std.resid
## 1          0.3136729      0.02177557             115                 115
## 2          0.4742964      0.03338553              98                  98
##   .ordering.cooks .non.extreme.leverage
## 1             124                  TRUE
## 2              96                  TRUE
```

dfCompile.glmnet

```
library(glmnet)
glmnet.object <- glmnet(y = iris$Sepal.Length,
                        x = model.matrix(Sepal.Length~., data = iris))
dfCompile(glmnet.object) %>% names # needs package dplyr for "%>%"
```

```
## [1] ".log.lambda"      "variable"         "beta.value"
## [4] ".norm"            ".dev"             ".number.non.zero"
```

```
dfCompile(glmnet.object) %>% head(2) # needs package dplyr for "%>%"
```

```
##   .log.lambda     variable beta.value      .norm      .dev
## 1  -0.3292550 X.Intercept.          0 0.00000000 0.0000000
## 2  -0.4222888 X.Intercept.          0 0.03632753 0.1290269
##   .number.non.zero
## 1                0
## 2                1
```

dfCompile.cv.glmnet

```
library(glmnet)
cv.glmnet.object <- cv.glmnet(y = iris$Sepal.Length,
                              x = model.matrix(Sepal.Length~., data = iris))
dfCompile(cv.glmnet.object) %>% names # needs package dplyr for "%>%"
```

```
## [1] "cross.validated.error"        "cross.validation.upper.error"
## [3] "cross.validation.lower.error" "number.non.zero"
## [5] ".log.lambda"
```

```
dfCompile(cv.glmnet.object) %>% head(2) # needs package dplyr for "%>%"
```

```
##    cross.validated.error cross.validation.upper.error
## s0             0.6830310                    0.7531277
## s1             0.6053838                    0.6700409
##    cross.validation.lower.error number.non.zero .log.lambda
## s0                    0.6129344               0  -0.3292550
## s1                    0.5407268               1  -0.4222888
```

dfCompile.Gam

```
library(gam)
gam.object <- gam::gam(Sepal.Length ~ gam::s(Sepal.Width) + Species,
                       data = iris)
dfCompile(gam.object) %>% names # needs package dplyr for "%>%"
```

```
##  [1] "Sepal.Length"
##  [2] "Sepal.Width"
##  [3] "Petal.Length"
##  [4] "Petal.Width"
##  [5] "Species"
##  [6] ".resid"
##  [7] ".smooth.gam.s.Sepal.Width."
##  [8] ".smooth.Species"
##  [9] ".se.smooth.gam.s.Sepal.Width..upper"
## [10] ".se.smooth.Species.upper"
## [11] ".se.smooth.gam.s.Sepal.Width..lower"
## [12] ".se.smooth.Species.lower"
```

```
dfCompile(gam.object) %>% head(2) # needs package dplyr for "%>%"
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species     .resid
## 1          5.1         3.5          1.4         0.2  setosa 0.03614362
## 2          4.9         3.0          1.4         0.2  setosa 0.23792407
##   .smooth.gam.s.Sepal.Width. .smooth.Species
## 1                 0.35570963       -1.135187
## 2                -0.04607082       -1.135187
##   .se.smooth.gam.s.Sepal.Width..upper .se.smooth.Species.upper
## 1                          0.44985507                -1.006952
## 2                         -0.03387729                -1.006952
##   .se.smooth.gam.s.Sepal.Width..lower .se.smooth.Species.lower
## 1                          0.26156418                -1.263422
## 2                         -0.05826436                -1.263422
```

dfCompile.tree

```
library(gam)
gam.object <- gam::gam(Sepal.Length ~ gam::s(Sepal.Width) + Species,
                       data = iris)
dfCompile(tree.object) %>% length # needs package dplyr for "%>%"
```

```
## [1] 4
```

```
dfCompile(tree.object)$segments %>% head # needs package dplyr for "%>%"
```

```
##      .x       .y .xend     .yend .n
## 2 2.250 39.49191 2.250 102.16833 73
## 3 1.500 33.64903 1.500  39.49191 53
## 4 1.000 31.29592 1.000  33.64903 20
## 5 2.000 31.29592 2.000  33.64903 33
## 6 3.000 33.64903 3.000  39.49191 20
## 7 6.125 39.49191 6.125 102.16833 77
```

```
dfCompile(tree.object)$labels %>% head # needs package dplyr for "%>%"
```

```
##       .x        .y            .label
## 1 4.1875 102.16833 Petal.Length<4.25
## 2 2.2500  39.49191  Petal.Length<3.4
## 3 1.5000  33.64903  Sepal.Width<3.25
## 4 6.1250  39.49191 Petal.Length<6.05
## 5 5.2500  27.04709 Petal.Length<5.15
## 6 4.5000  24.00201  Sepal.Width<3.05
```

```
dfCompile(tree.object)$leaf_labels %>% head # needs package dplyr for "%>%"
```

```
##    .x        .y .label    .yval
## 4   1 31.29592   4.74 4.735000
## 5   2 31.29592   5.17 5.169697
## 6   3 33.64903   5.64 5.640000
## 10  4 22.26715   6.05 6.054545
## 11  5 22.26715   6.53 6.530000
## 12  6 24.00201   6.60 6.604000
```