Plotting model residuals with `plotres`

Stephen Milborrow

August 26, 2016

Contents

1 Introduction

Residual plots are important for checking linear models, but they also are useful for other types of model. They can be used for an overview of the model's performance, to check for outliers, and to check if the response should be transformed.

The `plotres` function in the `plotmo` R package [8] makes it easy to plot residuals for a wide variety of R models. Figure 1 shows an example. It was was produced with the following code:

```
library(rpart); library(plotmo)             # plotres is in the plotmo package
library(earth); data(ozone1)                # get the ozone data
rpart.mod <- rpart(O3 ~ ., data = ozone1)   # generate an rpart model
plotres(rpart.mod)                          # plot its residuals
```

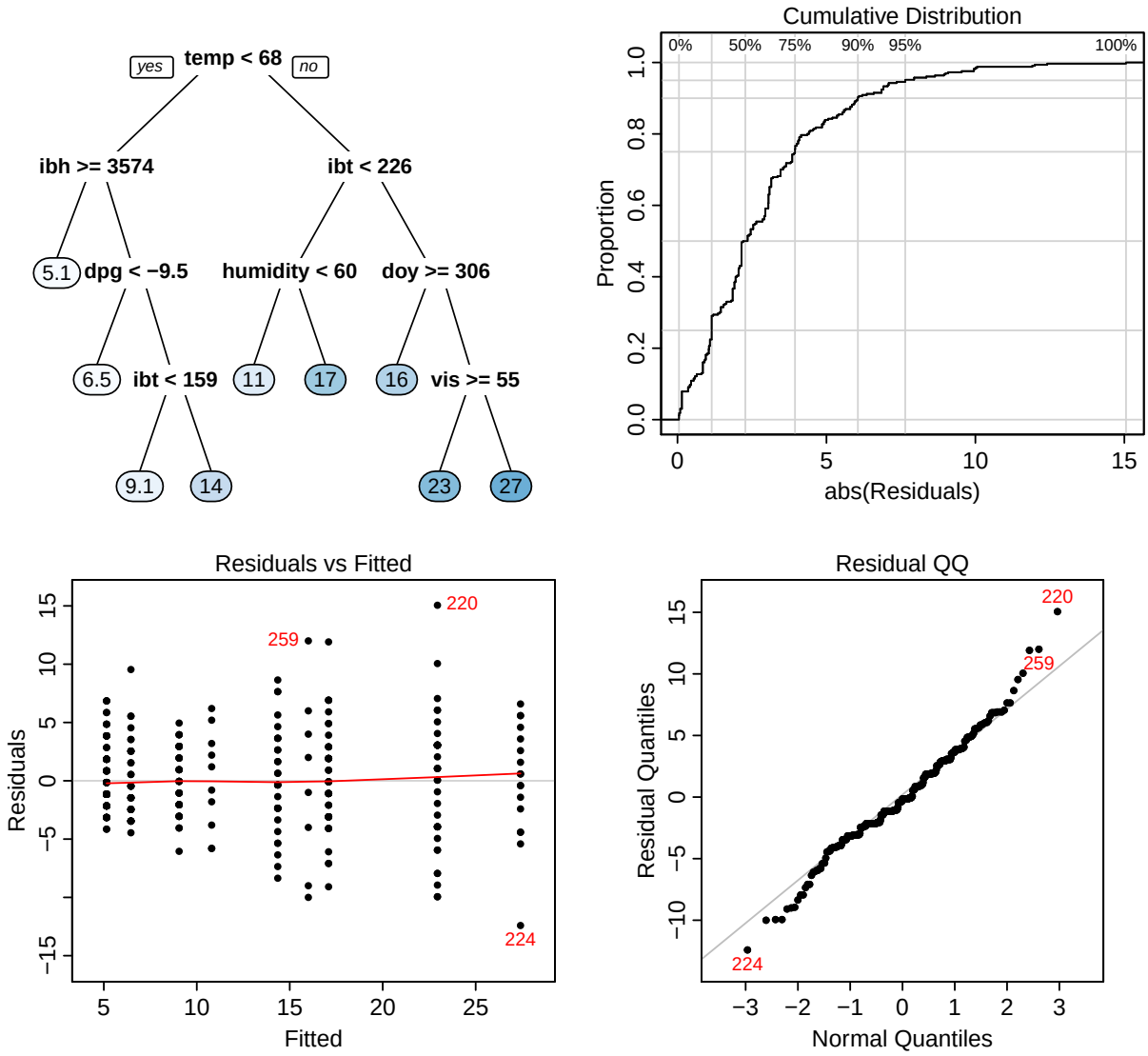

Figure 1: *An example* `plotres` *plot. This example is for an* `rpart` *model.*

Since this is an `rpart` model [13], `plotres` draws a tree at the top left [6]. Different figures will be drawn in the top left for other types of model (Section 5).

The bottom left plot is a canonical Residuals vs Fitted plot. In this example we see the quantized fits characteristic of RPART models. Each vertical line of points corresponds to a leaf of the tree. (We could use `plotres`'s `jitter` argument to reduce the overplotting caused by quantization here.) The red line is a lowess smooth.

In case 220, for example, we see that although the model estimates a high ozone level (the fitted value is 23), the observed level was actually a lot higher. From the QQ plot (bottom right) we see that this residual is indeed unusual. Since it is the biggest residual, it determines the right bound of the Cumulative Distribution plot (top right).

A variety of other residual plots can be generated by `plotres` as described on its help page.

2 Some examples

Here are some examples which illustrate `plotres` on various models.

```
library(earth) # for ozone1 data
data(ozone1)

lm.mod <- lm(O3 ~ ., data=ozone1)                          ## linear model
plotres(lm.mod)

earth.mod <- earth(O3 ~ ., data=ozone1, degree=2)          ## earth
plotres(earth.mod) # equivalent to plot.earth

library(rpart)                                             ## rpart
rpart.mod <- rpart(O3 ~ ., data=ozone1)
plotres(rpart.mod)

library(tree)                                              ## tree
tree.mod <- tree(O3~., data=ozone1)
plotres(tree.mod)

library(randomForest)                                      ## randomForest
set.seed(2015)
rf.mod <- randomForest(O3~., data=ozone1)
plotres(rf.mod)

library(gbm)                                               ## gbm
set.seed(2015)
gbm.mod <- gbm(O3~., data=ozone1, dist="gaussian",
               interact=2, shrink=.01, n.trees=1000)
plotres(gbm.mod)

library(nnet)                                              ## nnet
set.seed(2015)
nnet.mod <- nnet(O3~., data=scale(ozone1), size=2, decay=0.01, trace=FALSE)
plotres(nnet.mod, type="raw")

library(neuralnet)                                         ## neuralnet
set.seed(2015)
nn.mod <- neuralnet(O3~humidity+temp, data=scale(ozone1), hidden=2)
plotres(nn.mod)

library(caret)                                             ## caret
set.seed(2015)
caret.earth.mod <- train(O3~., data=ozone1, method="earth",
                         tuneGrid=data.frame(degree=2, nprune=10))
plotres(caret.earth.mod, type="raw")
```

This definitely isn't an exhaustive list of models supported by `plotres`. The packages used in the above code are [2–4, 9–11, 13, 14].

3 Limitations

Plotres is designed primarily for displaying standard `response - fitted` residuals for regression models with a single continuous response. For some model types it supports multiple responses and other kinds of residuals.

In general, it won't work on models that don't save the call or data with the model in a standard way. For further discussion please see *Accessing the model data* in the *plotmo* vignette. Please also see the vignette *Guidelines for S3 Regression Models* [7].

4 Generating the residuals and calling `predict`

Plotres first tries to get the residuals by calling the `residuals` method for the model. If the call fails (which it will for models that don't have a `residuals` method), `plotres` must figure out the residuals manually. It does that using `predict`.

Plotres tries to use sensible default arguments for `predict`, but they won't always be correct (`plotres` can't know about every kind of model). Change the defaults if necessary using arguments with a `predict.` prefix. Plotres passes any argument prefixed with `predict.` directly to `predict`, after removing the prefix.

For example, `predict.gbm` has a `n.trees` argument, which `plotres` defaults to the total number of trees. But that can be changed, for example:

```
library(gbm); library(plotmo)
example(gbm)                              # create gbm1, a gbm model
plotres(gbm1)                             # call predict.gbm with total n.trees
nbest <- gbm.perf(gbm1, method = "OOB",   # get "best" number of trees
                  plot.it=FALSE)
plotres(gbm1, predict.n.trees = nbest)    # pass n.trees = nbest to predict.gbm
```

Pass `trace = 1` to `plotres` to see the arguments passed to `predict` and friends.

5 The which=1 plot

The top left plot is a model-specific plot.[1] We call this plot the "`which=1`" plot. What gets plotted here depends on the model class. For example, for `earth` models this is a model selection plot and for `glmnet` models it's a coefficient profile plot.

Nothing will be displayed for some models. This isn't really an issue. You will see three instead of four plots when you call `plotres` with the default arguments.

For some models, the `which=1` plot is called with default arguments programmed into `plotres`. Use `trace = 1` to see those arguments. Change the arguments passed to this plot by using `plotres` arguments with a `w1.` prefix. Plotres passes any argument

[1] This plot is included by default, since by default `which = 1:4`.

prefixed with `w1.` directly to the `which=1` plot, after removing the prefix. See the example below.

It may happen that the `which=1` plot is only partially plotted, or messes up the page for further plots. In that case, call `plotres` with a `which` argument that excludes `1` (don't use the default `which`). We would be interested in hearing about models that cause this kind of bad behavior.

An example

Here's a `which=1` plot showing a `glmnet` [1] model taming the `longley` data (Figure 2):

```
library(glmnet); library(plotmo); data(longley)

mod <- glmnet(data.matrix(longley[,1:6]), longley[,7])

plotres(mod, which=1)                    # left side of the figure
```

Change the parameters passed to the plot using the `w1.` prefix:

```
plotres(mod, which=1,                    # right side of the figure
        w1.xvar="norm",                  # pass xvar="norm" to the plot
        w1.col=1:3)                      # change the color scheme
```

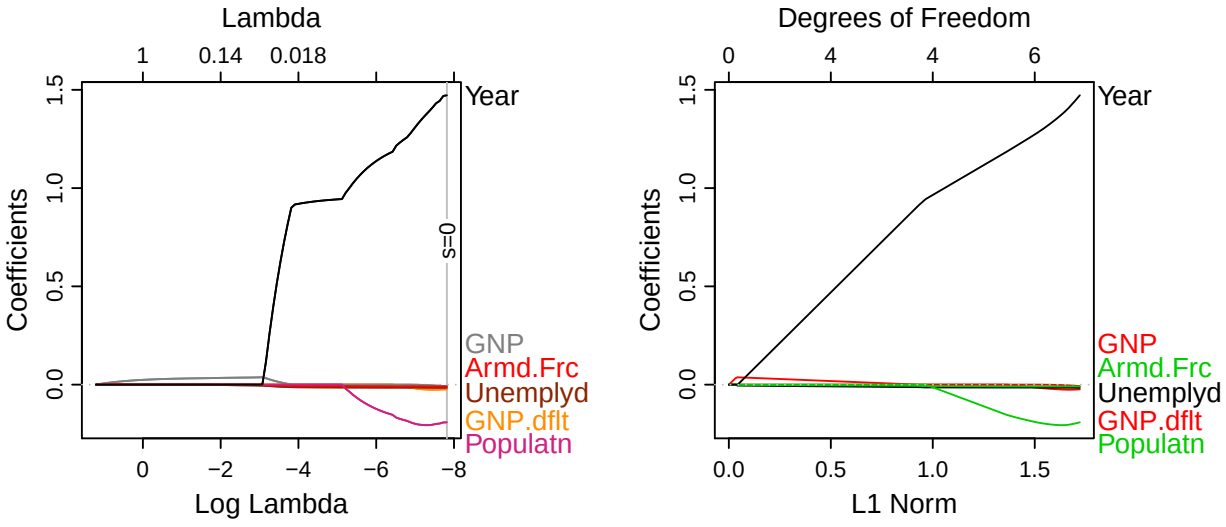

Figure 2: `glmnet` *on the* `longley` *data [5]. Just the* `which=1` *plot is shown.*

The variable names on the right of the plots are automatically spread out to prevent overplotting.

Left `plotres(glmnet.mod)`
Right `plotres(glmnet.mod, w1.xvar="norm", w1.col=1:3))`

6 Notes on glmnet models

The following code plots an example glmnet model (Figure 3). The previous page also has examples.

```
library(glmnet)
library(plotmo)
x <- matrix(rnorm(100 * 10), 100, 10)
y <- x[,1] + x[,2] + 3 * rnorm(100)     # y depends only on x[,1] and x[,2]
mod <- glmnet(x, y)
plotres(mod)
```

For the which=1 plot (top left), plotres uses an internal version of plot.glmnet. This is much like the plot.glmnet described in the glmnet help pages. It has an extra xvar option "rlambda", which plotres uses by default. (This allows better printing of the variable names next to their coefficient curves on the right of the plot.) Override that default by passing say w1.xvar="norm" to plotres.

The plot is annotated with a gray vertical line showing the s parameter. This is the penalty lambda passed internally to predict.glmnet when generating the residual plots. By default plotres uses s=0 (no penalty). Change that by passing say predict.s=0.02 to plotres. This can used to see how the residuals change for models at different points on the coefficient plot. (In fact, you can pass any argument to predict.glmnet by prefixing the argument with predict. as described in Section 4.)

Plot arguments like col and lty can be passed to this plot using a w1. prefix as described in Section 5, e.g. w1.col=c(1,2,5). These get recycled as usual if necessary.

By default, the names of the 10 variables with the largest final coefficients are printed on the right. This is done using spread.labs from the TeachingDemos package [12] to minimize overplotting. Some options:
Use w1.label=3 to print only 3 names.
Use w1.label=0 or FALSE to remove the names.
Use w1.label=TRUE to print all the names.

Use w1.s.col=0 to remove the s gray vertical line.
Use w1.s.col="red" to print it in red.

Use grid.col="gray" argument to add a gray grid to the plots.

For multiple response models, use plotres's nresponse argument to select which response is plotted (plot.glmnet's type.coef argument isn't supported).

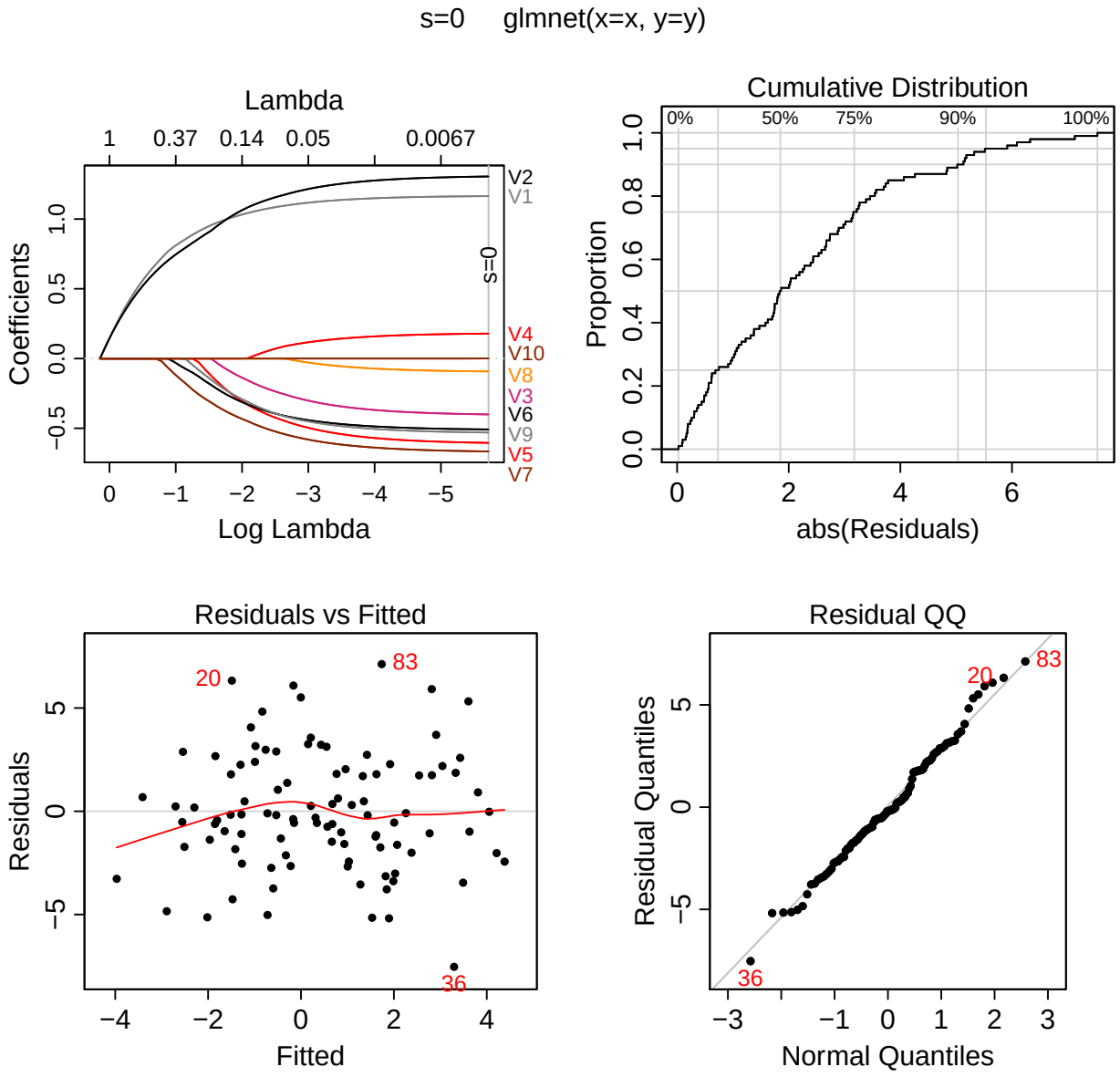

Figure 3: *A* `glmnet` *model plotted with* `plotres`.

In this example, all variables except V1 and V2 are noise variables. (The `x` *matrix has no column names, so the variables have been automatically named with a* `V` *prefix.)*

To calculate the residuals, `s=0` *was passed to* `predict.glmnet`*, as indicated by the vertical gray line in the top left plot. (Use* `w1.s.col=0` *to remove this line.)*

7 Notes on gbm models

Plotres uses an internal function for plotting gbm models. Figure 4 shows an example, generated with the following code (only the which=1 plot is shown):

```
library(earth); data(ozone1) # get the ozone data
gbm.mod <- gbm(O3~., data=ozone1, dist="gaussian", interact=2,
               shrink=.01, train.frac=.8, cv.folds=10, n.trees=1000)
plotres(gbm.mod)
```

The gray vertical line on the right shows that all 1000 trees were used when generating the residuals (Section 4), although the residual plots aren't shown here.

The vertical dotted lines and the corresponding colored numbers along the top of the plot show the number of trees selected by various criteria.

The OOB line is the out-of-bag improvement, similar to the gbm.perf plot. It is on a different scale to the other lines, so shouldn't really be shown on the same plot. We force it in by rescaling and shifting it. Although helpful for comparing model-selection criteria, this can also lead to confusion: the scale on the left doesn't apply to the OOB line. A dashed line is used as a reminder.

Pass arguments to the plot using the w1. prefix. For more resolution on the vertical axis use something like w1.ylim=c(10,30). Use w1.n.trees=NA to remove the vertical gray line.

TODO Allow the user to remove curves or specify their colors.

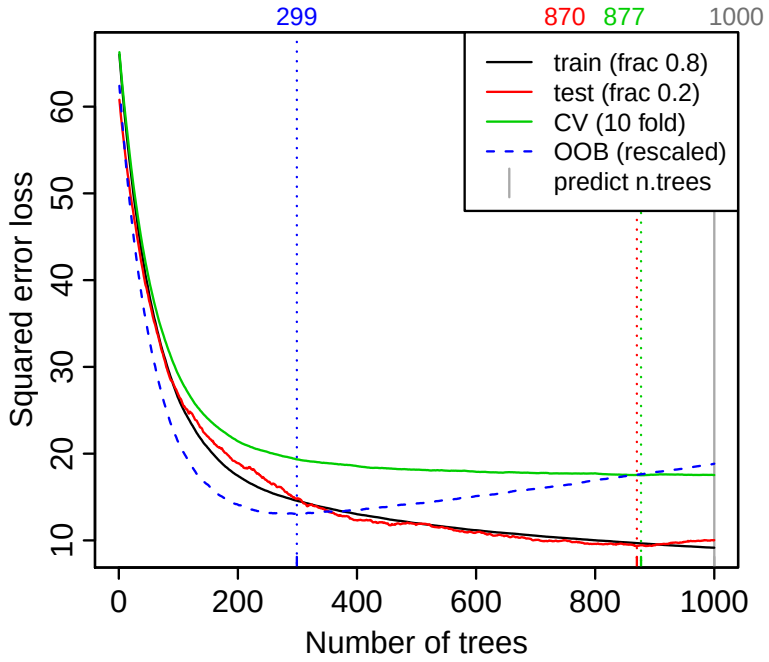

Figure 4: gbm *on the* ozone1 *data. Just the* which=1 *plot is shown.*
In this example the test and CV minima are almost coincident (and the 870 and 877 are automatically spread out to prevent overplotting).

8 Comparison to plot.lm

The function `plot.lm` automatically standardizes residuals for some of the plots. In contrast, with `plotres` we must explicitly specify when the residuals should be standardized. This is because standardization isn't possible or appropriate for most of the models that `plotres` is designed for.

The different `which` numbering scheme used by `plotres` is mostly an historical legacy.

As a somewhat academic exercise, here's how `plot.lm` can be emulated with `plotres` (Figure 8):

```
plotlm <- function(object) # similar to plot.lm
{                                                       # residuals vs fitted
    plotres(object, which=3, center=FALSE,
            caption=paste(deparse(object$call), collapse=" "))
                                                        # QQ plot
    plotres(object, which=4, standardize=TRUE)
                                                        # scale-location plot
    plotres(object, which=6, standardize=TRUE, main="Scale-Location")
                                                        # leverage plot
    plotres(object, which=3, versus=4, standardize=TRUE)
}
fit <- lm(Volume ~ ., data = trees)     # simple linear model
par(mfrow = c(2,2), oma = c(0,0,3,0))   # four plots on page, space for caption
plot(fit)                               # call plot.lm
plotlm(fit)                             # call our version of plot.lm
```

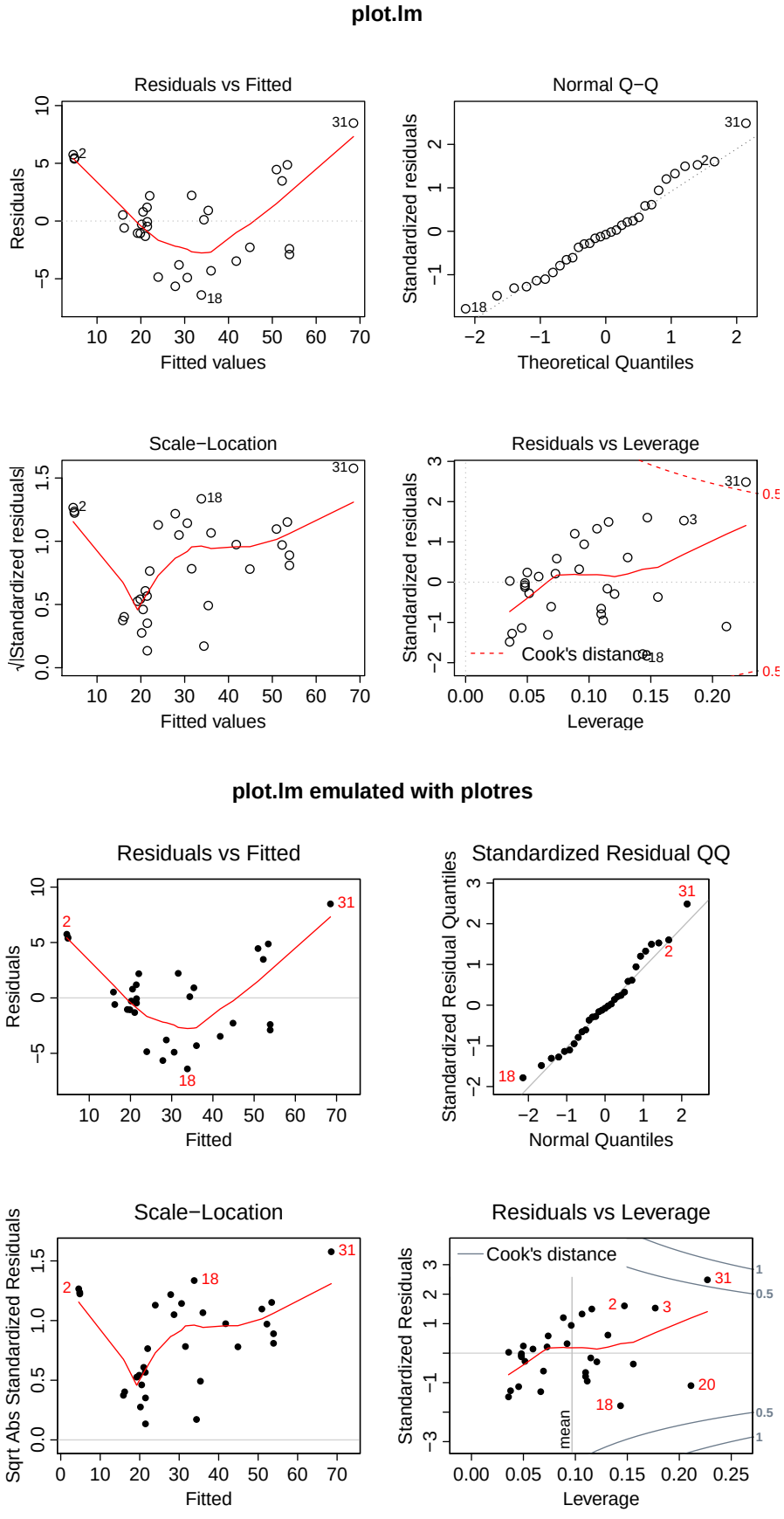

Figure 5: `plot.lm` *and* `plotres`.

9 FAQ and common error messages

Please see the FAQ and *Common error messages* in the `plotmo` vignette.

See also the *Notes on some packages* in that vignette.

Most of the difficulties associated with plotting residuals by functions like `plotres` arise because the model-building function neglected to include some essential fields in the model, for example the `call`. See the vignette *Guidelines for S3 Regression Models* [7].

Why is nothing displayed for which=1 ?

See Section 5. For some model classes, nothing will displayed for `which=1`. By default, `plotres` will plot three instead of four plots.

References

[1] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *Regularization Paths for Generalized Linear Models via Coordinate Descent*. JASS, 2010. Cited on page 5.

[2] Stefan Fritsch and Frauke Guenther; following earlier work by Marc Suling. *neuralnet: Training of neural networks*, 2012. R package, `http://CRAN.R-project.org/package=neuralnet`. Cited on page 3.

[3] Max Kuhn. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer, and Allan Engelhardt. *caret: Classification and Regression Training*, 2011. R package, `http://CRAN.R-project.org/package=caret`. Cited on page 3.

[4] Andy Liaw, Mathew Weiner; Fortran original by Leo Breiman, and Adele Cutler. *randomForest: Breiman and Cutler's random forests for regression and classification*, 2014. R package, `http://CRAN.R-project.org/package=randomForest`. Cited on page 3.

[5] J.W. Longley. *An appraisal of least-squares programs from the point of view of the user*. JASS, 1967. Cited on page 5.

[6] S. Milborrow. *rpart.plot: Plot rpart models. An enhanced version of plot.rpart*, 2011. R package, `http://www.milbo.org/rpart-plot`. Cited on page 2.

[7] S. Milborrow. *Guidelines for S3 regression models*, 2015. Vignette for R package plotmo, `http://www.milbo.org/doc/modguide.pdf`. Cited on pages 4 and 11.

[8] S. Milborrow. *plotmo: Plot a model's response and residuals*, 2015. R package, `http://CRAN.R-project.org/package=plotmo`. Cited on page 1.

[9] S. Milborrow. Derived from mda:mars by T. Hastie and R. Tibshirani. *earth: Multivariate Adaptive Regression Splines*, 2011. R package, `http://www.milbo.users.sonic.net/earth`. Cited on page 3.

[10] Greg Ridgeway et al. *gbm: Generalized Boosted Regression Models*, 2014. R package, `http://CRAN.R-project.org/package=gbm`. Cited on page 3.

[11] Brian Ripley. *tree: Classification and regression trees*, 2014. R package, `http://CRAN.R-project.org/package=tree`. Cited on page 3.

[12] Greg Snow. *TeachingDemos: Demonstrations for teaching and learning*, 2013. R package, `http://CRAN.R-project.org/package=TeachingDemos`. Cited on page 6.

[13] Terry Therneau and Beth Atkinson. *rpart: Recursive Partitioning and Regression Trees*, 2014. R package, `http://CRAN.R-project.org/package=rpart`. Cited on pages 2 and 3.

[14] W.N. Venables and B.D. Ripley. *nnet: Feed-forward Neural Networks and Multinomial Log-Linear Models*, 2014. R package, `http://CRAN.R-project.org/package=MASS`. Cited on page 3.