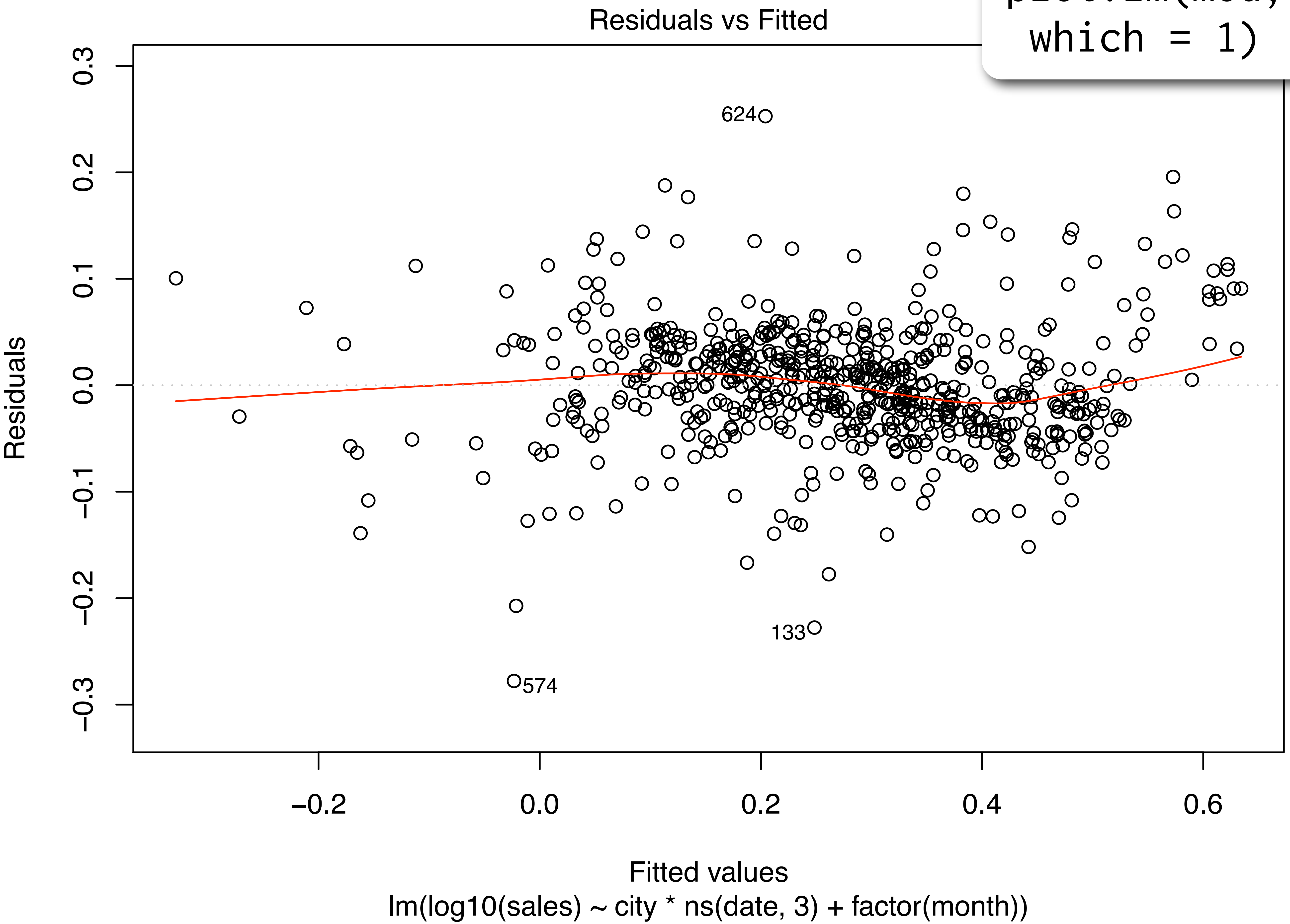
Model visualisation

(with ggplot2)

Hadley Wickham
Rice University

1. Introducing plot.lm
2. The current state of play. Why this is suboptimal.
3. A better strategy: separate data from representation.
4. Why a canned set of plots is not good enough.

```
plot.lm(mod,
 which = 1)
```



```
#  File src/library/stats/R/plot.lm.R
#  Part of the R package, http://www.R-project.org
#
#  This program is free software; you can redistribute it and/or modify
#  it under the terms of the GNU General Public License as published by
#  the Free Software Foundation; either version 2 of the License, or
#  (at your option) any later version.
#
#  This program is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.
#
#  A copy of the GNU General Public License is available at
#  http://www.r-project.org/Licenses/

plot.lm <-
function (x, which = c(1L:3,5), ## was which = 1L:4,
          caption = list(”Residuals vs Fitted”, ”Normal Q-Q”,
          ”Scale-Location”, ”Cook’s distance”,
          ”Residuals vs Leverage”,
          expression(”Cook’s dist vs Leverage  ” * h[ii] / (1 - h[ii]))),
          panel = if(add.smooth) panel.smooth else points,
          sub.caption = NULL, main = ””,
          ask = prod(par(”mfcol”)) < length(which) && dev.interactive(), ...,
          id.n = 3, labels.id = names(residuals(x)), cex.id = 0.75,
          qqline = TRUE, cook.levels = c(0.5, 1.0),
          add.smooth = getOption(”add.smooth”),
          label.pos = c(4,2), cex.caption = 1)
{
    dropInf <- function(x, h) {
        if(any(isInf <- h >= 1.0)) {
            warning(”Not plotting observations with leverage one:\n  ”,
                    paste(which(isInf), collapse=”, ”),
                    call.=FALSE)
            x[isInf] <- NaN
        }
        x
    }

    if (!inherits(x, ”lm”))
        stop(”use only with \”lm\” objects”)
    if(!is.numeric(which) || any(which < 1) || any(which > 6))
        stop(”’which’ must be in 1L:6”)
    isGlm <- inherits(x, ”glm”)
    show <- rep(FALSE, 6)
```

```
show[which] <- TRUE
r <- residuals(x)
yh <- predict(x) # != fitted() for glm
w <- weights(x)
if(!is.null(w)) { # drop obs with zero wt: PR#6640
    wind <- w != 0
    r <- r[wind]
    yh <- yh[wind]
    w <- w[wind]
    labels.id <- labels.id[wind]
}
n <- length(r)
if (any(show[2L:6L])) {
    s <- if (inherits(x, ”rlm”)) x$s
    else if(isGlm) sqrt(summary(x)$dispersion)
    else sqrt(deviance(x)/df.residual(x))
    hii <- lm.influence(x, do.coef = FALSE)$hat
    if (any(show[4L:6L])) {
        cook <- if (isGlm) cooks.distance(x)
        else cooks.distance(x, sd = s, res = r)
    }
}
if (any(show[2L:3L])) {
    ylab23 <- if(isGlm) ”Std. deviance resid.” else ”Standardized residuals”
    r.w <- if (is.null(w)) r else sqrt(w) * r
    ## NB: rs is already NaN if r=0, hii=1
    rs <- dropInf( r.w/(s * sqrt(1 - hii)), hii )
}

if (any(show[5L:6L])) { # using ’leverages’
    r.hat <- range(hii, na.rm = TRUE) # though should never have NA
    isConst.hat <- all(r.hat == 0) ||
        diff(r.hat) < 1e-10 * mean(hii, na.rm = TRUE)
}
if (any(show[c(1L, 3L)]))
    l.fit <- if (isGlm) ”Predicted values” else ”Fitted values”
if (is.null(id.n))
    id.n <- 0
else {
    id.n <- as.integer(id.n)
    if(id.n < 0L || id.n > n)
        stop(gettextf(”’id.n’ must be in {1,..,%d}”, n), domain = NA)
}
if(id.n > 0L) { ## label the largest residuals
    if(is.null(labels.id))
        labels.id <- paste(1L:n)
```

```
        iid <- 1L:id.n
        show.r <- sort.list(abs(r), decreasing = TRUE)[iid]
        if(any(show[2L:3L]))
            show.rs <- sort.list(abs(rs), decreasing = TRUE)[iid]
        text.id <- function(x, y, ind, adj.x = TRUE) {
            labpos <-
                if(adj.x) label.pos[1+as.numeric(x > mean(range(x)))] else 3
            text(x, y, labels.id[ind], cex = cex.id, xpd = TRUE,
                 pos = labpos, offset = 0.25)
        }
    }
    getCaption <- function(k) # allow caption = "" , plotmath etc
        as.graphicsAnnot(unlist(caption[k]))

    if(is.null(sub.caption)) { ## construct a default:
        cal <- x$call
        if (!is.na(m.f <- match("formula", names(cal)))) {
            cal <- cal[c(1, m.f)]
            names(cal)[2L] <- "" # drop        " formula = "
        }
        cc <- deparse(cal, 80) # (80, 75) are ``parameters''
        nc <- nchar(cc[1L], "c")
        abbr <- length(cc) > 1 || nc > 75
        sub.caption <-
            if(abbr) paste(substr(cc[1L], 1L, min(75L, nc)), "...") else cc[1L]
    }
    one.fig <- prod(par("mfcol")) == 1
    if (ask) {
        oask <- devAskNewPage(TRUE)
        on.exit(devAskNewPage(oask))
    }
    ##---------- Do the individual plots : ----------
    if (show[1L]) {
        ylim <- range(r, na.rm=TRUE)
        if(id.n > 0)
            ylim <- extendrange(r= ylim, f = 0.08)
        plot(yh, r, xlab = l.fit, ylab = "Residuals", main = main,
             ylim = ylim, type = "n", ...)
        panel(yh, r, ...)
        if (one.fig)
            title(sub = sub.caption, ...)
        mtext(getCaption(1), 3, 0.25, cex = cex.caption)
        if(id.n > 0) {
            y.id <- r[show.r]
            y.id[y.id < 0] <- y.id[y.id < 0] - strheight(" ")/3
            text.id(yh[show.r], y.id, show.r)
```

```
        }
        abline(h = 0, lty = 3, col = "gray")
    }
    if (show[2L]) { ## Normal
        ylim <- range(rs, na.rm=TRUE)
        ylim[2L] <- ylim[2L] + diff(ylim) * 0.075
        qq <- qqnorm(rs, main = main, ylab = ylab23, ylim = ylim, ...)
        if (qqline) qqline(rs, lty = 3, col = "gray50")
        if (one.fig)
            title(sub = sub.caption, ...)
        mtext(getCaption(2), 3, 0.25, cex = cex.caption)
        if(id.n > 0)
            text.id(qq$x[show.rs], qq$y[show.rs], show.rs)
    }
    if (show[3L]) {
        sqrtabsr <- sqrt(abs(rs))
        ylim <- c(0, max(sqrtabsr, na.rm=TRUE))
        yl <- as.expression(substitute(sqrt(abs(YL)), list(YL=as.name(ylab23))))
        yhn0 <- if(is.null(w)) yh else yh[w!=0]
        plot(yhn0, sqrtabsr, xlab = l.fit, ylab = yl, main = main,
             ylim = ylim, type = "n", ...)
        panel(yhn0, sqrtabsr, ...)
        if (one.fig)
            title(sub = sub.caption, ...)
        mtext(getCaption(3), 3, 0.25, cex = cex.caption)
        if(id.n > 0)
            text.id(yhn0[show.rs], sqrtabsr[show.rs], show.rs)
    }
    if (show[4L]) {
        if(id.n > 0) {
            show.r <- order(-cook)[iid]# index of largest 'id.n' ones
            ymx <- cook[show.r[1L]] * 1.075
        } else ymx <- max(cook, na.rm = TRUE)
        plot(cook, type = "h", ylim = c(0, ymx), main = main,
             xlab = "Obs. number", ylab = "Cook's distance", ...)
        if (one.fig)
            title(sub = sub.caption, ...)
        mtext(getCaption(4), 3, 0.25, cex = cex.caption)
        if(id.n > 0)
            text.id(show.r, cook[show.r], show.r, adj.x=FALSE)
    }
    if (show[5L]) {
        ylab5 <- if (isGlm) "Std. Pearson resid." else "Standardized residuals"
        r.w <- residuals(x, "pearson")
        if(!is.null(w)) r.w <- r.w[wind] # drop 0-weight cases
```

```
         rsp <- dropInf( r.w/(s * sqrt(1 - hii)), hii )
        ylim <- range(rsp, na.rm = TRUE)
        if (id.n > 0) {
            ylim <- extendrange(r= ylim, f = 0.08)
            show.rsp <- order(-cook)[iid]
        }
        do.plot <- TRUE
        if(isConst.hat) { ## leverages are all the same
            if(missing(caption)) # set different default
                caption[[5]] <- ”Constant Leverage:\n Residuals vs Factor Levels”
            ## plot against factor-level combinations instead
            aterms <- attributes(terms(x))
            ## classes w/o response
            dcl <- aterms$dataClasses[ -aterms$response ]
            facvars <- names(dcl)[dcl %in% c(”factor”, ”ordered”)]
            mf <- model.frame(x)[facvars]# better than x$model
            if(ncol(mf) > 0) {
                ## now re-order the factor levels *along* factor-effects
                ## using a ”robust” method {not requiring dummy.coef}:
                effM <- mf
                for(j in seq_len(ncol(mf)))
                    effM[, j] <- sapply(split(yh, mf[, j]), mean)[mf[, j]]
                ord <- do.call(order, effM)
                dm <- data.matrix(mf)[ord, , drop = FALSE]
                ## #{levels} for each of the factors:
                nf <- length(nlev <- unlist(unname(lapply(x$xlevels, length))))
                ff <- if(nf == 1) 1 else rev(cumprod(c(1, nlev[nf:2])))
                facval <- ((dm-1) %*% ff)
                ## now reorder to the same order as the residuals
                facval[ord] <- facval
                xx <- facval # for use in do.plot section.

                plot(facval, rsp, xlim = c(-1/2, sum((nlev-1) * ff) + 1/2),
                     ylim = ylim, xaxt = ”n”,
                     main = main, xlab = ”Factor Level Combinations”,
                     ylab = ylab5, type = ”n”, ...)
                axis(1, at = ff[1L]*(1L:nlev[1L] - 1/2) - 1/2,
                     labels= x$xlevels[[1L]][order(sapply(split(yh,mf[,1]),
mean))])
                mtext(paste(facvars[1L],”:”), side = 1, line = 0.25, adj=-.05)
                abline(v = ff[1L]*(0:nlev[1L]) - 1/2, col=”gray”, lty=”F4”)
                panel(facval, rsp, ...)
                abline(h = 0, lty = 3, col = ”gray”)
            }
            else { # no factors
                message(”hat values (leverages) are all = ”,
                        format(mean(r.hat)),
                        ”\n and there are no factor predictors; no plot no. 5”)
                frame()
                do.plot <- FALSE
            }
        }
        else { ## Residual vs Leverage
            xx <- hii
            ## omit hatvalues of 1.
            xx[xx >= 1] <- NA

            plot(xx, rsp, xlim = c(0, max(xx, na.rm = TRUE)), ylim = ylim,
                 main = main, xlab = ”Leverage”, ylab = ylab5, type = ”n”,
                 ...)
            panel(xx, rsp, ...)
            abline(h = 0, v = 0, lty = 3, col = ”gray”)
            if (one.fig)
                title(sub = sub.caption, ...)
            if(length(cook.levels)) {
                p <- length(coef(x))
                usr <- par(”usr”)
                hh <- seq.int(min(r.hat[1L], r.hat[2L]/100), usr[2L],
                              length.out = 101)
                for(crit in cook.levels) {
                    cl.h <- sqrt(crit*p*(1-hh)/hh)
                    lines(hh, cl.h, lty = 2, col = 2)
                    lines(hh,-cl.h, lty = 2, col = 2)
                }
                legend(”bottomleft”, legend = ”Cook’s distance”,
                       lty = 2, col = 2, bty = ”n”)
                xmax <- min(0.99, usr[2L])
                ymult <- sqrt(p*(1-xmax)/xmax)
                aty <- c(-sqrt(rev(cook.levels))*ymult,
                         sqrt(cook.levels)*ymult)
                axis(4, at = aty,
                     labels = paste(c(rev(cook.levels), cook.levels)),
                     mgp = c(.25,.25,0), las = 2, tck = 0,
                     cex.axis = cex.id, col.axis = 2)
            }
        } # if(const h_ii) .. else ..
        if (do.plot) {
            mtext(getCaption(5), 3, 0.25, cex = cex.caption)
            if (id.n > 0) {
                y.id <- rsp[show.rsp]
                y.id[y.id < 0] <- y.id[y.id < 0] - strheight(” ”)/3
```

```
                text.id(xx[show.rsp], y.id, show.rsp)
            }
        }
    }
    if (show[6L]) {
        g <- dropInf( hii/(1-hii), hii )
        ymx <- max(cook, na.rm = TRUE)*1.025
        plot(g, cook, xlim = c(0, max(g, na.rm=TRUE)), ylim = c(0, ymx),
             main = main, ylab = ”Cook’s distance”,
             xlab = expression(”Leverage  ” * h[ii]),
             xaxt = ”n”, type = ”n”, ...)
        panel(g, cook, ...)
        ## Label axis with h_ii values
        athat <- pretty(hii)
        axis(1, at = athat/(1-athat), labels = paste(athat))
        if (one.fig)
            title(sub = sub.caption, ...)
        p <- length(coef(x))
        bval <- pretty(sqrt(p*cook/g), 5)

        usr <- par(”usr”)
        xmax <- usr[2L]
        ymax <- usr[4L]
        for(i in 1L:length(bval)) {
            bi2 <- bval[i]^2
            if(ymax > bi2*xmax) {
                xi <- xmax + strwidth(” ”)/3
                yi <- bi2*xi
                abline(0, bi2, lty = 2)
                text(xi, yi, paste(bval[i]), adj = 0, xpd = TRUE)
            } else {
                yi <- ymax - 1.5*strheight(” ”)
                xi <- yi/bi2
                lines(c(0, xi), c(0, yi), lty = 2)
                text(xi, ymax-0.8*strheight(” ”), paste(bval[i]),
                     adj = 0.5, xpd = TRUE)
            }
        }

        ## axis(4, at=p*cook.levels, labels=paste(c(rev(cook.levels),
cook.levels)),
        ##        mgp=c(.25,.25,0), las=2, tck=0, cex.axis=cex.id)
        mtext(getCaption(6), 3, 0.25, cex = cex.caption)
        if (id.n > 0) {
            show.r <- order(-cook)[iid]
            text.id(g[show.r], cook[show.r], show.r)
```

```
        }
    }

    if (!one.fig && par(”oma”)[3L] >= 1)
        mtext(sub.caption, outer = TRUE, cex = 1.25)
    invisible()
}
```

Problems

Hard to understand.

Hard to extend.

Locked into set of pre-specified graphics.

Of no use to other graphics packages.

Alternative approach

What does this actually code do?

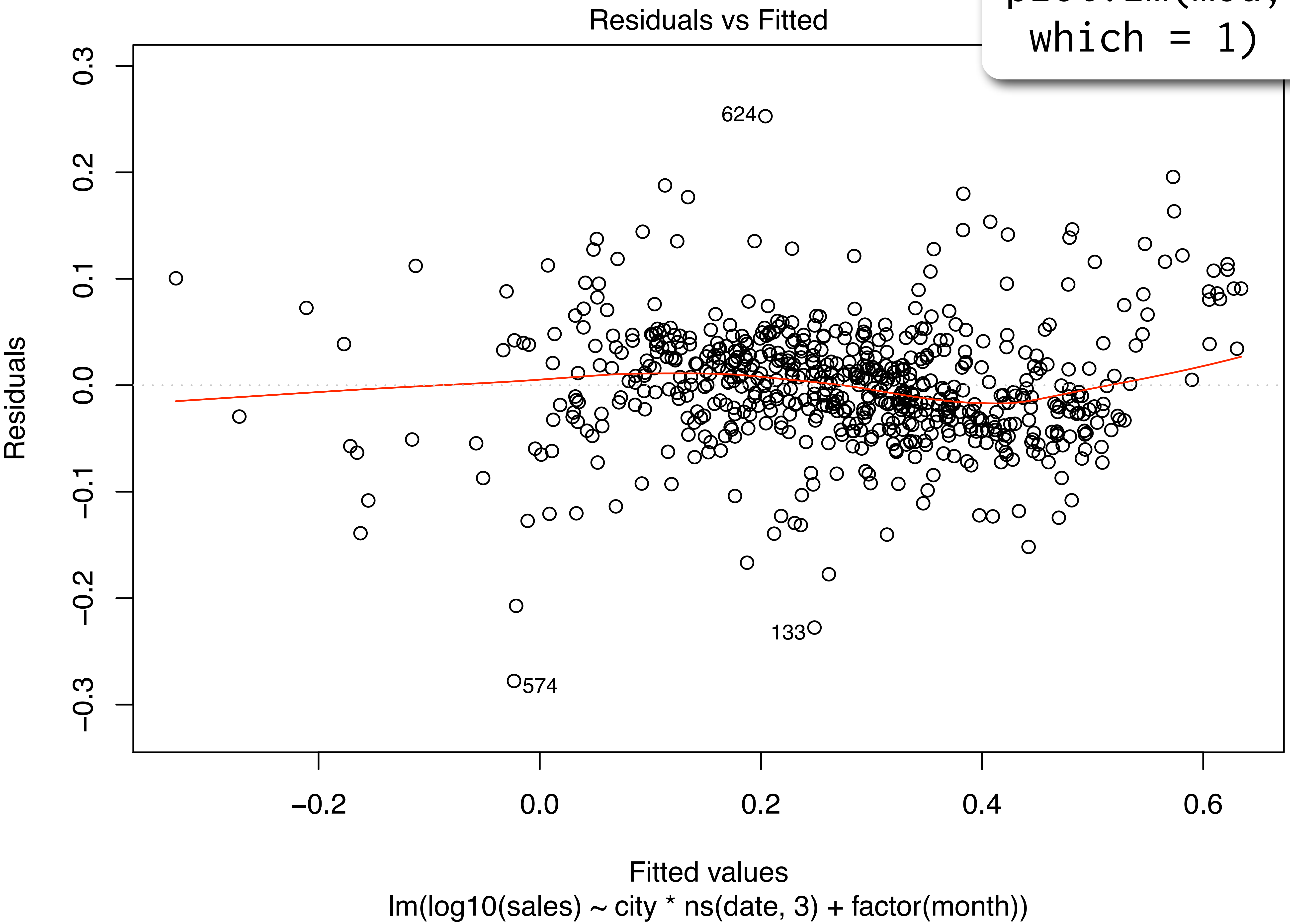
It **1)** extracts various quantities of interest from the model and then **2)** plots them

So why not perform those two tasks separately?

Quantities of interest

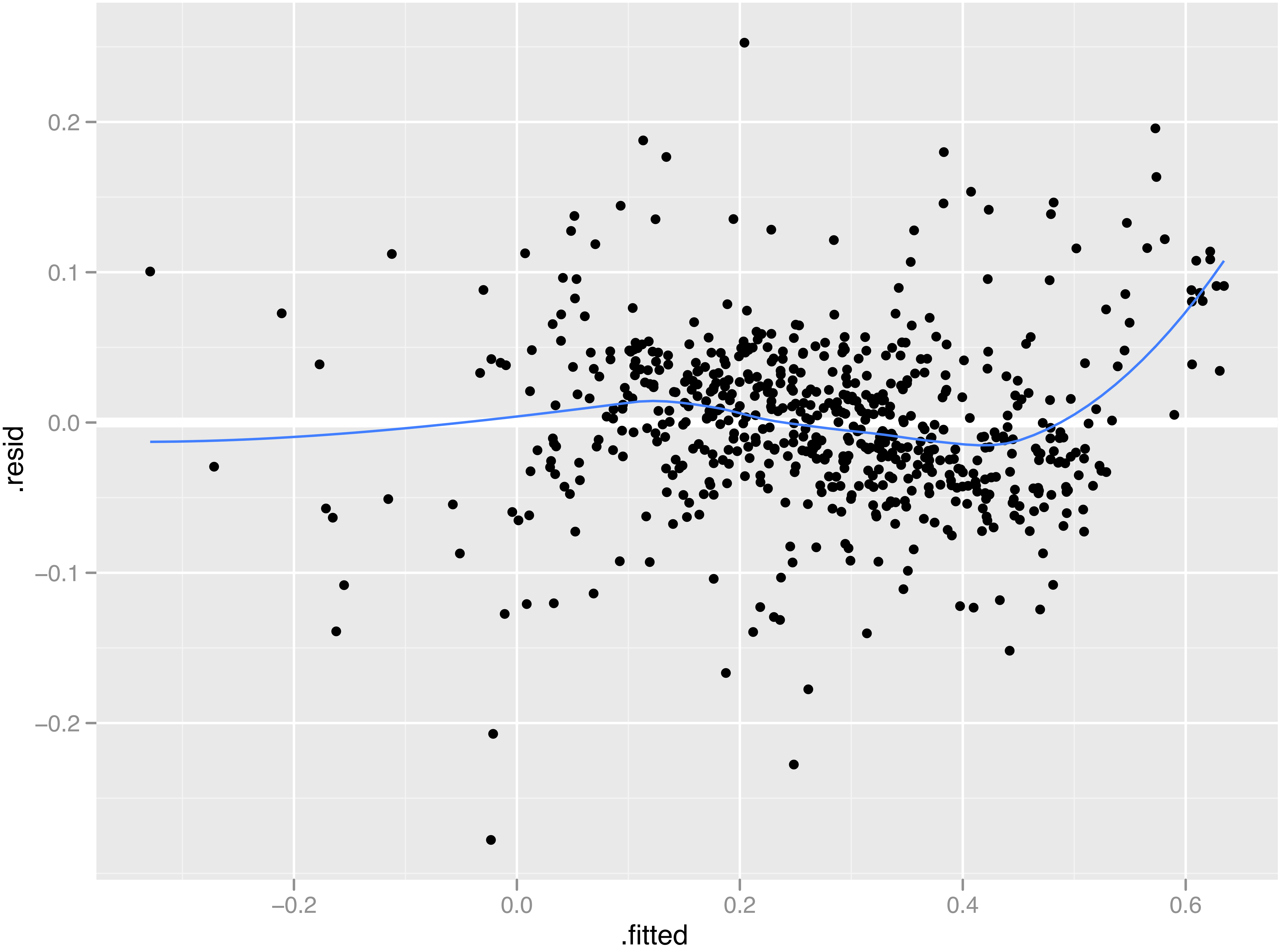
```
fortify.lm <- function(model, data = model$model, ...) {
  infl <- influence(model, do.coef = FALSE)
  data$.hat <- infl$hat
  data$.sigma <- infl$sigma
  data$.cooksd <- cooks.distance(model, infl)

  data$.fitted <- predict(model)
  data$.resid <- resid(model)
  data$.stdresid <- rstandard(model, infl)

  data
}
```

Note use of . prefix to avoid name clasehes

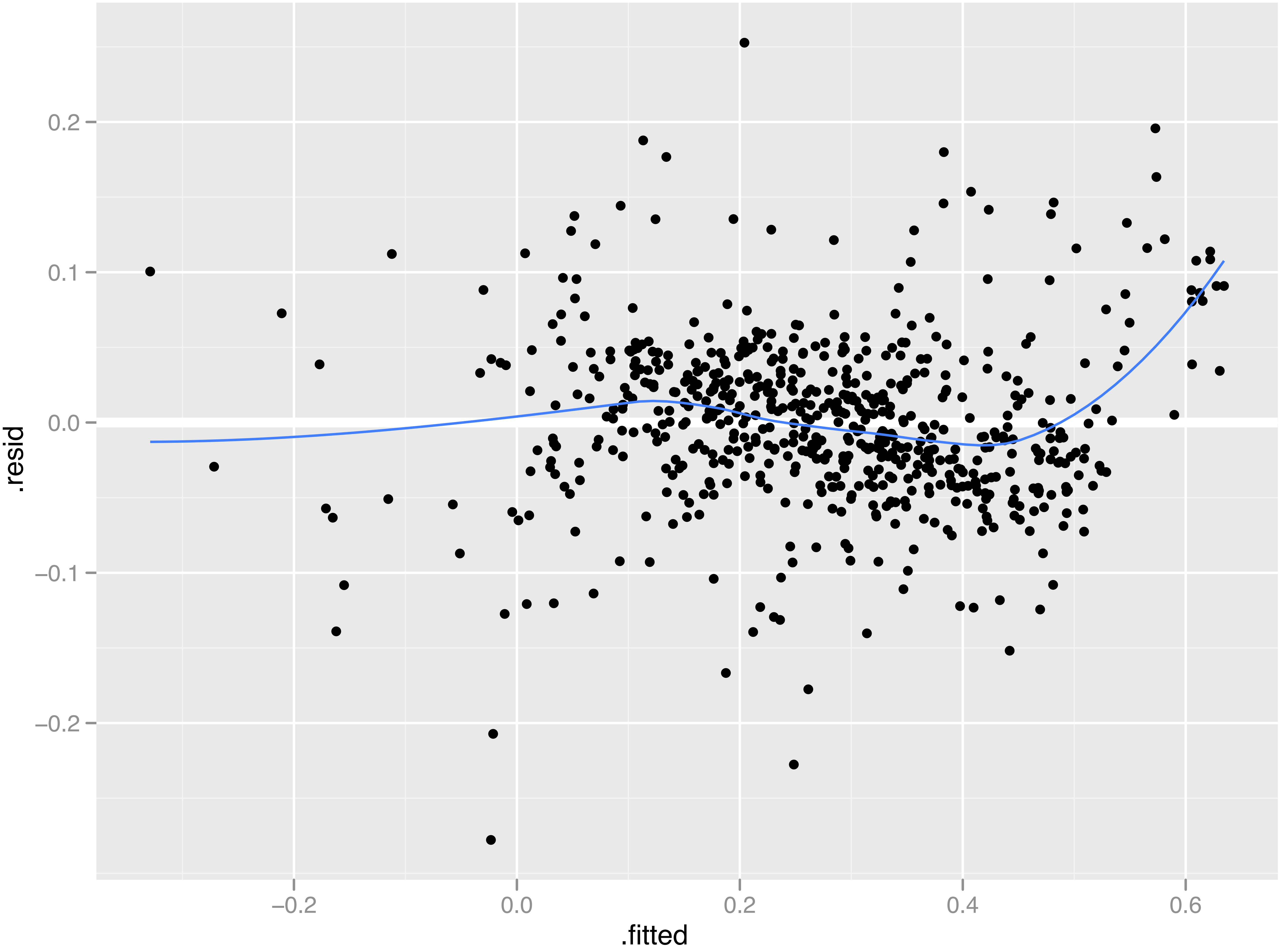


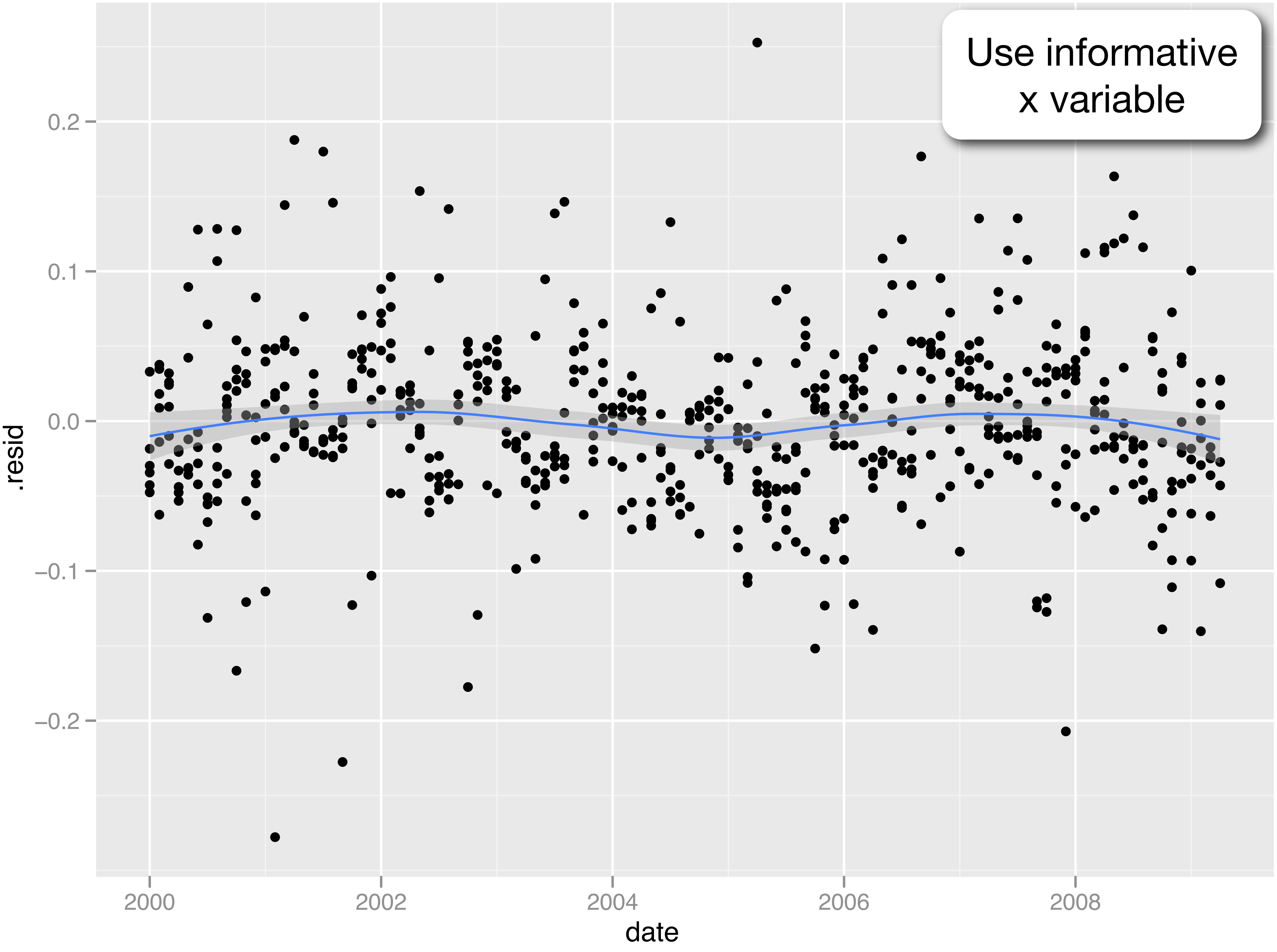
```
plot.lm(mod,
 which = 1)
```

```
ggplot(mod, aes(.fitted, .resid)) +
  geom_hline(yintercept = 0) +
  geom_point() +
  geom_smooth(se = F)
```


0.2
0.1
0.0
−0.1
−0.2
.resid
−0.2
0.0
0.2
0.4
0.6
.fitted

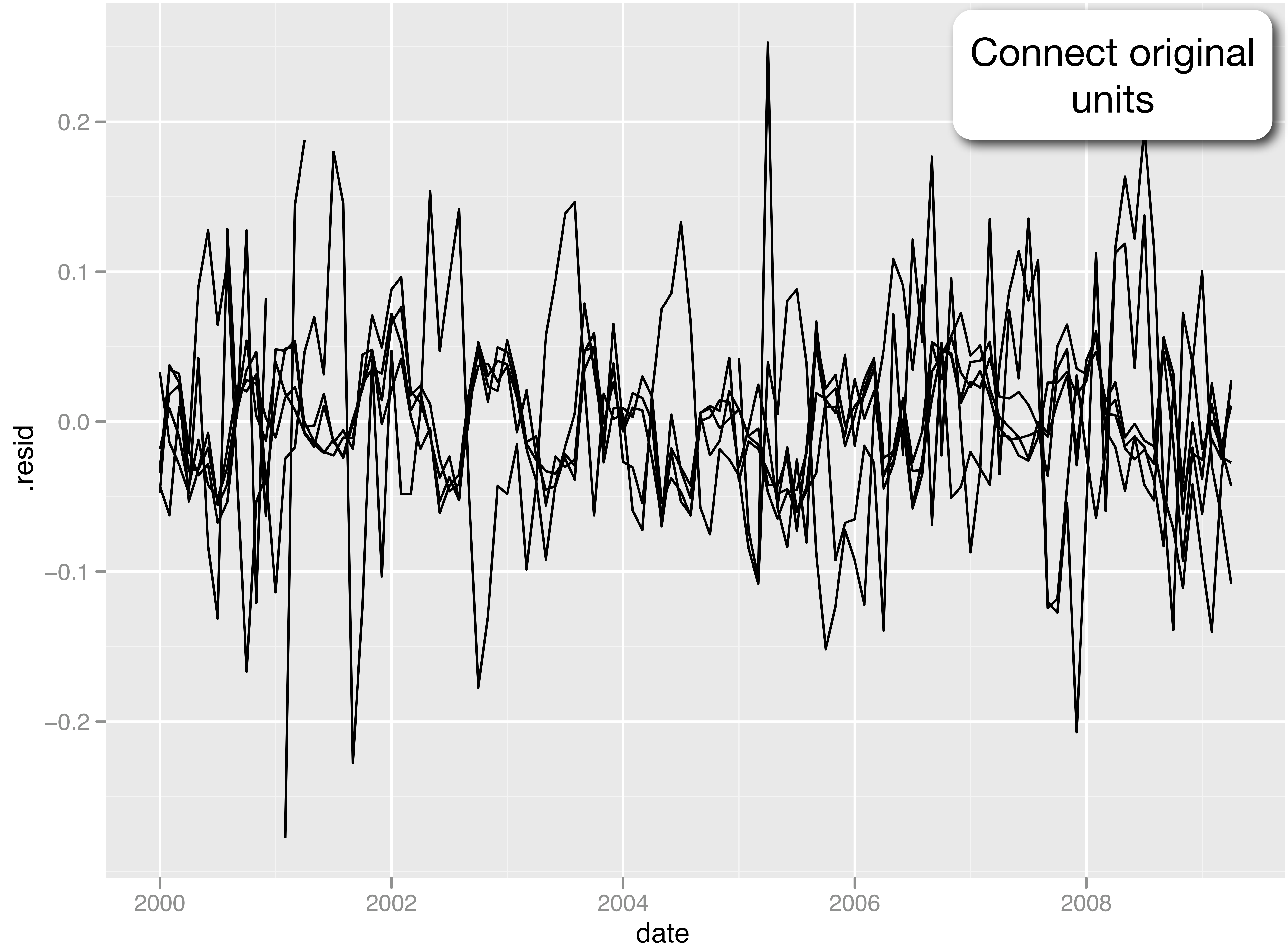
Diagnostics should reflect data


0.2
0.1
0.0
−0.1
−0.2
.resid
−0.2
0.0
0.2
0.4
0.6
.fitted



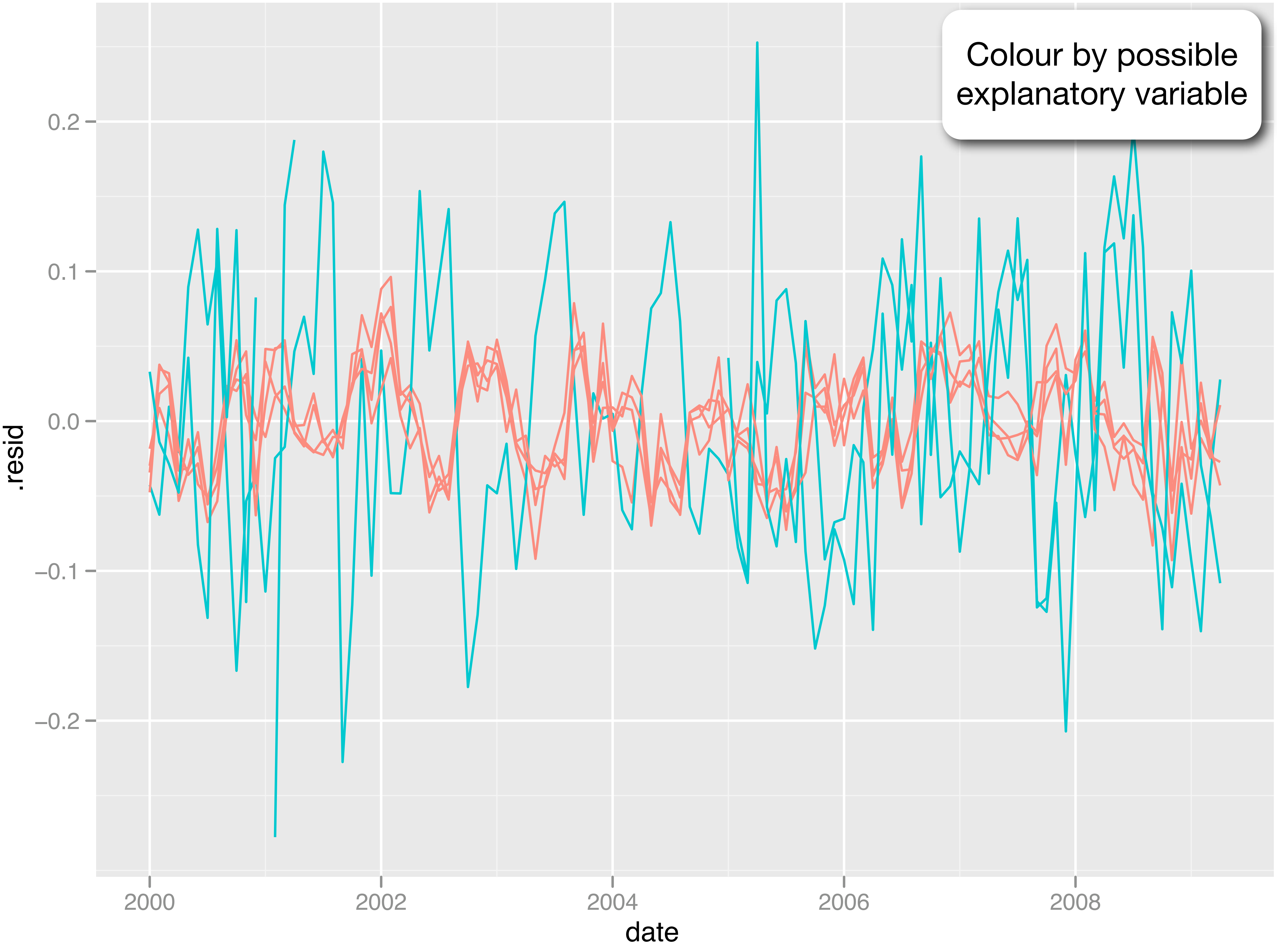
Use informative x variable

Monday, 13 July 2009

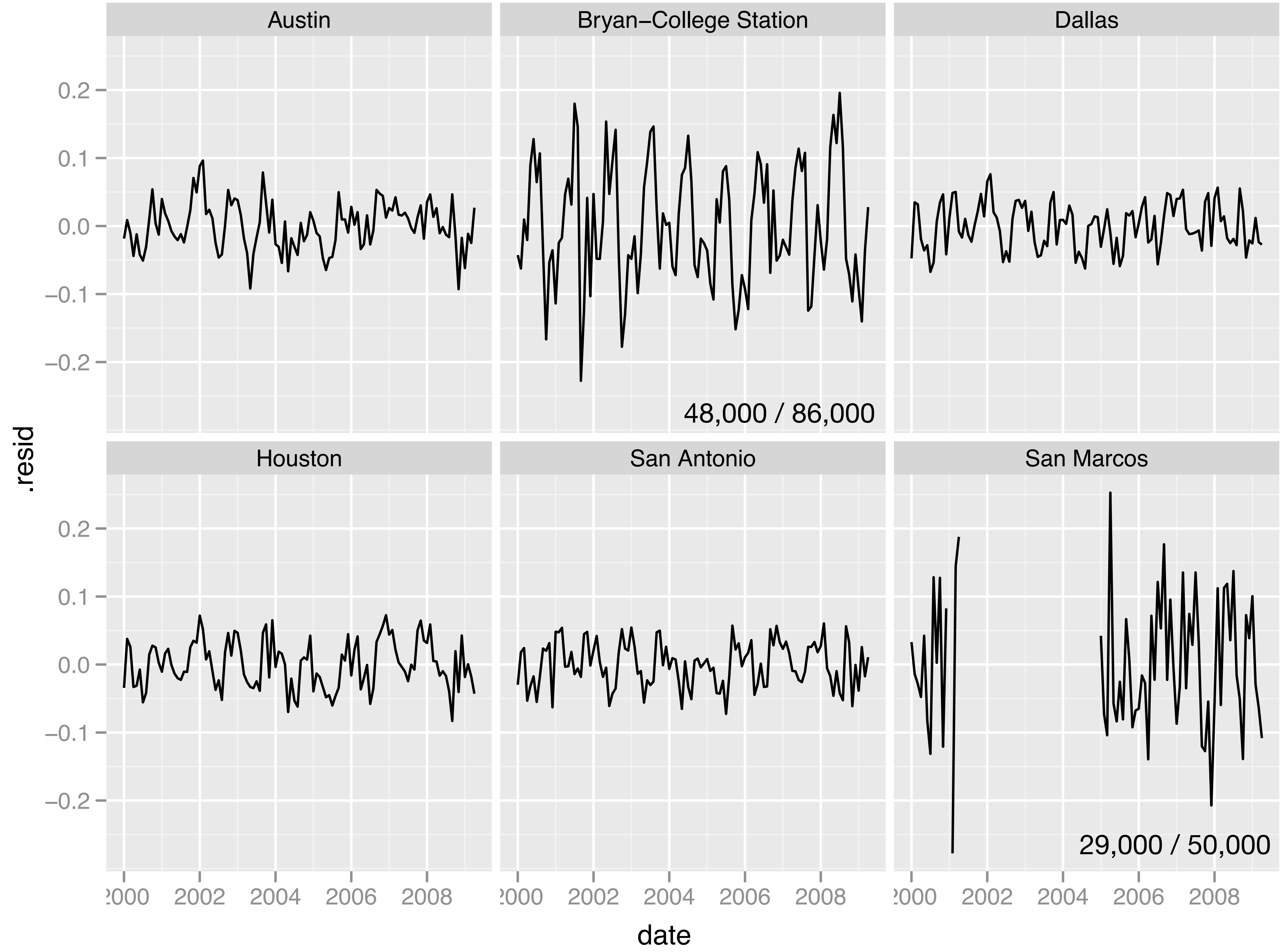


Connect original units

Monday, 13 July 2009



Colour by possible explanatory variable


Austin
Bryan−College Station
Dallas
Houston
San Antonio
San Marcos
0.2
0.1
0.0
−0.1
−0.2
.resid
48,000 / 86,000
29,000 / 50,000
2000
2002
2004
2006
2008
date

```
ggplot(modf, aes(date, .resid)) +
  geom_line(aes(group = city))

ggplot(modf, aes(date, .resid,
  colour = college_town)) +
  geom_line(aes(group = city))

ggplot(modf, aes(date, .resid)) +
  geom_line(aes(group = city)) +
  facet_wrap(~ city)
```

```r
fortify.lm <- function(model, data = model$model, ...) {
  infl <- influence(model, do.coef = FALSE)
  data$.hat <- infl$hat
  data$.sigma <- infl$sigma
  data$.cooksd <- cooks.distance(model, infl)

  data$.fitted <- predict(model)
  data$.resid <- resid(model)
  data$.stdresid <- rstandard(model, infl)

  data
}
```

```r
# Which = 1
ggplot(mod, aes(.fitted, .resid)) +
  geom_hline(yintercept = 0) +
  geom_point() +
  geom_smooth(se = F)

# Which = 2
ggplot(mod, aes(sample = .stdresid)) +
  stat_qq() +
  geom_abline()

# Which = 3
ggplot(mod, aes(.fitted, abs(.stdresid)) +
  geom_point() +
  geom_smooth(se = FALSE) +
  scale_y_sqrt()

# Which = 4
mod$row <- rownames(mod)
ggplot(mod, aes(row, .cooksd)) +
  geom_bar(stat = "identity")

# Which = 5
ggplot(mod, aes(.hat, .stdresid)) +
  geom_vline(size = 2, colour = "white", xintercept = 0) +
  geom_hline(size = 2, colour = "white", yintercept = 0) +
  geom_point() +
  geom_smooth(se = FALSE)

# Which = 6
ggplot(mod, aes(.hat, .cooksd, data = mod)) +
  geom_vline(colour = NA) +
  geom_abline(slope = seq(0, 3, by = 0.5), colour = "white") +
  geom_smooth(se = FALSE) +
  geom_point()
```

Other models

A work in progress: hard work because most of the functions are like plot.lm

Models: lm, tsdiag, survreg

Maps: maps, and sp classes. Much easier to work with data frames.

Conclusions

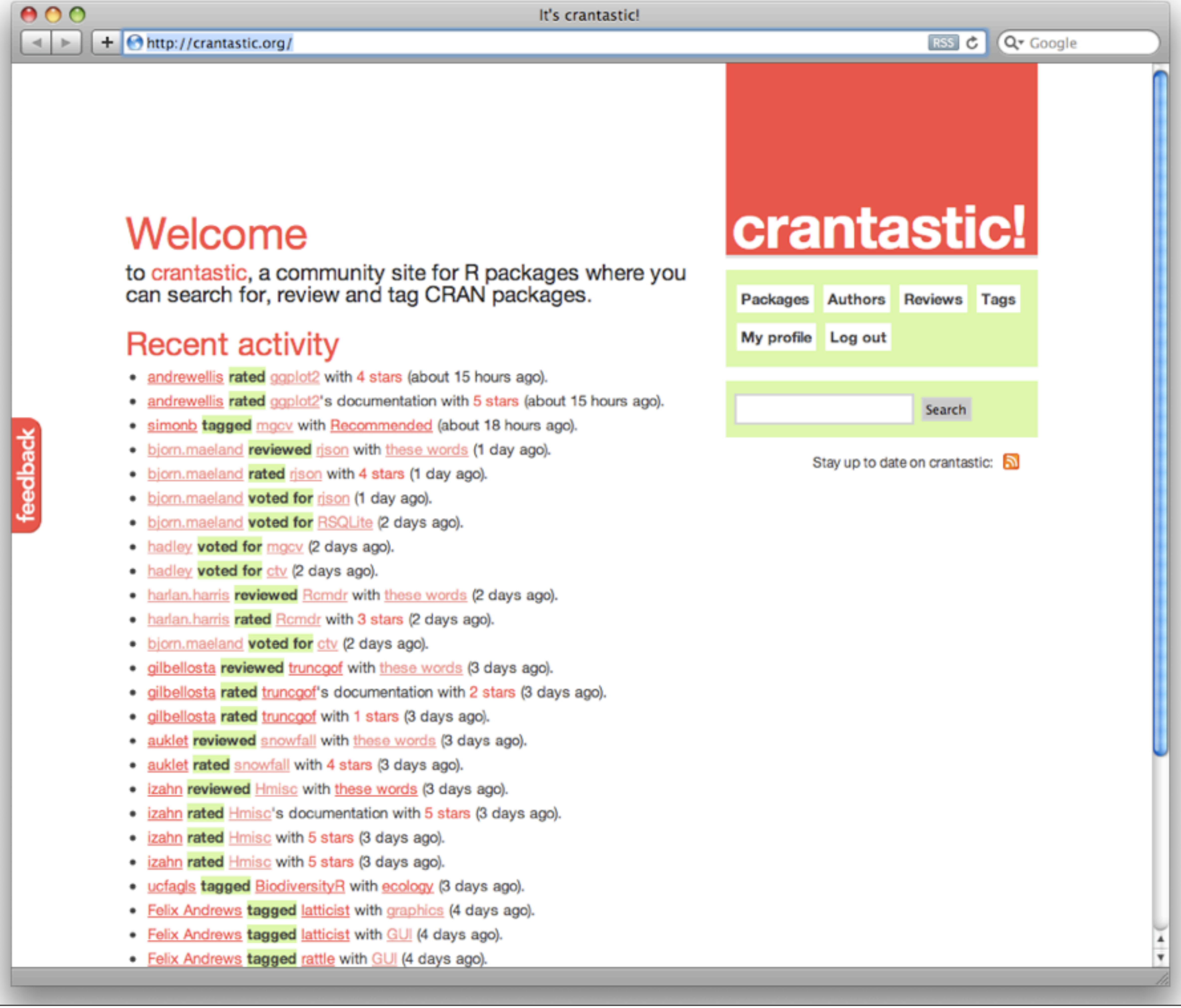
Separating data from visualisation improves clarity and reusability.

A pre-specified set of plots will not uncover many model problems. Should be easy custom diagnostics for your needs.

crantastic!

http://crantastic.org

A community site for finding, rating, and reviewing R packages.



Welcome

to crantastic, a community site for R packages where you can search for, review and tag CRAN packages.

Recent activity

- andrewellis **rated** ggplot2 with 4 stars (about 15 hours ago).
- andrewellis **rated** ggplot2's documentation with 5 stars (about 15 hours ago).
- simonb **tagged** mgcv with Recommended (about 18 hours ago).
- bjorn.maeland **reviewed** rjson with these words (1 day ago).
- bjorn.maeland **rated** rjson with 4 stars (1 day ago).
- bjorn.maeland **voted for** rjson (1 day ago).
- bjorn.maeland **voted for** RSQLite (2 days ago).
- hadley **voted for** mgcv (2 days ago).
- hadley **voted for** ctv (2 days ago).
- harlan.harris **reviewed** Rcmdr with these words (2 days ago).
- harlan.harris **rated** Rcmdr with 3 stars (2 days ago).
- bjorn.maeland **voted for** ctv (2 days ago).
- gilbellosta **reviewed** truncgof with these words (3 days ago).
- gilbellosta **rated** truncgof's documentation with 2 stars (3 days ago).
- gilbellosta **rated** truncgof with 1 stars (3 days ago).
- auklet **reviewed** snowfall with these words (3 days ago).
- auklet **rated** snowfall with 4 stars (3 days ago).
- izahn **reviewed** Hmisc with these words (3 days ago).
- izahn **rated** Hmisc's documentation with 5 stars (3 days ago).
- izahn **rated** Hmisc with 5 stars (3 days ago).
- izahn **rated** Hmisc with 5 stars (3 days ago).
- ucfagls **tagged** BiodiversityR with ecology (3 days ago).
- Felix Andrews **tagged** latticist with graphics (4 days ago).
- Felix Andrews **tagged** latticist with GUI (4 days ago).
- Felix Andrews **tagged** rattle with GUI (4 days ago).

Stay up to date on crantastic: