R Regression Methods

Interrogate R Output Objects

Paul E. Johnson

Center for Research Methods and Data Analysis
University of Kansas

2012

Outline

Methods: Things To Do “To” a Regression Object

```
bush1 <- glm(pres04 ~ partyid + sex + owngun, data=dat, family=
     binomial(link=logit))
```

pres04	Kerry, Bush
partyid	Factor with 7 levels, SD $\rightarrow$ SR
sex	Male, Female
owngun	Yes, No

Just for the Record, The Data Preparation Steps Were ...

```
preslev <- levels(dat$pres04)
dat$pres04[dat$pres04 %in% preslev[3:10]]<- NA
dat$pres04 <- factor(dat$pres04)
levels(dat$pres04) <- c("Kerry", "Bush")
plev <- levels(dat$partyid)
dat$partyid[dat$partyid %in% plev[8]] <- NA
dat$partyid <- factor(dat$partyid)
levels(dat$partyid) <- c("Strong Dem.", "Dem.", "Ind. Near Dem.", "
    Independent", "Ind. Near Repub.", "Repub.", "Strong Repub.")
dat$owngun[ dat$owngun == "REFUSED"] <- NA
levels(dat$sex) <- c("Male","Female")
dat$owngun <- relevel(dat$owngun, ref="NO")
```

First, Find Out What You Got I

```
attributes(bush1)
```

```
$names
 [1] "coefficients"      "residuals"
 [3] "fitted.values"     "effects"
 [5] "R"                 "rank"
 [7] "qr"                "family"
 [9] "linear.predictors" "deviance"
[11] "aic"               "null.deviance"
[13] "iter"              "weights"
[15] "prior.weights"     "df.residual"
[17] "df.null"           "y"
[19] "converged"         "boundary"
[21] "model"             "na.action"
[23] "call"              "formula"
[25] "terms"             "data"
[27] "offset"            "control"
[29] "method"            "contrasts"
[31] "xlevels"

$class
[1] "glm" "lm"
```

Understanding attributes

- If you see $, it means you have an S3 object
- That means you can just "take" values out of the object with the dollar sign operator using commands like

```
bush1$coefficients
```

```
              (Intercept)              partyidDem.
                   -3.571                    1.910
    partyidInd. Near Dem.      partyidIndependent
                    1.456                    3.464
  partyidInd. Near Repub.          partyidRepub.
                    5.468                    6.031
   partyidStrong Repub.              sexFemale
                    7.191                    0.049
               owngunYES
                    0.642
```

R Core Team Warns against $ Access

- A usage like this works

```
bush1$coefficients
```

- But it might not work in the future, if the internal contents of the glm object were to change
- We should instead use the "extractor method"

```
coefficients(bush1)
```

- Challenge: finding/remembering the extractor functions.
- Especially difficult because some VERY important extractor functions don't show up using usual methods of searching for them (AIC, coefficients)

Double-Check the glm Object's Class

- Ask the object what class it is from

```
class(bush1)
```

```
[1] "glm" "lm"
```

Ask R What Methods are declared to apply to a "glm" Object I

```
methods(class = "glm")
```

```
 [1] add1.glm*           anova.glm
 [3] confint.glm*        cooks.distance.glm*
 [5] deviance.glm*       drop1.glm*
 [7] effects.glm*        extractAIC.glm*
 [9] family.glm*         formula.glm*
[11] influence.glm*      logLik.glm*
[13] model.frame.glm     nobs.glm*
[15] predict.glm         print.glm
[17] residuals.glm       rstandard.glm
[19] rstudent.glm        summary.glm
[21] vcov.glm*           weights.glm*

   Non-visible functions are asterisked
```

Check methods for “lm” class I

```
methods(class = "lm")
```

```
 [1] add1.lm*            alias.lm*
 [3] anova.lm            case.names.lm*
 [5] confint.lm*         cooks.distance.lm*
 [7] deviance.lm*        dfbeta.lm*
 [9] dfbetas.lm*         drop1.lm*
[11] dummy.coef.lm*      effects.lm*
[13] extractAIC.lm*      family.lm*
[15] formula.lm*         hatvalues.lm
[17] influence.lm*       kappa.lm
[19] labels.lm*          logLik.lm*
[21] model.frame.lm      model.matrix.lm
[23] nobs.lm*            plot.lm
[25] predict.lm          print.lm
[27] proj.lm*            qr.lm*
[29] residuals.lm        rstandard.lm
[31] rstudent.lm         simulate.lm*
[33] summary.lm          variable.names.lm*
[35] vcov.lm*

   Non-visible functions are asterisked
```

Looking Into the Class Hierarchy

- Functions are always located inside packages. With R, several packages are supplied and are automatically searched for methods.
- Read the source code for some of your favorite functions.

```
lm
predict.lm
glm
predict.glm
```

- For functions in packages that are loaded, typing its name (without telling R what package it lives in) will show its contents.

Functions, Methods and Hidden Methods

- Methods are ALSO FOUND if we ask for them explicitly with their namespace (and two colons)..

  ```
  stats::lm
  stats::predict.lm
  stats::glm
  stats::predict.glm
  ```

  Result should be identical to previous code.
- Hidden methods: Functions that are not “exported” by the package writer remain hidden
- functions used by package author, but they don’t want create confusion by having users access them directly
- You can see code for hidden methods if you use three colons.

  ```
  stats:::confint.lm
  stats:::weights.glm
  ```

The First Method Used is usually `summary()` I

```
summary(bush1)
```

```
Call:
glm(formula = pres04 ~ partyid + sex + owngun, family = binomial(
    link = logit),
    data = dat)

Deviance Residuals:
   Min       1Q   Median       3Q      Max
-2.941   -0.488   0.163    0.390    2.683

Coefficients:
                        Estimate  Std. Error  z value
(Intercept)              -3.5712      0.3934    -9.08
partyidDem.               1.9103      0.3972     4.81
partyidInd. Near Dem.     1.4559      0.4348     3.35
partyidIndependent        3.4642      0.4105     8.44
partyidInd. Near Repub.   5.4677      0.5073    10.78
partyidRepub.             6.0307      0.4502    13.39
partyidStrong Repub.      7.1908      0.6213    11.57
sexFemale                 0.0488      0.1928     0.25
owngunYES                 0.6424      0.1937     3.32
                        Pr(>|z|)
(Intercept)              < 2e-16 ***
```

The First Method Used is usually summary() II

```
partyidDem.                  1.5e-06 ***
partyidInd. Near Dem.        0.00081 ***
partyidIndependent           < 2e-16 ***
partyidInd. Near Repub.      < 2e-16 ***
partyidRepub.                < 2e-16 ***
partyidStrong Repub.         < 2e-16 ***
sexFemale                    0.80006
owngunYES                    0.00091 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1721.9  on 1242  degrees of freedom
Residual deviance:  764.0  on 1234  degrees of freedom
  (3267 observations deleted due to missingness)
AIC: 782

Number of Fisher Scoring iterations: 6
```

Summary Object I

Create a Summary Object

```
sb1 <- summary(bush1)
attributes(sb1)
```

```
$names
 [1] "call"            "terms"           "family"
 [4] "deviance"        "aic"             "contrasts"
 [7] "df.residual"     "null.deviance"   "df.null"
[10] "iter"            "na.action"       "deviance.resid"
[13] "coefficients"    "aliased"         "dispersion"
[16] "df"              "cov.unscaled"    "cov.scaled"

$class
[1] "summary.glm"
```

My deviance is

```
sb1$deviance
```

```
[1] 764
```

The coef Enigma I

- `coef()` is the same as `coefficients()`
- Note the Bizarre Truth:
  1. that the “coef” function returns something different when it is applied to a model object

```
coef(bush1)
```

```
           (Intercept)              partyidDem.
                −3.571                    1.910
  partyidInd. Near Dem.      partyidIndependent
                 1.456                    3.464
partyidInd. Near Repub.           partyidRepub.
                 5.468                    6.031
   partyidStrong Repub.               sexFemale
                 7.191                    0.049
             owngunYES
                 0.642
```

Than is returned from a summary object.

```
coef(sb1)
```

The coef Enigma II

```
                           Estimate Std. Error z value
(Intercept)                  −3.571      0.39   −9.08
partyidDem.                   1.910      0.40    4.81
partyidInd. Near Dem.         1.456      0.43    3.35
partyidIndependent            3.464      0.41    8.44
partyidInd. Near Repub.       5.468      0.51   10.78
partyidRepub.                 6.031      0.45   13.39
partyidStrong Repub.          7.191      0.62   11.57
sexFemale                     0.049      0.19    0.25
owngunYES                     0.642      0.19    3.32
                           Pr(>|z|)
(Intercept)                 1.1e−19
partyidDem.                 1.5e−06
partyidInd. Near Dem.       8.1e−04
partyidIndependent          3.2e−17
partyidInd. Near Repub.     4.3e−27
partyidRepub.               6.5e−41
partyidStrong Repub.        5.6e−31
sexFemale                   8.0e−01
owngunYES                   9.1e−04
```

anova() I

- You can apply anova() to just one model
- That gives a "stepwise" series of comparisons (not very useful)

```
anova(bush1, test="Chisq")
```

```
Analysis of Deviance Table

Model: binomial, link: logit

Response: pres04

Terms added sequentially (first to last)


        Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                     1242       1722
partyid  6      947      1236        775  < 2e-16 ***
sex      1        0      1235        775  0.97862
owngun   1       11      1234        764  0.00087 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

But anova Very Useful to Compare 2 Models

Here's the basic procedure:

1. Fit 1 big model, "mod1"
2. Exclude some variables to create a smaller model, "mod2"
3. Run `anova()` to compare:
   anova(mod1, mod2, test="Chisq")
4. If resulting test statistic is far from 0, it means the big model really is better and you should keep those variables in there.

Quick Reminder:

- In an OLS model, this is would be an F test for the hypothesis that the coefficients for omitted parameters are all equal to 0.
- In a model estimated by maximum likelihood, it is a likelihood ratio test with df= number of omitted parameters.

But there's an anova "Gotcha" I

```
> anova(bush0, bush1, test="Chisq")
Error in anova.glmlist(c(list(object), dotargs),
  dispersion = dispersion,  :
  models were not all fitted to the same size of dataset
```

What the Heck?

anova() Gotcha, cont.

- Explanation: Listwise Deletion of Missing Values causes this. Missings cause sample sizes to differ when variables change.
- One Solution: Fit both models on same data.
  1. Fit the “big model” (one with most variables)

     ```
     mod1 <- glm(y~ x1+ x2 + x3 + (more variables), data=dat,
         family=binomial)
     ```

  2. Fit the “smaller Model” with the data extracted from the fit of the previous model (model.frame(mod1), extractor for mod1$model) as the data frame

     ```
     mod2 <- glm(y~ x3 + (some variables), data=model.frame(
         mod1), family=binomial)
     ```

  3. After that, anova() will work

Example anova()

- Here's the big model

```
bush3 <- glm(pres04 ~ partyid + sex + owngun + race + wrkslf +
      realinc + polviews , data=dat, family=binomial(link=
     logit))
```

- Here's the small model

```
bush4 <- glm(pres04 ~ partyid +  owngun + race + polviews ,
     data=model.frame(bush3), family=binomial(link=logit))
```

anova(): The Big Reveal!

- anova:

```
anova(bush3, bush4, test="Chisq")
```

```
Analysis of Deviance Table

Model 1: pres04 ~ partyid + sex + owngun + race + wrkslf +
    realinc + polviews
Model 2: pres04 ~ partyid + owngun + race + polviews
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1      1044        589
2      1047        593 -3     -4.1     0.25
```

- Conclusion: the big model is not statistically significantly better than the small model
- Same as: Can't reject the null hypothesis that $\beta_j = 0$ for all omitted parameters

Interesting Use of anova

- Consider the fit for "polviews" in bush3 (recall "extremely liberal" is the reference category, the intercept)

label:	lib.	slt. lib.	mod.	sl. con.	con.	extr. con.
mle($\hat{\beta}$):	0.41	1.3	1.8*	2.5*	2.6*	3.1*
se:	0.88	0.83	0.79	0.83	0.84	1.2

* $p \leq 0.05$

- I wonder: are all "conservatives" the same? Do we really need separate parameter estimates for those respondents?

Use anova() To Test the Recoding

1. Make a New Variable for the New Coding

```
dat$newpolv <- dat$polviews
(levnpv <- levels(dat$newpolv))
```

```
[1] "EXTREMELY LIBERAL"    "LIBERAL"
[3] "SLIGHTLY LIBERAL"     "MODERATE"
[5] "SLGHTLY CONSERVATIVE" "CONSERVATIVE"
[7] "EXTRMLY CONSERVATIVE"
```

```
dat$newpolv[dat$newpolv %in% levnpv[5:7]] <- levnpv[6]
```

- Effect is to set slight and extreme conservatives into the conservative category

Better Check newpolv

```
dat$newpolv <- factor(dat$newpolv)
table(dat$newpolv)
```

```
EXTREMELY LIBERAL             LIBERAL
              139                 524
 SLIGHTLY LIBERAL            MODERATE
              517                1683
     CONSERVATIVE
             1470
```

Neat anova thing, cont.

1. Fit a new regression model, replacing polviews with newpolv

```
bush5 <- glm(pres04 ~ partyid + sex + owngun + race + wrkslf +
    realinc + newpolv , data=dat, family=binomial(link=logit))
```

2. Use anova() to test:

```
anova(bush3, bush5, test="Chisq")
```

```
Analysis of Deviance Table

Model 1: pres04 ~ partyid + sex + owngun + race + wrkslf +
    realinc + polviews
Model 2: pres04 ~ partyid + sex + owngun + race + wrkslf +
    realinc + newpolv
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1      1044        589
2      1046        589 -2   -0.431     0.81
```

- Apparently, all conservatives really are alike :)
- A similar test for liberals is left to the reader!

drop1 Relieves Tedium

- drop1() repeats the anova() procedure, removing each variable one-at-a-time.

```
drop1(bush3, test="Chisq")
```

```
Single term deletions

Model:
pres04 ~ partyid + sex + owngun + race + wrkslf + realinc +
    polviews
         Df Deviance AIC LRT Pr(>Chi)
<none>           589 627
partyid   6      951 977 362  < 2e-16 ***
sex       1      589 625   0    0.991
owngun    1      592 628   4    0.050 .
race      2      618 652  30  3.6e-07 ***
wrkslf    1      592 628   4    0.054 .
realinc   1      589 625   0    0.761
polviews  6      628 654  40  5.7e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- Recall "Chisq" ⇔ L.L.R test.

Variance-Covariance Matrix of $\hat{\beta}$ I

```
bush1Vcov <- vcov(bush1)
round(bush1Vcov, 3)
```

```
                        (Intercept) partyidDem.
(Intercept)                   0.155      -0.130
partyidDem.                  -0.130       0.158
partyidInd. Near Dem.        -0.132       0.130
partyidIndependent           -0.133       0.130
partyidInd. Near Repub.      -0.137       0.130
partyidRepub.                -0.135       0.130
partyidStrong Repub.         -0.134       0.130
sexFemale                    -0.025      -0.001
owngunYES                    -0.019       0.001
                        partyidInd. Near Dem.
(Intercept)                            -0.132
partyidDem.                             0.130
partyidInd. Near Dem.                   0.189
partyidIndependent                      0.130
partyidInd. Near Repub.                 0.131
partyidRepub.                           0.130
partyidStrong Repub.                    0.130
sexFemale                               0.003
owngunYES                               0.000
                        partyidIndependent
```

Variance-Covariance Matrix of $\hat{\beta}$ II

```
(Intercept)                        −0.133
partyidDem.                         0.130
partyidInd. Near Dem.               0.130
partyidIndependent                  0.168
partyidInd. Near Repub.             0.131
partyidRepub.                       0.131
partyidStrong Repub.                0.130
sexFemale                           0.004
owngunYES                           0.001
                        partyidInd. Near Repub.
(Intercept)                              −0.137
partyidDem.                               0.130
partyidInd. Near Dem.                     0.131
partyidIndependent                        0.131
partyidInd. Near Repub.                   0.257
partyidRepub.                             0.132
partyidStrong Repub.                      0.131
sexFemale                                 0.006
owngunYES                                 0.007
                        partyidRepub.
(Intercept)                    −0.135
partyidDem.                     0.130
partyidInd. Near Dem.           0.130
partyidIndependent              0.131
partyidInd. Near Repub.         0.132
```

Variance-Covariance Matrix of $\hat{\beta}$ III

```
partyidRepub.                        0.203
partyidStrong Repub.                 0.131
sexFemale                            0.004
owngunYES                            0.006
                         partyidStrong Repub.
(Intercept)                                   -0.134
partyidDem.                                    0.130
partyidInd. Near Dem.                          0.130
partyidIndependent                             0.130
partyidInd. Near Repub.                        0.131
partyidRepub.                                  0.131
partyidStrong Repub.                           0.386
sexFemale                                      0.003
owngunYES                                      0.004
                         sexFemale owngunYES
(Intercept)                 -0.025    -0.019
partyidDem.                 -0.001     0.001
partyidInd. Near Dem.        0.003     0.000
partyidIndependent           0.004     0.001
partyidInd. Near Repub.      0.006     0.007
partyidRepub.                0.004     0.006
partyidStrong Repub.         0.003     0.004
sexFemale                    0.037     0.003
owngunYES                    0.003     0.038
```

Variance-Covariance Matrix of $\hat{\beta}$ IV

These will match the "SE" column in the summary of bush1

```
sqrt(diag(vcov(bush1)))
```

```
              (Intercept)                  partyidDem.
                   0.3934                       0.3972
   partyidInd. Near Dem.           partyidIndependent
                   0.4348                       0.4105
partyidInd. Near Repub.              partyidRepub.
                   0.5073                       0.4502
    partyidStrong Repub.                    sexFemale
                   0.6213                       0.1928
                ownqunYES
                   0.1937
```

Heteroskedasticity-consistent Standard Errors?

Variants of the Huber-White "heteroskedasticity-consistent" (slang: robust) covarance matrix are available in "car" and "sandwich".

- hccm() in car works for linear models only
- vcovHC in the "sandwich" package returns a matrix of estimates. One should certainly read ?vcovHC and the associated literature.

```
library(sandwich)
myvcovHC <- vcovHC(bush1)
```

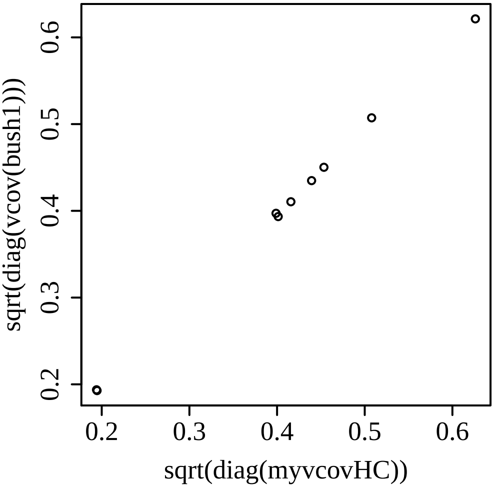
The heteroskedasticity consistent standard errors of the $\hat{\beta}$ are:

```
t(sqrt(diag(myvcovHC)))
```

```
     (Intercept) partyidDem.
[1,]      0.4013      0.3988
     partyidInd. Near Dem. partyidIndependent
[1,]            0.4394                 0.4158
     partyidInd. Near Repub. partyidRepub.
[1,]              0.5079         0.4535
     partyidStrong Repub. sexFemale owngunYES
[1,]               0.6262    0.1946    0.1941
```

Compare those: I

The HC and ordinary standard errors are almost identical:



Multicollinearity Diagnostics I

- VIF (Variance Inflation Factors) available in "car"
- rockchalk has "mcDiagnose"

```
library(rockchalk)
mcDiagnose(bush1)
```

```
The following auxiliary models are being estimated and returned
     in a list:
partyidDem. ~ `partyidInd. Near Dem.` + partyidIndependent +
    `partyidInd. Near Repub.` + partyidRepub. + `partyidStrong
         Repub.` +
    sexFemale + owngunYES
<environment: 0x3eb4560>
`partyidInd. Near Dem.` ~ partyidDem. + partyidIndependent +
    `partyidInd. Near Repub.` + partyidRepub. + `partyidStrong
         Repub.` +
    sexFemale + owngunYES
<environment: 0x3eb4560>
partyidIndependent ~ partyidDem. + `partyidInd. Near Dem.` +
    `partyidInd. Near Repub.` + partyidRepub. + `partyidStrong
         Repub.` +
    sexFemale + owngunYES
<environment: 0x3eb4560>
```

Multicollinearity Diagnostics II

```
`partyidInd. Near Repub.` ~ partyidDem. + `partyidInd. Near Dem.
    ` +
    partyidIndependent + partyidRepub. + `partyidStrong Repub.`
        +
    sexFemale + owngunYES
<environment: 0x3eb4560>
partyidRepub. ~ partyidDem. + `partyidInd. Near Dem.` +
    partyidIndependent +
    `partyidInd. Near Repub.` + `partyidStrong Repub.` +
        sexFemale +
    owngunYES
<environment: 0x3eb4560>
`partyidStrong Repub.` ~ partyidDem. + `partyidInd. Near Dem.` +
    partyidIndependent + `partyidInd. Near Repub.` +
        partyidRepub. +
    sexFemale + owngunYES
<environment: 0x3eb4560>
sexFemale ~ partyidDem. + `partyidInd. Near Dem.` +
    partyidIndependent +
    `partyidInd. Near Repub.` + partyidRepub. + `partyidStrong
        Repub.` +
    owngunYES
<environment: 0x3eb4560>
owngunYES ~ partyidDem. + `partyidInd. Near Dem.` +
    partyidIndependent +
```

Multicollinearity Diagnostics III

```
    `partyidInd. Near Repub.` + partyidRepub. + `partyidStrong
        Repub.` +
    sexFemale
<environment: 0x3eb4560>
Drum roll please!

And your R_j Squareds are (auxiliary Rsq)
             partyidDem.   partyidInd. Near Dem.
                 0.39471                 0.31465
      partyidIndependent partyidInd. Near Repub.
                 0.26782                 0.22589
           partyidRepub.    partyidStrong Repub.
                 0.40933                 0.38675
               sexFemale               owngunYES
                 0.02243                 0.03130
The Corresponding VIF, 1/(1-R_j^2)
             partyidDem.   partyidInd. Near Dem.
                   1.652                   1.459
      partyidIndependent partyidInd. Near Repub.
                   1.366                   1.292
           partyidRepub.    partyidStrong Repub.
                   1.693                   1.631
               sexFemale               owngunYES
                   1.023                   1.032
Bivariate Correlations for design matrix
```

Multicollinearity Diagnostics IV

```
                              partyidDem.
partyidDem.                        1.00
partyidInd. Near Dem.             −0.17
partyidIndependent                −0.15
partyidInd. Near Repub.           −0.13
partyidRepub.                     −0.23
partyidStrong Repub.              −0.21
sexFemale                          0.07
owngunYES                         −0.06
                              partyidInd. Near Dem.
partyidDem.                                  −0.17
partyidInd. Near Dem.                         1.00
partyidIndependent                           −0.11
partyidInd. Near Repub.                      −0.10
partyidRepub.                                −0.18
partyidStrong Repub.                         −0.16
sexFemale                                    −0.02
owngunYES                                    −0.04
                              partyidIndependent
partyidDem.                               −0.15
partyidInd. Near Dem.                     −0.11
partyidIndependent                         1.00
partyidInd. Near Repub.                   −0.08
partyidRepub.                             −0.15
partyidStrong Repub.                      −0.14
```

Multicollinearity Diagnostics V

```
sexFemale                                 -0.03
owngunYES                                  0.04
                          partyidInd. Near Repub.
partyidDem.                                    -0.13
partyidInd. Near Dem.                          -0.10
partyidIndependent                             -0.08
partyidInd. Near Repub.                         1.00
partyidRepub.                                  -0.13
partyidStrong Repub.                           -0.12
sexFemale                                      -0.04
owngunYES                                       0.00
                          partyidRepub.
partyidDem.                        -0.23
partyidInd. Near Dem.              -0.18
partyidIndependent                 -0.15
partyidInd. Near Repub.            -0.13
partyidRepub.                       1.00
partyidStrong Repub.               -0.22
sexFemale                          -0.04
owngunYES                           0.04
                          partyidStrong Repub.
partyidDem.                                -0.21
partyidInd. Near Dem.                      -0.16
partyidIndependent                         -0.14
partyidInd. Near Repub.                    -0.12
```

Multicollinearity Diagnostics VI

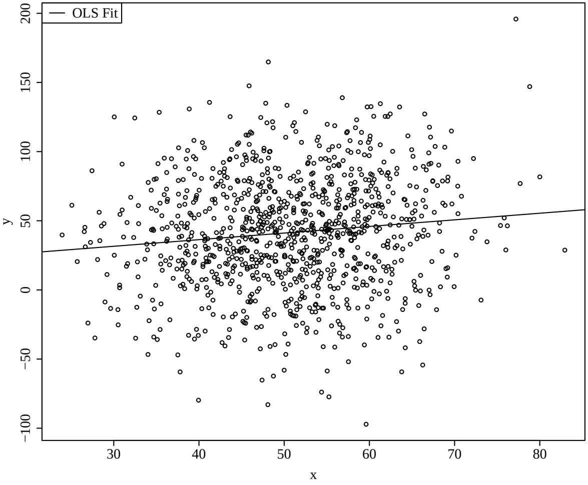
```
partyidRepub.                             −0.22
partyidStrong Repub.                       1.00
sexFemale                                 −0.03
owngunYES                                  0.11
                        sexFemale  owngunYES
partyidDem.                  0.07      −0.06
partyidInd. Near Dem.       −0.02      −0.04
partyidIndependent          −0.03       0.04
partyidInd. Near Repub.     −0.04       0.00
partyidRepub.               −0.04       0.04
partyidStrong Repub.        −0.03       0.11
sexFemale                    1.00      −0.11
owngunYES                   −0.11       1.00
```

plot.lm (plot.glm) produces Diagnostics

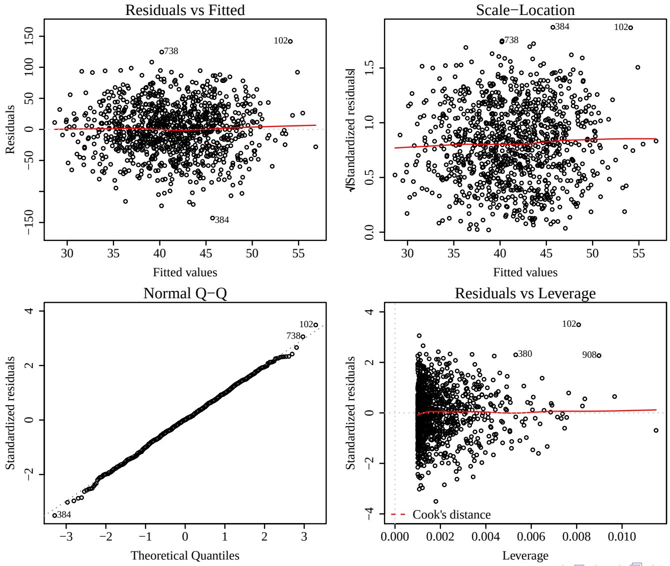
Run plot() on the model object for a quick diagnostic analysis.
Example:

```
myolsmod <- lm(y ~ x, data=datols)
plot(myolsmod)
```

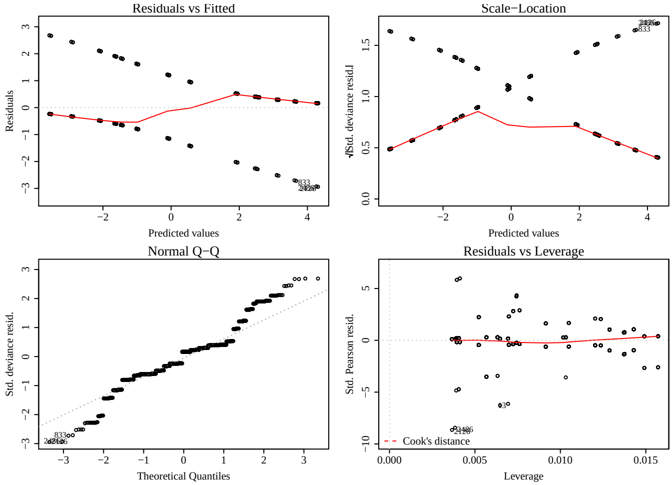
Here's a Scatterplot with OLS Fit



Output from plot(myolsmod)



Output from plot.glm Difficult To Read



influence() Function Digs up the Diagnostics I

```
ib1 <- influence(bush1)
head(ib1$hat)
```

```
        1         4         5         9        10
0.003941 0.003941 0.004117 0.003941 0.005226
       11
0.005226
```

```
head(ib1$coefficients)
```

```
   (Intercept) partyidDem. partyidInd. Near Dem.
1    -0.0052361    0.005286             0.0052149
4    -0.0052361    0.005286             0.0052149
5    -0.0059698    0.005023             0.0051036
9    -0.0052361    0.005286             0.0052149
10   -0.0005007    0.019143             0.0007462
11    0.0001594   -0.006095            -0.0002376
   partyidIndependent partyidInd. Near Repub.
1           0.0052232              0.0053054
4           0.0052232              0.0053054
5           0.0051290              0.0052763
9           0.0052232              0.0053054
10          0.0006130             -0.0007269
```

influence() Function Digs up the Diagnostics II

```
11          -0.0001952                    0.0002315
   partyidRepub. partyidStrong Repub.  sexFemale
1       0.0053094              5.274e-03 -0.0004822
4       0.0053094              5.274e-03 -0.0004822
5       0.0052130              5.165e-03  0.0009737
9       0.0053094              5.274e-03 -0.0004822
10     -0.0008014             -2.216e-04  0.0080812
11      0.0002552              7.056e-05 -0.0025732
   owngunYES
1    0.000635
4    0.000635
5    0.000730
9    0.000635
10  -0.010400
11   0.003312
```

```
head(ib1$sigma)
```

```
     1      4      5      9     10     11
0.7871 0.7871 0.7871 0.7871 0.7853 0.7870
```

```
head(ib1$dev.res)
```

influence() Function Digs up the Diagnostics III

```
      1       4       5       9      10      11
-0.2413 -0.2413 -0.2355 -0.2413  1.8942 -0.6031
```

```
head(ib1$pear.res)
```

```
      1       4       5       9      10      11
-0.1718 -0.1718 -0.1677 -0.1718  2.2390 -0.4466
```

`influence.measures()` A bigger collection of influence measures I

From influence.measures, DFBETAS for each parameter, DFFITS, covariance ratios, Cook's distances and the diagonal elements of the hat matrix.

```
imb1 <- influence.measures(bush1)
attributes(imb1)
```

```
$names
[1] "infmat" "is.inf" "call"

$class
[1] "infl"
```

```
colnames(imb1$infmat)
```

```
 [1] "dfb.1_"   "dfb.prD." "dfb.pIND" "dfb.prtI"
 [5] "dfb.pINR" "dfb.prR." "dfb.pSR." "dfb.sxFm"
 [9] "dfb.oYES" "dffit"    "cov.r"    "cook.d"
[13] "hat"
```

influence.measures() A bigger collection of influence measures II

```
head(imb1$infmat)
```

```
     dfb.1_  dfb.prD.    dfb.pIND    dfb.prtI
1  -0.016910  0.01691  0.0152357  0.0161655
4  -0.016910  0.01691  0.0152357  0.0161655
5  -0.019279  0.01607  0.0149105  0.0158739
9  -0.016910  0.01691  0.0152357  0.0161655
10 -0.001621  0.06137  0.0021851  0.0019015
11  0.000515 -0.01950 -0.0006943 -0.0006042
     dfb.pINR   dfb.prR.    dfb.pSR.   dfb.sxFm
1   0.0132875  0.0149821  0.0107838 -0.003177
4   0.0132875  0.0149821  0.0107838 -0.003177
5   0.0132145  0.0147101  0.0105602  0.006417
9   0.0132875  0.0149821  0.0107838 -0.003177
10 -0.0018248 -0.0022668 -0.0004541  0.053377
11  0.0005798  0.0007202  0.0001443 -0.016960
    dfb.oYES    dffit  cov.r    cook.d      hat
1   0.004164 -0.01932 1.0106 1.303e-05 0.003941
4   0.004164 -0.01932 1.0106 1.303e-05 0.003941
5   0.004787 -0.01928 1.0108 1.297e-05 0.004117
9   0.004164 -0.01932 1.0106 1.303e-05 0.003941
10 -0.068361  0.17528 0.9704 2.941e-03 0.005226
```

`influence.measures()` A bigger collection of influence measures III

```
11   0.021721  -0.05569  1.0083  1.170e-04  0.005226
```

```
summary(imb1)
```

```
Potentially influential observations of
   glm(formula = pres04 ~ partyid + sex + owngun, family = binomial(
        link = logit),        data = dat) :

      dfb.1_  dfb.prD.  dfb.pIND  dfb.prtI  dfb.pINR
10     0.00    0.06      0.00      0.00      0.00
13    -0.03    0.00      0.00      0.00      0.01
54     0.00    0.06      0.00      0.00      0.00
81     0.22   -0.18     -0.17     -0.18     -0.15
118    0.00    0.06      0.00      0.00      0.00
156    0.00    0.06      0.00      0.00      0.00
189    0.06    0.06      0.00     -0.01     -0.01
445    0.00    0.06      0.00      0.00      0.00
589    0.06    0.06      0.00     -0.01     -0.01
605    0.00    0.06      0.00      0.00      0.00
664    0.19   -0.19     -0.17     -0.18     -0.15
704    0.05    0.00      0.11     -0.01     -0.01
```

`influence.measures()` A bigger collection of influence measures IV

833	0.01	0.00	0.00	0.00	0.00
904	0.20	−0.23	−0.21	−0.22	−0.17
986	−0.04	0.00	0.00	0.00	0.01
987	−0.01	0.00	0.12	0.00	0.00
1120	−0.04	0.00	0.00	0.00	0.01
1161	0.06	0.06	0.00	−0.01	−0.01
1215	0.05	0.00	0.11	−0.01	−0.01
1227	0.01	0.00	0.00	0.00	0.00
1292	−0.04	0.00	0.00	0.00	−0.21
1298	−0.01	0.00	0.12	0.00	0.00
1322	−0.01	0.00	0.12	0.00	0.00
1564	−0.05	0.00	0.13	0.01	0.01
1603	0.19	−0.19	−0.17	−0.18	−0.15
1606	0.02	0.00	0.00	0.00	−0.22
1624	0.00	0.06	0.00	0.00	0.00
1737	0.02	0.00	0.00	0.00	−0.22
1758	−0.05	0.00	0.13	0.01	0.01
1784	0.01	0.00	0.00	0.00	0.00
1797	0.00	0.06	0.00	0.00	0.00
1805	0.01	0.00	0.00	0.00	0.00
1812	0.01	0.00	0.00	0.00	0.00
1846	0.00	0.06	0.00	0.00	0.00

influence.measures() A bigger collection of influence measures V

1943	−0.04	0.00	0.00	0.00	−0.21
2002	−0.05	0.00	0.13	0.01	0.01
2029	0.02	0.00	0.00	0.00	−0.22
2097	−0.04	0.00	0.00	0.00	−0.21
2119	0.00	0.06	0.00	0.00	0.00
2126	0.03	0.00	0.00	0.00	−0.01
2143	0.06	0.06	0.00	−0.01	−0.01
2146	0.00	0.00	0.00	0.00	0.00
2174	0.00	0.06	0.00	0.00	0.00
2259	0.05	0.00	0.11	−0.01	−0.01
2315	−0.01	0.00	0.12	0.00	0.00
2327	0.00	0.06	0.00	0.00	0.00
2405	0.02	0.00	0.00	0.00	−0.22
2486	0.00	0.00	0.00	0.00	0.00
2487	0.00	0.00	0.00	0.00	0.00
2508	−0.04	0.00	0.00	0.00	−0.21
2616	−0.01	0.00	0.12	0.00	0.00
2651	−0.05	0.00	0.13	0.01	0.01
2817	0.05	0.00	0.11	−0.01	−0.01
2823	−0.05	0.00	0.13	0.01	0.01
2832	0.00	0.06	0.00	0.00	0.00
2855	0.00	0.06	0.00	0.00	0.00

influence.measures() A bigger collection of influence measures VI

3057	0.20	−0.23	−0.21	−0.22	−0.17
3078	0.00	0.06	0.00	0.00	0.00
3180	0.06	0.06	0.00	−0.01	−0.01
3212	0.01	0.00	0.00	0.00	0.00
3282	0.01	0.00	0.12	0.00	0.00
3334	0.01	0.00	0.00	0.00	0.00
3415	0.01	0.00	0.00	0.00	0.00
3454	0.01	0.00	0.00	0.00	0.00
3510	0.06	0.06	0.00	−0.01	−0.01
3548	0.00	0.00	0.00	0.00	−0.19
3564	0.04	0.00	0.00	0.00	−0.01
3718	0.01	0.00	0.12	0.00	0.00
3769	−0.05	0.00	0.13	0.01	0.01
3823	−0.01	0.00	0.12	0.00	0.00
3890	−0.01	0.00	0.12	0.00	0.00
4113	0.24	−0.22	−0.21	−0.22	−0.18
4199	0.01	0.00	0.12	0.00	0.00
4225	0.24	−0.22	−0.21	−0.22	−0.18
4239	0.00	0.06	0.00	0.00	0.00
4274	0.00	0.06	0.00	0.00	0.00
4334	0.06	0.06	0.00	−0.01	−0.01
4364	0.00	0.00	0.00	0.00	0.00

`influence.measures()` A bigger collection of influence measures VII

```
4436  0.22  -0.18    -0.17    -0.18    -0.15
4471  0.01   0.00     0.00     0.00     0.00
     dfb.prR. dfb.pSR. dfb.sxFm dfb.oYES dffit
10    0.00     0.00     0.05    -0.07     0.18
13    0.01    -0.22     0.06     0.04    -0.29_*
54    0.00     0.00     0.05    -0.07     0.18
81   -0.17    -0.12    -0.07    -0.05     0.22
118   0.00     0.00     0.05    -0.07     0.18
156   0.00     0.00     0.05    -0.07     0.18
189  -0.01    -0.01    -0.12    -0.08     0.21
445   0.00     0.00     0.05    -0.07     0.18
589  -0.01    -0.01    -0.12    -0.08     0.21
605   0.00     0.00     0.05    -0.07     0.18
664  -0.17    -0.12     0.04    -0.05     0.21
704  -0.01     0.00    -0.10    -0.08     0.24
833   0.00    -0.22    -0.04     0.03    -0.28_*
904  -0.19    -0.14     0.05     0.08     0.27_*
986  -0.12     0.00     0.09     0.05    -0.23
987   0.00     0.00     0.07    -0.07     0.23
1120 -0.12     0.00     0.09     0.05    -0.23
1161 -0.01    -0.01    -0.12    -0.08     0.21
1215 -0.01     0.00    -0.10    -0.08     0.24
```

influence.measures() A bigger collection of influence measures VIII

```
1227  -0.12   0.00  -0.06   0.04  -0.22
1292   0.01   0.00   0.09   0.05  -0.33_*
1298   0.00   0.00   0.07  -0.07   0.23
1322   0.00   0.00   0.07  -0.07   0.23
1564   0.01   0.01   0.09   0.10   0.26_*
1603  -0.17  -0.12   0.04  -0.05   0.21
1606   0.00   0.00  -0.08   0.04  -0.32_*
1624   0.00   0.00   0.05  -0.07   0.18
1737   0.00   0.00  -0.08   0.04  -0.32_*
1758   0.01   0.01   0.09   0.10   0.26_*
1784  -0.12   0.00  -0.06   0.04  -0.22
1797   0.00   0.00   0.05  -0.07   0.18
1805  -0.12   0.00  -0.06   0.04  -0.22
1812  -0.12   0.00  -0.06   0.04  -0.22
1846   0.00   0.00   0.05  -0.07   0.18
1943   0.01   0.00   0.09   0.05  -0.33_*
2002   0.01   0.01   0.09   0.10   0.26_*
2029   0.00   0.00  -0.08   0.04  -0.32_*
2097   0.01   0.00   0.09   0.05  -0.33_*
2119   0.00   0.00   0.05  -0.07   0.18
2126  -0.01  -0.18  -0.04  -0.06  -0.23
2143  -0.01  -0.01  -0.12  -0.08   0.21
```

`influence.measures()` A bigger collection of influence measures IX

2146	−0.11	0.00	0.06	−0.08	−0.20
2174	0.00	0.00	0.05	−0.07	0.18
2259	−0.01	0.00	−0.10	−0.08	0.24
2315	0.00	0.00	0.07	−0.07	0.23
2327	0.00	0.00	0.05	−0.07	0.18
2405	0.00	0.00	−0.08	0.04	−0.32_*
2486	0.00	−0.18	0.04	−0.05	−0.23
2487	−0.11	0.00	0.06	−0.08	−0.20
2508	0.01	0.00	0.09	0.05	−0.33_*
2616	0.00	0.00	0.07	−0.07	0.23
2651	0.01	0.01	0.09	0.10	0.26_*
2817	−0.01	0.00	−0.10	−0.08	0.24
2823	0.01	0.01	0.09	0.10	0.26_*
2832	0.00	0.00	0.05	−0.07	0.18
2855	0.00	0.00	0.05	−0.07	0.18
3057	−0.19	−0.14	0.05	0.08	0.27_*
3078	0.00	0.00	0.05	−0.07	0.18
3180	−0.01	−0.01	−0.12	−0.08	0.21
3212	−0.12	0.00	−0.06	0.04	−0.22
3282	0.00	0.00	−0.09	0.09	0.26_*
3334	−0.12	0.00	−0.06	0.04	−0.22
3415	−0.12	0.00	−0.06	0.04	−0.22

`influence.measures()` A bigger collection of influence measures X

```
3454 -0.12     0.00     -0.06     0.04     -0.22
3510 -0.01    -0.01     -0.12    -0.08      0.21
3548  0.00     0.00      0.07    -0.10     -0.30_*
3564 -0.11     0.00     -0.06    -0.09     -0.20
3718  0.00     0.00     -0.09     0.09      0.26_*
3769  0.01     0.01      0.09     0.10      0.26_*
3823  0.00     0.00      0.07    -0.07      0.23
3890  0.00     0.00      0.07    -0.07      0.23
4113 -0.20    -0.14     -0.08     0.07      0.27_*
4199  0.00     0.00     -0.09     0.09      0.26_*
4225 -0.20    -0.14     -0.08     0.07      0.27_*
4239  0.00     0.00      0.05    -0.07      0.18
4274  0.00     0.00      0.05    -0.07      0.18
4334 -0.01    -0.01     -0.12    -0.08      0.21
4364 -0.11     0.00      0.06    -0.08     -0.20
4436 -0.17    -0.12     -0.07    -0.05      0.22
4471 -0.12     0.00     -0.06     0.04     -0.22
      cov.r   cook.d hat
10     0.97_*  0.00   0.01
13     0.93_*  0.03   0.01
54     0.97_*  0.00   0.01
81     0.93_*  0.02   0.00
```

influence.measures() A bigger collection of influence measures XI

```
118    0.97_*  0.00   0.01
156    0.97_*  0.00   0.01
189    0.97_*  0.00   0.01
445    0.97_*  0.00   0.01
589    0.97_*  0.00   0.01
605    0.97_*  0.00   0.01
664    0.93_*  0.01   0.00
704    0.96_*  0.01   0.01
833    0.93_*  0.03   0.01
904    0.95_*  0.01   0.01
986    0.95_*  0.01   0.01
987    0.96_*  0.01   0.01
1120   0.95_*  0.01   0.01
1161   0.97_*  0.00   0.01
1215   0.96_*  0.01   0.01
1227   0.95_*  0.01   0.01
1292   0.97_*  0.01   0.02
1298   0.96_*  0.01   0.01
1322   0.96_*  0.01   0.01
1564   0.98    0.01   0.01
1603   0.93_*  0.01   0.00
1606   0.97_*  0.01   0.01
```

influence.measures() A bigger collection of influence measures XII

```
1624   0.97_*  0.00   0.01
1737   0.97_*  0.01   0.01
1758   0.98    0.01   0.01
1784   0.95_*  0.01   0.01
1797   0.97_*  0.00   0.01
1805   0.95_*  0.01   0.01
1812   0.95_*  0.01   0.01
1846   0.97_*  0.00   0.01
1943   0.97_*  0.01   0.02
2002   0.98    0.01   0.01
2029   0.97_*  0.01   0.01
2097   0.97_*  0.01   0.02
2119   0.97_*  0.00   0.01
2126   0.91_*  0.03   0.00
2143   0.97_*  0.00   0.01
2146   0.94_*  0.01   0.00
2174   0.97_*  0.00   0.01
2259   0.96_*  0.01   0.01
2315   0.96_*  0.01   0.01
2327   0.97_*  0.00   0.01
2405   0.97_*  0.01   0.01
2486   0.91_*  0.03   0.00
```

influence.measures() A bigger collection of influence measures XIII

```
2487   0.94_*   0.01    0.00
2508   0.97_*   0.01    0.02
2616   0.96_*   0.01    0.01
2651   0.98     0.01    0.01
2817   0.96_*   0.01    0.01
2823   0.98     0.01    0.01
2832   0.97_*   0.00    0.01
2855   0.97_*   0.00    0.01
3057   0.95_*   0.01    0.01
3078   0.97_*   0.00    0.01
3180   0.97_*   0.00    0.01
3212   0.95_*   0.01    0.01
3282   0.98     0.01    0.01
3334   0.95_*   0.01    0.01
3415   0.95_*   0.01    0.01
3454   0.95_*   0.01    0.01
3510   0.97_*   0.00    0.01
3548   0.96_*   0.01    0.01
3564   0.94_*   0.01    0.00
3718   0.98     0.01    0.01
3769   0.98     0.01    0.01
3823   0.96_*   0.01    0.01
```

influence.measures() A bigger collection of influence measures XIV

```
3890  0.96_*  0.01   0.01
4113  0.95_*  0.02   0.01
4199  0.98    0.01   0.01
4225  0.95_*  0.02   0.01
4239  0.97_*  0.00   0.01
4274  0.97_*  0.00   0.01
4334  0.97_*  0.00   0.01
4364  0.94_*  0.01   0.00
4436  0.93_*  0.02   0.00
4471  0.95_*  0.01   0.01
```

Can get component columns directly with 'dfbetas', 'dffits', 'covratio' and 'cooks.distance'.

But if You Want dfbeta, Not dfbetas, Why Not Ask? I

```
dfb1 <- dfbeta(bush1)
colnames(dfb1)
```

```
[1] "(Intercept)"
[2] "partyidDem."
[3] "partyidInd. Near Dem."
[4] "partyidIndependent"
[5] "partyidInd. Near Repub."
[6] "partyidRepub."
[7] "partyidStrong Repub."
[8] "sexFemale"
[9] "owngunYES"
```

```
head(dfb1)
```

But if You Want dfbeta, Not dfbetas, Why Not Ask? II

```
   (Intercept) partyidDem. partyidInd. Near Dem.
1   -0.0052361    0.005286            0.0052149
4   -0.0052361    0.005286            0.0052149
5   -0.0059698    0.005023            0.0051036
9   -0.0052361    0.005286            0.0052149
10  -0.0005007    0.019143            0.0007462
11   0.0001594   -0.006095           -0.0002376
   partyidIndependent partyidInd. Near Repub.
1           0.0052232             0.0053054
4           0.0052232             0.0053054
5           0.0051290             0.0052763
9           0.0052232             0.0053054
10          0.0006130            -0.0007269
11         -0.0001952             0.0002315
   partyidRepub. partyidStrong Repub.  sexFemale
1      0.0053094            5.274e-03 -0.0004822
4      0.0053094            5.274e-03 -0.0004822
5      0.0052130            5.165e-03  0.0009737
9      0.0053094            5.274e-03 -0.0004822
10    -0.0008014           -2.216e-04  0.0080812
11     0.0002552            7.056e-05 -0.0025732
   owngunYES
1   0.000635
4   0.000635
```

But if You Want dfbeta, Not dfbetas, Why Not Ask? III

```
5    0.000730
9    0.000635
10  −0.010400
11   0.003312
```

I wondered what dfbetas does. You can see for yourself. Look at the code. Run:

```
> stats:::dfbetas.lm
```

`predict()` with newdata

- If you run this:
  `predict(bush5)`
  R calculates $X\hat{\beta}$, a "linear predictor" value for each row in your dataframe
- See "`?predict.glm`."
- We ask for predicted probabilities like so
  `predict(bush5, type="response")`
  and you still get one prediction for each line in the data.

Use predict to calculate with "for example" values

- Create "example" dataframes and get probabilities for hypothetical cases.

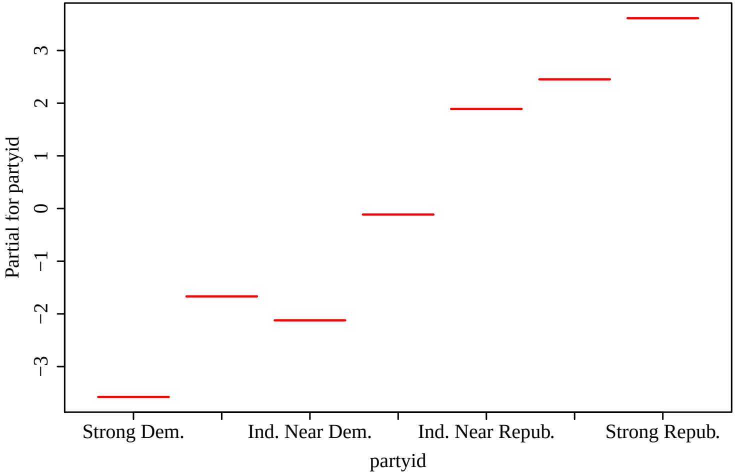
```
mydf <- # Pretend there are some commands, for example
```

- Run that new example data frame through the predict function

```
predict(bush5, newdata=mydf, type="response")
```

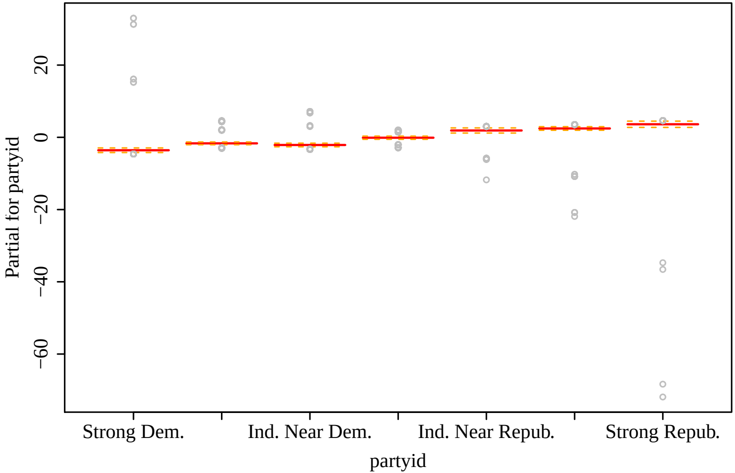
Termplot: Plotting The Linear Predictor

```
termplot(bush1,terms=c("partyid"))
```



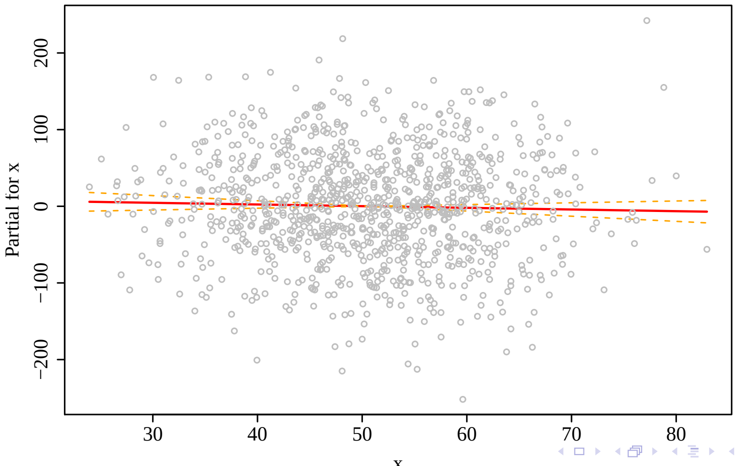
Termplot: Some of the Magic is Lost on a Logistic Model

```
termplot(bush1, terms=c("partyid"), partial.resid = T, se = T)
```



Termplot: But If You Had Some Continuous Data, Watch Out!

```
termplot(myolsmod, terms=c("x"), partial.resid = T, se = T)
```



`termplot()` works because . . .

- termplot doesn't make calculations, it uses the "`predict`" method associated with a model object.
- `predict` is a generic method, it doesn't do any work either!
- Actual work gets done by methods for models, `predict.lm` or `predict.glm`.
- You can leave out the "terms" option, termplot will cycle through all of the predictors in the model.

Why Termplot is Not the End of the Story

- Termplot draws $X\hat{\beta}$, the linear predictor.
- Maybe we want predicted probabilities instead.
- Maybe we want predictions for certain case types: `termplot` allows the predict implementation to decide which values of the inputs will be used.
- A regression expert will quickly conclude that a really great graph may require direct use of the `predict` method for the model object.