After Fitting Regressions

Paul E. Johnson[1] [2]

[1]Department of Political Science

[2]Center for Research Methods and Data Analysis, University of Kansas

2012

After Fitting Regressions

Paul E. Johnson[1] [2]

[1]Department of Political Science

[2]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

```
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"
```

First, Find Out What You Got ...

```
[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
```

- That "crude" approach is discouraged. We should instead use "extractor methods"

Just Making Sure About the Object's Class

- Ask the object what class it is from

```
class(bush1)
```

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

Ask What Methods Apply to a "glm" Object

```
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

```
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*
```

Check methods for “lm” class ...

```
[29] residuals.lm        rstandard.lm
[31] rstudent.lm         simulate.lm*
[33] summary.lm          variable.names.lm*
[35] vcov.lm*

   Non-visible functions are asterisked
```

Do You Wonder How "They" Do "That"?

- At some point, you realize that the help page is not detailed enough. You may need to see the Actual Code
- Darth said "Use the Source, Luke!"
  If you want to know "what a function does", the best option is to download the ACTUAL SOURCE CODE and read it!

Can See Some Code Within an R Session

- In the "old days", you could easily see a function's "code" by typing its name (i.e., omit the parentheses).
  Ex: q used to show all of the steps in shutting down.
- Today, in R 2.11, when I type q I see:

```
> q
function (save = "default", status = 0, runLast
    = TRUE)
.Internal(quit(save, status, runLast))
<environment: namespace:base>
```

Some Functions Still Show Their Code

- Some very informative examples. Try:
  - `> lm #(or stats::lm)`
  - `> glm #(or stats::glm)`
  - `> termplot`
- Generic method output not so useful. Try:
  - `> predict`
  - `> plot`

Looking Into the Class Hierarchy

- In many cases, you can only find what you need if you give the "function" name and the name of the "class" separated by a period.
- Try:
  - `> predict.lm`
  - `> predict.glm`
- Many methods are inside "namespaces" and you can't see their code without some extra effort.
  - namespace::method will often be useful
  - Three colons needed for "hidden methods" stats:::weights.glm
- Many times I have doublechecked this detailed posting by Prof. Brian Ripley on this question: `http://tolstoy.newcastle.edu.au/R/help/05/09/12506.html`

The First Method Used is usually summary()

```
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
```

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

```
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 ***
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 ***
---
```

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

```
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

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"
```

Summary Object ...

My deviance is

```
sb1$deviance
```

```
[1] 764
```

The coef Enigma

- `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.
```

The coef Enigma ...

```
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 ...

```
                          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
```

The coef Enigma ...

```
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()

- 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  < 2
    e-16 ***
```

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"

```
> 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 + . . ., data=dat, family=binomial)
  2. Fit the "smaller Model" with the data extracted from the fit of the previous model (mod1$model) as the data frame
     mod2 <- glm(y x3 + . . ., data=mod1$model, family=binomial)
  3. After that, anova() will work
- Hasten to add: more elaborate treatment of missingness is often called for.

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=bush3$model, 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 β_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 :)

drop1 Relieves Tedium

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

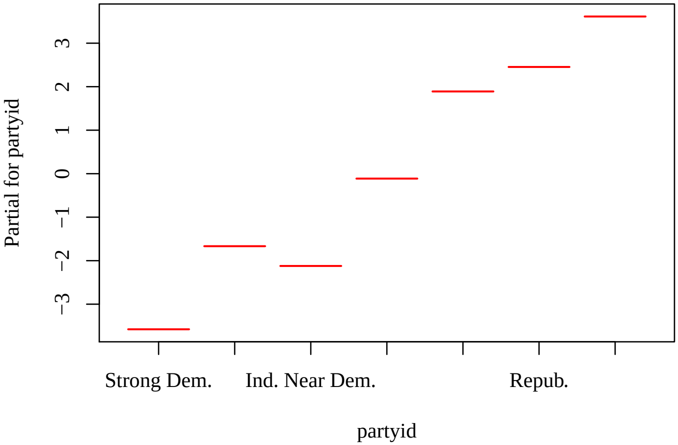
```
drop1(bush3, test="Chisq")
```

```
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 ***
```

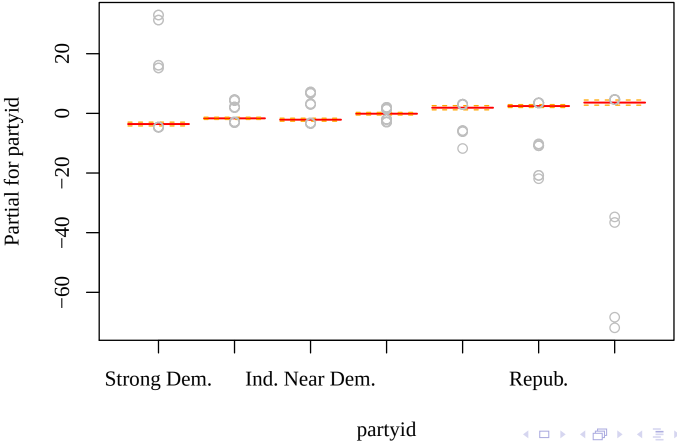
Termplot: Plotting The Linear Predictor

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



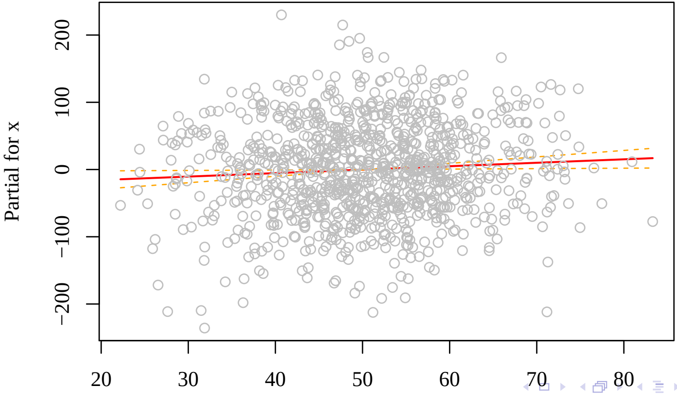
Termplot: Some of the Magic is Lost on a Logistic Model

```
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.

`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
#to create an example data frame
```

- Run that new example data frame through the predict function

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

Create the New Data Frame

```
nd <- bush5$model
colnames(nd)
```

```
[1] "pres04"   "partyid" "sex"      "owngun"
[5] "race"     "wrkslf"  "realinc" "newpolv"
```

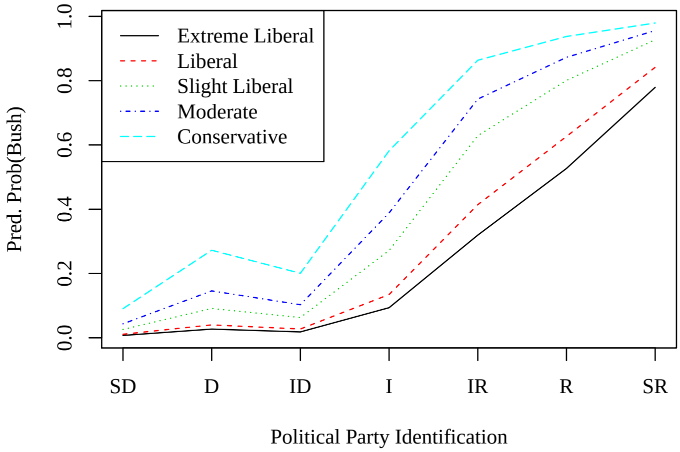
```
mynewdf <- expand.grid(levels(nd$partyid), levels(
    nd$newpolv))
colnames(mynewdf) <- c("partyid","newpolv")
mynewdf$sex <- levels(nd$sex)[1]
mynewdf$owngun <- levels(nd$owngun)[1]
mynewdf$race <- levels(nd$race)[1]
mynewdf$wrkslf <- levels(nd$wrkslf)[1]
mynewdf$realinc <- mean(nd$realinc)
mynewdf$newpred <- predict(bush5, newdata=mynewdf,
     type="response")
levels(mynewdf$newpolv) <- c("Ex.L","L","SL","M","
    C")
```

Make Table of Predicted Probabilities

```
library(gdata)
newtab <- aggregate.table(mynewdf$newpred, by1=
    mynewdf$partyid, by2=mynewdf$newpolv, FUN=I)
```

	Ex.L	L	SL	M	C
Strong Dem.	0.0073	0.0110	0.0260	0.0435	0.0906
Dem.	0.0270	0.0402	0.0912	0.1460	0.2724
Ind. Near Dem.	0.0183	0.0273	0.0631	0.1029	0.2008
Independent	0.0936	0.1346	0.2716	0.3884	0.5818
Ind. Near Repub.	0.3194	0.4141	0.6289	0.7427	0.8634
Repub.	0.5268	0.6264	0.8008	0.8726	0.9375
Strong Repub.	0.7791	0.8416	0.9272	0.9559	0.9794

Or Perhaps You Would Like A Figure?



How Could You Make That Figure?

```
prebynewpol <- unstack(mynewdf, newpred~newpolv)
matplot(prebynewpol,type="l",xaxt="n",xlab="
    Political Party Identification", ylab="Pred.
    Prob(Bush)")
axis(1, at=1:7, labels=c("SD","D","ID","I","IR","R
    ","SR"))
legend("topleft", legend=c("Extreme Liberal","
    Liberal","Slight Liberal","Moderate","
    Conservative"),col=1:5,lty=1:5)
```

Covariance of $\hat{\beta}$

```
vcov(bush1)
```

```
                          (Intercept)  partyidDem.
(Intercept)                   0.15475   -0.1302192
partyidDem.                  -0.13022    0.1577463
partyidInd. Near Dem.        -0.13230    0.1300411
partyidIndependent           -0.13296    0.1300573
partyidInd. Near Repub.      -0.13678    0.1302007
partyidRepub.                -0.13514    0.1301957
partyidStrong Repub.         -0.13388    0.1301365
sexFemale                    -0.02524   -0.0005279
owngunYES                    -0.01892    0.0010382
                          partyidInd. Near Dem.
(Intercept)                          -0.1323024
partyidDem.                           0.1300411
partyidInd. Near Dem.                 0.1890942
```

Covariance of $\hat{\beta}$ ...

```
partyidIndependent              0.1304249
partyidInd. Near Repub.         0.1305706
partyidRepub.                   0.1304179
partyidStrong Repub.            0.1303894
sexFemale                       0.0033138
owngunYES                       0.0002006
                        partyidIndependent
(Intercept)                      -0.132959
partyidDem.                       0.130057
partyidInd. Near Dem.             0.130425
partyidIndependent                0.168499
partyidInd. Near Repub.           0.130774
partyidRepub.                     0.130579
partyidStrong Repub.              0.130499
sexFemale                         0.003767
owngunYES                         0.001017
                        partyidInd. Near Repub.
```

Covariance of $\hat{\beta}$ ...

```
(Intercept)                                    -0.136777
partyidDem.                                     0.130201
partyidInd. Near Dem.                           0.130571
partyidIndependent                              0.130774
partyidInd. Near Repub.                         0.257308
partyidRepub.                                   0.131613
partyidStrong Repub.                            0.131170
sexFemale                                       0.005551
owngunYES                                       0.006971
                          partyidRepub.
(Intercept)                   -0.135138
partyidDem.                    0.130196
partyidInd. Near Dem.          0.130418
partyidIndependent             0.130579
partyidInd. Near Repub.        0.131613
partyidRepub.                  0.202702
partyidStrong Repub.           0.130920
```

Covariance of $\hat{\beta}$ ...

```
sexFemale                 0.003812
owngunYES                 0.005802
                        partyidStrong Repub.
(Intercept)                      -0.133884
partyidDem.                       0.130136
partyidInd. Near Dem.             0.130389
partyidIndependent                0.130499
partyidInd. Near Repub.           0.131170
partyidRepub.                     0.130920
partyidStrong Repub.              0.386045
sexFemale                         0.003435
owngunYES                         0.003547
                         sexFemale   owngunYES
(Intercept)             -0.0252418  -0.0189238
partyidDem.             -0.0005279   0.0010382
partyidInd. Near Dem.    0.0033138   0.0002006
partyidIndependent       0.0037667   0.0010175
```

Covariance of $\hat{\beta}$ ...

```
partyidInd. Near Repub.   0.0055510  0.0069708
partyidRepub.             0.0038122  0.0058016
partyidStrong Repub.      0.0034348  0.0035474
sexFemale                 0.0371676  0.0032171
owngunYES                 0.0032171  0.0375305
```

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
```

Covariance of $\hat{\beta}$ ...

```
                0.6213                          0.1928
             owngunYES
                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:

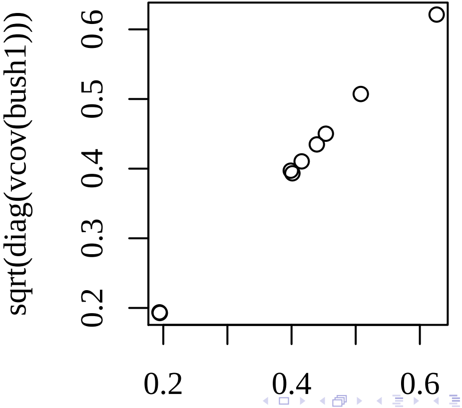
```
t(sqrt(diag(myvcovHC)))
```

```
     (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 ownggunYES
[1,]               0.6262    0.1946     0.1941
```

Compare those:

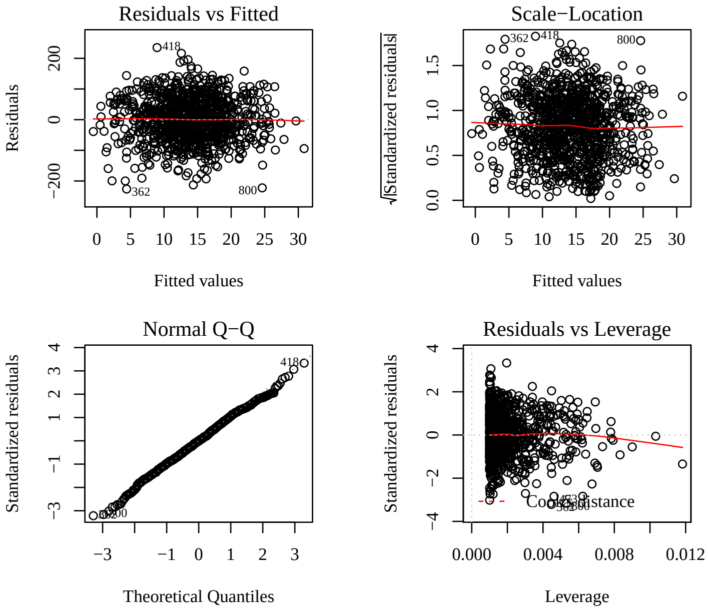
```
plot(sqrt(diag(myvcovHC)),sqrt(diag(
    vcov(bush1)))    )
```

The HC and ordinary standard errors are almost identical:

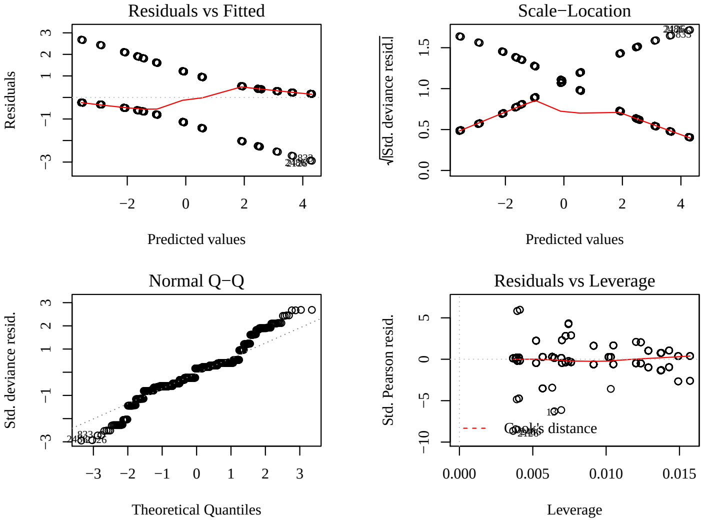


Tons of Diagnostic Information

Run plot() on the model object for a quick view.
Example: plot(myolsmod)


Residuals vs Fitted
Residuals
Fitted values
418
362
800
Scale−Location
√|Standardized residuals|
Fitted values
362
418
800
Normal Q−Q
Standardized residuals
Theoretical Quantiles
418
Residuals vs Leverage
Standardized residuals
Leverage
Cook's distance

Tough to read the glm plot, IMHO...



influence() Function Digs up the Diagnostics

```
ib1 <- influence(bush1)
 colnames(ib1)
```

```
NULL
```

```
 str(ib1)
```

```
List of 5
 $ hat            : Named num [1:1243] 0.00394 0.00394
      0.00412 0.00394 0.00523 ...
  ..- attr, "names")= chr [1:1243] "1" "4" "5" "9" ...coefficients :
```

num[1 : 1243, 1 : 9] − 0.005236 − 0.005236 − 0.00597 − 0.005236 − 0.000501..... − *attr*(∗, " *dimnames*") = *Listof* 2.... : chr [1:1243] "1" "4" "5" "9" ....: *chr*[1 : 9]" (*Intercept*)"" *partyidDem*."" *partyidInd*.*NearDem*."" *partyidIndependen* sigma : Named num [1:1243] 0.787 0.787 0.787 0.787 0.785 .....- attr(*, "names")= chr [1:1243] "1" "4" "5" "9"

influence() Function Digs up the Diagnostics ...

$...dev.res : Namednum[1:1243]-0.241-0.241-0.236-0.2411.894.....-attr(*,"names")=chr[1:1243]"1""4""5""9"...$
pear.res : Named num [1:1243] -0.172 -0.172 -0.168 -0.172 2.239 .....- attr(*, "names")= chr [1:1243] "1" "4" "5" "9" ...

```
summary(ib1)
```

```
             Length  Class    Mode
hat           1243  -none-  numeric
coefficients 11187  -none-  numeric
sigma         1243  -none-  numeric
dev.res       1243  -none-  numeric
pear.res      1243  -none-  numeric
```

influence.measures() A bigger collection of influence measures

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)
```

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

```
 [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"
```

```
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
```

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

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

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

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

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

```
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? ...

```
   (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
```

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

```
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
```

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

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

You Will Want to Use LaTeX After You See This

- How do you get regression tables out of your project?
- Do you go through error-prone copying, pasting, typing, tabling, etc?
- What if your software could produce a finished publishable table?

- Years ago, I wrote a function “outreg”
- This command:

```
outreg(bush1, tight=F, modelLabels=c("Bush
    Logistic"))
```

- Produces the output on the next slide

	Bush Logistic	
	Estimate	(S.E.)
(Intercept)	-3.571*	(0.393)
partyidDem.	1.91*	(0.397)
partyidInd. Near Dem.	1.456*	(0.435)
partyidIndependent	3.464*	(0.41)
partyidInd. Near Repub.	5.468*	(0.507)
partyidRepub.	6.031*	(0.45)
partyidStrong Repub.	7.191*	(0.621)
sexFemale	0.049	(0.193)
owngunYES	0.642*	(0.194)
N	1243	
Deviance	763.996	
$-2LLR(Model\chi^2)$	957.944*	

* $p \leq 0.05$

Polish that up

- you can beautify the variable labels, either by specifying them in the outreg command or editing the table output.
- outreg produces Latex that looks like this in the R session output.

```
\begin{center}
 \begin{tabular}{*{3}{l}}
 \hline
      &\multicolumn{2}{c}{Bush Logistic} \\
                    & Estimate & (S.E.)   \\
\hline
\hline
  (Intercept)   &     -3.571*    &    (0.393) \\
  partyidDem.   &     1.91*    &    (0.397) \\
  partyidInd. Near Dem.    &     1.456*    &    (0
     .435) \\
  partyidIndependent    &     3.464*    &    (0.41)
      \\
  partyidInd. Near Repub.    &     5.468*    &
     (0.507) \\
```

Push Several Models Into One Wide Table

```
outreg(list(bush1,bush4,bush5),  modelLabels=c("
    bush1","bush4","bush5"))
```

Sorry, I had to split this manually across 3 slides :(

	bush1 Estimate (S.E.)	bush4 Estimate (S.E.)	bush5 Estimate (S.E.)
(Intercept)	-3.571*	-4.196*	-4.861*
	(0.393)	(0.854)	(0.96)
partyidDem.	1.91*	1.356*	1.324*
	(0.397)	(0.424)	(0.423)
partyidInd. Near Dem.	1.456*	0.937*	0.925*
	(0.435)	(0.461)	(0.464)
partyidIndependent	3.464*	2.613*	2.637*
	(0.41)	(0.442)	(0.444)
partyidInd. Near Repub.	5.468*	4.114*	4.151*
	(0.507)	(0.538)	(0.54)
partyidRepub.	6.031*	4.985*	5.015*
	(0.45)	(0.479)	(0.483)
partyidStrong Repub.	7.191*	5.999*	6.168*
	(0.621)	(0.738)	(0.742)
sexFemale	0.049	.	-0.006
	(0.193)		(0.224)
owngunYES	0.642*	0.417	0.449*
	(0.194)	(0.221)	(0.224)
raceBLACK	.	-2.067*	-2.11*
		(0.45)	(0.45)
raceOTHER	.	-0.483	-0.497
		(0.391)	(0.394)
polviewsLIBERAL	.	0.303	.
		(0.866)	
polviewsSLIGHTLY LIBERAL	.	1.173	.
		(0.819)	

R Packages for Producing Regression Output

- memisc: works well, further from final form than outreg
- xtable: incomplete output, but latex or HTML works
- apsrtable: very similar to outreg
- Hmisc “latex” function

```
library(xtable)
tabout1 <- xtable(bush1)
print(tabout1, type="latex")
```

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-3.5712	0.3934	-9.08	0.0000
partyidDem.	1.9103	0.3972	4.81	0.0000
partyidInd. Near Dem.	1.4559	0.4348	3.35	0.0008
partyidIndependent	3.4642	0.4105	8.44	0.0000
partyidInd. Near Repub.	5.4677	0.5073	10.78	0.0000
partyidRepub.	6.0307	0.4502	13.39	0.0000
partyidStrong Repub.	7.1908	0.6213	11.57	0.0000
sexFemale	0.0488	0.1928	0.25	0.8001
owngunYES	0.6424	0.1937	3.32	0.0009

If you Can't Shake the MS Word "Habit"

The best you can do is HTML output, which you can copy paste-special into a document.

```
print(xtable(summary(bush1)), type="HTML")
```

```
<!-- html table generated in R 2.15.0 by xtable 1
    .7-0 package -->
<!-- Thu Jun  7 00:59:30 2012 -->
<TABLE border=1>
<TR> <TH>  </TH> <TH> Estimate </TH> <TH> Std.
    Error </TH> <TH> z value </TH> <TH> Pr(&gt |z|)
    </TH>  </TR>
  <TR> <TD align="right"> (Intercept) </TD> <TD
      align="right"> -3.5712 </TD> <TD align="right
      "> 0.3934 </TD> <TD align="right"> -9.08 </TD
      > <TD align="right"> 0.0000 </TD> </TR>
  <TR> <TD align="right"> partyidDem. </TD> <TD
      align="right"> 1.9103 </TD> <TD align="right"
      > 0.3972 </TD> <TD align="right"> 4.81 </TD>
```

memisc mtable is nice for comparing models (except for verbosity of parameter labels)

```
library(memisc)
mtable(bush1, bush4, bush5)
```

```
Calls:
bush1: glm(formula = pres04 ~ partyid + sex +
    owngun, family = binomial(link = logit),
     data = dat)
bush4: glm(formula = pres04 ~ partyid + owngun +
    race + polviews, family = binomial(link = logit
    ),
     data = bush3$model)
bush5: glm(formula = pres04 ~ partyid + sex +
    owngun + race + wrkslf +
     realinc + newpolv, family = binomial(link =
         logit), data = dat)
```

memisc mtable is nice for comparing models (except for verbosity of parameter labels) ...

bush

b

b

```
(Intercept)
    −3.571***  −4.196***  −4.861***
```

memisc mtable is nice for comparing models (except for verbosity of parameter labels) ...

(0
.39
)
(0
.85
)
(0
.96
)

memisc mtable is nice for comparing models (except for verbosity of parameter labels) ...

```
partyid: Dem./Strong Dem.
    1.910***   1.356**    1.324**
                                                            (0
                                                               .39
                                                               )
                                                               (0
                                                               .42
                                                               )
                                                               (0
                                                               .42
```

memisc mtable is nice for comparing models (except for verbosity of parameter labels) ...

```
                                                                )
partyid: Ind. Near Dem./Strong Dem.
    1.456***   0.937*     0.925*
                                                              (0
                                                                .43
                                                                )
                                                                (0
                                                                .46
                                                                )
                                                                (0
```

memisc mtable is nice for comparing models (except for verbosity of parameter labels) ...

```
partyid: Independent/Strong Dem.
    3.464***   2.613***   2.637***
```

.46
)

(0

.41
)

(0
.44
)

memisc mtable is nice for comparing models (except for verbosity of parameter labels) ...

```
partyid: Ind. Near Repub./Strong Dem.
    5.468***   4.114***   4.151***
```

(0
.44
)

(0
.50
)

(0
.53
)

memisc mtable is nice for comparing models (except for verbosity of parameter labels) ...

```
partyid: Repub./Strong Dem.
    6.031***   4.985***   5.015***
```

(0
.54
)

memisc mtable is nice for comparing models (except for verbosity of parameter labels) ...

(0
.45
)
(0
.47
)
(0
.48
)

memisc mtable is nice for comparing models (except for verbosity of parameter labels) ...

```
partyid: Strong Repub./Strong Dem.
    7.191***   5.999***   6.168***
                                                  (0
                                                     .62
                                                     )
                                                     (0
                                                     .73
                                                     )
                                                     (0
                                                     .74
```

memisc mtable is nice for comparing models (except for verbosity of parameter labels) ...

```
sex: Female/Male
    0.049          −0.006
                                  (0
                                   .19
                                   )
                                   (0
                                   .22
                                   )
owngun: YES/NO
    0.642***   0.417     0.449*
```

)

memisc mtable is nice for comparing models (except for verbosity of parameter labels) ...

(0
.19
)

(0
.22
)

(0
.22
)

memisc mtable is nice for comparing models (except for verbosity of parameter labels) ...

```
race: BLACK/WHITE

   −2.067***  −2.110***










race: OTHER/WHITE

   −0.483     −0.497
```

memisc mtable is nice for comparing models (except for verbosity of parameter labels) ...

```
polviews: LIBERAL/EXTREMELY LIBERAL
                                   0.303
```

memisc mtable is nice for comparing models (except for verbosity of parameter labels) ...

```
polviews: SLIGHTLY LIBERAL/EXTREMELY LIBERAL
                              1.173




polviews: MODERATE/EXTREMELY LIBERAL
                                        1.761*




polviews: SLGHTLY CONSERVATIVE/EXTREMELY LIBERAL
                         2.443**
```

memisc mtable is nice for comparing models (except for verbosity of parameter labels) ...

```
polviews: CONSERVATIVE/EXTREMELY LIBERAL
                           2.542**



polviews: EXTRMLY CONSERVATIVE/EXTREMELY LIBERAL
                  3.028*
```

memisc mtable is nice for comparing models (except for verbosity of parameter labels) ...

```
wrkslf: SOMEONE ELSE/SELF−EMPLOYED
                                          0.696



realinc

   −0.000
```

memisc mtable is nice for comparing models (except for verbosity of parameter labels) ...

```
newpolv: LIBERAL/EXTREMELY LIBERAL
                                          0.409
newpolv: SLIGHTLY LIBERAL/EXTREMELY LIBERAL
                                  1.284
```

memisc mtable is nice for comparing models (except for verbosity of parameter labels) ...

```
newpolv: MODERATE/EXTREMELY LIBERAL
                                                  1.816*

newpolv: CONSERVATIVE/EXTREMELY LIBERAL
                                             2.600**
```

memisc mtable is nice for comparing models (except for verbosity of parameter labels) ...

```
Aldrich−Nelson R−sq.
                                    0.435     0.453
        0.454
McFadden R−sq.
                                         0.556
       0.597     0.600
Cox−Snell R−sq.
                                        0.537
   0.563     0.564
```

memisc mtable is nice for comparing models (except for verbosity of parameter labels) ...

```
Nagelkerke R-sq.
                                                    0.717      0
    .751      0.753
phi

    1.000      1.000      1.000
Likelihood-ratio
    957.944    879.756    883.424
p

    0.000      0.000      0.000
Log-likelihood
    -381.998   -296.361   -294.527
Deviance
    763.996    592.722    589.054
```

memisc mtable is nice for comparing models (except for verbosity of parameter labels) ...

```
AIC
    781.996    624.722    623.054
BIC
    828.124    704.224    707.525
N
    1243       1063       1063
```

memisc toLatex

```
toLatex(mtable(bush1))
```

(Intercept)	-3.571^{***}
	(0.393)
partyid: Dem./Strong Dem.	1.910^{***}
	(0.397)
partyid: Ind. Near Dem./Strong Dem.	1.456^{***}
	(0.435)
partyid: Independent/Strong Dem.	3.464^{***}
	(0.410)
partyid: Ind. Near Repub./Strong Dem.	5.468^{***}
	(0.507)
partyid: Repub./Strong Dem.	6.031^{***}
	(0.450)
partyid: Strong Repub./Strong Dem.	7.191^{***}
	(0.621)
sex: Female/Male	0.049
	(0.193)
owngun: YES/NO	0.642^{***}
	(0.194)
Aldrich-Nelson R-sq.	0.435
McFadden R-sq.	0.556

Relable Levels to Truncate Output

- We could have to edit that output A LOT
- Hack the Labels down

```
levels(dat$partyid) <- c("SD","D","ID","I","IR
    ","R","SR")
levels(dat$polviews) <- c("EL","L","SL","M","
    SC","C","EC")
levels(dat$newpolv) <- c("EL","L","SL","M","C"
    )
levels(dat$wrkslf) <- c("Yes","No")
```

- Re-run the models

```
bush1 <- glm(pres04 ~ partyid + sex + owngun,
    data=dat, family=binomial(link=logit))
 bush3 <- glm(pres04 ~ partyid + sex + owngun
     + race + wrkslf + realinc + polviews ,
     data=dat, family=binomial(link=logit))
 bush4 <- glm(pres04 ~ partyid +  owngun +
     race + polviews , data=bush3$model, family
```

```
toLatex(mtable(bush1, bush4, bush5))
```

	bush1	bush4	bush5
(Intercept)	-3.571^{***}	-4.196^{***}	-4.861^{***}
	(0.393)	(0.854)	(0.960)
partyid: D/SD	1.910^{***}	1.356^{**}	1.324^{**}
	(0.397)	(0.424)	(0.423)
partyid: ID/SD	1.456^{***}	0.937^{*}	0.925^{*}
	(0.435)	(0.461)	(0.464)
partyid: I/SD	3.464^{***}	2.613^{***}	2.637^{***}
	(0.410)	(0.442)	(0.444)
partyid: I/SDR	5.468^{***}	4.114^{***}	4.151^{***}
	(0.507)	(0.538)	(0.540)
partyid: R/SD	6.031^{***}	4.985^{***}	5.015^{***}
	(0.450)	(0.479)	(0.483)
partyid: SR/SD	7.191^{***}	5.999^{***}	6.168^{***}
	(0.621)	(0.738)	(0.742)
sex: Female/Male	0.049		-0.006
	(0.193)		(0.224)
owngun: YES/NO	0.642^{***}	0.417	0.449^{*}
	(0.194)	(0.221)	(0.224)
race: BLACK/WHITE		-2.067^{***}	-2.110^{***}
		(0.450)	(0.450)
race: OTHER/WHITE		-0.483	-0.497
		(0.391)	(0.394)
polviews: L/EL		0.303	
		(0.866)	
polviews: SL/EL		1.173	
		(0.819)	
polviews: M/EL		1.761^{*}	