Linear Regression

Paul E. Johnson[1] [2] and Terrence Jorgensen[2]

[1]Department of Political Science

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

2018

KU CENTER FOR RESEARCH METHODS & DATA ANALYSIS

College of Liberal Arts & Sciences

Outline

Outline ...

- Multi-Category factor
- Numerical Interaction

5 Conclusion

Outline

Check your packages

- Recall that the R (R Core Team, 2017) packages “stats” “graphics” “datasets” “base” “utils” and “grDevices” are loaded by default.

```
sessionInfo()
```

```
R version 3.6.0 (2019-04-26)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 19.04

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/atlas/libblas.so.3.10.3
LAPACK: /usr/lib/x86_64-linux-gnu/atlas/liblapack.so.3.10.3

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
     LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=en_US.UTF-8     LC_MONETARY=en_US.UTF-8
     LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8
    LC_IDENTIFICATION=C

attached base packages:
```

Check your packages ...

```
[1] stats     graphics  grDevices utils     datasets  methods   base

loaded via a namespace (and not attached):
[1] compiler_3.6.0 tools_3.6.0
```

Check your packages

- We'll use some addons today

If you don't already have these R packages, install them on your computer

```
install.packages(c("car", "lmtest", "rockchalk"))
```

Don't forget to check documentation

You can browse a list of all functions in a particular package (e.g., rockchalk)

```
library(rockchalk)
help(package = rockchalk)
```

or look up a help page for a specific function

```
?plotSlopes
```

Outline

Got Data?

- The example data is saved in "data/affect.dat"
- Unusually, this data does not have column names in row 1.

```
dat <- read.table("data/affect.dat", header =
    FALSE)
colnames(dat) <- c("Agency1", "Agency2",
    "Agency3",
                  "Intrin1", "Intrin2", "Intrin3",
                  "Extrin1", "Extrin2", "Extrin3",
                  "PosAFF1", "PosAFF2", "PosAFF3",
                  "NegAFF1", "NegAFF2", "NegAFF3",
                  "Sex", "Ethnic2", "Ethnic3",
                     "Ethnic4")
```

- View first few rows of data

```
options("width" = 70)
head(dat)
```

Got Data? ...

```
  Agency1 Agency2 Agency3 Intrin1 Intrin2 Intrin3 Extrin1 Extrin2
1  3.5000  4.0000  4.0000  4.0000     4.0       4  1.0000  1.0000
2  2.5000  3.1667  3.0000  3.2123     2.0       3  1.8333  2.6667
3  1.8333  2.0000  1.5000  3.0000     3.0       2  1.0000  1.0000
4  2.7714  3.0602  2.3639  3.1337     4.0       3  1.0774  1.1667
5  3.1667  3.3333  2.8333  3.5000     4.0       4  1.8333  2.0000
6  2.3333  2.8333  2.3333  3.0000     2.5       3  3.0588  2.4125
  Extrin3 PosAFF1 PosAFF2 PosAFF3 NegAFF1 NegAFF2 NegAFF3 Sex
      Ethnic2
1  1.5000  4.0000     4.0     4.0     1.0  1.0000     1.0   1
             0
2  1.8333  3.0000     3.5     2.5     1.5  1.6858     1.5   1
             0
3  1.0000  3.0184     2.5     3.0     1.0  1.0000     1.0   1
             0
4  1.0000  3.0000     2.5     3.0     2.5  2.5000     1.5   1
             0
5  1.8333  3.7804     3.5     3.0     2.5  2.0000     3.0   1
             0
6  2.6667  4.0000     3.0     3.0     2.0  1.5000     2.0   1
             0
  Ethnic3 Ethnic4
1       1       0
2       0       0
3       0       0
```

Got Data? ...

```
4           0           0
5           0           0
6           0           0
```

```
options("width" = 80)
```

recodes

- Create scales by calculating means of the indicator variables

```
dat$agency <- rowMeans(dat[ , c("Agency1","Agency2","Agency3")], na.rm = TRUE)
dat$intMotiv <- rowMeans(dat[ , c("Intrin1","Intrin2","Intrin3")], na.rm = TRUE)
dat$extMotiv <- rowMeans(dat[ , c("Extrin1","Extrin2","Extrin3")], na.rm = TRUE)
dat$posAffect <- rowMeans(dat[ , c("PosAFF1","PosAFF2","PosAFF3")], na.rm = TRUE)
dat$negAffect <- rowMeans(dat[ , c("NegAFF1","NegAFF2","NegAFF3")], na.rm = TRUE)
```

recodes ...

- Recode dummy variables

```
table(dat$Sex)
```

```
  1   2
195 185
```

```
dat$gender <- factor(dat$Sex, levels = c(1,2),
    labels = c("male", "female"))
dat$ethnicity <- as.factor(ifelse(dat$Ethnic2,
    "Black",
                            ifelse(dat$Ethnic3,
                                "Hispanic",
                            ifelse(dat$Ethnic4,
                                "Asian",
                                "White"))))
```

recodes ...

```
dat$race <-
    rockchalk::combineLevels(dat$ethnicity, levs
    = c("Black", "Hispanic", "Asian"), newLabel =
    "Nonwhite")
```

```
The original levels Asian Black Hispanic White
have been replaced by White Nonwhite
```

```
options("width" = 60)
head(dat)
```

recodes ...

```
  Agency1 Agency2 Agency3 Intrin1 Intrin2 Intrin3 Extrin1
1  3.5000  4.0000  4.0000  4.0000     4.0       4  1.0000
2  2.5000  3.1667  3.0000  3.2123     2.0       3  1.8333
3  1.8333  2.0000  1.5000  3.0000     3.0       2  1.0000
4  2.7714  3.0602  2.3639  3.1337     4.0       3  1.0774
5  3.1667  3.3333  2.8333  3.5000     4.0       4  1.8333
6  2.3333  2.8333  2.3333  3.0000     2.5       3  3.0588
  Extrin2 Extrin3 PosAFF1 PosAFF2 PosAFF3 NegAFF1 NegAFF2
1  1.0000  1.5000  4.0000     4.0     4.0     1.0  1.0000
2  2.6667  1.8333  3.0000     3.5     2.5     1.5  1.6858
3  1.0000  1.0000  3.0184     2.5     3.0     1.0  1.0000
4  1.1667  1.0000  3.0000     2.5     3.0     2.5  2.5000
5  2.0000  1.8333  3.7804     3.5     3.0     2.5  2.0000
6  2.4125  2.6667  4.0000     3.0     3.0     2.0  1.5000
  NegAFF3 Sex Ethnic2 Ethnic3 Ethnic4   agency intMotiv
1     1.0   1       0       1       0 3.833333 4.000000
2     1.5   1       0       0       0 2.888900 2.737433
3     1.0   1       0       0       0 1.777767 2.666667
4     1.5   1       0       0       0 2.731833 3.377900
5     3.0   1       0       0       0 3.111100 3.833333
6     2.0   1       0       0       0 2.499967 2.833333
  extMotiv posAffect negAffect gender ethnicity     race
1 1.166667  4.000000  1.000000   male  Hispanic Nonwhite
2 2.111100  3.000000  1.561933   male     White    White
3 1.000000  2.839467  1.000000   male     White    White
```

recodes ...

```
4 1.081367   2.833333   2.166667   male      White     White
5 1.888867   3.426800   2.500000   male      White     White
6 2.712667   3.333333   1.833333   male      White     White
```

```
options("width" = 80)
```

- Save a copy of that in the workingdata folder

```
saveRDS(dat, file = "workingdata/affect.rds")
```

Outline

R formula

- Almost all model fitting functions in R use the Wilkinson and Rogers notation

```
dependent_variable ~ predictor_variable
```

- Can omit the estimation of the intercept
  - Old fashioned way

```
dependent_variable ~ -1 + predictor_variable
```

  - The old fashioned way confused school children, hence the new fashioned way

```
dependent_variable ~ 0 + predictor_variable
```

- There are special symbols in R formula notation, "+",":", "*", "/", "^", "|".

The lm() function

- Suppose we want to explain Positive Affect with sense of Agency

```
reg.mod.1 <- lm(posAffect ~ agency, data = dat)
```

Returns "silently" unless there is an error

- Print method for lm objects offers minimal information

```
reg.mod.1
```

```
Call:
lm(formula = posAffect ~ agency, data = dat)

Coefficients:
(Intercept)       agency
     2.1883       0.3491
```

- There is quite a bit of structure in there, however. Run "str()" you will see. I'll run the briefer "attributes".

The lm() function ...

```
attributes(reg.mod.1)
```

```
$names
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"

$class
[1] "lm"
```

Direct Retrieval versus Accessor functions

- The lm object "reg.mod.1" is of class "lm", S3 object.

```
class(reg.mod.1)
```

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

- If you ran "str(reg.mod.1)", a big structure inside there would be apparent.
- I'll just ask for names

```
names(reg.mod.1)
```

```
 [1] "coefficients"  "residuals"     "effects"
 [4] "rank"          "fitted.values" "assign"
 [7] "qr"            "df.residual"   "xlevels"
[10] "call"          "terms"         "model"
```

Everybody Loves $

- S3 list objects allow a shortcut access with the dollar sign
  - data.frame access like dat$x1
- Notice that inside the fitted model object there is an element named "df.residual". Get that:

```
reg.mod.1$df.residual
```

```
[1] 378
```

Since the dollar sign is a shortcut notation, we could go the long form as well

```
reg.mod.1[["df.residual"]]
```

```
[1] 378
```

- Any of the elements in reg.mod.1's internal structure can be retrieved in that way.

Everybody Loves $ ...

- If this were an S4 class object, then we would use the "@" sign rather than the "$" sign as a shortcut.
- The R Core team does NOT encourage us to pull pieces out in that way.
  - They reserve the right to rename those internal bits.
- Instead, it is recommended to use "**accessor**" functions that R provides

Coefficients, retrieved both ways

- Point estimates of parameters (regression coefficients)
  1. The accessor function “coef()” (short for coefficients)

```
coef(reg.mod.1)
```

```
(Intercept)      agency
  2.1882796   0.3491155
```

  2. Use the dollar sign access

```
reg.mod.1$coefficients
```

```
(Intercept)      agency
  2.1882796   0.3491155
```

- The predicted value vector
  1. The accessor function “fitted()”

```
head(fitted(reg.mod.1))
```

Coefficients, retrieved both ways ...

```
       1        2        3        4        5        6
3.526556 3.196839 2.808925 3.142005 3.274413 3.061057
```

2. The dollar sign avenue (note element name “fitted.values” is different than accessor function name “fitted()”)

```
head(reg.mod.1$fitted.values)
```

```
       1        2        3        4        5        6
3.526556 3.196839 2.808925 3.142005 3.274413 3.061057
```

Coefficients, retrieved both ways ...

- The residual vector
  1. The accessor function "resid()"

```
head(resid(reg.mod.1), 4)
```

```
         1          2          3          4
0.47344443 -0.19683927  0.03054124 -0.30867153
```

  2. The dollar sign avenue (note element name is different than accessor function)

```
head(reg.mod.1$residuals, 4)
```

```
         1          2          3          4
0.47344443 -0.19683927  0.03054124 -0.30867153
```

Some functions offer much more elaborate information

- Every useful object in R is supposed to have a summary() method!
- The "summary()" function is as close as we get to a "big standard output"

```
summary(reg.mod.1)
```

```
Call:
lm(formula = posAffect ~ agency, data = dat)

Residuals:
     Min       1Q   Median       3Q      Max
-2.06107 -0.37515  0.04591  0.45144  1.39929

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.18828    0.15963  13.708  < 2e-16 ***
agency       0.34912    0.06233   5.601  4.1e-08 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6077 on 378 degrees of freedom
```

Some functions offer much more elaborate information ...

```
Multiple R-squared:  0.07664, Adjusted R-squared:  0.0742
F-statistic: 31.37 on 1 and 378 DF,  p-value: 4.103e-08
```

- Confidence intervals for regression coefficients

```
confint(reg.mod.1, level = .99)
```

```
                0.5 %    99.5 %
(Intercept) 1.7750122 2.6015469
agency      0.1877538 0.5104771
```

Math functions in Formulas

- If we wanted to predict with the logarithm of a variable,
  1. We could create a new variable by recoding, Or
  2. use the symbol for the logarithm in the formula

```
dependent_variable ~ log(predictor)
```

Any of the mathematical transformations in R could be used in place of log.

```
dependent_variable ~ sqrt(predictor)
```

- I don't usually write math transformations into formulas, it complicates plotting and table-making duties later on.

Special Symbols in Formulas

- Multiple regression: "+" for more predictors

```
dependent_variable ~ predictor1 + predictor2
```

- Interaction: "*"

```
dependent_variable ~ predictor1 * predictor2
```

Means you want a regression to estimate 3 coefficients,
$\beta_1 predictor1 + \beta_2 predictor2 + \beta_3 predictor1 \times predictor2$

Be cautious with ^

- You may think to yourself, "I'll add a squared term":

```
dependent_variable ~ predictor + predictor^2
```

- However, there is a gotcha
  1. "^" has a special meaning in the formula notation.
  2. If we are trying to make a predictive equation like

     $$dependent\_variable = \beta_0 + \beta_1 predictor + \beta_2 predictor^2$$

     Wrap the math inside the capital **I()**.

```
dependent_variable ~ predictor
          + I(predictor^2)
```

  3. But don't do that, better ways exist (orthogonal polynomials).

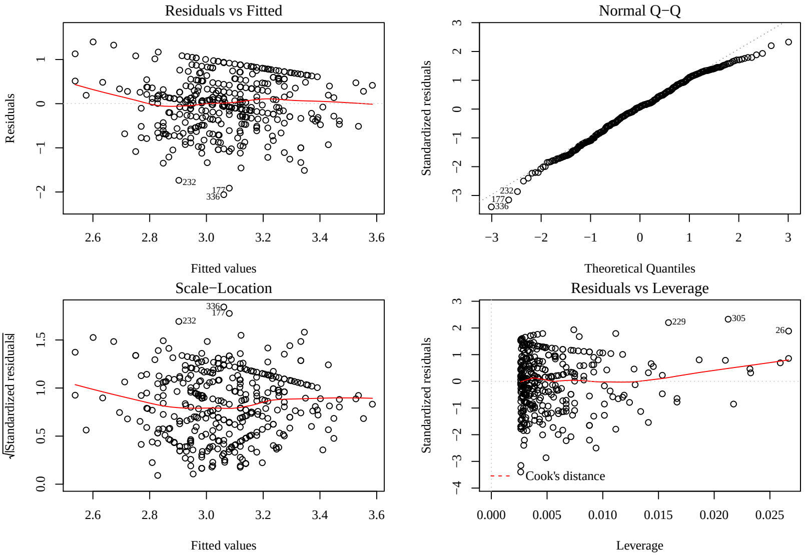
plot diagnostics

The `lm` class has a plot method (`plot.lm`)

```
plot(reg.mod.1)
```

defaults to offer 4 graphs (can be adjusted, see ?`plot.lm`)

plot diagnostics ...



plot diagnostics ...

Plot diagnostics

1. residuals are (not) related to fitted values
2. residuals are (not) approximately normally distributed
3. residuals are (not) homoscedastic
4. highly influential (leverage) observations do/not exist (Cook's Distance)

Influence Diagnostics

- The influence.measures() function collects a great deal of information and displays information for each row in the data for the fitted model:

```
inf1 <- influence.measures(reg.mod.1)
```

- Output is immense, does not fit within these notes

```
inf1
```

generates such a massive outflow that the presentation software fails.

- A tidy summary function show the problematic cases

```
summary(inf1)
```

Influence Diagnostics ...

```
Potentially influential observations of
   lm(formula = posAffect ~ agency, data = dat) :

    dfb.1_ dfb.agnc dffit   cov.r   cook.d hat
1   -0.10   0.11     0.12    1.02_*  0.01   0.02_*
25  -0.02   0.03     0.03    1.02_*  0.00   0.02
26   0.31  -0.30     0.31_*  1.01    0.05   0.03_*
38   0.07  -0.06     0.07    1.02_*  0.00   0.01
95   0.11  -0.12    -0.13    1.02_*  0.01   0.02_*
129 -0.06   0.07     0.08    1.02_*  0.00   0.01
131  0.08  -0.09    -0.10    1.02_*  0.01   0.02_*
136  0.14  -0.13     0.14    1.03_*  0.01   0.03_*
146 -0.10   0.11     0.11    1.03_*  0.01   0.03_*
177 -0.02  -0.01    -0.16    0.96_*  0.01   0.00
196  0.02  -0.04    -0.13    0.98_*  0.01   0.00
211 -0.06   0.04    -0.12    0.98_*  0.01   0.00
229  0.27  -0.26     0.28_*  1.00    0.04   0.02_*
230  0.07  -0.08    -0.08    1.02_*  0.00   0.02_*
232 -0.16   0.14    -0.20    0.97_*  0.02   0.00
245  0.11  -0.10     0.11    1.02_*  0.01   0.02_*
252 -0.06   0.07     0.07    1.03_*  0.00   0.02_*
256  0.04  -0.05    -0.06    1.02_*  0.00   0.01
261 -0.03   0.03     0.04    1.02_*  0.00   0.01
280  0.05  -0.05     0.05    1.03_*  0.00   0.02_*
294  0.05  -0.05     0.05    1.02_*  0.00   0.01
```

Influence Diagnostics ...

```
305  0.34  -0.32     0.35_*  1.00    0.06   0.02_*
336 -0.04   0.00    -0.18    0.95_*  0.02   0.00
353  0.05  -0.05    -0.06    1.02_*  0.00   0.02
368  0.18  -0.21    -0.25_*  0.98_*  0.03   0.01
373  0.01  -0.01    -0.01    1.02_*  0.00   0.01
```

Thumbnail sketch

- In case you wonder what those things are, I wrote out a longer lecture about it http://pj.freefaculty.org/guides/stat/Regression/RegressionDiagnostics

```
 dfb.1_ dfb.agnc    dffit     cov.r    cook.d      hat
 -0.098    0.108    0.115     1.024     0.007    0.021
```

- Thumbnail sketch is as follows

  dfb "df beta" (one for each coefficient) shows how the coefficient estimate would change if that row were dropped

  dffit change in predicted value if that row were dropped

  cook.d A summary of how far the vector of parameter estimates $(\hat{\beta}_0, \hat{\beta}_1)$ would change if that row were dropped.

  hat The "Hat matrix" value for the row. If this value is large, it means the case is influential in changing the overall regression line

The predict function accepts a "newdata" argument

- `predict(reg.mod.1)` is the same result as `fitted(reg.mod.1)`: the predicted values of the observed cases.
- Often, we want predicted values for particular, substantively interesting values of the predictors.
- Obtain predicted values for a new data set:

```
predict(reg.mod.1, newdata =
    some_data_frame_we_make_up)
```

- And if we want confidence intervals, we add

```
predict(reg.mod.1, newdata =
    some_data_frame_we_make_up, interval =
    "confidence")
```

- This makes it possible to calculate "marginal effects", the change in the outcome due to any given change in a predictor.

rockchalk::newdata

- The newdata object MUST include
  - all predictors, with exactly same names as used in the formula, and
  - values of factors within the newdata object must match the data used to fit the model
- In rockchalk, I needed this often and wrote a "newdata" function.
- For example, I notice the variable "agency" varies between 1 and 4.

```
library(rockchalk)
nd <- newdata(reg.mod.1, predVals =
    c("agency"), n = 5)
nd
```

```
    agency
1 1.000000
2 2.175525
3 2.499983
4 2.832975
5 4.000000
```

rockchalk::newdata ...

```
## n = 5 is 5 evenly spaced quartile values
```

- Then we use that with the predict function

```
reg.mod.1.pred <- predict(reg.mod.1, newdata
    = nd)
reg.mod.1.pred
```

```
       1        2        3        4        5
2.537395 2.947789 3.061062 3.177315 3.584741
```

Stash Predictions into the newdata frame

- Usually, you want to save the fitted values

```
nd$reg.mod.1.pred <- predict(reg.mod.1, newdata =
    nd)
```

- Because I became tired of that, in rockchalk I wrote `predictOMatic()`. It creates the new data and also saves the predictions:

```
reg.mod.1.pm <- predictOMatic(reg.mod.1, predVals
    = c("agency"), n = 5)
reg.mod.1.pm
```

```
    agency      fit
1 1.000000 2.537395
2 2.175525 2.947789
3 2.499983 3.061062
4 2.832975 3.177315
5 4.000000 3.584741
```

Stash Predictions into the newdata frame ...

- A more-or-less "automatic" graphing routine, " plotSlopes ", will do all of this and draw a plot. Before I show that, I need to show about confidence intervals for predictions

Confidence Interval on Predicted Values

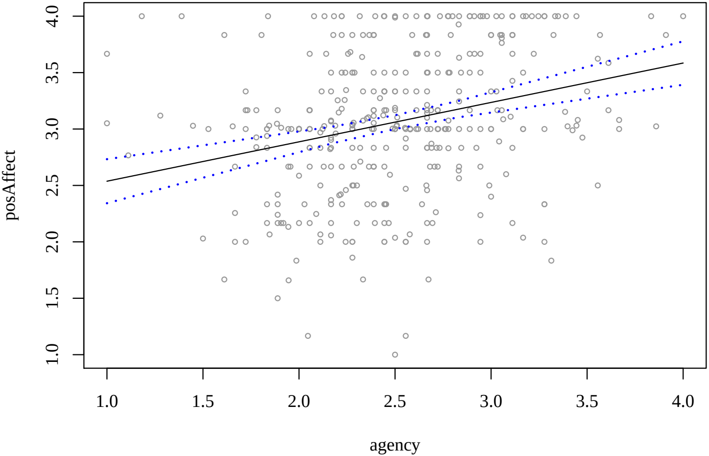
- The R predict() function has a confidence interval argument. It defaults to none, but can be either "confidence" or "prediction".
- The returned data structure is a matrix with 3 columns

```
reg.mod.1.pred2 <- predict(reg.mod.1, newdata =
    nd, interval = "confidence")
reg.mod.1.pred2
```

```
       fit      lwr      upr
1 2.537395 2.342241 2.732549
2 2.947789 2.873926 3.021652
3 3.061062 2.999750 3.122375
4 3.177315 3.104471 3.250159
5 3.584741 3.392334 3.777149
```

- For the 5 example values of agency, we have a value
  1. "on" the line, and
  2. 95% range below ("lwr")
  3. 95% range above ("upr")

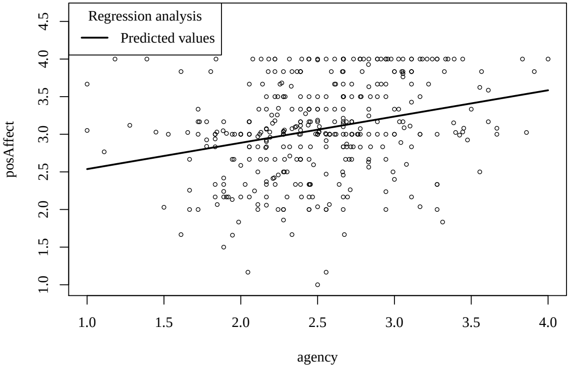
The CI lines should be a "smooth hourglass"



Those CI lines are "connect-the-dots" curves. In this case, they don't look so bad

plotSlopes with no confidence interval

```
plotSlopes(reg.mod.1, plotx = "agency")
```



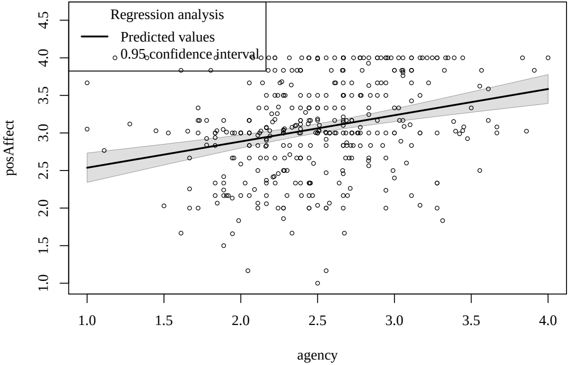
predictOMatic understands the interval argument

```
reg.mod.1.pm <- predictOMatic(reg.mod.1, predVals
    = c("agency"), n = 5, interval = "confidence")
reg.mod.1.pm
```

```
    agency      fit      lwr      upr
1 1.000000 2.537395 2.342241 2.732549
2 2.175525 2.947789 2.873926 3.021652
3 2.499983 3.061062 2.999750 3.122375
4 2.832975 3.177315 3.104471 3.250159
5 4.000000 3.584741 3.392334 3.777149
```

```
plotSlopes(reg.mod.1, plotx = "agency", n = 5,
    interval = "confidence")
```

predictOMatic understands the interval argument ...



- plotSlopes creates an output object that has the newdata in it:

```
reg.mod.ps <- plotSlopes(reg.mod.1, plotx =
    "agency", n = 5, interval = "confidence")
```

predictOMatic understands the interval argument ...

- To obtain smooth curve, I need to calculate predictions for more values of X. plotSlopes calculates predicted values for, n = 40, values of agency

```
reg.mod.ps$newdata
```

```
      agency      fit      lwr      upr
1   1.000000 2.537395 2.342241 2.732549
2   1.076923 2.564250 2.378022 2.750478
3   1.153846 2.591105 2.413752 2.768458
4   1.230769 2.617960 2.449422 2.786498
5   1.307692 2.644815 2.485022 2.804608
6   1.384615 2.671670 2.520540 2.822801
7   1.461538 2.698525 2.555961 2.841090
8   1.538462 2.725380 2.591266 2.859495
9   1.615385 2.752235 2.626432 2.878039
10  1.692308 2.779090 2.661430 2.896751
11  1.769231 2.805945 2.696222 2.915669
12  1.846154 2.832800 2.730760 2.934841
13  1.923077 2.859655 2.764982 2.954329
14  2.000000 2.886511 2.798809 2.974212
15  2.076923 2.913366 2.832139 2.994592
16  2.153846 2.940221 2.864843 3.015598
```

predictOMatic understands the interval argument ...

```
17 2.230769 2.967076 2.896766 3.037385
18 2.307692 2.993931 2.927727 3.060134
19 2.384615 3.020786 2.957539 3.084032
20 2.461538 3.047641 2.986036 3.109245
21 2.538462 3.074496 3.013113 3.135878
22 2.615385 3.101351 3.038755 3.163947
23 2.692308 3.128206 3.063041 3.193371
24 2.769231 3.155061 3.086123 3.223998
25 2.846154 3.181916 3.108186 3.255646
26 2.923077 3.208771 3.129414 3.288128
27 3.000000 3.235626 3.149972 3.321280
28 3.076923 3.262481 3.169996 3.354966
29 3.153846 3.289336 3.189595 3.389077
30 3.230769 3.316191 3.208857 3.423525
31 3.307692 3.343046 3.227847 3.458245
32 3.384615 3.369901 3.246618 3.493184
33 3.461538 3.396756 3.265210 3.528302
34 3.538462 3.423611 3.283655 3.563567
35 3.615385 3.450466 3.301978 3.598955
36 3.692308 3.477321 3.320198 3.634444
37 3.769231 3.504176 3.338332 3.670020
38 3.846154 3.531031 3.356393 3.705670
39 3.923077 3.557886 3.374391 3.741382
40 4.000000 3.584741 3.392334 3.777149
```

R factors are recoded into "dummy variables" in regression models

- gender is a dichotomous variable, coded "male" or "female". Check the levels:

```
levels(dat$gender)
```

```
[1] "male"   "female"
```

R factors are recoded into ”dummy variables” in regression models ...

- Include gender as a predictor

```
reg.mod.2 <- lm(posAffect ~ gender, data = dat)
summary(reg.mod.2)
```

```
Call:
lm(formula = posAffect ~ gender, data = dat)

Residuals:
     Min       1Q   Median       3Q      Max
-2.10992 -0.35609 -0.01197  0.47724  0.97724

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.02276    0.04518  66.906   <2e-16 ***
genderfemale  0.08717    0.06475   1.346    0.179
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6309 on 378 degrees of freedom
Multiple R-squared:  0.004772,  Adjusted R-squared:  0.002139
F-statistic: 1.812 on 1 and 378 DF,  p-value: 0.179
```

R factors are recoded into "dummy variables" in regression models ...

- One gender, in this case "males", is treated as a baseline group. There are 2 categories, we can only estimate 2 coefficients. The default model include an intercept, then only 1 coefficient is left over for one of the groups.
- In this case, the predicted values would be
  - males: 3.0227
  - females: $3.0227 + 0.08717$

R factors are recoded into "dummy variables" in regression models ...

- There are 2 alternatives to this coding scheme

1. Get rid of the intercept, in which case we get one estimate for males, one for females

```
reg.mod.2b <- lm(posAffect ~ 0 + gender, data
    = dat)
summary(reg.mod.2b)
```

```
Call:
lm(formula = posAffect ~ 0 + gender, data = dat)

Residuals:
     Min        1Q    Median        3Q       Max
-2.10992 -0.35609 -0.01197  0.47724  0.97724

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
gendermale    3.02276    0.04518   66.91   <2e-16 ***
genderfemale  3.10992    0.04638   67.05   <2e-16 ***
---
```

R factors are recoded into "dummy variables" in regression models ...

```
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

Residual standard error: 0.6309 on 378 degrees of freedom
Multiple R-squared:  0.9596,  Adjusted R-squared:  0.9594
F-statistic:  4486 on 2 and 378 DF,  p-value: < 2.2e-16
```

The disadvantage of this coding is that we cannot directly say whether the 2 values are statistically significantly different from one another.

2. Reverse the levels on the gender variable

```
dat$gender2 <- factor(dat$gender, levels =
   c("female", "male"))
reg.mod.2c <- lm(posAffect ~ gender2, data = dat)
summary(reg.mod.2c)
```

R factors are recoded into "dummy variables" in regression models ...

```
Call:
lm(formula = posAffect ~ gender2, data = dat)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.10992 -0.35609 -0.01197  0.47724  0.97724 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  3.10992    0.04638  67.047   <2e-16 ***
gender2male -0.08717    0.06475  -1.346    0.179    
---
Signif. codes:  
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6309 on 378 degrees of freedom
Multiple R-squared:  0.004772,  Adjusted R-squared:  0.002139 
F-statistic: 1.812 on 1 and 378 DF,  p-value: 0.179
```

Test Homogeneity of Variances

- Use the Levene test, which is in John Fox's car package.

```
library(car)
leveneTest(reg.mod.2)
```

```
Levene's Test for Homogeneity of Variance (center = median)
       Df F value Pr(>F)
group   1  2.3963 0.1225
      378
```

- This suggests we were not wrong to assume the error variances for males and females are the same.

A multi-category factor

```
levels(dat$ethnicity)
```

```
[1] "Asian"    "Black"    "Hispanic" "White"
```

```
reg.mod.3 <- lm(posAffect ~ ethnicity, data = dat)
summary(reg.mod.3)
```

```
Call:
lm(formula = posAffect ~ ethnicity, data = dat)

Residuals:
     Min       1Q   Median       3Q      Max
-2.00610 -0.40094 -0.01907  0.49390  1.06360

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)         2.9364     0.1026  28.628   <2e-16 ***
ethnicityBlack      0.0697     0.1777   0.392    0.695
ethnicityHispanic   0.1237     0.1284   0.964    0.336
ethnicityWhite      0.1536     0.1099   1.398    0.163
---
Signif. codes:
```

A multi-category factor ...

```
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6323 on 376 degrees of freedom
Multiple R-squared:  0.005664,  Adjusted R-squared:  -0.00227
F-statistic: 0.7139 on 3 and 376 DF,  p-value: 0.5442
```

Finding out what's going wrong

- Sometimes you'll have confusing output that you can't understand
- I often snoop on data as R sees it "inside" lm() :
  1. model.frame: a data.frame with output and predictors that R creates when you run lm.

```
rm1.mf <- model.frame(reg.mod.1)
head(rm1.mf)
```

```
  posAffect   agency
1  4.000000 3.833333
2  3.000000 2.888900
3  2.839467 1.777767
4  2.833333 2.731833
5  3.426800 3.111100
6  3.333333 2.499967
```

  2. Suppose the regression fails, so there is no object from which to obtain a frame. No problem! Give the formula to model.frame.

Finding out what's going wrong ...

```
rm1.mf <- model.frame(posAffect ~
    log(agency), data = dat)
head(rm1.mf)
```

```
  posAffect log(agency)
1  4.000000   1.3437347
2  3.000000   1.0608758
3  2.839467   0.5753579
4  2.833333   1.0049729
5  3.426800   1.1349764
6  3.333333   0.9162774
```

3. model.matrix shows the "design matrix", the numeric columns used in estimation

```
rm1.dm <- model.matrix(reg.mod.1)
head(rm1.dm)
```

Finding out what's going wrong ...

```
  (Intercept)   agency
1           1 3.833333
2           1 2.888900
3           1 1.777767
4           1 2.731833
5           1 3.111100
6           1 2.499967
```

- This is especially revealing if there is a factor as a predictor

```
rm2.dm <- model.matrix(posAffect ~ ethnicity,
    data = dat)
head(rm2.dm)
```

Finding out what's going wrong ...

```
  (Intercept) ethnicityBlack ethnicityHispanic
1           1              0                 1
2           1              0                 0
3           1              0                 0
4           1              0                 0
5           1              0                 0
6           1              0                 0
  ethnicityWhite
1              0
2              1
3              1
4              1
5              1
6              1
```

Outline

Addition sign "+"

- Can insert math transformations "on the fly"

```
dep_var ~ log(predictor1) + sqrt(predictor2) +
    predictor3 + predictor4
```

but this makes creating a newdata object somewhat more complicated

- However, `rockchalk::newdata()` and predictOMatic can work together to avoid problems for us!

Multiplication sign "*" is not exactly like multiplication

- A multiplicative interaction between two continuous predictors can be entered like so

```
dep_var ~ predictor1 * predictor2 + predictor3
```

- It adds predictor1 and predictor2 as "additive" (or "main") effects, plus their product.

```
dep_var ~ predictor1 + predictor2 +
    predictor1:predicor2 + predictor3
```

- COLON! The symbol "predictor1:predictor2" represents "$predictor1 \times predictor2$".

With categorical predictors, "*" does something else

- Because factor variables are broken into dummy variables, an interactive term like

```
posAffect ~ gender * agency
```

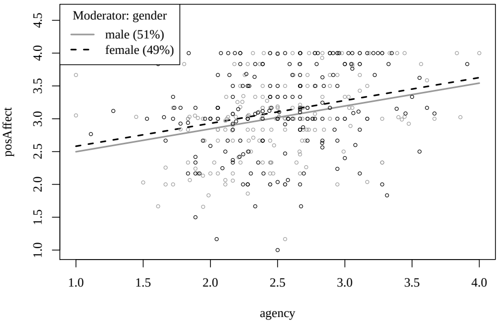
will have the effect of estimating a different slope and a different intercept for each of the sexes. Will illustrate in next section.

Our first guess might be "everything is additive"

```
reg.mod.5 <- lm(posAffect ~ agency + gender, data
    = dat)
summary(reg.mod.5)
```

```
Call:
lm(formula = posAffect ~ agency + gender, data = dat)

Residuals:
     Min       1Q   Median       3Q      Max
-2.10427 -0.39890  0.05395  0.44156  1.35513

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.14912    0.16207  13.260  < 2e-16 ***
agency        0.34839    0.06226   5.596 4.24e-08 ***
genderfemale  0.08417    0.06230   1.351    0.177
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.607 on 377 degrees of freedom
Multiple R-squared:  0.08109, Adjusted R-squared:  0.07621
F-statistic: 16.63 on 2 and 377 DF,  p-value: 1.194e-07
```

Our first guess might be "everything is additive" ...



This asserts "parallel lines" for males and females

Agency effect depends on gender?

One might imagine that rather than an additive effect of gender, as in

$$posAffect_i = \beta_0 + \beta_1 agency_i + \beta_2 female_i$$

it is more likely that the effect of agency differs between males and females

$$posAffect_i = \beta_0 + \beta_1 agency_i + \beta_2 female_i + \beta_3 agency_i \times female$$

```
reg.mod.6 <- lm(posAffect ~ agency*gender, data =
    dat)
```

The results are

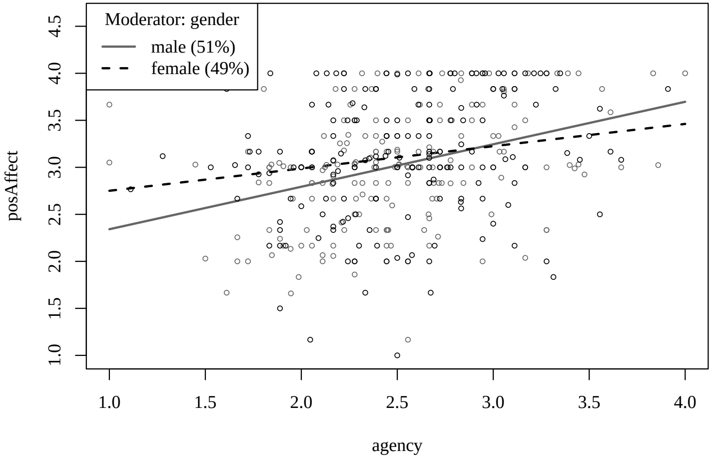
```
summary(reg.mod.6)
```

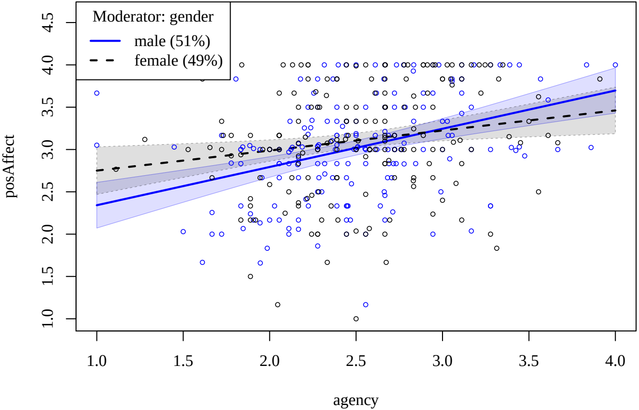
Agency effect depends on gender? ...

```
Call:
lm(formula = posAffect ~ agency * gender, data = dat)

Residuals:
     Min       1Q   Median       3Q      Max
-2.10608 -0.40401  0.02606  0.44460  1.32508

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)          1.88976    0.22056   8.568 2.75e-16
agency               0.45182    0.08624   5.239 2.69e-07
genderfemale         0.62379    0.31834   1.960   0.0508
agency:genderfemale -0.21481    0.12428  -1.728   0.0847

(Intercept)         ***
agency              ***
genderfemale        .
agency:genderfemale .
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6054 on 376 degrees of freedom
Multiple R-squared:  0.08833, Adjusted R-squared:  0.08106
F-statistic: 12.14 on 3 and 376 DF,  p-value: 1.331e-07
```

Agency effect depends on gender? ...

Visualize the interaction




Moderator: gender
male (51%)
female (49%)
posAffect
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
agency

How to best plot that?

- My tendency has been to draw several groups on one plot.
- Others prefer "trellis" graphics, which make smaller pictures, one for each group
- In the base of R, the lattice package is provided for this purpose
- Hadley Wickham's ggplot2 package is a little bit easier to use, so we will test that.

ggplot thumbnail sketch

- ggplot is similar in many ways to concept of R base graphics,
- We can
  - draw a "blank" figure
  - add pieces to it
- However,
  - it uses an entirely different vocabulary, such as "geom" and "aes".
  - additional graph commands do not just "draw" pieces can fundamentally alter the display.
  - variable names need not be quoted (I find this confusing)

ggplot thumbnail sketch

- The plot is initiated by a call to `ggplot()`, which must specify an "aesthetic", the fundamental nature of the plot
  - An interesting difference with base graphics is that we think of "adding" graph elements

```
p1 <- ggplot(data.frame, aes(...))
p1 <- p1 + new features here
p1
```

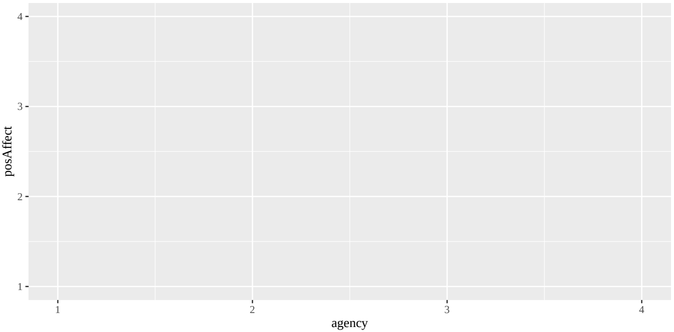
- The last `p1` causes the result to be drawn in a graphic window.
- People often write a string of added-together features, but I usually test the new features one at a time.

```
p1 <- ggplot(data.frame, aes(...))
          + new feature here
          + more features
p1
```

ggplot thumbnail sketch ...

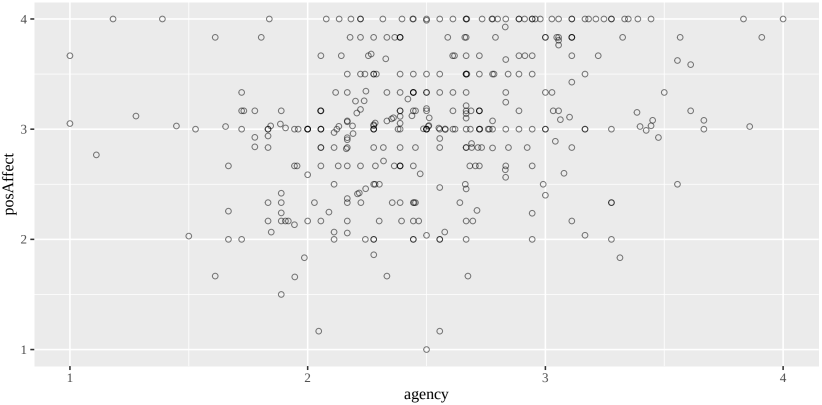
- Sometimes the ordering of new features will make a little difference in the final display.

ggplot blank page



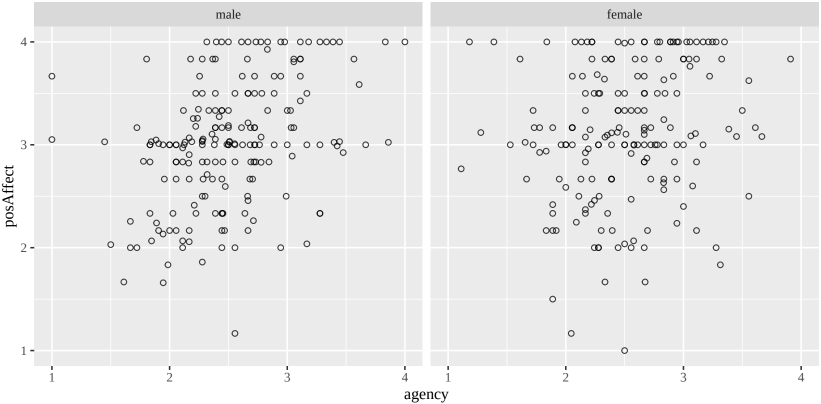

```
library(ggplot2)
p1 <- ggplot(dat, aes(x = agency, y = posAffect))
p1
```

geom_point is for points in a scatter



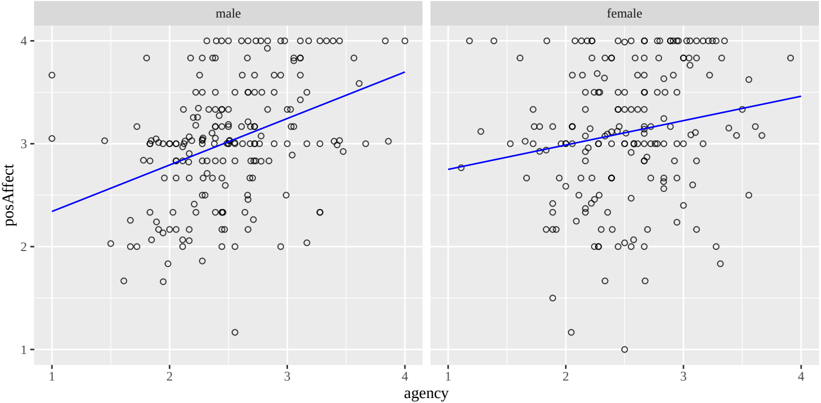

```
p1 <- p1 + geom_point(shape = 1, alpha = 0.5)
p1
```

facet_grid() divides plot into sections



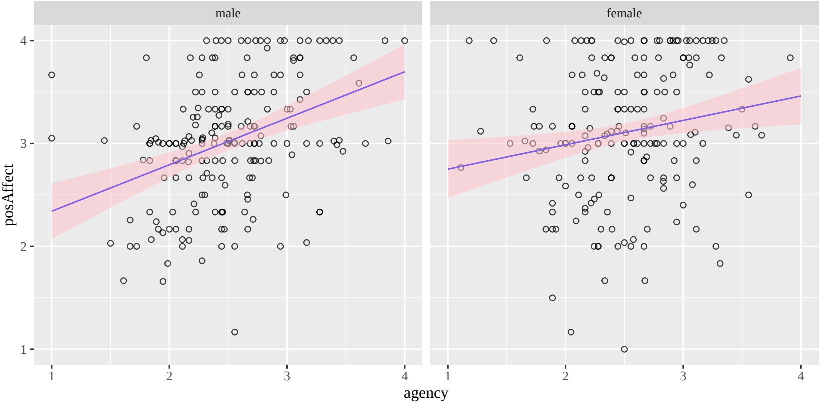

```
p1 <- p1 + geom_point(shape = 1, alpha = 0.5)
p1 <- p1 + facet_grid(. ~ gender)
p1
```

geom_line will get line data from plotSlopes object



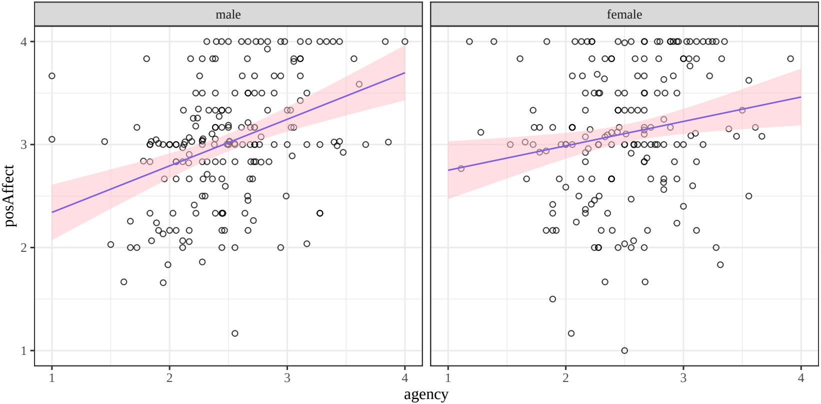

```
ps31$newdata$posAffect <- ps31$newdata$fit
p1 <- p1 + geom_line(data = ps31$newdata, color =
    "blue")
p1
```

geom_ribbon() can draw the confidence intervals



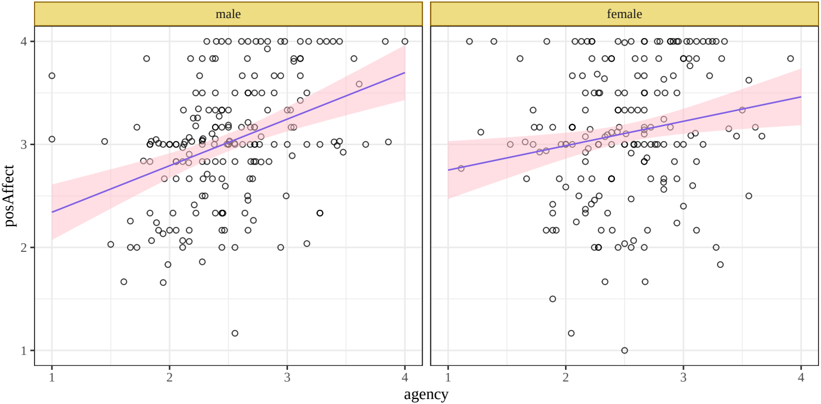

```
p1 <- p1 + geom_ribbon(data = ps31$newdata,
                aes(ymin = lwr, ymax = upr),
        fill = "pink", alpha = 0.5)
p1
```

I don't want gray boxes in background!




```
p1 <- p1 + theme_bw()
p1
```

More beautful group labels




```
p1 <- p1 + theme(strip.background =
    element_rect(color="darkgoldenrod4",
    fill="lightgoldenrod"))
p1
```

Include ethnicity

- Previous seems to indicate there is not a "statistically significant" difference between males and females, so instead we consider ethnicity

```
reg.mod.7 <- lm(posAffect ~ agency*ethnicity +
    gender, data = dat)
summary(reg.mod.7)
```

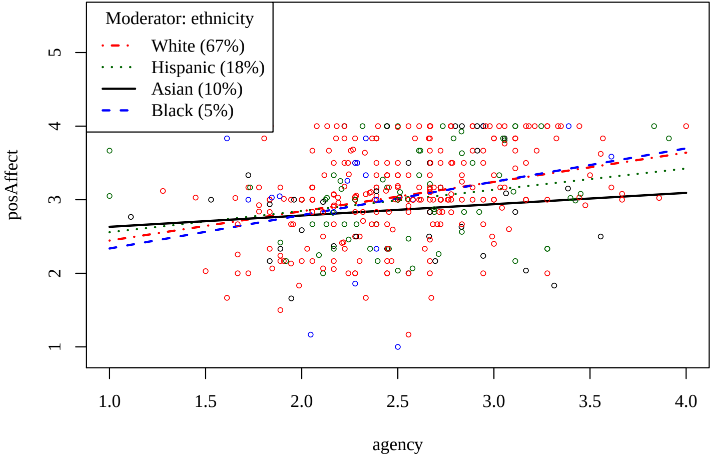
```
Call:
lm(formula = posAffect ~ agency * ethnicity + gender, data = dat)

Residuals:
     Min       1Q   Median       3Q      Max
-2.11727 -0.38900  0.04804  0.44363  1.38301

Coefficients:
                         Estimate Std. Error t value
(Intercept)               2.47790    0.47151   5.255
agency                    0.15378    0.18164   0.847
ethnicityBlack           -0.59417    0.82482  -0.720
ethnicityHispanic        -0.20871    0.59406  -0.351
ethnicityWhite           -0.43142    0.51081  -0.845
genderfemale              0.09997    0.06343   1.576
```

Include ethnicity ...

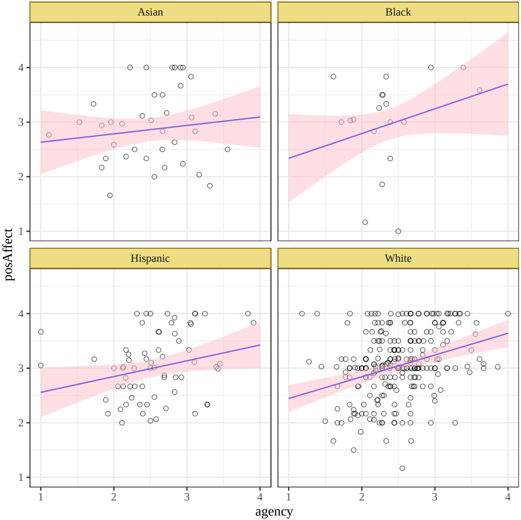
```
agency:ethnicityBlack      0.29965    0.33543   0.893
agency:ethnicityHispanic  0.13503    0.22878   0.590
agency:ethnicityWhite      0.24453    0.19813   1.234
                          Pr(>|t|)
(Intercept)                2.5e-07 ***
agency                       0.398
ethnicityBlack               0.472
ethnicityHispanic            0.726
ethnicityWhite               0.399
genderfemale                 0.116
agency:ethnicityBlack        0.372
agency:ethnicityHispanic     0.555
agency:ethnicityWhite        0.218
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6078 on 371 degrees of freedom
Multiple R-squared:  0.0933,  Adjusted R-squared:  0.07375
F-statistic: 4.772 on 8 and 371 DF,  p-value: 1.347e-05
```

- Again, this example is a little disappointing

Include ethnicity ...



ggplot trellis plot for quantile-based groups



ggplot trellis plot for quantile-based groups ...

```r
## Data must be subdivided by groups
ps71 <- plotSlopes(reg.mod.7, plotx = "agency",
   modx = "ethnicity", interval = "confidence")
ps71$newdata$posAffect <- ps71$newdata$fit
p1 <- ggplot(dat, aes(x = agency, y = posAffect))
   + geom_point(shape = 1, alpha = 0.5) +
 facet_wrap( ~ ethnicity, ncol = 2) +
    geom_line(data = ps71$newdata, color =
    "blue") +
 geom_ribbon(data = ps71$newdata, aes(ymin = lwr,
    ymax = upr), fill = "pink", alpha = 0.5) +
 theme_bw() +
 theme(strip.background =
    element_rect(color="darkgoldenrod4",
    fill="lightgoldenrod"))
p1
```

Additive Model

```r
reg.mod.10 <- lm(posAffect ~ agency + intMotiv +
    extMotiv, data = dat)
summary(reg.mod.10)
```

```
Call:
lm(formula = posAffect ~ agency + intMotiv + extMotiv, data = dat)

Residuals:
     Min       1Q   Median       3Q      Max
-1.88002 -0.35067  0.01655  0.42346  1.16862

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.85865    0.19367   9.597  < 2e-16 ***
agency       0.22583    0.06950   3.249  0.00126 **
intMotiv     0.25207    0.05110   4.932 1.22e-06 ***
extMotiv    -0.07459    0.06629  -1.125  0.26126
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5877 on 376 degrees of freedom
Multiple R-squared:  0.1409,  Adjusted R-squared:  0.1341
```

Additive Model ...

```
F-statistic: 20.56 on 3 and 376 DF,  p-value: 2.333e-12
```

Explore interactions

- Based on a comprehensive literature review and exhaustive theoretical analysis, the PI proposes an interaction between agency and extMotiv

```
reg.mod.11 <- lm(posAffect ~ intMotiv +
    agency*extMotiv, data = dat)
summary(reg.mod.11)
```

```
Call:
lm(formula = posAffect ~ intMotiv + agency * extMotiv, data = dat)

Residuals:
     Min       1Q   Median       3Q      Max
-1.88992 -0.35422  0.01966  0.42660  1.17393

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      2.75054    0.51547   5.336 1.65e-07 ***
intMotiv         0.25260    0.05094   4.959 1.08e-06 ***
agency          -0.11041    0.19305  -0.572   0.5677
extMotiv        -0.65218    0.31650  -2.061   0.0400 *
agency:extMotiv  0.21506    0.11525   1.866   0.0628 .
---
```

Explore interactions ...

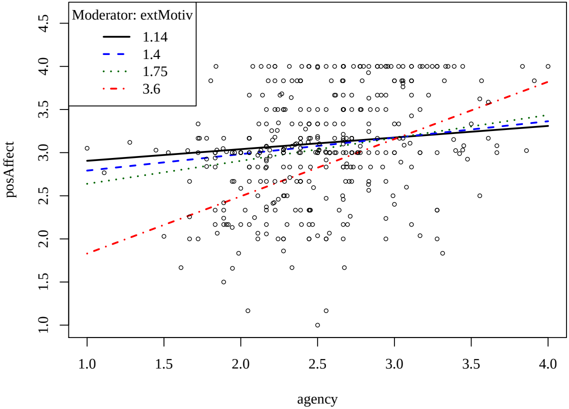
```
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5858 on 375 degrees of freedom
Multiple R-squared:  0.1488,  Adjusted R-squared:  0.1398
F-statistic: 16.39 on 4 and 375 DF,  p-value: 2.175e-12
```

Explore interactions ...

- Visualize that by choosing center points of the 4 quantiles of extMotiv

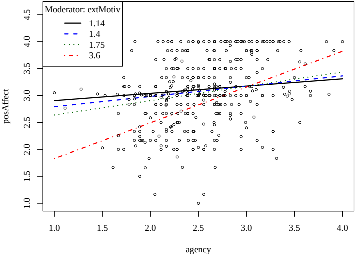
```
ps80 <- plotSlopes(reg.mod.11, plotx = "agency",
    modx = "extMotiv", modxVals = c(1.14, 1.4,
    1.75, 3.6))
```

Explore interactions ...



Follow-Up 1: The J-N Analysis

- When you looked at this



did you wonder the following:

1. It looks like the black line's slope is not different from 0, but the blue line slope certainly is.
2. That means the "statistical significance of agency depends on the value of extMotiv."

Follow-Up 1: The J-N Analysis ...

- Instead of asking "is agency significant?", an interaction modeler should as "are there values of extMotiv for which agency might be significant?"

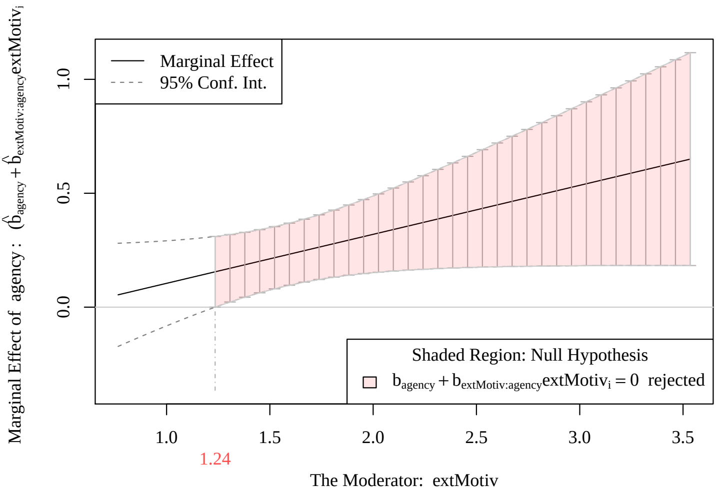
That is known as a Jersey-Neyman (J-N) hypothesis analysis.

In rockchalk, find the function "testSlopes"

```
ps80ts <- testSlopes(ps80)
```

```
Values of extMotiv INSIDE this interval:
       lo        hi
 1.235092 20.920372
cause the slope of (b1 + b2*extMotiv)agency to be statistically
    significant
```

Follow-Up 1: The J-N Analysis ...



Followup 2: Nested Model Hypo Test

- A competing research team insists that we need to check interactions with intMotiv as well. This includes all interaction terms

```
reg.mod.12 <- lm(posAffect ~ agency * intMotiv *
    extMotiv, data = dat)
summary(reg.mod.12)
```

```
Call:
lm(formula = posAffect ~ agency * intMotiv * extMotiv, data = dat)

Residuals:
     Min       1Q   Median       3Q      Max
-1.88373 -0.36143  0.03298  0.40975  1.14755

Coefficients:
                          Estimate Std. Error t value
(Intercept)                 0.2944     2.1812   0.135
agency                      1.0015     0.8970   1.117
intMotiv                    0.9717     0.6850   1.419
extMotiv                    1.1748     1.4165   0.829
agency:intMotiv            -0.3254     0.2685  -1.212
agency:extMotiv            -0.5861     0.5547  -1.057
```

Followup 2: Nested Model Hypo Test ...

```
intMotiv:extMotiv           -0.5460     0.4473  -1.221
agency:intMotiv:extMotiv     0.2386     0.1680   1.420
                         Pr(>|t|)
(Intercept)                 0.893
agency                      0.265
intMotiv                    0.157
extMotiv                    0.407
agency:intMotiv             0.226
agency:extMotiv             0.291
intMotiv:extMotiv           0.223
agency:intMotiv:extMotiv    0.156

Residual standard error: 0.5861 on 372 degrees of freedom
Multiple R-squared:  0.1548,  Adjusted R-squared:  0.1389
F-statistic: 9.735 on 7 and 372 DF,  p-value: 3.806e-11
```

- The research question is this: Is the model with more predictors better?
- These are NESTED models (the smaller one is a simplification of the larger one).

Followup 2: Nested Model Hypo Test ...

- A classical approach to test that is an F test, which examines the change in the sum-of-squares between the two models. The R team has bundled together a number of tests of that sort in the anova() function.

```
anova(reg.mod.10, reg.mod.11, reg.mod.12,  test =
    "F")
```

```
Analysis of Variance Table

Model 1: posAffect ~ agency + intMotiv + extMotiv
Model 2: posAffect ~ intMotiv + agency * extMotiv
Model 3: posAffect ~ agency * intMotiv * extMotiv
  Res.Df    RSS Df Sum of Sq      F  Pr(>F)
1    376 129.87
2    375 128.67  1   1.19479 3.4787 0.06295 .
3    372 127.77  3   0.90459 0.8779 0.45261
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- The comparison of models 1 and 2 is statistically significant, meaning we should keep the additional coefficients in the model

Followup 2: Nested Model Hypo Test ...

- The comparison of models 2 and 3 is not. So the enemy research team was wrong.

Centering and Standardizing

- We notice that many in psychology enjoy "standardized regression" or "mean-centered" regressions.
- Can do "manually", but I do that so often while teaching I created shortcuts.
- Do you want a model in which all numeric variables are centered at their means? The `meanCenter` function defaults to only change variables involved in interactions

```
## The mean-centered model sets the predictors at
    (x - xmean)
reg.mod.14 <- meanCenter(reg.mod.11)
summary(reg.mod.14)
```

Centering and Standardizing ...

```
These variables were mean-centered before any transformations were made
     on the design matrix.
[1] "agencyc"   "extMotivc"
The centers and scale factors were
       agencyc extMotivc
mean  2.511814  1.644807
scale 1.000000  1.000000
The summary statistics of the variables in the design matrix (after
     centering).
                      mean  std.dev.
posAffect         3.065193 0.6315674
intMotiv          3.022962 0.6607460
agencyc           0.000000 0.5008115
extMotivc         0.000000 0.4760930
agencyc:extMotivc 0.058399 0.2740184

The following results were produced from:
meanCenter.default(model = reg.mod.11)

Call:
lm(formula = posAffect ~ intMotiv + agencyc * extMotivc, data = stddat)

Residuals:
     Min       1Q   Median       3Q      Max
-1.88992 -0.35422  0.01966  0.42660  1.17393
```

Centering and Standardizing ...

```
Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)         2.28902    0.15707  14.574  < 2e-16 ***
intMotiv            0.25260    0.05094   4.959 1.08e-06 ***
agencyc             0.24333    0.06990   3.481 0.000559 ***
extMotivc          -0.11198    0.06905  -1.622 0.105687
agencyc:extMotivc   0.21506    0.11525   1.866 0.062818 .
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5858 on 375 degrees of freedom
Multiple R-squared:  0.1488,  Adjusted R-squared:  0.1398
F-statistic: 16.39 on 4 and 375 DF,  p-value: 2.175e-12
```

- Change centerOnlyInteractors = FALSE

```
reg.mod.14b <- meanCenter(reg.mod.11,
    centerOnlyInteractors = FALSE)
summary(reg.mod.14b)
```

Centering and Standardizing ...

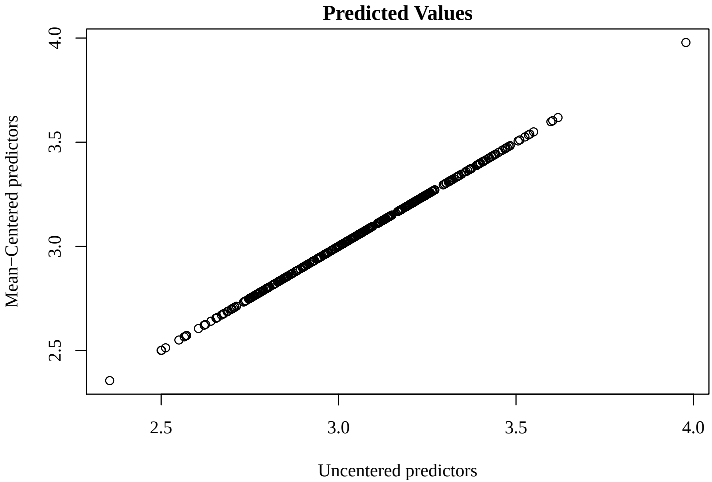
```
These variables were mean-centered before any transformations were made
     on the design matrix.
[1] "intMotivc" "agencyc"   "extMotivc"
The centers and scale factors were
      intMotivc  agencyc extMotivc
mean   3.022962 2.511814  1.644807
scale  1.000000 1.000000  1.000000
The summary statistics of the variables in the design matrix (after
     centering).
                         mean   std.dev.
posAffect            3.065193 0.6315674
intMotivc            0.000000 0.6607460
agencyc              0.000000 0.5008115
extMotivc            0.000000 0.4760930
agencyc:extMotivc    0.058399 0.2740184

The following results were produced from:
meanCenter.default(model = reg.mod.11, centerOnlyInteractors = FALSE)

Call:
lm(formula = posAffect ~ intMotivc + agencyc * extMotivc, data = stddat)

Residuals:
     Min        1Q    Median        3Q       Max
-1.88992 -0.35422   0.01966   0.42660   1.17393
```

Centering and Standardizing ...

```
Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)         3.05263    0.03079  99.131  < 2e-16 ***
intMotivc           0.25260    0.05094   4.959 1.08e-06 ***
agencyc             0.24333    0.06990   3.481 0.000559 ***
extMotivc          -0.11198    0.06905  -1.622 0.105687
agencyc:extMotivc   0.21506    0.11525   1.866 0.062818 .
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5858 on 375 degrees of freedom
Multiple R-squared:  0.1488,  Adjusted R-squared:  0.1398
F-statistic: 16.39 on 4 and 375 DF,  p-value: 2.175e-12
```

- The coefficients hop about because of the transformation, but don't let anybody fool you. The mean-centered regression is absolutely identical to the un-centered one. Note the predicted values are identical

Centering and Standardizing ...



Centering and Standardizing ...

```
plot(fitted(reg.mod.14), fitted(reg.mod.11), xlab
    = "Uncentered predictors", ylab =
    "Mean-Centered predictors", main = "Predicted
    Values")
```

- The results have “seemed” different to some authors.

Outline

Modern Applied Statistics

- The famous book by Wm. Venables and Brian Ripley, *Modern Applied Statistics with S*, advances the theme that statistical analysis has entered a new phase characterized the idea that
  - We "interact" with estimated objects (rather than just printing output about them)
- These notes focus on linear regression modeling
- SPSS & SAS users should notice the difference, because R makes it possible to "see inside" output objects and interrogate them in various ways

Other regression functions

- R base also includes
  - glm: generalized linear models (logit, probit, poisson, gamma)
- Recommended packages include additional regression functions
  - MASS: negative binomial, Box-Cox transformation
  - mgcv: generalized additive models (smoothing functions for "wiggly" model fits)
  - rpart: partitioned regression trees
- Other contributed packages add many models, many of which are written in a similar style.

References

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Session

```
sessionInfo()
```

```
R version 3.6.0 (2019-04-26)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 19.04

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/atlas/libblas.so.3.10.3
LAPACK: /usr/lib/x86_64-linux-gnu/atlas/liblapack.so.3.10.3

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] ggplot2_3.2.0     car_3.0-2         carData_3.0-2
    rockchalk_1.8.144
```

Session ...

```
loaded via a namespace (and not attached):
 [1] lavaan_0.6-3       tidyselect_0.2.5   purrr_0.3.2
     reshape2_1.4.3
 [5] splines_3.6.0      haven_2.1.0        lattice_0.20-38
     colorspace_1.4-1
 [9] stats4_3.6.0       rlang_0.4.0        nloptr_1.2.1       pillar_1.4.2
[13] foreign_0.8-71     glue_1.3.1         withr_2.1.2        readxl_1.3.1
[17] plyr_1.8.4         stringr_1.4.0      munsell_0.5.0      gtable_0.3.0
[21] cellranger_1.1.0   zip_2.0.2          kutils_1.69        labeling_0.3
[25] rio_0.5.16         forcats_0.4.0      curl_3.3           Rcpp_1.0.1
[29] xtable_1.8-4       scales_1.0.0       abind_1.4-5        lme4_1.1-21
[33] mnormt_1.5-5       hms_0.4.2          stringi_1.4.3
     openxlsx_4.1.0
[37] dplyr_0.8.3        grid_3.6.0         tools_3.6.0        magrittr_1.5
[41] lazyeval_0.2.2     tibble_2.1.3       crayon_1.3.4
     pbivnorm_0.6.0
[45] pkgconfig_2.0.2    MASS_7.3-51.4      Matrix_1.2-17
     data.table_1.12.2
[49] assertthat_0.2.1   minqa_1.2.4        R6_2.4.0           boot_1.3-22
[53] nlme_3.1-140       compiler_3.6.0
```