Intro to generalized linear models in R

Rose Hartman

Office of the Vice President for Research and Innovation
Center for Assessment Statistics & Evaluation

May 18th, 2016

Workshop Overview

1. Background and overview
   - Why do we need generalized linear models?
   - Tips for presenting model results
2. Example: Logistic regression
   - Prepping the data
   - Running the model
   - Summarizing the results
   - Plotting
   - Additional predictors
3. Other glms

This workshop

- Focus on the R code rather than the stats — if you'd like to learn about the stats behind the general linear model more deeply, I can recommend several excellent classes and texts.

This workshop

- Focus on the R code rather than the stats — if you'd like to learn about the stats behind the general linear model more deeply, I can recommend several excellent classes and texts.
- Lots of practice. Learn R by using R!

This workshop

- Focus on the R code rather than the stats — if you'd like to learn about the stats behind the general linear model more deeply, I can recommend several excellent classes and texts.
- Lots of practice. Learn R by using R!
- We'll be using ggplot2 for plotting and dplyr for data wrangling

This workshop

- Focus on the R code rather than the stats — if you'd like to learn about the stats behind the general linear model more deeply, I can recommend several excellent classes and texts.
- Lots of practice. Learn R by using R!
- We'll be using ggplot2 for plotting and dplyr for data wrangling
- Color coded content to help you keep track of the important stuff

Keep an eye out for...

Keep an eye out for...

How would you do this?

Keep an eye out for...

How would you do this?

What will this do?

Keep an eye out for...

How would you do this?

What will this do?

Learn more: resources to check out

Keep an eye out for...

How would you do this?

What will this do?

Learn more: resources to check out

Key idea: the big ideas you need to hold on to

Keep an eye out for...

How would you do this?

What will this do?

Learn more: resources to check out

Key idea: the big ideas you need to hold on to

See also: other functions or packages that do a similar thing

If you don't already have ggplot2 installed, do that now:

```
install.packages("ggplot2")
install.packages("dplyr")

library("ggplot2")
library("dplyr")
```

Why do we need generalized linear models?

Linear models are only appropriate when your outcome variable(s) are continuous and unbounded.
If you want to model a categorical outcome, or one that's bounded (like a proportion or percent), then usually can't rely on regular linear modeling to get the job done.

Why do we need generalized linear models?

We can coerce problematic outcome variables into nice, continuous ones so that we can still model them with a line.
The strategy for transforming your outcome depends on what its distribution is like to begin with — if it's binary (or bounded at 0 and 1), then a logit works well. If it's count data, then a log is the right transformation, etc.

Why do we need generalized linear models?

We can coerce problematic outcome variables into nice, continuous ones so that we can still model them with a line.
The strategy for transforming your outcome depends on what its distribution is like to begin with — if it's binary (or bounded at 0 and 1), then a logit works well. If it's count data, then a log is the right transformation, etc.

Key idea: Generalized linear models are just linear models on data that's been transformed with a link function.

Special considerations for generalized linear models

- Unlike linear models, there is not a single analytic solution to glms

Special considerations for generalized linear models

- Unlike linear models, there is not a single analytic solution to glms
- The computer uses brute force to find a solution, iterating over tons of combinations of parameter estimates and seeing which gives the bets model fit

Special considerations for generalized linear models

- Unlike linear models, there is not a single analytic solution to glms
- The computer uses brute force to find a solution, iterating over tons of combinations of parameter estimates and seeing which gives the bets model fit
- This means the results are not exact — they depend on the sampling algorithm being used — so you might see very small changes if you run the model again, or if you run it on other software

Special considerations for generalized linear models

- Unlike linear models, there is not a single analytic solution to glms
- The computer uses brute force to find a solution, iterating over tons of combinations of parameter estimates and seeing which gives the bets model fit
- This means the results are not exact — they depend on the sampling algorithm being used — so you might see very small changes if you run the model again, or if you run it on other software
- There's a risk that your model won't converge if it's too complicated and you don't have enough data

Overview

1. Background and overview
   - Why do we need generalized linear models?
   - Tips for presenting model results
2. Example: Logistic regression
   - Prepping the data
   - Running the model
   - Summarizing the results
   - Plotting
   - Additional predictors
3. Other glms

Building dynamic documents

Let's use an .rmd file to keep track of all of the steps in the analysis and prepare the results. If you don't already have knitr installed, get it now.

```
install.packages("knitr")
library("knitr")
```

Building dynamic documents

Let's use an .rmd file to keep track of all of the steps in the analysis and prepare the results. If you don't already have knitr installed, get it now.

```
install.packages("knitr")
library("knitr")
```

You can use knitr tools built right into R Studio to write r-markdown documents that include both code and text, and then "knit" them up into .docx, .pdf, or .html.
This makes it easy to get your R output into (close to) the table formatting you'll need in your document.

Building dynamic documents

Markdown is a super simple language for formatting text. It can't do much, but it handles most of what you'll need for a basic document, and it's really quick to learn.

Building dynamic documents

Markdown is a super simple language for formatting text. It can't do much, but it handles most of what you'll need for a basic document, and it's really quick to learn.

Learn more: `https://daringfireball.net/projects/markdown/basics`

Building dynamic documents

Markdown is a super simple language for formatting text. It can't do much, but it handles most of what you'll need for a basic document, and it's really quick to learn.

Learn more: `https://daringfireball.net/projects/markdown/basics`

Try opening a new .rmd file in R Studio now!

Building dynamic documents

We'll use pander to automatically format model summaries into lovely tables.

```
install.packages("pander")
```

Overview

1 Background and overview
- Why do we need generalized linear models?
- Tips for presenting model results

2 Example: Logistic regression
- Prepping the data
- Running the model
- Summarizing the results
- Plotting
- Additional predictors

3 Other glms

Getting your data into R

Check your working directory:

```
getwd()
```

Getting your data into R

Check your working directory:

```
getwd()
```

If you want R to find something on your computer, you have three options:

1. Put the file in R's working directory
2. Move R's working directory to where ever the file is saved using setwd()
3. Specify the file path when you tell R to look for the file

Getting your data into R

Check your working directory:

```
getwd()
```

If you want R to find something on your computer, you have three options:

1. Put the file in R's working directory
2. Move R's working directory to where ever the file is saved using setwd()
3. Specify the file path when you tell R to look for the file

See also: You can also use the menu options in R Studio to change R's working directory.

Getting your data into R

I'll use option 3, specifying the file path for the file when I tell R to read it in.

Getting your data into R

I'll use option 3, specifying the file path for the file when I tell R to read it in. Find the file on your computer, get its location, and add

that file path to your read_sav command:

```
osf <- read.csv("data/OSF_badges.csv")
```

Getting your data into R

I'll use option 3, specifying the file path for the file when I tell R to read it in. Find the file on your computer, get its location, and add

that file path to your read_sav command:

```
osf <- read.csv("data/OSF_badges.csv")
```

Key idea: Remember, if your data file isn't in R's working directory you need to tell R where to look for it.

Check your data

These are data from

```
str(osf)
head(osf)
summary(osf)
```

Check your data

These are data from

```
str(osf)
head(osf)
summary(osf)
```

```
osf$date <- as.Date(osf$date)
str(osf)
```

Check your data

These are data from

```
str(osf)
head(osf)
summary(osf)
```

```
osf$date <- as.Date(osf$date)
str(osf)
```

Key idea: Always check your data frame using str() and View() or head() before you run any tests.

Check your data

We'll want to test our glm against a null model, which means any missing data on our predictor (Year) will be problematic — it means the null model and the test model won't be on the same data.

```
?na.omit
```

```
osf.lgt <- osf %>%
  select(statement.included, date, Journal) %>%
  filter(Journal != "Infant behavior & development") %>%
  na.omit()

summary(osf.lgt)
```

Overview

1. Background and overview
   - Why do we need generalized linear models?
   - Tips for presenting model results
2. Example: Logistic regression
   - Prepping the data
   - Running the model
   - Summarizing the results
   - Plotting
   - Additional predictors
3. Other glms

glm()

```
?glm
```

Writing the formula

Did it become more likely for articles to provide a data statement (statement.included) over time (Year)?

Writing the formula

Did it become more likely for articles to provide a data statement (statement.included) over time (Year)? In modeling functions in R,

the model is usually provided as a formula, like: outcome predictor1 + predictor2 etc. How should we specify the formula for this test?

Writing the formula

Did it become more likely for articles to provide a data statement (statement.included) over time (Year)? In modeling functions in R,

the model is usually provided as a formula, like: outcome predictor1 + predictor2 etc. How should we specify the formula for this test?

```
glm(statement.included ~ date, data=osf.lgt,
     family=binomial(link="logit"),
     na.action=na.exclude)
```

Link functions

```
?binomial
```

The model

Let's run the model.

```
logit.model1 <- glm(statement.included ~ date, data=osf.lgt,
                    family=binomial(link="logit"),
                    na.action=na.exclude)
```

The model

Let's look at that object we just created:

```
logit.model1

str(logit.model1)
```

Overview

1 Background and overview
- Why do we need generalized linear models?
- Tips for presenting model results

2 Example: Logistic regression
- Prepping the data
- Running the model
- Summarizing the results
- Plotting
- Additional predictors

3 Other glms

Summarize the model

```
summary(logit.model1)
```

Summarize the model

```
summary(logit.model1)
```

Note that R prints the results such that the first level of the outcome variable is a failure, so these results articulate the probability of the second level of statement.included happening. To see the order of the levels in the data frame, use levels():

```
levels(osf.lgt$statement.included)
```

Summarize the model

add this code to your .rmd file, and make sure you've set your knitr options to results='asis'

```
model.sum <- summary(logit.model1)
pander(model.sum)
```

Summarize the model

add this code to your .rmd file, and make sure you've set your knitr options to results='asis'

```
model.sum <- summary(logit.model1)
pander(model.sum)
```

See also: the stargazer package, which makes lovely model tables and works for knitting to pdf but not Word

Summarize the model

add this code to your .rmd file, and make sure you've set your knitr options to results='asis'

```
model.sum <- summary(logit.model1)
pander(model.sum)
```

See also: the stargazer package, which makes lovely model tables and works for knitting to pdf but not Word

See also: knitr has a function for making tables, kable(), but it doesn't work for model summaries

Testing a series of models

It's typical to test a series of glms, beginning with a null model (no predictors).

```
logit.model0 <- glm(statement.included ~ 1, data=osf.lgt,
                    family=binomial(link="logit"),
                    na.action=na.exclude)
```

Testing a series of models

It's typical to test a series of glms, beginning with a null model (no predictors).

```
logit.model0 <- glm(statement.included ~ 1, data=osf.lgt,
                    family=binomial(link="logit"),
                    na.action=na.exclude)
```

Use anova() to test for a significant improvement in model fit from the null model to our test model:

```
anova(logit.model0, logit.model1, test="Chisq")
```

Since we want to test for a significant change in model fit (deviance), we'll used a chi-squared test.

Classification tables

For logistic regression, it's nice to have a classification table showing your model's predicted outcomes compared to the actual outcome.

Classification tables

For logistic regression, it's nice to have a classification table showing your model's predicted outcomes compared to the actual outcome.

```
osf.lgt$pred1 <- predict(logit.model1,
                                 osf.lgt,
                                 type="response")

osf.lgt$pred0 <- predict(logit.model0,
                                  osf.lgt,
                                  type="response")
```

Note that if you leave out the type="response" argument, or enter type="link" instead, it will give you predicted logits instead of predicted probabilities.

Classification tables

Set a probability cutoff (let's use .5). Probabilities above this will be classified as "yes", and below will be classified as "no".

```
osf.lgt$clas0 <- ifelse(osf.lgt$pred0 >= .5, 1,
                        ifelse(osf.lgt$pred0 < .5, 0,
                               NA))
osf.lgt$clas1 <- ifelse(osf.lgt$pred1 >= .5, 1,
                        ifelse(osf.lgt$pred1 < .5, 0,
                               NA))
```

Classification tables

Convert the new classification variables to factors, for cleaner output in the crosstabs.

```
osf.lgt$clas0 <- factor(osf.lgt$clas0,
                                levels=c(1,0),
                                labels=c("yes", "no"))
osf.lgt$clas1 <- factor(osf.lgt$clas1,
                                levels=c(1,0),
                                labels=c("yes", "no"))
```

Classification tables

Convert the new classification variables to factors, for cleaner output in the crosstabs.

```
osf.lgt$clas0 <- factor(osf.lgt$clas0,
                                levels=c(1,0),
                                labels=c("yes", "no"))
osf.lgt$clas1 <- factor(osf.lgt$clas1,
                                levels=c(1,0),
                                labels=c("yes", "no"))
```

```
xtabs(~ statement.included + clas0, data=osf.lgt)

xtabs(~ statement.included + clas1, data=osf.lgt)
```

Classification tables

Convert the new classification variables to factors, for cleaner output in the crosstabs.

```
osf.lgt$clas0 <- factor(osf.lgt$clas0,
                                levels=c(1,0),
                                labels=c("yes", "no"))
osf.lgt$clas1 <- factor(osf.lgt$clas1,
                                levels=c(1,0),
                                labels=c("yes", "no"))
```

```
xtabs(~ statement.included + clas0, data=osf.lgt)

xtabs(~ statement.included + clas1, data=osf.lgt)
```

See also: gmodels::CrossTable(), which makes fancier crosstabs

Overview

1. Background and overview
   - Why do we need generalized linear models?
   - Tips for presenting model results
2. Example: Logistic regression
   - Prepping the data
   - Running the model
   - Summarizing the results
   - Plotting
   - Additional predictors
3. Other glms

Plotting

The relationship between publication date and whether there was a data statement available

```
ggplot(osf.lgt, aes(x=date, y=statement.included)) +
  geom_point(alpha=.3)
```

Plotting the line from your model

Note that R counts factor levels starting at 1, so "no" and "yes" are plotted at 1 and 2 respectively. To line them up with the predicted values from the model, we need to subtract 1.

Plotting the line from your model

Note that R counts factor levels starting at 1, so "no" and "yes" are plotted at 1 and 2 respectively. To line them up with the predicted values from the model, we need to subtract 1.

```
ggplot(osf.lgt, aes(x=date, y=as.numeric(statement.included)-1)) +
  geom_point( alpha=.3 ) +
  geom_line( aes(y=pred, x=date) ) +
  labs(y="Probability of providing a data statement")
```

Additional resources for learning ggplot2

- Read Jenny Bryan's ggplot tutorials — tons of great examples and code! Click on the files that have the file extension .md (those will be the easiest to read)
  `https://github.com/jennybc/ggplot2-tutorial`
- All of the geoms, with pictures
  `http://docs.ggplot2.org/current/`
- For more in-depth material on ggplot2, see the resources at
  `http://ggplot2.org/`

Additional resources for learning ggplot2

- Read Jenny Bryan's ggplot tutorials — tons of great examples and code! Click on the files that have the file extension .md (those will be the easiest to read)
  `https://github.com/jennybc/ggplot2-tutorial`
- All of the geoms, with pictures
  `http://docs.ggplot2.org/current/`
- For more in-depth material on ggplot2, see the resources at
  `http://ggplot2.org/`

Learn more: For a tutorial on controlling colors in ggplot, see an old R Club post of mine: `http://blogs.uoregon.edu/rclub/2015/02/17/picking-pretty-plot-palates/`

The model

What will this do?
What question would this model test?

```
logit.model2 <- glm(statement.included ~ date*Journal, data=osf.lgt2,
                    family=binomial(link="logit"),
                    na.action=na.exclude)
```

Testing against simpler models

```
anova(logit.model0, logit.model1, logit.model2, test="Chisq")
```

Getting predicted values

```
osf.lgt$pred2 <- predict(logit.model2,
                         osf.lgt,
                         type="response")
osf.lgt$clas2 <- ifelse(osf.lgt$pred2 >= .5, 1,
                         ifelse(osf.lgt$pred2 < .5, 0,
                                NA))
osf.lgt$clas2 <- factor(osf.lgt$clas2,
                         levels=c(1,0),
                         labels=c("yes", "no"))

xtabs(~ statement.included + clas2, data=osf.lgt)
```

Plotting

```
ggplot(osf.lgt, aes(x=date, y=as.numeric(statement.included)-1,
                    color=Journal)) +
  geom_point( alpha=.3 ) +
  geom_line( aes(y=pred2, x=date) ) +
  labs(y="Probability of providing a data statement")
```

Other glms

Let's say you wanted to see if the number of experiments reported in each article varies by journal.

Other glms

Let's say you wanted to see if the number of experiments reported in each article varies by journal.

```
summary(osf$Number.of.experiments)

hist(osf$Number.of.experiments)
```

Other glms

Let's say you wanted to see if the number of experiments reported in each article varies by journal.

```
summary(osf$Number.of.experiments)

hist(osf$Number.of.experiments)
```

There are a couple reasonable options for a link function here. Let's go with Poisson.

Other glms

Let's say you wanted to see if the number of experiments reported in each article varies by journal.

```
summary(osf$Number.of.experiments)

hist(osf$Number.of.experiments)
```

There are a couple reasonable options for a link function here. Let's go with Poisson.

```
osf.pois <- osf %>%
  select(Number.of.experiments, Journal) %>%
  filter(Journal != "Infant behavior & development") %>%
  na.omit()
```

Other glms

How would you do this? Write the glm call to model number of experiments by journal, using a Poisson link.

```
pois.model <- glm(Number.of.experiments ~ Journal, data=osf.pois,
                  family=poisson(link = "log"),
                  na.action=na.exclude)
```

Other glms

```
ggplot(osf.pois, aes(x=Number.of.experiments)) +
  geom_histogram()
```

Other glms

```
ggplot(osf.pois, aes(x=Number.of.experiments)) +
  geom_histogram()
```

Show Journal info as well

```
ggplot(osf.pois, aes(x=Number.of.experiments, fill=Journal)) +
  geom_histogram()

ggplot(osf.pois, aes(x=Number.of.experiments, fill=Journal)) +
  geom_density(alpha=.3, adjust=2)
```

Other glms

```
ggplot(osf.pois, aes(x=Number.of.experiments)) +
  geom_histogram()
```

Show Journal info as well

```
ggplot(osf.pois, aes(x=Number.of.experiments, fill=Journal)) +
  geom_histogram()

ggplot(osf.pois, aes(x=Number.of.experiments, fill=Journal)) +
  geom_density(alpha=.3, adjust=2)
```

Try adding facet wrap by Journal as well, to put each density plot in its own facet.