REVOLUTION ANALYTICS

Generalized Linear Models with ScaleR

Revolution Analytics

Goals and Agenda

We will cover the Generalized Linear Model in this session.

We will focus on how to use the `rxGlm()` function, and we will review how to interact with results once they have been estimated.

Directory config

```
## dir config
big.data.path <- Sys.getenv("ACADEMYR_BIG_DATA_PATH")
if (big.data.path == "") {
  Sys.setenv(ACADEMYR_BIG_DATA_PATH = "/usr/share/BigData")
  big.data.path <- Sys.getenv("ACADEMYR_BIG_DATA_PATH")
}
data.path <- "../data"
output.path <- "../output/xdf"
if (!file.exists(output.path)) dir.create(output.path, recursive = TRUE)
sample.data.dir <- rxOptions()[["sampleDataDir"]]
```

Outline

What is Generalized Linear Modeling?

Generalized linear models (GLMs) are a framework for a wide range of analyses.

They relax the assumptions for a standard linear model in two ways.

1. First, a functional form can be specified for the conditional mean of the predictor. This is referred to as the *link* function.
2. Second, you can specify a distribution for the response variable.

The `rxGlm()` function in `RevoScaleR` provides the ability to estimate generalized linear models on large data sets.

rxGlm Usage

```
args(rxGlm)
```

```
## function (formula, data, family = gaussian(), pweights = NULL,
##     fweights = NULL, offset = NULL, cube = FALSE, variableSelection = list(),
##     rowSelection = NULL, transforms = NULL, transformObjects = NULL,
##     transformFunc = NULL, transformVars = NULL, transformPackages = NULL,
##     transformEnvir = NULL, dropFirst = FALSE, dropMain = rxGetOption("dropMain"),
##     covCoef = FALSE, computeAIC = FALSE, initialValues = NA,
##   coefLabelStyle = rxGetOption("coefLabelStyle"), blocksPerRead = rxGetOption("blocksPerRead"),
...
```

family argument

Main Difference: `family` argument

GLMs let us specify a functional form for the conditional mean of the predictor and a distribution for the response variable. We use the `family` argument to specify both.

rxGlm() families

Any valid R family object that can be used with `glm()` can be used with `rxGlm()`, including those that are user-defined. Some common familes / links are:

binomial	logit
binomial	probit
poisson	log
Gamma	inverse

`rxGlm()` implementation detail

The following family / link combinations are implemented in `C++` for performance enhancements:

- binomial / logit
- gamma / log, poisson / log
- Tweedie

Others are implemented in a combination of `C` and `R`

rxGlm example

In this example, you use a subsample from the 5% sample of the U.S. 2000 census.

- Consider the annual cost of property insurance for heads of household ages 21 through 89, and its relationship to age, sex, and region.
- A column `perwt` in the data set represents the probability weight for that observation.

Subsetting the Census Data

We won't be working with all 265 variables or 14 million observations.

Create a working file, `Census5PCT2000_insurance.xdf`, for this analysis.

```
census2000 <- file.path(big.data.path, "Census5PCT2000.xdf")
census_insurance <- file.path(output.path, "Census5PCT2000_insurance.xdf")
if (!file.exists(census_insurance)) {
  rxDataStep(inData = census2000, outFile = census_insurance, rowSelection = (related ==
    "Head/Householder") & (age > 20) & (age < 90), varsToKeep = c("propinsr",
    "age", "sex", "region", "perwt"), blocksPerRead = 1, overwrite = TRUE)
}
```

Check the subset

```
rxGetVarInfo(census_insurance)
```

```
## Var 1: propinsr, Annual property insurance cost
##        Type: integer, Low/High: (0, 3700)
## Var 2: age, Age
##        Type: integer, Low/High: (21, 89)
## Var 3: sex, Sex
##        2 factor levels: Male Female
## Var 4: region, Census region and division
...
```

Examining the Regions

Let's do one more step in data cleaning. The variable `region` has some very long factor level character strings, and it also has a number of levels for which there are no observations.

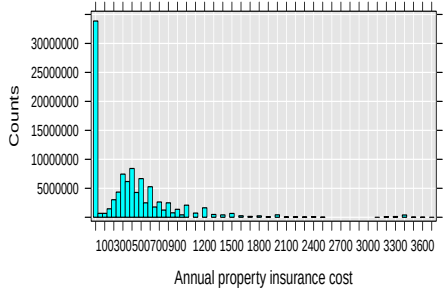
```
p <- rxSummary(~region, data = census_insurance)
print(p, head = FALSE)
```

```
##
## Category Counts for region
## Number of categories: 17
## Number of valid observations: 5175270
## Number of missing observations: 0
##
##  region                                          Counts
...
```

Checking the Distribution

As a first step in our analysis, look at a histogram of property insurance cost.

```
rxHistogram(~propinsr, data = census_insurance, pweights = "perwt")
```



Tweedie Distributions

A random variable Y is said to have a Tweedie distribution if its variance is related to it's mean in the following fashion:

$$\sigma^2 = a\mu^p$$

where σ^2 is its variance and μ corresponds to that variable's expected value.

Tweedie Distributions

This definition encompasses a lot of well known distributions:

Distribution	p	Function	Description
Normal	0	$\sigma^2 = a$	Variance and Mean are independent
Poisson	1	$\sigma^2 = \mu$	Variance = Mean
Gamma	2	$\sigma^2 = \frac{\mu^2}{k}$	Variance = Mean^2 / shape
Inverse Gaussian	3	$\sigma^2 = \frac{\mu^3}{\lambda}$	Variance = Mean^3 / shape

Another Set of Distributions

There are also another class of distributions that are defined by situations in which $ 1 < p < 2 $.

With $[1 < p < 2]$, the Tweedie family characterizes a compound Possion-gamma distribution:

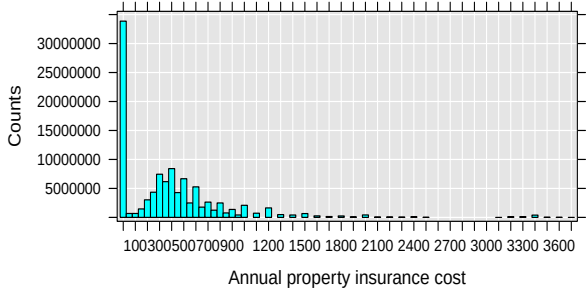
$$Y \sim \Sigma_{i=1}^{T} X_i$$

$$T \sim Poisson(\lambda)$$

$$X_i \sim Gamma(k, \Theta)$$

Utility

These are very useful for estimating models in which there is substantial density at an exact value of 0, and the rest of the distribution is positive. Which corresponds to our data!



Estimating a Model

Let's estimate a model using a Tweedie family.

```
propinGlm <- rxGlm(propinsr ~ sex + F(age) + region, pweights = "perwt",
  data = census_insurance, family = rxTweedie(var.power = 1.5),
  dropFirst = TRUE)

p <- summary(propinGlm)
print(p, head = FALSE)
```

```
## Coefficients:
##                                                  Estimate
## (Intercept)                                   0.12313163
## sex=Male                                         Dropped
## sex=Female                                    0.00902634
## F_age=21                                         Dropped
## F_age=22                                     -0.00920814
...
```

Exercise

We would typically use `rxLogit()` to estimate a logistic regression.

In order to get used to the `family` argument, use `rxGlm` to estimate a logistic regression for the `mortDefaultSmall.xdf` predicting `default` by `year` (as a factor), `ccDebt`, `creditScore`, `houseAge`, and `yearsEmploy`.

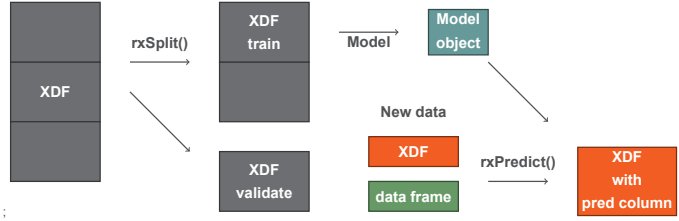
Include the computation of the covariance matrix of the coefficients.

```
mortgages <- file.path(sample.data.dir, "mortDefaultSmall.xdf")
```

Outline

Predicting with new data

Using a new data set for the prediction



Examine the results of the estimated model by looking at predicted values for a set of explanatory characteristics.

Creating a prediction data set

Create a prediction dataset for the `South Atlantic` region for all ages and sexes and a similar prediction dataset for the `Middle Atlantic` region.

```
varInfo <- rxGetVarInfo(census_insurance)
regionLabels <- varInfo$region$levels
ages <- 21:89
regionIndices <- c(grep("South Atlantic", regionLabels), grep("Middle Atlantic",
  regionLabels))
predData <- data.frame(age = rep(ages, times = 4), sex = gl(n = 2,
  k = length(ages), length = length(ages) * 2 * 2, labels = c("Male",
    "Female")), region = factor(rep(regionIndices, each = length(ages) *
  2), levels = 1:length(regionLabels), labels = regionLabels))
```

View the prediction data set

View the new datasets:

```
head(predData, 3)
```

```
##   age  sex                  region
## 1  21 Male South Atlantic Division
## 2  22 Male South Atlantic Division
## 3  23 Male South Atlantic Division
```

```
tail(predData, 3)
```

```
##     age    sex                  region
## 274  87 Female Middle Atlantic Division
## 275  88 Female Middle Atlantic Division
## 276  89 Female Middle Atlantic Division
```

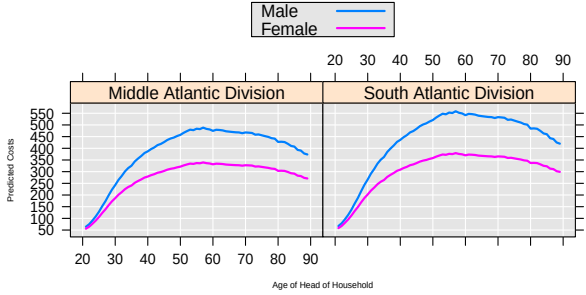
Making the Prediction

```
outData <- rxPredict(propinGlm, data = predData)
head(outData)
```

```
##   propinsr_Pred
## 1      67.29176
## 2      78.73753
## 3      95.92762
## 4     114.75933
## 5     137.16096
## 6     163.53119
```

Visualization

Last, we can combine the predicted values with our prediction data frame and plot:



More on Prediction

```
args(rxPredict)
```

```
## function (modelObject, data = NULL, ...)
## NULL
```

Argument: `type`

response produces predictions on the same scale as your predictor

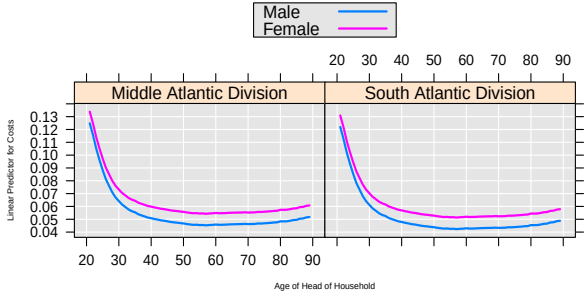
link Produces predictions on the scale of the linear predictor

Predicting `link`

```
outDataLink <- rxPredict(propinGlm, data = predData, type = "link")
head(outDataLink)
```

```
##   propinsr_Pred
## 1    0.12190431
## 2    0.11269616
## 3    0.10210057
## 4    0.09334821
## 5    0.08538562
## 6    0.07819873
```

Plotting the Linear Predictor



Exercise

Practice generating predictions by creating a new `XDF` file that contains predictions for the `mortgages` data. Do the following:

- Include the model variables in the new file
- Generate `response` predictions, and name those predictions `default_rxGlmPredResponse`,
- Compute residuals, standard errors, and 99% confidence intervals.

Summary

- Use `rxGlm` to estimate a generalized linear model.
- Key argument is `family`.
- Use `rxPredict` to generate predicted values.

Very similar usage to `rxLinMod` and `rxLogit`

Questions?

REVOLUTION ANALYTICS

Thank you

Revolution Analytics is the leading commercial provider of software and support for the popular open source R statistics language.

www.revolutionanalytics.com
1.855.GET.REVO
Twitter: @RevolutionR