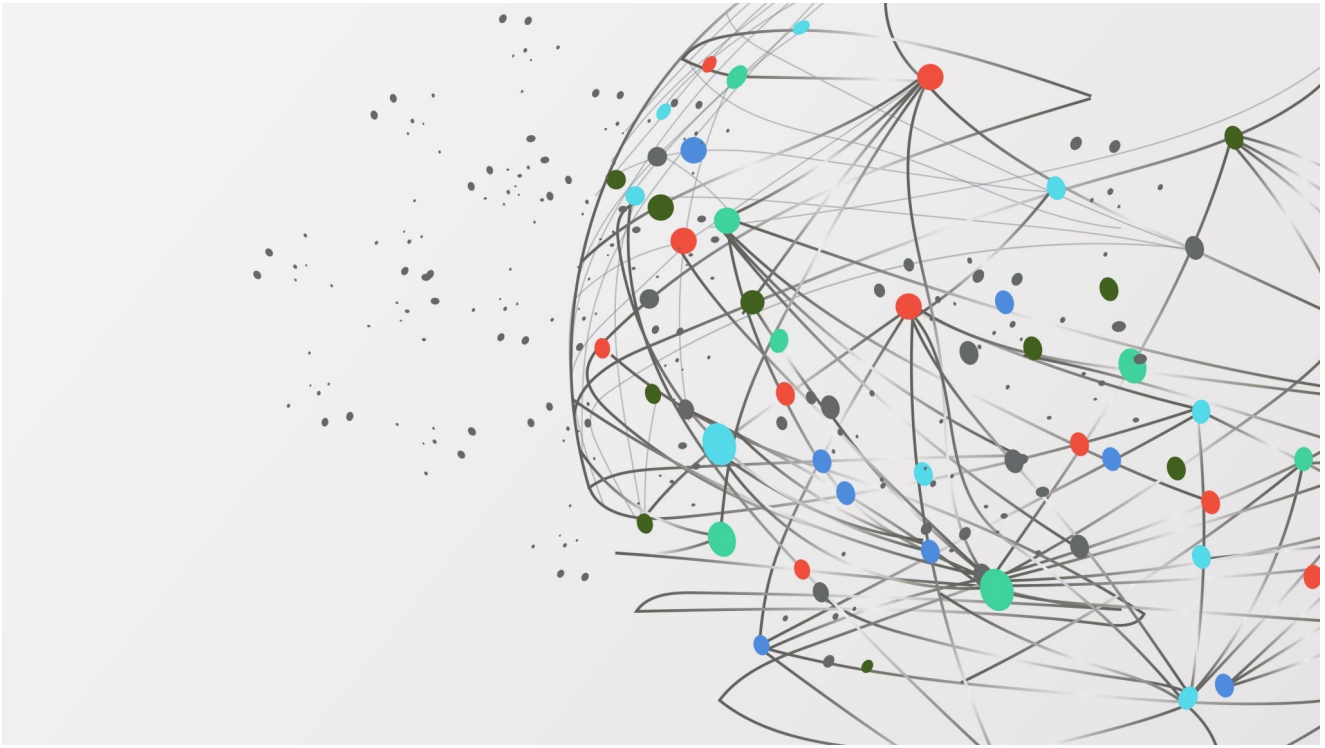
INTRODUCTION TO R

Logistic Regression Notebook

Danny Lumian, Ph.D.

Data Science Training Specialist

NIH Office of Data Science Strategy



Binomial Logistic Regression

- Probability that an observation falls into one of two categories
- A type of classification analysis with two classes
- Examples:
  - Is an email spam?
  - Is a tumor malignant?
  - Will a visit to a website convert into a sale?

mlbench package in R

- https://cran.r-project.org/web/packages/mlbench/index.html
- A collection of artificial and real-world machine learning benchmark problems, including, e.g., several data sets from the UCI repository.
- Includes a dataset on breast cancer
  - Features are set as factors
    - Factors by default are dummy coded
    - So we will convert them to numeric for analysis
  - Target is whether a tumor is malignant or benign

Breast Cancer Data Set

```r
str(bc)
```

```
'data.frame':   683 obs. of  11 variables:
 $ Id              : chr  "1000025" "1002945" "1015425" "1016277" ...
 $ Cl.thickness    : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 5 5 3 6 4 8 1 2 2 4 ...
 $ Cell.size       : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 4 1 8 1 10 1 1 1 2 ...
 $ Cell.shape      : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 4 1 8 1 10 1 2 1 1 ...
 $ Marg.adhesion   : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 5 1 1 3 8 1 1 1 1 ...
 $ Epith.c.size    : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 2 7 2 3 2 7 2 2 2 2 ...
 $ Bare.nuclei     : Factor w/ 10 levels "1","2","3","4",..: 1 10 2 4 1 10 10 1 1 1 ...
 $ Bl.cromatin     : Factor w/ 10 levels "1","2","3","4",..: 3 3 3 3 3 9 3 3 1 2 ...
 $ Normal.nucleoli: Factor w/ 10 levels "1","2","3","4",..: 1 2 1 7 1 7 1 1 1 1 ...
 $ Mitoses         : Factor w/ 9 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 5 1 ...
 $ Class           : Factor w/ 2 levels "benign","malignant": 1 1 1 1 1 2 1 1 1 1 ...
```

Creating the target as a factor

- bc$Class = ifelse(bc$Class == "malignant", 1, 0)
- bc$Class = factor(bc$Class, levels = c(0, 1))
- summary(bc$Class)

Train-test Split

- It is common in machine learning to create a train-test split
- The train data is used to train the model
  - This data is “seen” by the model
- The test data is used to evaluate model performance
  - This data is “unseen” by the model
- Test metrics are usually more informative than train metrics as it shows how well the model generalizes its predictions
- Common splits are 80/20 or 70/30
- Can be useful to stratify data based on target

Creating the train-test split

- trainDataIndex = createDataPartition(bc$Class, p=0.7, list = FALSE)
- trainData = bc[trainDataIndex,]
- testData = bc[-trainDataIndex,]

Training the model

- logitmod = glm(Class ~ Cl.thickness + Cell.size + Cell.shape, family = "binomial", data = trainData)

- Class is the target variable
- Other columns listed are features
- Family is binomial because it is a two-class classification
- Data is our training data

summary(logitmod)

- **Call**: our model definition
- **Deviance Residuals**: Describe fit of data to model
- **Coefficients**: Impact of each factor on the model fit
- **Null deviance:** tells us how well we can predict our output only using the intercept. Smaller is better.
- **Residual deviance:** tells us how well we can predict our output using the intercept and our inputs. Smaller is better. The bigger the difference between the null deviance and residual deviance is, the more helpful our input variables were for predicting the output variable.
- **AIC:** The AIC is the "Akaike information criterion" and it's an estimate of how well your model is describing the patterns in your data. It's mainly used for comparing models trained on the same dataset. If you need to pick between models, the model with the lower AIC is doing a better job describing the variance in the data.

```
Call:
glm(formula = Class ~ Cl.thickness + Cell.size + Cell.shape,
    family = "binomial", data = trainData)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.8866  -0.1777  -0.0848   0.0222   2.0456

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   -7.6402     0.8184  -9.336  < 2e-16 ***
Cl.thickness   0.4552     0.1157   3.935 8.31e-05 ***
Cell.size      0.8143     0.2221   3.667 0.000246 ***
Cell.shape     0.7434     0.2002   3.713 0.000205 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 620.69  on 478  degrees of freedom
Residual deviance: 122.81  on 475  degrees of freedom
AIC: 130.81

Number of Fisher Scoring iterations: 7
```

For more info see: https://stats.stackexchange.com/questions/86351/interpretation-of-rs-output-for-binomial-regression

Making Predictions

- pred = predict(logitmod, newdata = testData, type = "response")
- Predict on testData
- Type=“response”
  - Used for class predictions
  - Default is linear predictions

Convert probabilities to classes

- y_pred_class = ifelse(pred > .5, 1, 0)
- y_pred_factor = factor(y_pred_class, levels=c(0,1))
- testData$preds = y_pred_factor

confusionMatrix(testData$Class, y_pred_factor)

- Confusion matrix at top
- Followed by test metrics
  - Key metrics
    - Accuracy
    - Sensitivity
    - Specificity
- Positive class helps you interpret the metrics

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 129   4
         1   6  65

               Accuracy : 0.951
                 95% CI : (0.9117, 0.9762)
    No Information Rate : 0.6618
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8913

 Mcnemar's Test P-Value : 0.7518

            Sensitivity : 0.9420
            Specificity : 0.9556
         Pos Pred Value : 0.9155
         Neg Pred Value : 0.9699
             Prevalence : 0.3382
         Detection Rate : 0.3186
   Detection Prevalence : 0.3480
      Balanced Accuracy : 0.9488

       'Positive' Class : 1
```

Visualize Confusion Matrix