

---

# Title

---

April 3, 2024

Leo Kraushaar & Ali Hay  
STAT 413

# 1 Introduction

## **2 Model Building**

### **2.1 Fitting Process**

### **2.2 Model Overview**

## 3 Bootstrapping for Feature Significance

### 3.1 Resampling Bootstrap

#### Procedure

Using  $B = 10000$ , an algorithm was written to perform a resampling bootstrap on the dataset. On each iteration, a sample of size  $n$  (where  $n$  was the number of rows in the dataset) was taken with replacement. Using this sample, a new model was fit. The parameter estimates were recorded. Relevant code is shown below.

```
1 resampBoot <- function(df, B) {
2   # Get sample size
3   n <- nrow(df)
4   # Initialize empty dataframe
5   params <- c()
6   # Initialize progress bar
7   bar <- txtProgressBar(min=0, max=B, style=1)
8   # Perform B iterations
9   for (b in 1:B) {
10    # Select a sample of size n
11    indices <- sample(1:n, replace = TRUE)
12    samp <- df[indices, ]
13    # Fit the model with the sample
14    boot_model <- glm.nb(protests ~., data=samp, init.theta =
15                          10)
16    boot_params <- coef(boot_model)
17    params <- rbind(params, boot_params)
18    setTxtProgressBar(bar, b)
19  }
20  close(bar)
21  return(params)
}
```

#### Results

	mean	sd	2.5%	50%	97.5%	sig
intercept	3.66	0.53	2.70	3.64	4.77	TRUE
Spr	-0.06	0.08	-0.23	-0.06	0.09	FALSE
Sum	-0.55	0.08	-0.72	-0.55	-0.38	TRUE
Win	-0.24	0.10	-0.42	-0.23	-0.05	TRUE
BC	0.82	0.14	0.55	0.82	1.09	TRUE
M	-1.89	0.93	-3.85	-1.85	-0.18	TRUE
NB	-2.60	1.06	-4.84	-2.55	-0.66	TRUE
NL	-3.04	1.13	-5.43	-2.98	-0.99	TRUE
NT	-5.40	1.37	-8.20	-5.34	-2.94	TRUE
NS	-2.38	1.00	-4.53	-2.34	-0.54	TRUE
N	-4.98	1.28	-7.68	-4.92	-2.61	TRUE
O	5.54	2.37	1.18	5.40	10.58	TRUE
PEI	-4.10	1.23	-6.73	-4.04	-1.82	TRUE
Q	2.17	0.88	0.55	2.12	4.04	TRUE
S	-2.56	0.96	-4.57	-2.52	-0.77	TRUE
Y	-4.11	1.26	-6.77	-4.06	-1.81	TRUE
retail	-1.83	1.04	-4.05	-1.77	0.08	FALSE

## 3.2 Parametric Bootstrap

### Procedure

Using  $B = 10000$ , an algorithm was written to perform a parametric bootstrap on the dataset.

Using the estimated dispersion parameter  $\theta \approx 8.36$ , each iteration sampled a random vector from a negative binomial distribution. The distribution had dispersion parameter  $\theta$  and mean  $\hat{y}$ , where  $\hat{y}$  was the predicted mean value for the corresponding input values.

Using these new estimates, a model was fit on each iteration and the parameter estimates were recorded. Relevant code is shown below.

```

1 conditionalNegBinom <- function(theta, mu) {
2   nb_sample <- rnbinom(size=theta, mu=mu, n=1)
3   return(nb_sample)
4 }
5
6 paramBoot <- function(B, X, yhat, theta, func) {
7
8   # Initialize empty vector
9   params <- c()
10  # Iterate B times
11  for (b in 1:B) {

```

```

12     # Simulate NB given means
13     sim_y <- sapply(yhat, function(y) func(theta, y))
14     # Add to the dataframe
15     sim_data <- cbind(X, protests=sim_y)
16     # Fit the model to the simulated data
17     sim_model <- glm.nb(protests ~., data=sim_data, init.theta =
18         theta)
19     # Access the coefficients and store
20     parameters <- coef(sim_model)
21     params <- rbind(params, parameters)
22 }
23 return(params)

```

## Results

	mean	sd	2.5%	50%	97.5%	sig
intercept	3.68	0.51	2.68	3.68	4.69	TRUE
Spr	-0.07	0.08	-0.23	-0.07	0.10	FALSE
Sum	-0.55	0.09	-0.72	-0.55	-0.38	TRUE
Win	-0.23	0.09	-0.41	-0.23	-0.05	TRUE
BC	0.82	0.16	0.51	0.82	1.14	TRUE
M	-1.92	0.92	-3.75	-1.91	-0.11	TRUE
NB	-2.63	1.05	-4.72	-2.63	-0.57	TRUE
NL	-3.08	1.11	-5.31	-3.06	-0.89	TRUE
NT	-5.41	1.27	-7.92	-5.39	-2.91	TRUE
NS	-2.42	1.00	-4.39	-2.41	-0.46	TRUE
N	-5.01	1.27	-7.52	-5.01	-2.53	TRUE
O	5.61	2.45	0.80	5.59	10.51	TRUE
PEI	-4.13	1.22	-6.54	-4.12	-1.73	TRUE
Q	2.20	0.93	0.36	2.19	4.07	TRUE
S	-2.59	0.94	-4.46	-2.57	-0.75	TRUE
Y	-4.15	1.25	-6.61	-4.14	-1.71	TRUE
retail	-1.86	1.07	-3.98	-1.85	0.23	FALSE

### 3.3 Smooth Bootstrap

The smooth bootstrap is not an “ideal” method for the given dataset, as only one predictor (**retail**) was continuous and real-valued. However, results were consistent with other methods, as discussed later.

Again using  $B = 10000$ , an algorithm was written to perform a smooth bootstrap. The **retail** column was found to have a sample variance of 1, due to the fact that it was standardized prior to model building. A reasonable value for the noise term was chosen, that is,  $\frac{1}{\sqrt{n}} \approx 0.05783$ . On each iteration, some  $\varepsilon_i$  was added to each row  $i$ , where  $\varepsilon \sim N(0, 0.05783)$ . Using this “new” dataset, a model was fit and the parameter estimates were recorded. Relevant code is shown below.

## Procedure

```

1 addNoise <- function(X) {
2
3   cols <- colnames(X)
4   new_X <- X
5   for (col in cols) {
6     Xi <- X[, col]
7     if (class(data[, col]) != "factor") {
8       n <- length(Xi)
9       S_sq <- var(Xi)
10      noise_var <- S_sq / n
11      new_X[, col] <- Xi + rnorm(n=n, mean=0, sd=sqrt(noise_
12                                var))
13    } else {
14      new_X[, col] <- Xi
15    }
16  }
17  return(new_X)
18 }
19
20 smoothBoot <- function(X, y, B, noisefunc) {
21
22   # Get sample size
23   n <- nrow(X)
24   # Initialize empty vector
25   params <- c()
26
27   # Initialize progress bar
28   pb <- txtProgressBar(min = 0, max = B, style = 3)

```

```

29   # Perform B iterations
30   for (b in 1:B) {
31       # Update progress bar
32       setTxtProgressBar(pb, b)
33
34       # Get new dataset
35       new_X <- noisefunc(X)
36       new_data <- data.frame(protests=y, new_X)
37
38       # Fit the model with the simulated data
39       smoothboot_model <- glm.nb(protests ~., data=new_data, init.
40                                   theta = 5)
41       boot_params <- coef(smoothboot_model)
42       params <- rbind(params, boot_params)
43   }
44
45   # Close progress bar
46   close(pb)
47
48   return(params)
49 }

```

## Results

### 3.4 Error-Sampling Bootstrap

Another bootstrap method was implemented, in which the error terms from the fitted model were randomly sampled with replacement, and added to the fitted values. Notably, some resulting simulated counts were rounded up to zero in the case where a negative value was produced. This was required both logically; as protests counts cannot be negative, and mathematically; as the negative binomial glm cannot be fit with negative training outputs. As a result, the integrity of the simulated datasets was not assumed to be completely intact. That being said, the results were quite consistent with the previous methods.

## Procedure

```

1  epsilonBoot <- function(X, model, B, errors) {
2

```



	mean	sd	2.5%	50%	97.5%	sig
intercept	3.00	0.21	2.58	3.01	3.42	TRUE
Spr	-0.04	0.01	-0.07	-0.04	-0.03	TRUE
Sum	-0.55	0.01	-0.56	-0.55	-0.54	TRUE
Win	-0.20	0.01	-0.23	-0.20	-0.18	TRUE
BC	0.67	0.05	0.57	0.67	0.76	TRUE
M	-0.67	0.39	-1.43	-0.68	0.10	FALSE
NB	-1.21	0.44	-2.08	-1.22	-0.33	TRUE
NL	-1.56	0.47	-2.49	-1.57	-0.62	TRUE
NT	-3.68	0.52	-4.70	-3.68	-2.63	TRUE
NS	-1.07	0.42	-1.89	-1.08	-0.23	TRUE
N	-3.30	0.53	-4.33	-3.30	-2.24	TRUE
O	2.24	1.05	0.14	2.25	4.29	TRUE
PEI	-2.47	0.51	-3.48	-2.48	-1.45	TRUE
Q	0.92	0.40	0.13	0.92	1.69	TRUE
S	-1.31	0.40	-2.09	-1.32	-0.52	TRUE
Y	-2.46	0.52	-3.49	-2.47	-1.41	TRUE
retail	-0.39	0.45	-1.29	-0.40	0.52	FALSE

```

3  # Get sample size
4  n <- nrow(X)
5  # Initialize empty vector
6  params <- c()
7  # Perform B iterations
8  for (b in 1:B) {
9      # Get errors
10     errs <- sample(errors, size=n, replace=TRUE)
11     # Get fitted values
12     yhat <- fitted(model)
13     # Get simulated y
14     ystar <- yhat + errs
15     # round up negative values
16     ystar <- pmax(rep(0, n), ystar)
17     # Turn into DataFrame
18     sim_data <- data.frame(protests=ystar, X)
19     # Fit the model with the simulated data
20     paramboot_model <- glm.nb(protests ~., data=sim_data, init.
        theta = 5)
21     boot_params <- coef(paramboot_model)
22     params <- rbind(params, boot_params)
23 }

```

```

24     return(params)
25 }

```

## Results

	mean	sd	2.5%	50%	97.5%	sig
intercept	3.68	0.51	2.68	3.68	4.69	TRUE
Spr	-0.07	0.08	-0.23	-0.07	0.10	FALSE
Sum	-0.55	0.09	-0.72	-0.55	-0.38	TRUE
Win	-0.23	0.09	-0.41	-0.23	-0.05	TRUE
BC	0.82	0.16	0.51	0.82	1.14	TRUE
M	-1.92	0.92	-3.75	-1.91	-0.11	TRUE
NB	-2.63	1.05	-4.72	-2.63	-0.57	TRUE
NL	-3.08	1.11	-5.31	-3.06	-0.89	TRUE
NT	-5.41	1.27	-7.92	-5.39	-2.91	TRUE
NS	-2.42	1.00	-4.39	-2.41	-0.46	TRUE
N	-5.01	1.27	-7.52	-5.01	-2.53	TRUE
O	5.61	2.45	0.80	5.59	10.51	TRUE
PEI	-4.13	1.22	-6.54	-4.12	-1.73	TRUE
Q	2.20	0.93	0.36	2.19	4.07	TRUE
S	-2.59	0.94	-4.46	-2.57	-0.75	TRUE
Y	-4.15	1.25	-6.61	-4.14	-1.71	TRUE
retail	-1.86	1.07	-3.98	-1.85	0.23	FALSE

## 3.5 Method Comparison

## 4 Monte Carlo Estimation

## 5 Conclusion