Model Building

In [ ]:
```
library(MASS)
library(dplyr)
```

```
Attaching package: ‘dplyr’


The following object is masked from ‘package:MASS’:

    select


The following objects are masked from ‘package:stats’:

    filter, lag


The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union


The following object is masked from ‘package:MASS’:

    select


The following objects are masked from ‘package:stats’:

    filter, lag


The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union
```

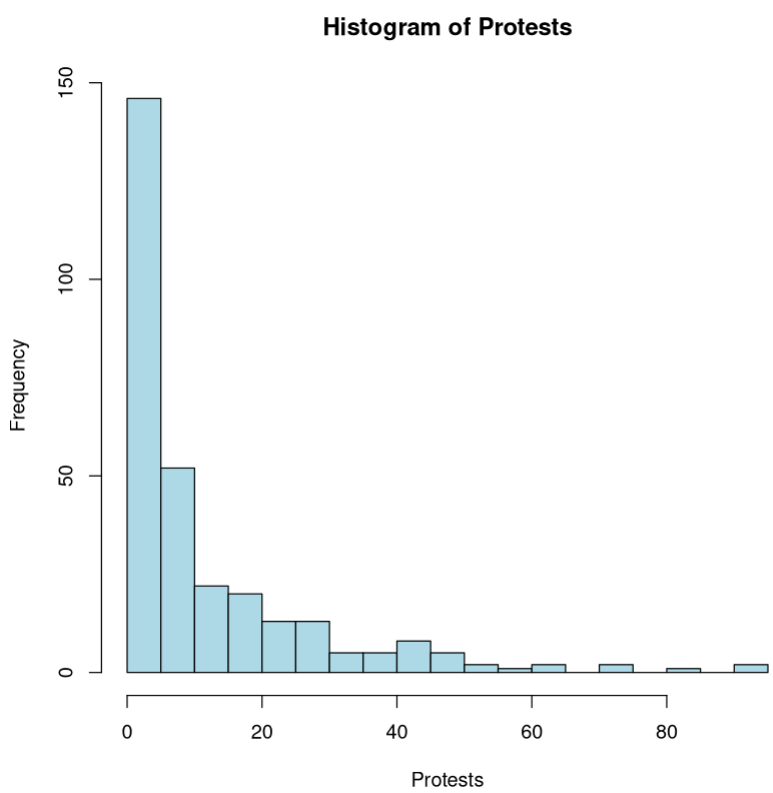
In [ ]:
```
setwd("/home/leoKraushaar/Documents/School/Year 3/Semester 2/STAT 413/Project/protests/")
set.seed(42)
```

In [ ]:
```
B <- 1e4
init_data <- read.csv("data/merged_data.csv")[, -1]
```

In [ ]:
```
no_na <- init_data[!is.na(init_data$food), ]
```

In [ ]:
```

```

In [ ]:
```
hist(init_data$protests, breaks = 20, main = "Histogram of Protests", xlab = "Protests", col="light blue")
```



In [ ]:
```
means <- aggregate(no_na, by=list(GEO=no_na$GEO), mean)
```

```
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
Warning message in mean.default(X[[i]], ...):
“argument is not numeric or logical: returning NA”
```

In [ ]:

```r
replaceFood <- function(food, prov, means) {

    if (is.na(food)) {
        new_val <- means[means$GEO == prov, "food"]
        return(as.numeric(new_val))
    } else {
        return(as.numeric(food))
    }
}
```

In [ ]:

```r
init_data$food <- apply(init_data, MARGIN = 1, function(row) {replaceFood(row["food"], row["GEO"], means)})
```

In [ ]:

```r
init_data$manufac <- NULL

# init_data <- init_data[, 1:5]

init_data
```

A data.frame: 299 × 9

year	month	GEO	pop	protests	retail	oil	food	power
<int>	<chr>	<chr>	<int>	<int>	<dbl>	<int>	<dbl>	<int>
2022	April	Alberta	4480956	17	7989056	3983	868863	6069621
2022	April	British Columbia	5310164	42	8959229	77433	1222442	5240902
2022	April	Manitoba	1405197	2	2083495	6290	194206	2168371
2022	April	New Brunswick	801778	5	1340707	1818	116742	1171958
2022	April	Newfoundland and Labrador	529249	2	920444	77160	78292	686123
2022	April	Northwest Territories	44828	0	76390	0	5724	58889
2022	April	Nova Scotia	1014827	2	1689162	47821	164055	899107
2022	April	Nunavut	40489	0	48635	0	1861	16071
2022	April	Ontario	15046211	46	24616762	267687	2682207	10717875
2022	April	Prince Edward Island	165524	0	269014	0	30470	129091
2022	April	Quebec	8627524	26	13896378	228362	1314059	17514950
2022	April	Saskatchewan	1173366	5	2048833	1285	178681	1869556
2022	April	Yukon	43454	2	88900	0	7819	43778
2022	August	Alberta	4510891	6	8312320	4388	906253	6783590
2022	August	British Columbia	5356284	19	9144939	151630	1233372	4870866
2022	August	Manitoba	1413409	6	2231006	4489	198691	2054341
2022	August	New Brunswick	809568	4	1368487	2311	116593	997796
2022	August	Newfoundland and Labrador	531583	5	958844	150263	78940	684606
2022	August	Northwest Territories	44685	0	75755	36714	5832	51834
2022	August	Nova Scotia	1025445	8	1714108	57149	166828	847062
2022	August	Nunavut	40485	0	48163	0	2375	14405
2022	August	Ontario	15145006	28	24320748	256796	2754474	12987686
2022	August	Prince Edward Island	167188	1	281978	0	33239	115815
2022	August	Quebec	8672185	8	14351635	358902	1395697	15068395
2022	August	Saskatchewan	1178422	4	2110437	1606	181882	2063730
2022	August	Yukon	43905	0	92845	0	7493	38095
2022	December	Alberta	4561350	4	8431294	2616	937555	7146188
2022	December	British Columbia	5403528	16	8960346	167172	1285432	6876391
2022	December	Manitoba	1423596	8	2275146	3608	208406	2876953
2022	December	New Brunswick	817766	7	1376248	1233	121964	1415122
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
2023	November	Prince Edward Island	175853	2	306554.00	0	36342	139348
2023	November	Quebec	8948540	43	14861184.00	311386	1550373	18782951
2023	November	Saskatchewan	1218976	5	2118073.00	1314	203970	2199838
2023	November	Yukon	45148	4	97695.00	0	9460	63017
2023	October	Alberta	4756408	17	8524706.00	2004	1020938	6576535
2023	October	British Columbia	5581127	31	9116046.00	144566	1360250	5481719
2023	October	Manitoba	1465440	14	2263706.00	4114	214117	1883566
2023	October	New Brunswick	842725	4	1470290.00	1137	129902	982231
2023	October	Newfoundland and Labrador	540418	7	953907.00	0	86490	817346
2023	October	Northwest Territories	44760	0	83353.50	0	6509	47285
2023	October	Nova Scotia	1066416	8	1774644.00	63643	180974	825066
2023	October	Nunavut	40817	0	54694.50	0	3715	16165
2023	October	Ontario	15801768	91	24940255.00	110879	3042934	11422274
2023	October	Prince Edward Island	175853	4	302505.00	0	35466	118057
2023	October	Quebec	8948540	36	15090182.50	267734	1534894	15244058
2023	October	Saskatchewan	1218976	14	2180903.50	1628	198505	1979795

year	month	GEO	pop	protests	retail	oil	food	power
<int>	<chr>	<chr>	<int>	<int>	<dbl>	<int>	<dbl>	<int>
2023	October	Yukon	45148	3	94912.00	0	9268	50892
2023	September	Alberta	4695290	19	8548094.33	2956	1004084	6298895
2023	September	British Columbia	5519013	40	9073433.00	127585	1351233	4957719
2023	September	Manitoba	1454902	20	2255683.67	5158	212430	1744014
2023	September	New Brunswick	834691	12	1444919.67	24572	124672	905474
2023	September	Newfoundland and Labrador	538605	6	943546.33	123003	84021	718676
2023	September	Northwest Territories	44972	0	68868.67	24660	5178	48440
2023	September	Nova Scotia	1058694	13	1772434.00	42237	178627	773087
2023	September	Nunavut	40673	0	54499.00	0	3058	14630
2023	September	Ontario	15608369	73	24758223.67	149100	3009578	11407506
2023	September	Prince Edward Island	173787	4	304682.00	0	35896	117725
2023	September	Quebec	8874683	22	14978844.33	327837	1528758	14563138
2023	September	Saskatchewan	1209107	7	2151378.33	1250	196160	1892052
2023	September	Yukon	44975	3	97186.67	0	9221	37029

In [ ]:
```
newMonth <- function(x) {
    if (x %in% c("December", "January", "February")) {
        return("Winter")
    } else if (x %in% c("March", "April", "May")) {
        return("Spring")
    } else if (x %in% c("June", "July", "August")) {
        return("Summer")
    } else {
        return("Fall")
    }
}

newProv <- function(x) {
    if (x %in% c("Yukon", "Nunavut", "Northwest Territories")) {
        return("Northern")
    } else {
        return(x)
    }
}
```

In [ ]:
```
colnames(init_data)
```

'year' · 'month' · 'GEO' · 'pop' · 'protests' · 'retail' · 'oil' · 'food' · 'power'

In [ ]:
```
init_data$month <- sapply(init_data$month, newMonth)
colnames(init_data)[2] <- "season"

head(init_data)
```

A data.frame: 6 × 9

	year	season	GEO	pop	protests	retail	oil	food	power
	<int>	<chr>	<chr>	<int>	<int>	<dbl>	<int>	<dbl>	<int>
1	2022	Spring	Alberta	4480956	17	7989056	3983	868863	6069621
2	2022	Spring	British Columbia	5310164	42	8959229	77433	1222442	5240902
3	2022	Spring	Manitoba	1405197	2	2083495	6290	194206	2168371
4	2022	Spring	New Brunswick	801778	5	1340707	1818	116742	1171958
5	2022	Spring	Newfoundland and Labrador	529249	2	920444	77160	78292	686123
6	2022	Spring	Northwest Territories	44828	0	76390	0	5724	58889

In [ ]:
```
standardize <- function(x, mu, std) {
    return((x-mu)/std)
}
```

In [ ]:
```
colnames(init_data)[colnames(init_data) == "GEO"] <- "prov"
head(init_data)
```

A data.frame: 6 × 9

	year	season	prov	pop	protests	retail	oil	food	power
	<int>	<chr>	<chr>	<int>	<int>	<dbl>	<int>	<dbl>	<int>
1	2022	Spring	Alberta	4480956	17	7989056	3983	868863	6069621
2	2022	Spring	British Columbia	5310164	42	8959229	77433	1222442	5240902
3	2022	Spring	Manitoba	1405197	2	2083495	6290	194206	2168371
4	2022	Spring	New Brunswick	801778	5	1340707	1818	116742	1171958
5	2022	Spring	Newfoundland and Labrador	529249	2	920444	77160	78292	686123
6	2022	Spring	Northwest Territories	44828	0	76390	0	5724	58889

In [ ]:
```
init_data$prov  <- as.factor(init_data$prov)
init_data$season <- as.factor(init_data$season)
init_data$year <- as.factor(init_data$year)
```

In [ ]:
```
head(init_data)
```

A data.frame: 6 × 9

	year	season	prov	pop	protests	retail	oil	food	power
	<fct>	<fct>	<fct>	<int>	<int>	<dbl>	<int>	<dbl>	<int>
1	2022	Spring	Alberta	4480956	17	7989056	3983	868863	6069621
2	2022	Spring	British Columbia	5310164	42	8959229	77433	1222442	5240902
3	2022	Spring	Manitoba	1405197	2	2083495	6290	194206	2168371
4	2022	Spring	New Brunswick	801778	5	1340707	1818	116742	1171958
5	2022	Spring	Newfoundland and Labrador	529249	2	920444	77160	78292	686123
6	2022	Spring	Northwest Territories	44828	0	76390	0	5724	58889

Fitting the Model

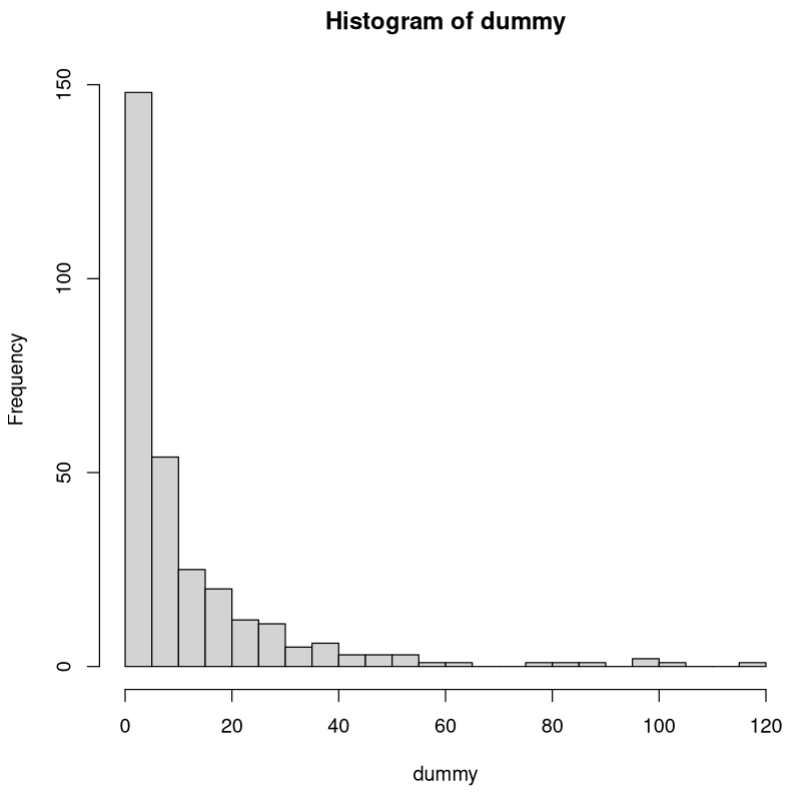
In [ ]:
```
init_data$pop <- sapply(init_data$pop, function(x) {standardize(x, mean(init_data$pop), sd(init_data$pop))})
init_data$retail <- sapply(init_data$retail, function(x) {standardize(x, mean(init_data$retail), sd(init_data$retai
init_data$power <- sapply(init_data$power, function(x) {standardize(x, mean(init_data$power), sd(init_data$power))}
init_data$oil <- sapply(init_data$oil, function(x) {standardize(x, mean(init_data$oil), sd(init_data$oil))})
init_data$food <- sapply(init_data$food, function(x) {standardize(x, mean(init_data$food), sd(init_data$food))})
```

In [ ]:
```
model1 <- glm.nb(protests ~., data=init_data, init.theta = 1)
```

In [ ]:
```
summary(model1)
```

```
Call:
glm.nb(formula = protests ~ ., data = init_data, init.theta = 8.787356117,
    link = log)

Coefficients:
                            Estimate Std. Error z value Pr(>|z|)
(Intercept)                  3.49134    0.82959   4.208 2.57e-05 ***
year2023                     0.09127    0.07835   1.165   0.2441
seasonSpring                -0.03918    0.08668  -0.452   0.6513
seasonSummer                -0.53066    0.08765  -6.054 1.41e-09 ***
seasonWinter                -0.17789    0.09594  -1.854   0.0637 .
provBritish Columbia         0.70090    0.37307   1.879   0.0603 .
provManitoba                -1.61586    1.63901  -0.986   0.3242
provNew Brunswick           -2.21134    1.93107  -1.145   0.2522
provNewfoundland and Labrador -2.57320    2.06609  -1.245   0.2130
provNorthwest Territories   -4.84238    2.34607  -2.064   0.0390 *
provNova Scotia             -2.03238    1.82298  -1.115   0.2649
provNunavut                 -4.46958    2.34348  -1.907   0.0565 .
provOntario                  3.68766    5.29595   0.696   0.4862
provPrince Edward Island    -3.61895    2.27313  -1.592   0.1114
provQuebec                   1.81488    2.17220   0.836   0.4034
provSaskatchewan            -2.20521    1.73856  -1.268   0.2046
provYukon                   -3.61983    2.33373  -1.551   0.1209
pop                          1.75023    2.06369   0.848   0.3964
retail                      -2.41493    1.21992  -1.980   0.0478 *
oil                         -0.04936    0.06776  -0.728   0.4663
food                        -0.34791    0.33713  -1.032   0.3021
power                       -0.20999    0.14441  -1.454   0.1459
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for Negative Binomial(8.7874) family taken to be 1)

    Null deviance: 2151.08  on 298  degrees of freedom
Residual deviance:  350.72  on 277  degrees of freedom
AIC: 1589.5

Number of Fisher Scoring iterations: 1


              Theta:  8.79
          Std. Err.:  1.59

 2 x log-likelihood:  -1543.477
```

In [ ]:
```
dummy <- rnegbin(fitted(model1), theta = 8.68)
hist(dummy, breaks=25)
```



In [ ]:
```
anova(model1)
```

```
Warning message in anova.negbin(model1):
“tests made without re-estimating 'theta'”
```

A anova: 9 × 5

	Df	Deviance	Resid. Df	Resid. Dev	Pr(>Chi)
	<int>	<dbl>	<int>	<dbl>	<dbl>
NULL	NA	NA	298	2151.0784	NA
year	1	1.022458e+00	297	2150.0560	3.119367e-01
season	3	5.355482e+01	294	2096.5012	1.396212e-11
prov	12	1.737473e+03	282	359.0283	0.000000e+00
pop	1	6.022725e-03	281	359.0222	9.381413e-01
retail	1	5.208708e+00	280	353.8135	2.247403e-02
oil	1	3.983980e-01	279	353.4151	5.279178e-01
food	1	5.640837e-01	278	352.8511	4.526195e-01
power	1	2.134495e+00	277	350.7166	1.440179e-01

In [ ]:
```
data <- init_data[, -c(1, 4, 7, 8)]
model <- glm.nb(protests ~., data=data, init.theta = 1)

summary(step(model))
# names(summary(model))
# summary(model)
```

```
Start:  AIC=1584.06
protests ~ season + prov + retail + power

         Df Deviance    AIC
- power   1   350.06 1583.0
<none>        349.12 1584.1
- retail  1   352.66 1585.6
- season  3   397.71 1626.7
- prov   12   740.51 1951.5

Step:  AIC=1583
protests ~ season + prov + retail

         Df Deviance    AIC
<none>        349.20 1583.0
- retail  1   352.33 1584.1
- season  3   397.19 1625.0
- prov   12   745.36 1955.2
Call:
glm.nb(formula = protests ~ season + prov + retail, data = data,
    init.theta = 8.30561596, link = log)

Coefficients:
                            Estimate Std. Error z value Pr(>|z|)
(Intercept)                  3.68364    0.50985   7.225 5.01e-13 ***
seasonSpring                -0.06502    0.08264  -0.787 0.431441
seasonSummer                -0.55217    0.08629  -6.399 1.57e-10 ***
seasonWinter                -0.22871    0.08873  -2.578 0.009946 **
provBritish Columbia         0.81551    0.16310   5.000 5.73e-07 ***
provManitoba                -1.91505    0.92108  -2.079 0.037605 *
provNew Brunswick           -2.62399    1.04409  -2.513 0.011965 *
provNewfoundland and Labrador -3.06967    1.11340  -2.757 0.005833 **
provNorthwest Territories   -5.35839    1.26866  -4.224 2.40e-05 ***
provNova Scotia             -2.41344    0.99351  -2.429 0.015133 *
provNunavut                 -4.98454    1.26290  -3.947 7.92e-05 ***
provOntario                  5.59695    2.44944   2.285 0.022314 *
provPrince Edward Island    -4.11279    1.21600  -3.382 0.000719 ***
provQuebec                   2.19032    0.93156   2.351 0.018711 *
provSaskatchewan            -2.58187    0.94178  -2.741 0.006116 **
provYukon                   -4.13848    1.24460  -3.325 0.000884 ***
retail                      -1.85334    1.06448  -1.741 0.081671 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for Negative Binomial(8.3056) family taken to be 1)

    Null deviance: 2091.2  on 298  degrees of freedom
Residual deviance:  349.2  on 282  degrees of freedom
AIC: 1585

Number of Fisher Scoring iterations: 1


              Theta:  8.31
          Std. Err.:  1.46

 2 x log-likelihood:  -1548.997
```

Bootstrap Methods

1. Resampling Bootstrap

In [ ]:
```r
resampBoot <- function(df, B) {

    # Get sample size
    n <- nrow(df)
    # Initialize empty dataframe
    params <- c()
    # Initialize progress bar
    bar <- txtProgressBar(min=0, max=B, style=1)
    # Perform B iterations
    for (b in 1:B) {
        # Select a sample of size n
        indices <- sample(1:n, replace = TRUE)
        samp <- df[indices, ]
        # Fit the model with the sample
        boot_model <- glm.nb(protests ~., data=samp, init.theta = 10)
        boot_params <- coef(boot_model)
        params <- rbind(params, boot_params)
        setTxtProgressBar(bar, b)
    }
    close(bar)
    return(params)
}
```

In [ ]:
```r
# boot_models <- resampBoot(data, B)
# boot_models <- as.data.frame(boot_models)
# write.csv(boot_models, "data/results/resamp_boot_results.csv")
```

2. Parametric Bootstrap

In [ ]:
```r
conditionalNegBinom <- function(theta, mu) {
    nb_sample <- rnbinom(size=theta, mu=mu, n=1)
    return(nb_sample)
}
```

In [ ]:
```r
paramBoot <- function(B, X, yhat, theta, func) {

    # Initialize empty vector
    params <- c()
    # Iterate B times
    for (b in 1:B) {
        # Simulate NB given means
        sim_y <- sapply(yhat, function(y) func(theta, y))
        # Add to the dataframe
        sim_data <- cbind(X, protests=sim_y)
        # Fit the model to the simulated data
        sim_model <- glm.nb(protests ~., data=sim_data, init.theta = theta)
        # Access the coefficients and store
        parameters <- coef(sim_model)
        params <- rbind(params, parameters)
    }
    return(params)
}
```

In [ ]:
```r
model <- glm.nb(protests ~., data=data, init.theta = 1)

theta <- summary(model)$theta
X <- data[, c(1,2,4)]
yhat <- fitted(model)
```

In [ ]:
```r
# parametric_results <- paramBoot(B, X, yhat, theta, func=conditionalNegBinom)
# param_results <- as.data.frame(parametric_results)
# write.csv(param_results, "data/results/param_boot_results.csv")
```

3. Error Bootstrap

In [ ]:
```r
epsilonBoot <- function(X, model, B, errors) {

    # Get sample size
    n <- nrow(X)
    # Initialize empty vector
    params <- c()
    # Perform B iterations
    for (b in 1:B) {
        # Get errors
        errs <- sample(errors, size=n, replace=TRUE)
        # Get fitted values
        yhat <- fitted(model)
```

```
        # Get simulated y
        ystar <- yhat + errs
        # round up negative values
        ystar <- pmax(rep(0, n), ystar)
        # Turn into DataFrame
        sim_data <- data.frame(protests=ystar, X)
        # Fit the model with the simulated data
        paramboot_model <- glm.nb(protests ~., data=sim_data, init.theta = 5)
        boot_params <- coef(paramboot_model)
        params <- rbind(params, boot_params)
    }
    return(params)
}
```

In [ ]:
```
model <- glm.nb(protests ~., data=data, init.theta = 10)
```

In [ ]:
```
names(summary(model))
```

'call' · 'terms' · 'family' · 'deviance' · 'aic' · 'contrasts' · 'df.residual' · 'null.deviance' · 'df.null' · 'iter' · 'deviance.resid' · 'coefficients' · 'aliased' · 'dispersion' · 'df' · 'cov.unscaled' · 'cov.scaled' · 'theta' · 'SE.theta' · 'twologlik' · NA

In [ ]:
```
summary(model)
```

```
Call:
glm.nb(formula = protests ~ ., data = data, init.theta = 8.362454949,
    link = log)

Coefficients:
                            Estimate Std. Error z value Pr(>|z|)
(Intercept)                  3.80300    0.53069   7.166 7.71e-13 ***
seasonSpring                -0.06231    0.08254  -0.755 0.450316
seasonSummer                -0.55664    0.08631  -6.450 1.12e-10 ***
seasonWinter                -0.20141    0.09358  -2.152 0.031377 *
provBritish Columbia         0.80242    0.16309   4.920 8.65e-07 ***
provManitoba                -2.12929    0.95809  -2.222 0.026254 *
provNew Brunswick           -2.88075    1.09019  -2.642 0.008232 **
provNewfoundland and Labrador -3.33992    1.16180  -2.875 0.004043 **
provNorthwest Territories   -5.66552    1.32286  -4.283 1.85e-05 ***
provNova Scotia             -2.66872    1.04101  -2.564 0.010360 *
provNunavut                 -5.29236    1.31782  -4.016 5.92e-05 ***
provOntario                  6.02503    2.50916   2.401 0.016341 *
provPrince Edward Island    -4.41369    1.27031  -3.475 0.000512 ***
provQuebec                   2.58647    1.03400   2.501 0.012370 *
provSaskatchewan            -2.80100    0.97986  -2.859 0.004255 **
provYukon                   -4.44513    1.29975  -3.420 0.000626 ***
retail                      -1.97897    1.07824  -1.835 0.066452 .
power                       -0.13459    0.13888  -0.969 0.332499
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for Negative Binomial(8.3625) family taken to be 1)

    Null deviance: 2098.42  on 298  degrees of freedom
Residual deviance:  349.12  on 281  degrees of freedom
AIC: 1586.1

Number of Fisher Scoring iterations: 1


              Theta:  8.36
          Std. Err.:  1.47

 2 x log-likelihood:  -1548.062
```

In [ ]:
```
summary(model)$coefficients
```

A matrix: 18 × 4 of type dbl

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	3.80299853	0.53068688	7.1661816	7.711845e-13
seasonSpring	-0.06230567	0.08253626	-0.7548885	4.503159e-01
seasonSummer	-0.55663825	0.08630699	-6.4495156	1.122082e-10
seasonWinter	-0.20140616	0.09357885	-2.1522615	3.137676e-02
provBritish Columbia	0.80242485	0.16308973	4.9201434	8.648084e-07
provManitoba	-2.12929244	0.95808858	-2.2224380	2.625372e-02
provNew Brunswick	-2.88074548	1.09019082	-2.6424232	8.231514e-03
provNewfoundland and Labrador	-3.33992495	1.16179654	-2.8747933	4.042920e-03
provNorthwest Territories	-5.66552282	1.32286008	-4.2827831	1.845701e-05
provNova Scotia	-2.66871606	1.04100790	-2.5635887	1.035962e-02
provNunavut	-5.29236331	1.31782058	-4.0159969	5.919504e-05
provOntario	6.02502639	2.50915535	2.4012170	1.634064e-02
provPrince Edward Island	-4.41369386	1.27030554	-3.4745136	5.117804e-04
provQuebec	2.58646624	1.03399957	2.5014191	1.236967e-02
provSaskatchewan	-2.80099782	0.97985749	-2.8585767	4.255462e-03
provYukon	-4.44513193	1.29974974	-3.4199906	6.262329e-04
retail	-1.97897057	1.07824473	-1.8353631	6.645191e-02
power	-0.13458713	0.13887966	-0.9690917	3.324994e-01

In [ ]:
```
X <- data[, c(1,2,4)]
epsilon <- resid(model)
err_mean <- mean(epsilon)
err_sd <- sd(epsilon)
```

In [ ]:
```
suppressWarnings({
  error_models <- epsilonBoot(X, model, B, epsilon)
})
error_results <- as.data.frame(error_models)
# write.csv(error_results, "data/results/error_boot_results.csv")
```

In [ ]:
```
var(data$retail)
```

1

3. Smooth Bootstrap

In [ ]:
```
addNoise <- function(X) {

    cols <- colnames(X)
    new_X <- X
    for (col in cols) {
        Xi <- X[, col]
        if (class(data[, col]) != "factor") {
            n <- length(Xi)
            S_sq <- var(Xi)
            noise_var <- S_sq / n
            new_X[, col] <- Xi + rnorm(n=n, mean=0, sd=sqrt(noise_var))
        } else {
            new_X[, col] <- Xi
        }
    }
    return(new_X)
}
```

In [ ]:
```
smoothBoot <- function(X, y, B, noisefunc) {

    # Get sample size
    n <- nrow(X)
    # Initialize empty vector
    params <- c()

    # Initialize progress bar
    pb <- txtProgressBar(min = 0, max = B, style = 3)

    # Perform B iterations
    for (b in 1:B) {
        # Update progress bar
        setTxtProgressBar(pb, b)
```

```
        # Get new dataset
        new_X <- noisefunc(X)
        new_data <- data.frame(protests=y, new_X)

        # Fit the model with the simulated data
        smoothboot_model <- glm.nb(protests ~., data=new_data, init.theta = 5)
        boot_params <- coef(smoothboot_model)
        params <- rbind(params, boot_params)
    }

    # Close progress bar
    close(pb)

    return(params)
}
```

In [ ]:
```
X <- data[, c(1,2,4)]
y <- data$protests
```

In [ ]:
```
# smooth_data <- smoothBoot(X, y, B, addNoise)
# smooth_results <- as.data.frame(smooth_data)
# write.csv(smooth_results, "data/results/smooth_boot_results.csv")
```

Monte Carlo Prediction

Make Predictions