Bicycle Counts for East River Bridges

Llasmin Lopez

March 10, 2021

1 Introduction

New York City is the most populated city in America with an approximate 8.6 million residents, more than double the second most populated city [3]. The need for efficiency in transportation throughout the city is evident. Thus we investigate one of the more environmentally friendly and inexpensive modes of transportation: cycling. In this analysis, we use 'Bicycle Counts for East River Bridges'[2] to build model and investigate the relationship between total count of cyclists occupation and Day of the week, temperature, and precipitation. We apply several model building methods to produce a model and make predictions.

2 Data

Our data set,'Bicycle Counts for East River Bridges' is a daily total of bike counts conducted monthly on the Brooklyn Bridge, Manhattan Bridge, Williamsburg Bridge, and Queensboro Bridge, collected by The Traffic Information Management System. The entire data set consists of 214 observations and ten variables: Date, Day of the week, Highest Temperature of the day in F, Lowest Temperature of the day in F, Daily Precipitation, daily cyclist count on the Brooklyn Bridge, daily cyclist count on the Manhattan Bridge, daily cyclist count on Williamsburg Bridge, daily cyclist count on the Queensboro Bridge, and total daily cyclist count. We consider Total daily cyclist count as the response variable.

With our goal of prediction, this data presents a challenge in the response variable. Although this data is described as count data, the range of the observed counts is too large to be modeled by typical count models. Also, although there are nine possible covariates, the inclusion of individual bridge counts introduces multicollinearity issues so these are excluded from possible covariates for the model. Thus we are left with only five covariates. Also, 14 of the 214 observations are are evaluated as 'T' meaning the precipitation of the day was too small to measure, however is not zero. Due to the uncertainty of this measurement, we remove these fourteen observations.

3 Methods

In order to build a model for prediction, we randomly split the remaining 200 observations into two data sets: Train and Test. We used the data set Train with 160 observations to construct a model. As previously mentioned, the response variable is a count so we initially fit Poisson, Negative Binomial, and Quasi-Poisson models. However, we quickly ran into problems as our count ranges from 2,374 to 26,969, which is too large for the aforementioned models.

We then fit models using Log-Normal, Gamma Additive, and Cross-Validation. We used step-wise AIC model selection in attempt to reduce the parameters of our model, however they all returnned the full model. Based on R^2 values, we determined the best performing model among our selection was a Gamma Additive model with a smoothing parameter.

3.1 Model

Generalized Additive Models are additive extensions of their generalized linear models counterparts, i.e. for link function g, $g(E(Y|X = x)) = \alpha + \beta_1 x_1 + ... + \beta_{p-1} x_{p-1}$ is extended to

$$g(E(Y|X = x)) = \alpha + f_1(x_1) + ... + f_{p-1}(x_{p-1})$$

with the requirement of $E(f_j(X_j)) = 0, j = 1, ..., p - 1$.

We fit a GAM with a log link fuction under the Gamma family to four variables: Day, High Temp., Low Temp., and Precipitation. We chose a cubic spline to smooth both High Temp. and Precipitation.

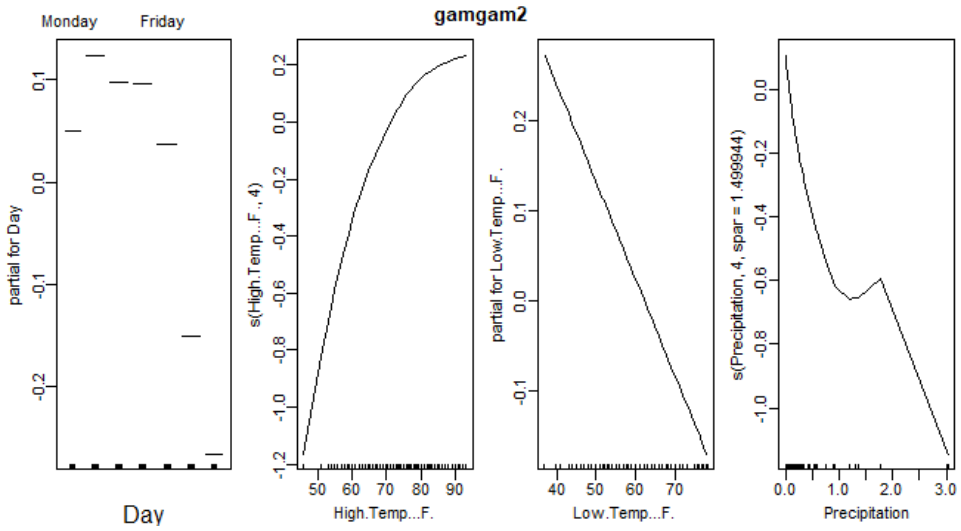
Non Parametric Effects	Non Parametric df	Non Parametric χ^2	P($> \chi^2$)
High Temp.	3	16.115	4.367e-09
Precipitation	3	15.136	1.294e-08

Table 1: Non-Parametric GAM Fit Results

Parametric Effects	Parametric df	F-value	P(¿ F)
Day	6	21.481	$<$ 2.2e-16
High Temp.	1	208.569	$<$ 2.2e-16
Low Temp.	1	56.275	5.961e-12
Precipitation	1	221.889	$<$ 2.2e-16

Table 2: Parametric GAM Fit Results

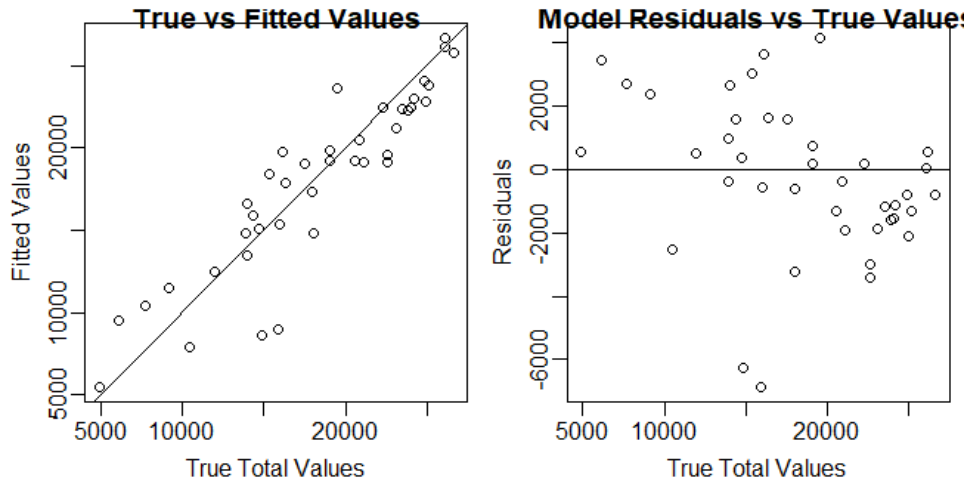
Our estimates additive functions, f_j j = 1,...,4, are plotted below



From the plots of fitted functions, we see that there are some gaps in the range of the predictor Precipitation which may impact the results for smoothing in a significant way. The minimum bandwidth must be large enough to cover the gap in order to provide well defined estimates at the gaps, which may lead to over-smoothing for other regions. This could be why we observe a sudden changes at the boundaries of the range of the predictor. Over the range of predictors however, the functions appear to have a linear trend apart from the variable High Temp.

4 Results

We use our model and function `predict.GAM()` to obtain predictions of Total values for the data set Test, described earlier. We found our model had an R^2 value of 0.7757 and a Root Mean Squared error of 2,396.1929448. The True versus Fitted values indicate a linear trend and suggest out model works

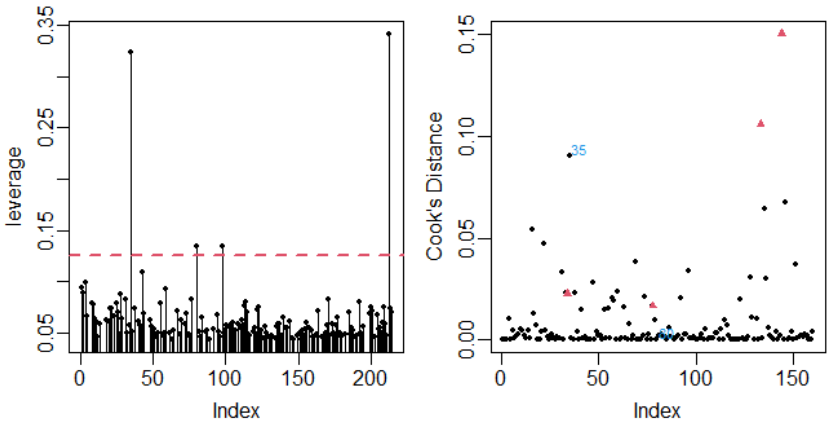


better for larger values of Total count. Our residual plot does not appear to indicate a trend, however our

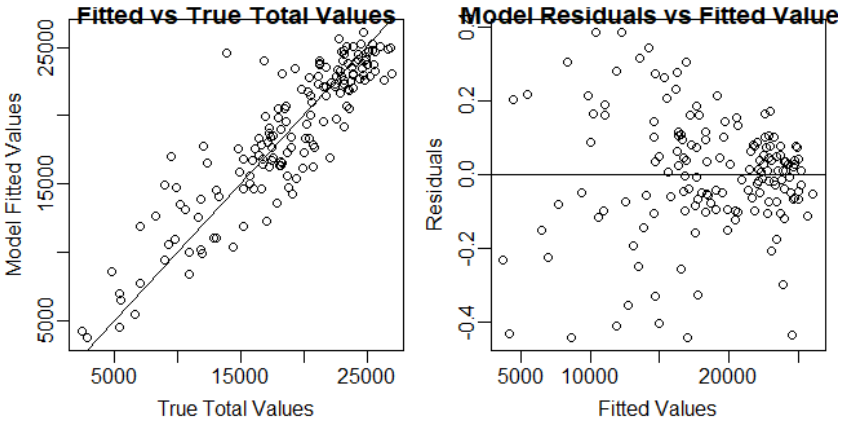
residuals and large root mean squared error indicate a lack of good fit.

5 Discussion

Although we considered an array of models, our selection was ultimately not a good fit. Recall the fitted plot for f_2, High Temp. We attempted to correct the curved trend found but the resulting model had a smaller R^2 than the chosen model. Otherwise, the model showed no indication of influential points:



Taking another look at the fitted versus true values on the training data set, we see a cluster of points near the higher end of the range, which may explain why our model seems to perform better for larger values.



6 Future Work

Our next steps could be to investigate the performance of GAM models with different smoothing methods or consider other extensions of the additive model such as the additive model with interactions. We may also pursue models under Log-Normal.

References

[1] 'Appendix A:Required Pre-knowledge: A Linear Regression and Additive Modelling Example' https://link.springer.com/content/pdf/bbm

[2] Department of Transportation, NYC Open Data https://data.cityofnewyork.us/Transportation/Bicycle-Counts-for-East-River-Bridges/gua4-p9wg

[3] U.S.A. Population Statistic: https://worldpopulationreview.com/us-cities

7 Appendix: R Code

```
###############
library(tidyverse)
library(MASS)
library(nnet)
library(pscl)
library(gam)

### DATA-----------------------------------------------------------------

my.data = read.csv("C:/Users/Llasmin/OneDrive/New One
Drive/OneDrive/Documents/STA 223/2017 Monthly Bike Count Totals for East
River Bridges_04 April 2017 Cyclist Numbers for Web.csv",header = TRUE)
  my.data = my.data[-c(31:47),-11]
may = read.csv("C:/Users/Llasmin/OneDrive/New One
Drive/OneDrive/Documents/STA 223/Copy of 2017 Monthly Bike Count Totals for
East River Bridges_05 May 2017 Cyclist Numbers for Web.csv",header = TRUE)
  may = may[-32,-11]
june = read.csv("C:/Users/Llasmin/OneDrive/New One
Drive/OneDrive/Documents/STA 223/2017 Monthly Bike Count Totals for East
River Bridges_06 June 2017 Cyclist Numbers for Web.csv",header = TRUE)
  june = june[-c(31,32),-c(11:13)]
july = read.csv("C:/Users/Llasmin/OneDrive/New One
Drive/OneDrive/Documents/STA 223/Copy of 2017 Monthly Bike Count Totals for
East River Bridges_07 July 2017 Cyclist Numbers for Web.csv",header = TRUE)
  july = july[-c(32:37),-c(11:13)]
aug = read.csv("C:/Users/Llasmin/OneDrive/New One
Drive/OneDrive/Documents/STA 223/Copy of 2017 Monthly Bike Count Totals for
East River Bridges_08 August 2017 Cyclist Numbers for Web.csv",header = TRUE)
  aug = aug[-c(32:40),-c(11:13)]
sept = read.csv("C:/Users/Llasmin/OneDrive/New One
Drive/OneDrive/Documents/STA 223/Copy of 2017 Monthly Bike Count Totals for
East River Bridges_09 September 2017 Cyclist Numbers for Web.csv",header =
TRUE)
  sept = sept[-c(31:38),-c(11:13)]
oct = read.csv("C:/Users/Llasmin/OneDrive/New One
Drive/OneDrive/Documents/STA 223/Copy of 2017 Monthly Bike Count Totals for
East River Bridges_10 October 2017 Cyclist Numbers.csv",header = TRUE)
  oct = oct[-c(32:38),-c(11:13)]

  cnames = colnames(my.data)
```

```
  colnames(may) = cnames;colnames(june) = cnames;colnames(july) = cnames
  colnames(aug) = cnames;colnames(sept) = cnames;colnames(oct) = cnames

my.data = rbind(my.data, may, june, july,aug,sept,oct)

# Formatting
my.data$Day = factor(my.data$Day,
            levels = c('Monday','Tuesday','Wednesday','Thursday','Friday','Saturday',
            'Sunday'))
my.data$Brooklyn.Bridge = as.numeric(gsub(",", "", my.data$Brooklyn.Bridge))
my.data$Manhattan.Bridge = as.numeric(gsub(",", "", my.data$Manhattan.Bridge))
my.data$Williamsburg.Bridge = as.numeric(gsub(",", "", my.data$Williamsburg.Bridge))
my.data$Queensboro.Bridge = as.numeric(gsub(",", "", my.data$Queensboro.Bridge))
my.data$Total = as.numeric(gsub(",", "", my.data$Total))
  # Percipitation = T implies trace amounts, not zero but too small to be measured
  # some say under 0.01
Ts = which(my.data$Precipitation == "T")

# Can't use value T for percipitation so I will remove these dates
my.data = my.data[-c(Ts),]
my.data$Precipitation = as.numeric(gsub(",", "", my.data$Precipitation))

# Split the dataset ( samples) into a training set (80\%, ) and testing set (20\%)
train_size = floor(0.80 * dim(my.data)[1])
set.seed(2021)
train_ind = sample(seq_len(dim(my.data)[1]), size = train_size)
train = my.data[train_ind,]
test = my.data[-train_ind,]


### Exploratory-----------------------------------------------------------
  summary(train)
  boxplot(train$Precipitation)
  # many outliers for percipitation but this data is from April-Oct
  # so spikes in percipitation across this range is expected
  boxplot(train$High.Temp...F.)
  boxplot(train$Low.Temp...F.)
  boxplot(train$Brooklyn.Bridge)
  boxplot(train$Manhattan.Bridge)
  boxplot(train$Williamsburg.Bridge)
  boxplot(train$Queensboro.Bridge)
  # everything looks pretty much normal
```

```
  boxplot(Total~Day, data = train)
  # Tuesday followed by Monday have the largest spread,
  # Sunday and Saturday have the lowest means

  boxplot(train$Brooklyn.Bridge, train$Manhattan.Bridge,
          train$Williamsburg.Bridge, train$Queensboro.Bridge)
  # not as much traffic through Brooklyn Bridge

  pairs(cbind(train$Total,train$High.Temp...F.))
  pairs(cbind(train$Total,train$Low.Temp...F.))
  pairs(cbind(train$Total,train$Precipitation))
  pairs(cbind(log(train$Total),train$High.Temp...F.))
  pairs(cbind(log(train$Total),train$Low.Temp...F.))
  pairs(cbind(log(train$Total),train$Precipitation))
  #percipitation is the problem child


# Possible Parametric Models ---------------------------------------
n <- nrow(train)

  # Poisson------
  poisson <- glm(Total ~ Day + High.Temp...F.+ Low.Temp...F.+Precipitation,
                 data = train, family = poisson())
  sigma2 <- poisson$deviance/(n-length(coef(poisson)))
  sigma2 # 636.8974 inflation parameter
  summary(poisson)
    #R^2
  1-(poisson$deviance/poisson$null.deviance) # 0.6941645

    #poisson with interactions
    poisson2 <- glm(Total ~ Day + High.Temp...F.*Low.Temp...F.*Precipitation,
                 data = train, family = poisson())
    sigma22 <- poisson2$deviance/(n-length(coef(poisson2)))
    sigma22 # 542.0782 inflation parameter
    stepAIC(poisson2,
              scope = list(upper = ~Day+ High.Temp...F.*Low.Temp...F.*Precipitation,
                           lower = ~1), trace = FALSE)
    summary(poisson2)
    mean(poisson2$residuals^2) # 0.0513
```

```
# NegBin without specifying theta-----------------
  nb1 <- glm.nb(Total ~ Day +High.Temp...F.*Low.Temp...F.*Precipitation,
              data=train)
  summary(nb1)$dispersion #1
  nb1$deviance / (n-length(coef(nb1)))
  # 1.103299 closer to 1 compared to Poisson so exclude poisson model as option
  1-(nb1$deviance/nb1$null.deviance) # 0.72632

# Quasi-Poisson---------------
  quasip <- glm(Total ~ Day + High.Temp...F.* Low.Temp...F.*Precipitation,
              data = train, family = quasipoisson())
  summary(quasip)
  quasip$deviance / (n-length(coef(quasip)))
  # 542.0782, significantly larger than previous models (same as poisson).
  summary(quasip)$dispersion # 521.993, not 1 as fixed in Poisson model
  1-(quasip$deviance/quasip$null.deviance) # 0.7466377

  # Zero Inflation ----------------------------------------------
  which(train$Total == 0) # 0
  # Zero inflation issue not expected in this data set.


#-----------------------------------------------------------------
# My data doesn't necessarily fit count models....

  # Log-Normal----------------------
  train_log = train
  train_log$Total = log(train_log$Total)

  log.fit = glm(Total ~ Day +High.Temp...F.*Low.Temp...F.*Precipitation,
                data=train_log, family = 'gaussian')
  summary(log.fit)
  stepAIC(log.fit,trace = TRUE, direction = 'both')
  1-(log.fit$deviance/log.fit$null.deviance) #0.7711252
  log.fit$deviance / (n-length(coef(log.fit))) # 0.4363

    log2.fit = glm(Total ~ Day +High.Temp...F.+Low.Temp...F.+Precipitation,
                data=train_log, family = 'gaussian')
    summary(log2.fit)
    stepAIC(log2.fit,trace = TRUE)
    1-(log2.fit$deviance/log2.fit$null.deviance) # 0.718, worse
```

```
# Quasi-gamma, gamma additive----------
  # fit a GLM as a baseline model
  glm <- glm(Total ~ Day +High.Temp...F.+Low.Temp...F.+Precipitation,
             family = Gamma(link=log), data = train)
  summary(glm)
  1-(glm$deviance/glm$null.deviance) #0.66428

  # GAM 1-----------------------
  gamgam1 <- gam::gam(Total ~ Day +s(High.Temp...F.,4)+
                        s(Low.Temp...F.,4)+s(Precipitation,4),
                      family = Gamma(link=log), data = train)
  # smoothing spline with df=4 for all covariates
  summary(gamgam1)
  # additive effect of Low.Temp is not significant
  1-(gamgam1$deviance/gamgam1$null.deviance) # 0.795105
  gamgam1$deviance / (n-length(coef(gamgam1))) # 0.02945
  rsq(gamgam1) #0.7752

  par(mfrow=c(1,2), mgp = c(1.8,0.5,0), mar = c(3,3,.5,0.5), oma = c(0,0,2,0))
  plot(gamgam1) # plot the fitted smoothing function. See Figure 3.
  title(main = "gamgam1", outer = TRUE)

  # GAM 2 : Selected Model -------------------------
  # additive effect Low.Temp was not significant in previous so we removed here
  gamgam2 <- gam::gam(Total ~ Day +s(High.Temp...F.,4)+Low.Temp...F.+
                        s(Precipitation,4,spar = 1.499944),
                      family = Gamma(link=log), data = train)
  summary(gamgam2)
  1-(gamgam2$deviance/gamgam2$null.deviance) # 0.7925508
  rsq(gamgam2) # 0.77
  gamgam2$deviance / (n-length(coef(gamgam2))) # 0.0298

  par(mfrow=c(1,4), mgp = c(1.8,0.5,0), mar = c(3,3,.5,0.5), oma = c(0,0,2,0))
  plot(gamgam2) # plot the fitted smoothing function. See Figure 3.
  title(main = "gamgam2", outer = TRUE)


  gamgam3 <- gam::gam(Total ~ Day +sqrt(High.Temp...F.)+
                        Low.Temp...F.+
                        s(Precipitation,4),family = Gamma(link=log), data = train)
  summary(gamgam3)
  # effect of High Temp no longer significant but mse sighlty worse
```

```
    1-(gamgam3$deviance/gamgam3$null.deviance) # 0.7491259, worse
    rsq(gamgam3)
    gamgam3$deviance / (n-length(coef(gamgam3))) # 0.03606

    par(mfrow=c(1,2), mgp = c(1.8,0.5,0), mar = c(3,3,.5,0.5), oma = c(0,0,2,0))
    plot(gamgam3) # plot the fitted smoothing function. See Figure 3.
    title(main = "gamgam3", outer = TRUE)

    # CV--------
    library(boot)
    glm.1 <- glm(Total ~ Day +High.Temp...F.+Low.Temp...F.+Precipitation,
                 data = train)
    cv.err <- cv.glm(train, glm.1)$delta # leave-one-out CV
    cv.err
    rsq(glm.1)


    # gamgam2 has largest R^2 so we select this as our model for prediction

# LEVERAGE POINTS & COOKS DISTANCE-----------------------------------
  leverage = hatvalues(gamgam2)

  W = diag(gamgam2$weights)
  X = cbind(rep(1,nrow(train)), train[['Day']]=='Tuesday',
            train[['Day']]=='Wednesday',train[['Day']]=='Thursday',
            train[['Day']]=='Friday',train[['Day']]=='Saturday',train[['Day
            ]]=='Sunday',
            train[['High.Temp...F.']], train[["Low.Temp...F."]], train[["Precipitation"]])
  Hat = sqrt(W) %*% X %*% solve(t(X) %*% W %*% X) %*% t(X) %*% sqrt(W)
  all(abs(leverage - diag(Hat)) < 1e-15)

  plot(names(leverage), leverage, xlab="Index", type="h")
  points(names(leverage), leverage, pch=16, cex=0.6)
  p <- length(coef(gamgam2))
  n <- nrow(train)
  abline(h=2*p/n,col=2,lwd=2,lty=2)
  infPts <- which(leverage>2*p/n)

  #Cook's Distance

  # high Cook's distance => influential points/outliers
  # leverage points with high Cook's distance => suspicious influential points & outliers
```

```
    #                    may need to be deleted -> check scatterplots

    cooks = cooks.distance(gamgam2)
    plot(cooks, ylab="Cook's Distance", pch=16, cex=0.6)
    points(infPts, cooks[infPts], pch=17, cex=0.8, col=2)
    susPts <- as.numeric(names(sort(cooks[infPts], decreasing=TRUE)[1:3]))
    text(susPts, cooks[susPts], susPts, adj=c(-0.1,-0.1), cex=0.7, col=4)

    dispersion <- 1
    res.P = residuals(gamgam2, type="pearson")
    all(abs(cooks - (res.P/(1 - leverage))^2 * leverage/(dispersion * p) < 1e-15))
    # no overlap between leverage points and sig cooks distance

    plot(train$Total,gamgam2$fitted.values, xlab = 'True Total Values',
         ylab = 'Model Fitted Values', main = 'Fitted vs True Total Values')
    abline(coef = c(0,1))
    plot(gamgam2$fitted.values, gamgam2$residuals, xlab = 'Fitted Values',
         ylab = 'Residuals', main = 'Model Residuals vs Fitted Values')
    abline(h=0)

# PREDICTION-------------
    predict1 = predict.Gam(gamgam2,newdata = test, type = 'response')
    rmse2 = sqrt(sum((predict1 - test$Total)^2)/length(test$Total))
    c(RMSE = rmse2, R.sq =rsq(gamgam2))

    err = (predict1 - test$Total)

    plot(test$Total, predict1,xlab = 'True Total Values',
         ylab = 'Fitted Values', main = 'True vs Fitted Values')
    abline(coef = c(0,1))
    plot(test$Total,err,xlab = 'True Total Values',
         ylab = 'Residuals', main = 'Model Residuals vs True Values')
    abline(h=0)
```