W241 Final Project - Spring 2019

Erico Cruz Lemus, Jun Jun Peh, Ava Rezvani

```
#setwd("~/Desktop/EDU/W241/final_project/scriptsforthereport_v3")
library(sandwich)

## Warning: package 'sandwich' was built under R version 3.4.4

library(dplyr)
# install.packages('pwr')
library(pwr)

## Warning: package 'pwr' was built under R version 3.4.3

library(lmtest)

## Warning: package 'lmtest' was built under R version 3.4.4

## Warning: package 'zoo' was built under R version 3.4.3

library(stargazer)

## Warning: package 'stargazer' was built under R version 3.4.4

library(readxl)
library(car)

## Warning: package 'car' was built under R version 3.4.4

## Warning: package 'carData' was built under R version 3.4.4

library(ggplot2)
library(tidyr)
```

This R markdown file is separated into 3 different sections, each associated to Experiment 1,2,3 respectively as mentioned in our final report. We analyze our data with:

- Data collected in data table format
- Statistical Power
- Effect Size
- ATE (Avg diff between control and treatment)
- Linear Regression Model
- Robust Standard Error

Experiment 1: Craigslist TV

We selected Electronics - TV as our product category in Craigslist listing. This is because TV is a household electronics that can be generalized to the public regardless of the buyers age, gender, and locations. In this experiment, we created 4 different combinations of listings below:

- Control: bad quality photo and single line description.
- Treatment 1: bad quality photo with full product description (bold and highlighted words)
- Treatment 2: good quality photo with single line description
- Treatment 3: good quality photo with full product description

```
# Read in data
crg <- read.csv('craigslist_data.csv')

# Note, I randomly generated numbers to add to the 'Responses' column
# I simply drew from a random normal distribution, with mean 25 and sd
8
# and converted the responses to integers.

set.seed(42)
random_normal_data <- floor(rnorm(n=nrow(crg), mean=25, sd=8))
random_normal_data

##  [1] 35 20 27 30 28 24 37 24 41 24 35 43 13 22 23 30 22  3  5 35

crg$Responses <- random_normal_data
```

ANOVA:

```
# (1) Statistical Power
summary(lm(Responses ~ PhotoQuality*Description, data=crg))

##
## Call:
## lm(formula = Responses ~ PhotoQuality * Description, data = crg)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -22.20  -4.50  -0.60   8.75  17.80
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|
t|)
## (Intercept)                          26.400      5.167   5.110 0.000
105
## PhotoQualityGood                     -1.200      7.307  -0.164 0.871
607
## DescriptionSingle                     0.400      7.307   0.055 0.957
```

```
021
## PhotoQualityGood:DescriptionSingle    0.200     10.333   0.019 0.984
798
##
## (Intercept)                          ***
## PhotoQualityGood
## DescriptionSingle
## PhotoQualityGood:DescriptionSingle
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.55 on 16 degrees of freedom
## Multiple R-squared:  0.00343,    Adjusted R-squared:  -0.1834
## F-statistic: 0.01836 on 3 and 16 DF,  p-value: 0.9965
```

```
# For power test on full model, need numerator df and denom df and effe
ct size (R2/(1-R2))
effect.size <- 0.00343/(1-0.00343)
pwr.f2.test(u=3, v=16, f2=0.003441805, sig.level=.05, power = NULL)
```

```
##
##      Multiple regression power calculation
##
##               u = 3
##               v = 16
##              f2 = 0.003441805
##       sig.level = 0.05
##           power = 0.05321927
```

Power is 0.05321927.

```
# (2) Observation from control group
crg %>%
  group_by(Type) %>%
  summarize(avg = mean(Responses),
            sd = sd(Responses))
```

```
## # A tibble: 4 x 3
##          Type   avg        sd
##        <fctr> <dbl>     <dbl>
## 1     Control  26.8 10.329569
## 2 Treatment 1  26.4  5.128353
## 3 Treatment 2  25.8 12.477981
## 4 Treatment 3  25.2 15.658863
```

The control group (bad photo with single line description) had an average response count of 26.8. and a sd of 10.3.

(3) Baseline Model Next we are going to look at a baseline model. In a 2x2 ANOVA we have 3 total hypotheses, 2 main effects and 1 interaction Baseline model for:

- Photo quality: mean response rate is the same across levels of photo quality

- Description length: mean response rate is the same across levels of description length
- Interaction: mean response rate for photo quality does not depend on levels of description length
- Full model: there are no significant main effects nor an interactive effect

```
# (4) ATE analysis
means_df <- crg %>%
  group_by(Type) %>%
  summarize(avg = mean(Responses)) %>%
  as.data.frame()

ATE1 <- means_df[1, 2] - means_df[2,2]
ATE2 <- means_df[1, 2] - means_df[3,2]
ATE3 <- means_df[1, 2] - means_df[4,2]
```

ATE for: * Treatment 1: 0.4 * Treatment 2: 1 * Treatment 3: 1.6

```
# (5) Regression
anova(aov(Responses ~ PhotoQuality*Description, data=crg))

## Analysis of Variance Table
##
## Response: Responses
##                            Df  Sum Sq Mean Sq F value Pr(>F)
## PhotoQuality                1    6.05    6.05  0.0453 0.8341
## Description                 1    1.25    1.25  0.0094 0.9241
## PhotoQuality:Description    1    0.05    0.05  0.0004 0.9848
## Residuals                  16 2135.60  133.47

anova(lm(aov(Responses ~ PhotoQuality*Description, data=crg)))

## Analysis of Variance Table
##
## Response: Responses
##                            Df  Sum Sq Mean Sq F value Pr(>F)
## PhotoQuality                1    6.05    6.05  0.0453 0.8341
## Description                 1    1.25    1.25  0.0094 0.9241
## PhotoQuality:Description    1    0.05    0.05  0.0004 0.9848
## Residuals                  16 2135.60  133.47

# You can model a 2x2 ANOVA as a multiple regression with
# dummy-coded variables for each experimental condition.
# Here, in R passing in 'aov' to 'lm' will convert
# the ANOVA to a multiple linear regression model output.
# Statistically, ANOVA and regression give equivalent results and concl
usions.
```

There are no significant effects in this model. There are no significant differences in response rate based on photo quality. There are no significant differences in

response rate based on description length. There is no significant interaction between photo quality and description length.

(6) Modeling With Photo Quality as a 2-level categorical factor (Good vs Bad) and Description as a 2-level categorical factor (Long vs short). We also included an interaction term. Response count was our dependent variable.

```
# (7) Calculate f1 score and robust standard error

# We are using the `vcovHC` function from the library `sandwich`
# to estimate the white heteroskedastic-consistent standard errors
m1 <- lm(Responses ~ PhotoQuality * Description, data = crg)
m1.vcovHC <- vcovHC(m1)  # from library(sandwich)

# With these, we can use the `coeftest` function from the `lmtest`
# package to perform hypothesis tests.
# these are the `robust` standard errors.
coeftest(m1, vcov = m1.vcovHC)

##
## t test of coefficients:
##
##                                    Estimate Std. Error t value  Pr(>
|t|)
## (Intercept)                         26.4000     2.5642 10.2957 1.828
e-08
## PhotoQualityGood                    -1.2000     8.2386 -0.1457    0.
8860
## DescriptionSingle                    0.4000     5.7663  0.0694    0.
9456
## PhotoQualityGood:DescriptionSingle   0.2000    11.5531  0.0173    0.
9864
##
## (Intercept)                         ***
## PhotoQualityGood
## DescriptionSingle
## PhotoQualityGood:DescriptionSingle
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# To print more nicely,  we are taking the square-root of the diagonals
# of this heteroskedastic consistent variance covariance matrix, which
# provides the standard errors for each of the coefficients.
rse1 <- sqrt(diag(m1.vcovHC))
rse1

##                         (Intercept)                       PhotoQualityGoo
d
##                            2.564176                              8.23862
9
##                   DescriptionSingle PhotoQualityGood:DescriptionSingl
```

```
e
##                           5.766281                           11.55313
8

# Compares robust vs non-robust standard errors
r1 <- coeftest(m1, vcov = vcovHC(m1, type = "const"))
r2 <- coeftest(m1, vcov = vcovHC(m1, type = "HC3"))
stargazer(r1, r2, type = "text")

##
## ===============================================================
##                                         Dependent variable:
##                                     ---------------------------
##
##                                           (1)            (2)
## ---------------------------------------------------------------
## PhotoQualityGood                         -1.200        -1.200
##                                         (7.307)        (8.239)
##
## DescriptionSingle                         0.400          0.400
##                                         (7.307)        (5.766)
##
## PhotoQualityGood:DescriptionSingle        0.200          0.200
##                                         (10.333)       (11.553)
##
## Constant                                26.400***      26.400***
##                                          (5.167)        (2.564)
##
## ===============================================================
## ===============================================================
## Note:                                 *p<0.1; **p<0.05; ***p<0.01
```

Poisson

```
# (1) Statistical Power
pos.mod <- glm(Responses ~ PhotoQuality * Description,
               data=crg, family=poisson)
summary(pos.mod)

##
## Call:
## glm(formula = Responses ~ PhotoQuality * Description, family = poiss
on,
##     data = crg)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -5.6241  -0.9008  -0.1211   1.6243   3.2179
##
## Coefficients:
##                                     Estimate Std. Error z value Pr(>
```

```
|z|)
## (Intercept)                          3.273364   0.087039  37.608   <2
e-16
## PhotoQualityGood                    -0.046520   0.124548  -0.374    0
.709
## DescriptionSingle                    0.015038   0.122631   0.123    0
.902
## PhotoQualityGood:DescriptionSingle   0.008493   0.175291   0.048    0
.961
##
## (Intercept)                         ***
## PhotoQualityGood
## DescriptionSingle
## PhotoQualityGood:DescriptionSingle
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 100.68  on 19  degrees of freedom
## Residual deviance: 100.40  on 16  degrees of freedom
## AIC: 207.66
##
## Number of Fisher Scoring iterations: 5
```

Note: In Poisson GLM we need to test for overdispersion and correct if it is present. Because we randomly drew from a normal and NOT a Poisson with counts (e.g. rpois(10, 20)). We are getting overdispersion (greater variability than expected). In Poisson, mean parameter is supposed to be equal to the variance parameter. https://en.wikipedia.org/wiki/Overdispersion.

```
#install.packages('AER')
library(AER)

## Warning: package 'AER' was built under R version 3.4.4

## Loading required package: survival

dispersiontest(pos.mod,trafo=1)

##
##  Overdispersion test
##
## data:  pos.mod
## z = 2.3482, p-value = 0.009433
## alternative hypothesis: true alpha is greater than 0
## sample estimates:
##   alpha
## 3.14852
```

```
# Caution: power tests here assume a R2 value.
# Since Poisson doesn't produce R2, we use a pseudo-R2
# 1-(Residual Deviance/Null Deviance)
# Ref: (https://stats.stackexchange.com/questions/11676/pseudo-r-squared-formula-for-glms)
pseudo_r2 <- 1-(100.40/110.68)
pseudo_r2/(1-pseudo_r2)

## [1] 0.1023904

pwr.f2.test(u=3, v=16, f2=pseudo_r2/(1-pseudo_r2), sig.level = .05)

##
##      Multiple regression power calculation
##
##               u = 3
##               v = 16
##              f2 = 0.1023904
##       sig.level = 0.05
##           power = 0.1622645
```

Power is 0.5503953

```
# 2) Observation from control group
crg %>%
  group_by(Type) %>%
  summarize(avg = mean(Responses),
            sd = sd(Responses))

## # A tibble: 4 x 3
##          Type   avg        sd
##        <fctr> <dbl>     <dbl>
## 1     Control  26.8 10.329569
## 2 Treatment 1  26.4  5.128353
## 3 Treatment 2  25.8 12.477981
## 4 Treatment 3  25.2 15.658863

# The control group (bad photo with single line description)
# had an average response count of 26.8.
```

3) Baseline model. In this Poisson regression we have 3 total hypotheses: 2 main effects and 1 interaction. Baseline model for:

- Photo quality: mean response rate is the same across levels of photo quality
- Description length: mean response rate is the same across levels of description length
- Interaction: mean response rate for photo quality does not depend on levels of description length
- Full model: there are no significant main effects nor an interactive effect

```
# 4) ATE analysis
means_df <- crg %>%
```

```
  group_by(Type) %>%
  summarize(avg = mean(Responses)) %>%
  as.data.frame()

ATE1 <- means_df[1, 2] - means_df[2,2]
ATE2 <- means_df[1, 2] - means_df[3,2]
ATE3 <- means_df[1, 2] - means_df[4,2]
```

ATE for: * Treatment 1: 0.4 * Treatment 2: 1 * Treatment 3: 1.6

```
# 6) Regression
pos.mod <- glm(Responses ~ PhotoQuality * Description,
               data=crg, family=poisson)
summary(pos.mod)

##
## Call:
## glm(formula = Responses ~ PhotoQuality * Description, family = poiss
on,
##     data = crg)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -5.6241  -0.9008  -0.1211   1.6243   3.2179
##
## Coefficients:
##                                    Estimate Std. Error z value Pr(>
|z|)
## (Intercept)                         3.273364   0.087039  37.608   <2
e-16
## PhotoQualityGood                   -0.046520   0.124548  -0.374    0
.709
## DescriptionSingle                   0.015038   0.122631   0.123    0
.902
## PhotoQualityGood:DescriptionSingle  0.008493   0.175291   0.048    0
.961
##
## (Intercept)                        ***
## PhotoQualityGood
## DescriptionSingle
## PhotoQualityGood:DescriptionSingle
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 100.68  on 19  degrees of freedom
## Residual deviance: 100.40  on 16  degrees of freedom
## AIC: 207.66
```

```
##
## Number of Fisher Scoring iterations: 5
```

You can model the experiment as a generalized linear model (GLM) with dummy-coded variables for each experimental condition. Here, in R we specify the outcome is Poisson distributed as is a common error distribution for integer count data

- There are no significant effects in this model.
- There are no significant differences in response rate based on * There are no significant differences in response rate based on description length.
- There is no significant interaction between photo quality and description length.

(7) Modeling We used a GLM with Photo Quality as a binary independent variable (Good vs Bad) and Description as a binary independent variable (Long vs short). We also included an interaction term. Response count was our dependent variable. We specified the error distribution to be Poisson, and chose a logarithm as the link function. In other words, the mean of the response is mapped to the linear combination of features via the logarithm function.

```
# 8) Calculate f1 score and robust standard error

# We are using the `vcovHC` function from the library `sandwich`
# to estimate the white heteroskedastic-consistent standard errors
pos.mod.vcovHC <- vcovHC(pos.mod)  # from library(sandwich)

# With these, we can use the `coeftest` function from the `lmtest`
# package to perform hypothesis tests.
# these are the `robust` standard errors.
coeftest(pos.mod, vcov = pos.mod.vcovHC)

##
## z test of coefficients:
##
##                                      Estimate Std. Error z value Pr(
>|z|)
## (Intercept)                          3.2733640  0.0971279 33.7016   <
2e-16
## PhotoQualityGood                    -0.0465200  0.3255199 -0.1429   0
.8864
## DescriptionSingle                    0.0150379  0.2158083  0.0697   0
.9444
## PhotoQualityGood:DescriptionSingle   0.0084926  0.4489768  0.0189   0
.9849
##
## (Intercept)                          ***
## PhotoQualityGood
## DescriptionSingle
## PhotoQualityGood:DescriptionSingle
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# To print more nicely,  we are taking the square-root of the diagonals
# of this heteroskedastic consistent variance covariance matrix, which
# provides the standard errors for each of the coefficients.
rse1 <- sqrt(diag(pos.mod.vcovHC))
rse1
```

```
##                          (Intercept)                    PhotoQualityGoo
d
##                           0.09712789                          0.3255198
6
##                    DescriptionSingle PhotoQualityGood:DescriptionSingl
e
##                           0.21580830                          0.4489767
6
```

```r
# Compares robust vs non-robust standard errors
r1 <- coeftest(pos.mod, vcov = vcovHC(pos.mod, type = "const"))
r2 <- coeftest(pos.mod, vcov = vcovHC(pos.mod, type = "HC3"))
stargazer(r1, r2, type = "text")
```

```
##
## ===============================================================
##                                         Dependent variable:
##                                     ---------------------------
##
##                                           (1)            (2)
## ---------------------------------------------------------------
## PhotoQualityGood                         -0.047        -0.047
##                                         (0.283)        (0.326)
##
## DescriptionSingle                         0.015          0.015
##                                         (0.275)        (0.216)
##
## PhotoQualityGood:DescriptionSingle        0.008          0.008
##                                         (0.397)        (0.449)
##
## Constant                                3.273***       3.273***
##                                         (0.196)        (0.097)
##
## ===============================================================
## ===============================================================
## Note:                                 *p<0.1; **p<0.05; ***p<0.01
```

Experiment 2: Craigslist Camera Lens

There were several complications after Experiment 1, including many posts being marked as spam and being removed by Craigslist. Additionally, the Electronics - TV market was highlight saturated and therefore for posts that remained, we received few responses.

We decided to create a simpler study, which comprises only a control and a treatment listing using the same camera lens in pilot study. Instead of experimenting with the photo quality, that created 2 extra permutations in the treatments, we controlled the photos. This experiment studied the effect in an unsaturated market compared to TV, as camera lens listing is only targeting specific group of buyers. Variation on description below were used in this study:

- Control: Normal photos and description with typos
- Treatment: Normal photos and descriptions without typo

```
# Read in data
exp2_df <- read_excel('Experiments 1 and 2.xlsx', sheet=2)
exp2_df <- as.data.frame(exp2_df[, c('Condition', 'Response_Count')])

# To generate more data, you can do as follows:
# set.seed(42)
# new_data <- data.frame('Condition' = rep(c('Control', 'Treatment'), e
ach = 10),
#                        'Response_Count' = c(floor(rnorm(10, 15, 3)), f
loor(rnorm(10, 20, 6))))
# This randomly generates responses for control (mean ~15) and treateme
nt (mean ~20)
```

T.Test

```
# 1) Statistical Power
t.test(exp2_df$Response ~ exp2_df$Condition, var.equal = TRUE)

##
##  Two Sample t-test
##
## data:  exp2_df$Response by exp2_df$Condition
## t = -1.1191, df = 18, p-value = 0.2778
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -4.316072  1.316072
## sample estimates:
##   mean in group Control mean in group Treatment
##                     2.2                     3.7

t_obtained <- t.test(exp2_df$Response ~ exp2_df$Condition, var.equal =
TRUE)$statistic
```

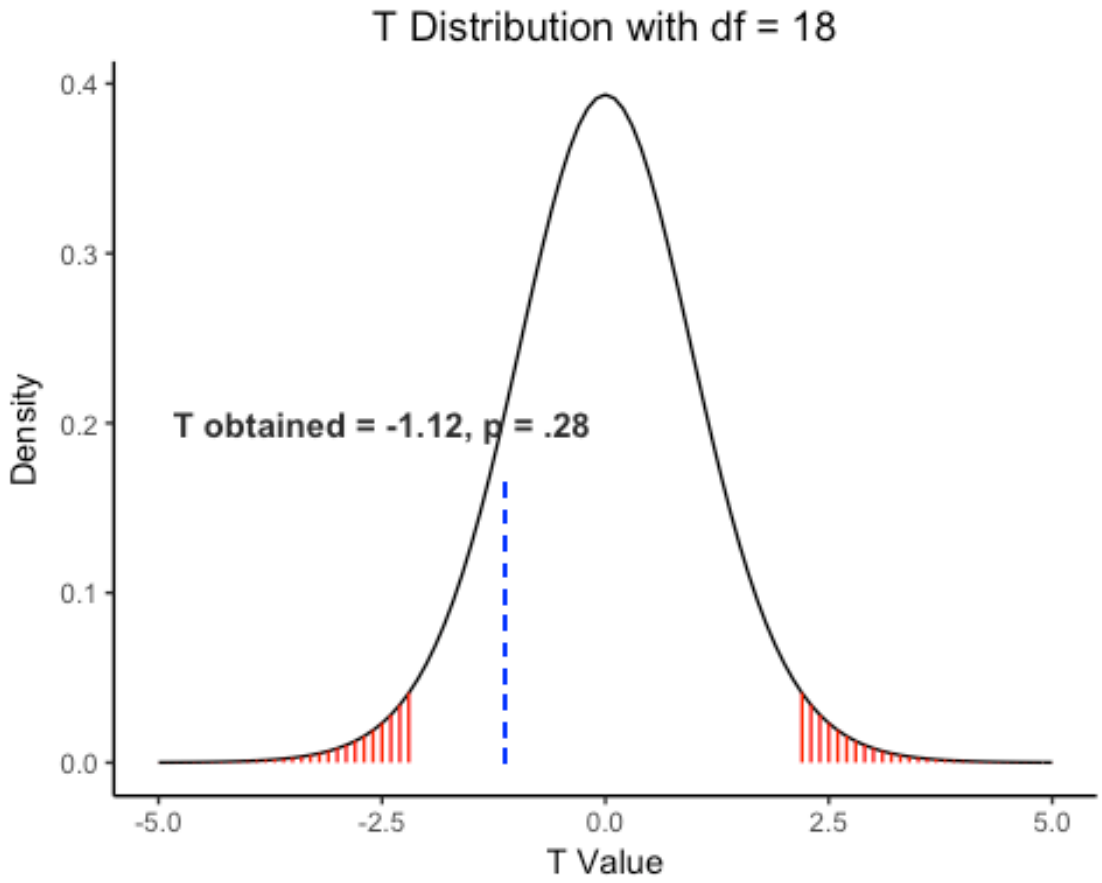
```
theoretical_ts <- data.frame(x = seq(-5, 5, by = .001),
                             t = dt(seq(-5, 5, by = .001), 18))
```

This plots the theoretical T distribution. In a two-tailed test, the rejection region is at -2.10092 or + 2.10092. As you can see, our obtained T-value falls outside the rejection region, which means we do NOT reject the null. Here, we have no evidence of a significant effect.

```
qt(.975, 18)
```

```
## [1] 2.100922
```

```
df   <- 18
gg   <- data.frame(x=seq(-5,5, 0.1))
gg$y <- dt(gg$x,df)

ggplot(gg) +
  geom_path(aes(x,y)) +
  geom_linerange(data=gg[gg$x < -qt(.975, 18) | gg$x > qt(.975, 18),],
                 aes(x, ymin=0, ymax=y),
                 colour="red") +
  ggtitle('T Distribution with df = 18') +
  xlab('T Value') +
  ylab('Density') +
  theme_bw() +
  theme(panel.border = element_blank(), panel.grid.major = element_blan
k(),
        panel.grid.minor = element_blank(), axis.line = element_line(co
lour = "black"),
        plot.title = element_text(hjust = 0.5)) +
  annotate(geom="text", x = -2.5, y = 0.2,
           label = "T obtained = -1.12, p = .28",
           fontface="bold", color='grey17') +
  geom_segment(aes(x = t_obtained, y = 0, xend = t_obtained, yend = .17
),
               data = gg, linetype='dashed', color='blue')
```




```r
# In order to calculate power on a 2-group comparison,
# we first need to calculate the Cohen's effect size.

# Calculate effect size
cohens_d <- function(x, y) {
  lx <- length(x)- 1
  ly <- length(y)- 1
  md  <- abs(mean(x) - mean(y))        ## mean difference (numerator)
  csd <- lx * var(x) + ly * var(y)
  csd <- csd/(lx + ly)
  csd <- sqrt(csd)                     ## common sd computation

  cd  <- md/csd                        ## cohen's d
}

exp2_cohens_d <- cohens_d(
  exp2_df[exp2_df$Condition == 'Control', 'Response_Count'],
  exp2_df[exp2_df$Condition == 'Treatment', 'Response_Count']
)
exp2_cohens_d

## [1] 0.5004636
```

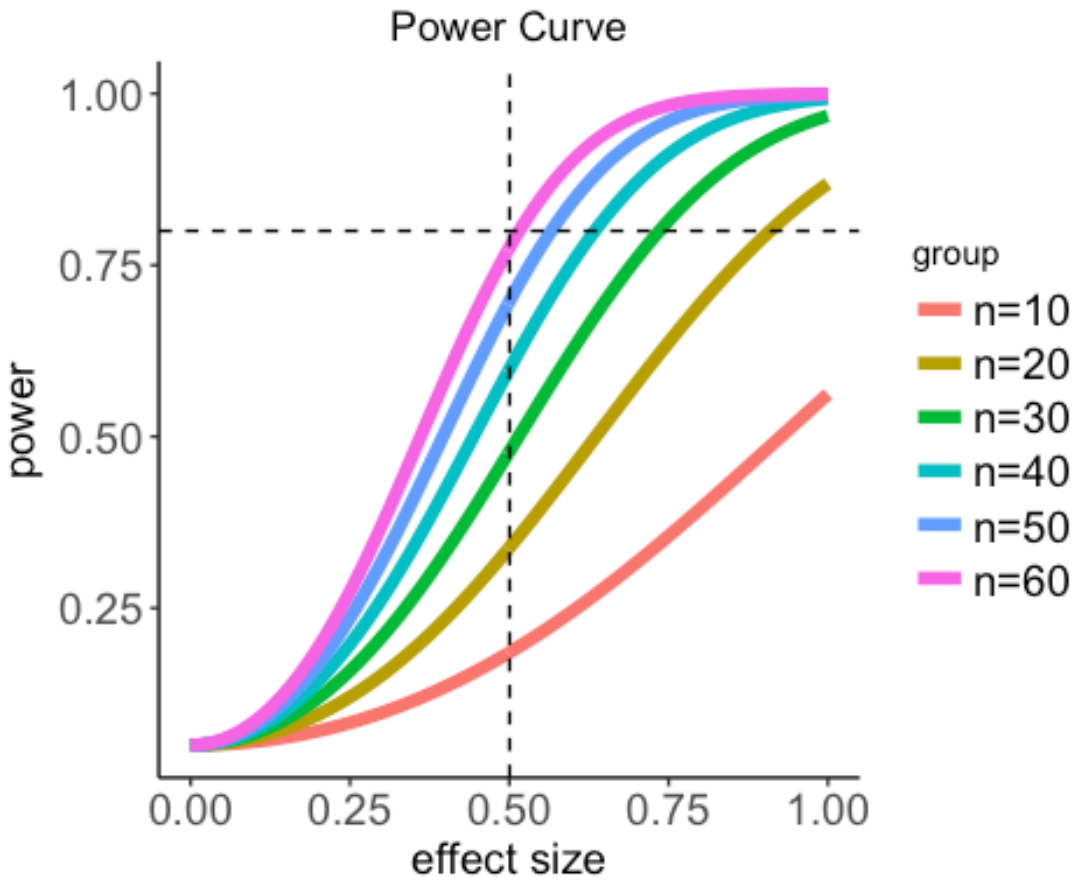
```r
# Use the pwr package with appropriate test
# Because we are using a t-test with 2 groups
# we use the pwr.t.test function

pwr.t.test(n = 10, d = exp2_cohens_d, sig.level = 0.05,
           type = 'two.sample', alternative = 'two.sided')

##
##      Two-sample t test power calculation
##
##               n = 10
##               d = 0.5004636
##       sig.level = 0.05
##           power = 0.1853525
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

Power Vis

Here I plot a curve for an effect size of .5004536. Assuming this effect size is the true population effect size. Here are the lines with power levels needed to detect an effect 80% power is recommended.Here you can see, with our sample size of n = 10 each in the current experiment, we are underpowered. The curve shows a recommendation of n=60 for each sample.

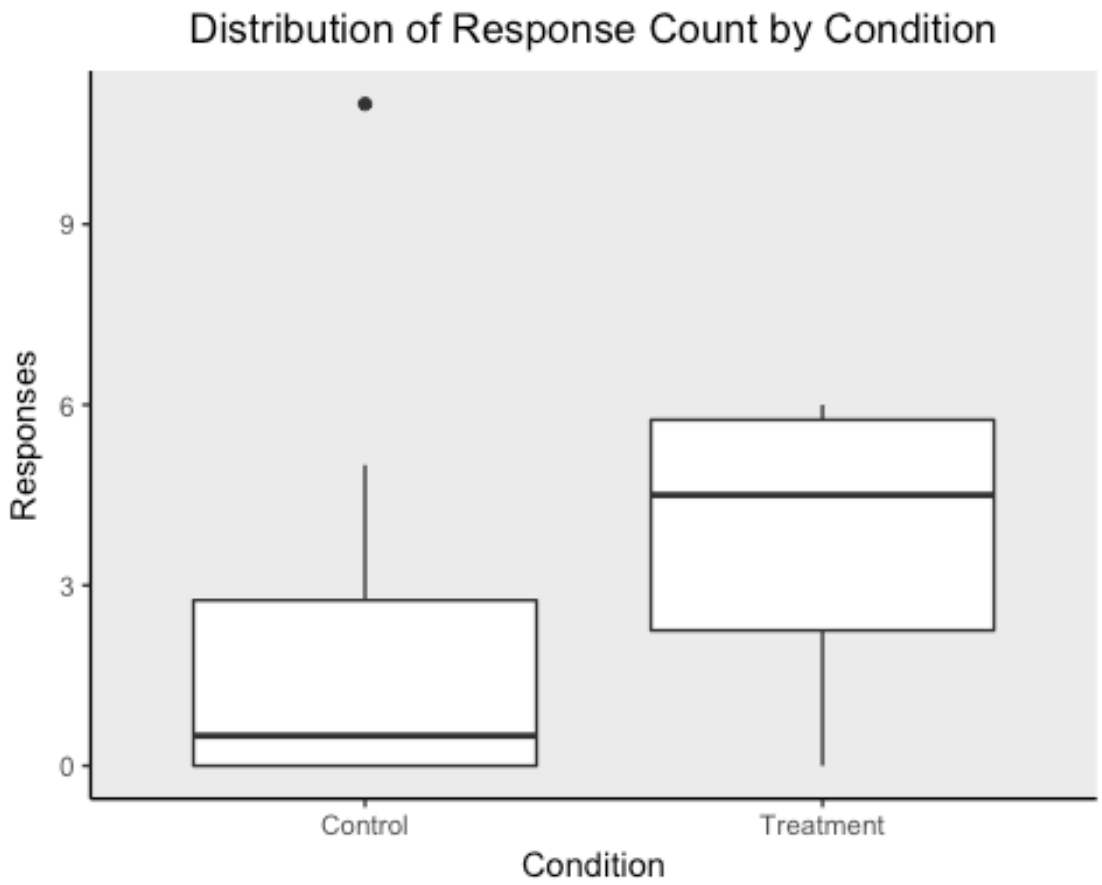
```r
# Generate power calculations
ptab <- cbind(NULL, NULL)

for (i in seq(0,1, length.out = 200)){
  pwrt1 <- pwr.t.test(n = 10, d = i, sig.level = 0.05,
                      type = 'two.sample',
                      alternative = 'two.sided')

  pwrt2 <- pwr.t.test(n = 20, d = i, sig.level = 0.05,
                      type = 'two.sample',
                      alternative = 'two.sided')

  pwrt3 <- pwr.t.test(n = 30, d = i, sig.level = 0.05,
                      type = 'two.sample',
                      alternative = 'two.sided')

  pwrt4 <- pwr.t.test(n = 40, d = i, sig.level = 0.05,
                      type = 'two.sample',
                      alternative = 'two.sided')

  pwrt5 <- pwr.t.test(n = 50, d = i, sig.level = 0.05,
                      type = 'two.sample',
                      alternative = 'two.sided')
```

```r
  pwrt6 <- pwr.t.test(n = 60, d = i, sig.level = 0.05,
                      type = 'two.sample',
                      alternative = 'two.sided')

  ptab <- rbind(ptab, cbind(pwrt1$d, pwrt1$power,
                            pwrt2$d, pwrt2$power,
                            pwrt3$d, pwrt3$power,
                            pwrt4$d, pwrt4$power,
                            pwrt5$d, pwrt5$power,
                            pwrt6$d, pwrt6$power))
}

ptab <- cbind(seq_len(nrow(ptab)), ptab)

colnames(ptab) <- c("id","n=10.effect size","n=10.power",
                    "n=20.effect size","n=20.power",
                    "n=30.effect size","n=30.power",
                    "n=40.effect size","n=40.power",
                    "n=50.effect size","n=50.power",
                    "n=60.effect size","n=60.power")

# get data into right format for ggplot2
temp <- ptab %>%
  as.data.frame() %>%
  gather(key = name, value = val, 2:13) %>%
  separate(col = name, into = c("group", "var"), sep = "\\.") %>%
  spread(key = var, value = val)

# factor group
temp$sample_group <- factor(temp$group,
                            levels = c("n=10", "n=20", "n=30",
                                       "n=40", "n=50", "n=60"))



# plot
p <- ggplot(temp, aes(x = `effect size`, y = power, color = group))
p + geom_line(size=2) +
  theme_bw() +
  theme(axis.text=element_text(size=14),
        axis.title=element_text(size=14),
        legend.text=element_text(size=14)) +
  geom_vline(xintercept = exp2_cohens_d, linetype = 2) +
  geom_hline(yintercept = 0.80, linetype = 2) +
  ggtitle('Power Curve') +
  theme(panel.border = element_blank(), panel.grid.major = element_blan
k(),
        panel.grid.minor = element_blank(), axis.line = element_line(co
lour = "black"),
        plot.title = element_text(hjust = 0.5))
```




```
# Power is 0.1853525

# 2) Observation from control group
exp2_df %>%
  group_by(Condition) %>%
  summarize(avg = mean(Response_Count),
            sd = sd(Response_Count))

## # A tibble: 2 x 3
##   Condition   avg       sd
##       <chr> <dbl>    <dbl>
## 1   Control   2.2 3.521363
## 2 Treatment   3.7 2.359378

# Boxplots of response counts
ggplot(exp2_df, aes(factor(Condition), Response_Count)) +
  geom_boxplot() +
  ggtitle('Distribution of Response Count by Condition') +
  xlab('Condition') +
  ylab('Responses') +
  theme(panel.border = element_blank(), panel.grid.major = element_blan
k(),
        panel.grid.minor = element_blank(), axis.line = element_line(co
```

```
lour = "black"),
        plot.title = element_text(hjust = 0.5))
```



The control group (normal descriptions with typo) had an average response count of 2.20 and standard deviation of 3.52

(3) Baseline model In a two-sample T.test (independent groups) we have one hypothesis Null hypothesis: there are no significant differences in response count to Craigslist camera lens ads between ad descriptions that have a typo vs don't.

```
# 4) ATE analysis
exp2_means_df <- exp2_df %>%
  group_by(Condition) %>%
  summarize(avg = mean(Response_Count)) %>%
  as.data.frame()

exp2_ATE <- exp2_means_df[2, 2] - exp2_means_df[1,2]
exp2_ATE

## [1] 1.5

# ATE for Treatment condition: 1.5

# 6) Regression
```

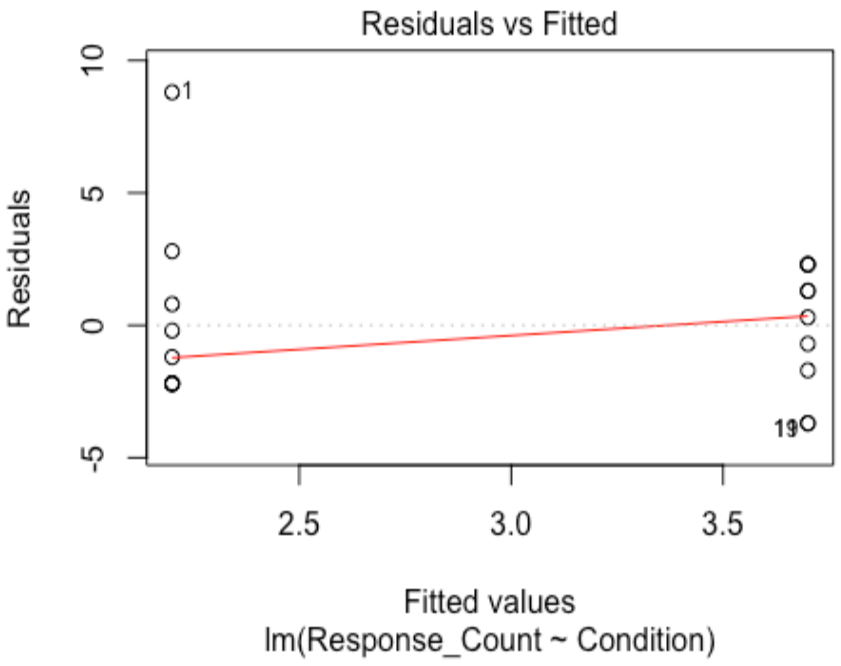
```r
summary(lm(Response_Count ~ Condition, data=exp2_df))
```

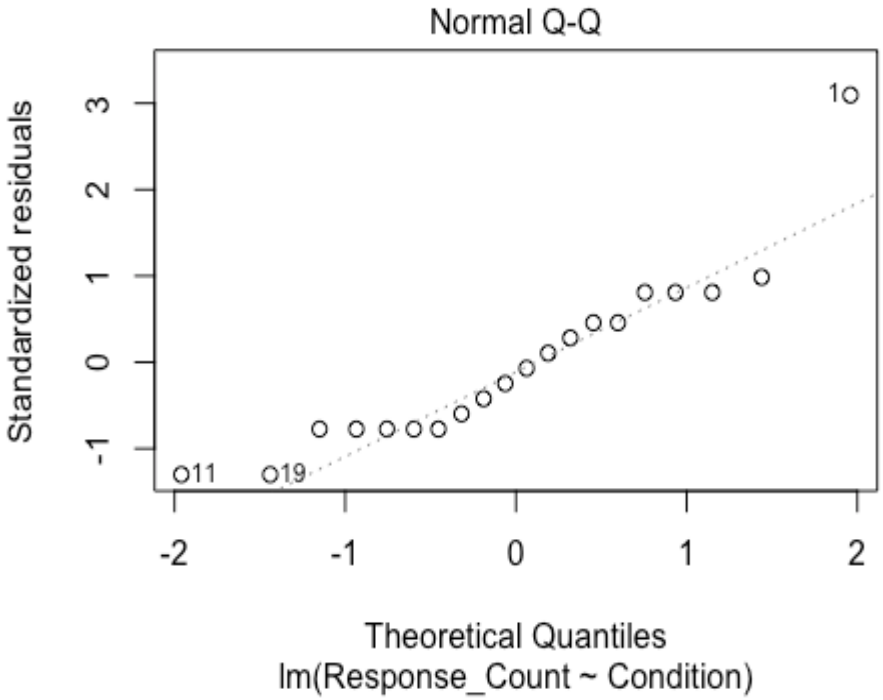
```
##
## Call:
## lm(formula = Response_Count ~ Condition, data = exp2_df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -3.70  -2.20  -0.45   1.55   8.80
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)          2.2000     0.9478   2.321   0.0322 *
## ConditionTreatment   1.5000     1.3404   1.119   0.2778
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.997 on 18 degrees of freedom
## Multiple R-squared:  0.06505,    Adjusted R-squared:  0.01311
## F-statistic: 1.252 on 1 and 18 DF,  p-value: 0.2778
```

(Optional... though not interesting).

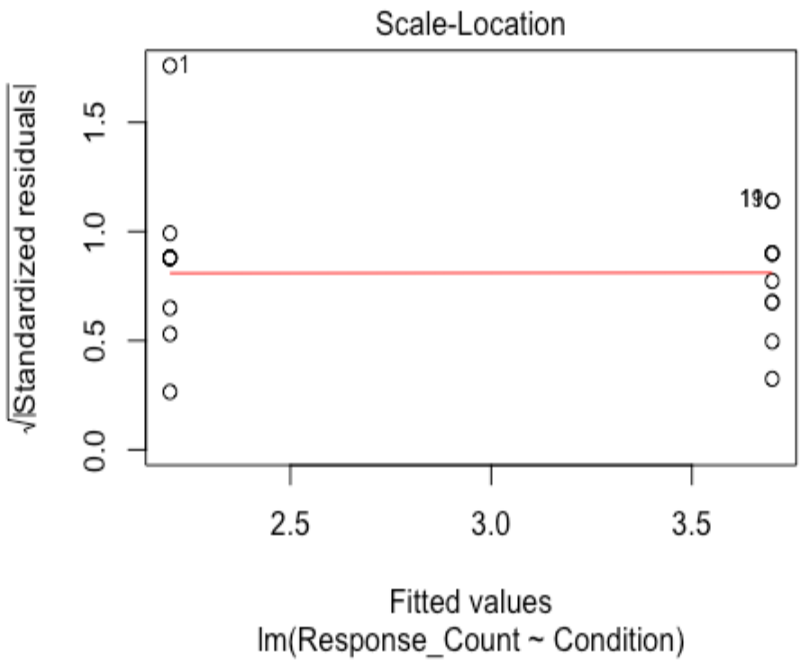
Plotting residuals of the linear model

```r
plot(lm(Response_Count ~ Condition, data=exp2_df))
```






```
## hat values (leverages) are all = 0.1
##  and there are no factor predictors; no plot no. 5
```



In regression output, when we have a single binary predictor (0 vs 1) the intercept is just the mean of the 0 group and the coefficient for the predictor is the average difference between being in a 0 group vs a 1 group. For each 1-unit increase in the predictor, Y changes by the amount of the coefficient weight. So, going from 0 to 1 in this case, means Y changes by 1.5.

```
# This is equivalent to a two-sample T-test
t.test(exp2_df$Response ~ exp2_df$Condition, var.equal = TRUE)

##
##  Two Sample t-test
##
## data:  exp2_df$Response by exp2_df$Condition
## t = -1.1191, df = 18, p-value = 0.2778
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -4.316072  1.316072
## sample estimates:
##   mean in group Control mean in group Treatment
##                     2.2                     3.7

# Example of T-test in R that does not assume equal variances
t.test(exp2_df$Response_Count ~ exp2_df$Condition)

##
##  Welch Two Sample t-test
##
```

```
## data:  exp2_df$Response_Count by exp2_df$Condition
## t = -1.1191, df = 15.725, p-value = 0.2799
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -4.345557  1.345557
## sample estimates:
##   mean in group Control mean in group Treatment
##                     2.2                     3.7
```

In all cases, results show no significant difference between conditions on the response count. Regression output shows the coefficient for condition is not significantly different from 0, t(18) = 1.119, p = 0.2778.

In most experiments, a minimum sample size of 30 is recommended per condition. To assess whether response count differs by condition, we need to know 3 pieces of information: * Means of the groups (here 2.2 and 3.7) * Sample size of each group * Standard deviation of Y (Response_Count) in each group

T stat is computed as mean(group1)-mean(group2) / standard error. Standard error is sqrt(N)/ pooled standard deviation

We can increase our ability to detect an effect (aka power) By either: * Increasing our effect size (e.g. mean differences) * Decreasing noise (e.g. minimizing standard deviations in Y by group) * Increasing sample size

Modeling

We used an independent groups T-Test to support the experiment of assigning subjects (cities) to condition with Condition as a 2-level categorical factor (Control vs Treatment). The control condition had no typo in the ad description, whereas the treatment condition did have a typo. Response count was our dependent variable.

```
# 8) Calculate f1 score and robust standard error

# We are using the `vcovHC` function from the library `sandwich`
# to estimate the white heteroskedastic-consistent standard errors
m1 <- lm(Response_Count ~ Condition, data = exp2_df)
m1.vcovHC <- vcovHC(m1)  # from library(sandwich)

# With these, we can use the `coeftest` function from the `lmtest`
# package to perform hypothesis tests.
# these are the `robust` standard errors.
coeftest(m1, vcov = m1.vcovHC)

##
## t test of coefficients:
##
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)          2.2000     1.1738  1.8743  0.07722 .
## ConditionTreatment   1.5000     1.4129  1.0616  0.30243
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The following plots the coefficients and predictors of the mode, visualizing the standard error and the confidence intervals. As a rule of thumb, if 0 is contained in the confidence interval the effect is not significant. Here we see a clear non-significant treatment effect and a marginally significant intercept.This is consistent with the p values on the model output.

source("https://www.r-statistics.com/wp-content/uploads/2010/07/coefplot.r.txt")

```
#coefplot(m1, main='Confidence Intervals by Predictor')

# To print more nicely,  we are taking the square-root of the diagonals
# of this heteroskedastic consistent variance covariance matrix, which
# provides the standard errors for each of the coefficients.
rse1 <- sqrt(diag(m1.vcovHC))
rse1

##        (Intercept) ConditionTreatment
##           1.173788           1.412903

# Compares robust vs non-robust standard errors
r1 <- coeftest(m1, vcov = vcovHC(m1, type = "const"))
r2 <- coeftest(m1, vcov = vcovHC(m1, type = "HC3"))
stargazer(r1, r2, type = "text")

##
## ===============================================
##                         Dependent variable:
##                     ---------------------------
##
##                           (1)            (2)
## -----------------------------------------------
## ConditionTreatment     1.500          1.500
##                       (1.340)        (1.413)
##
## Constant              2.200**        2.200*
##                       (0.948)        (1.174)
##
## ===============================================
## ===============================================
## Note:               *p<0.1; **p<0.05; ***p<0.01
```

In this case, using robust standard errors. Our T-statistic decreased from 1.11 to 1.061571. Practically, this means it weakened our effect.

Non-parametric for test on counts

Mann-Whitney U-test tests if counts in one group tend to be higher than counts in another group. Given our sample size is low here (10). We cannot be confident our T statistics calculated with parametric tests are reliable. Also, T tests require that data be distributed normally

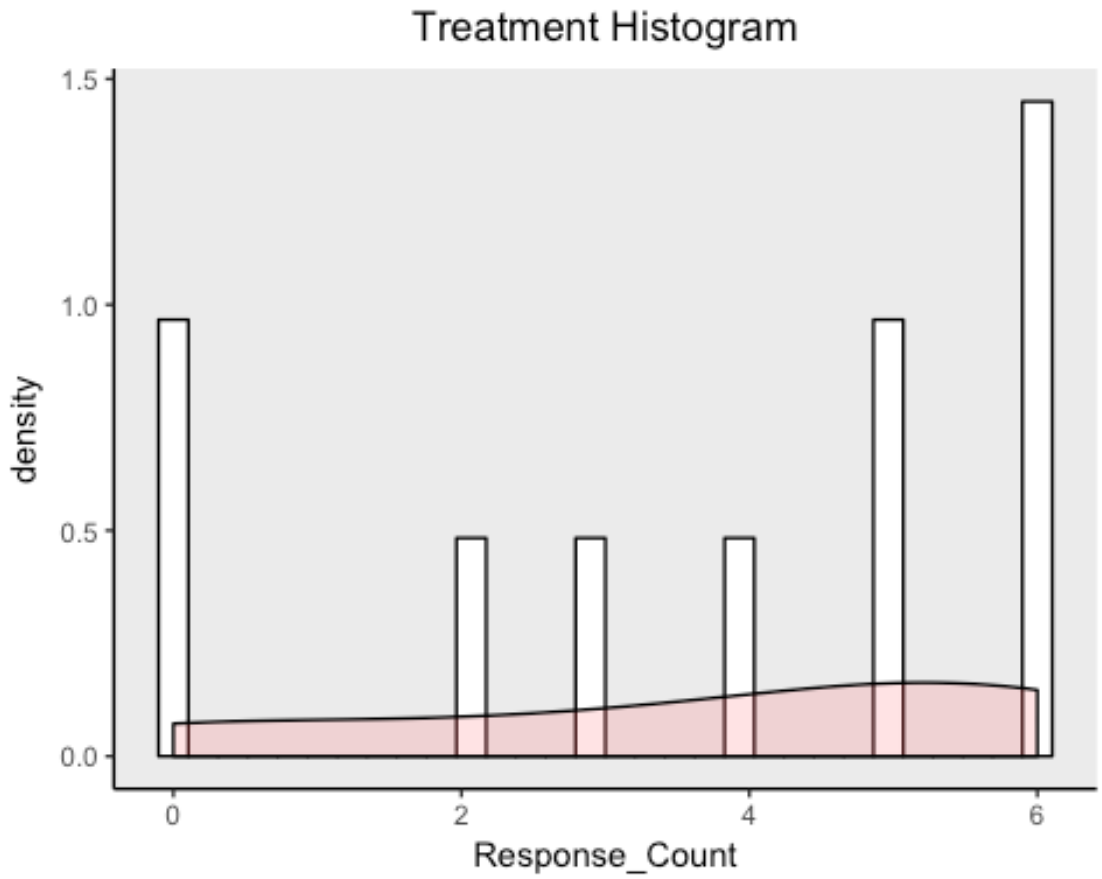
```
shapiro.test(exp2_df[exp2_df$Condition == 'Control', 'Response_Count'])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  exp2_df[exp2_df$Condition == "Control", "Response_Count"]
## W = 0.70729, p-value = 0.00108
```

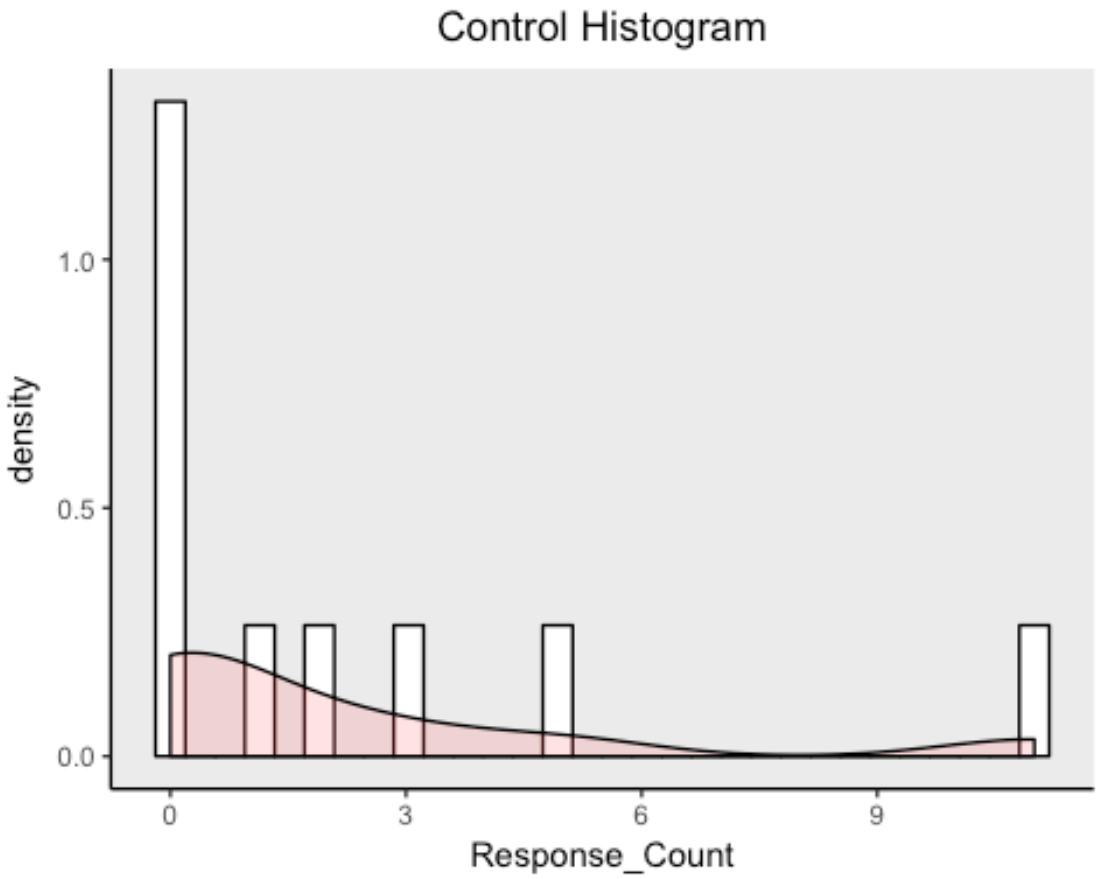
```
# These plots show the non-normal distribution of the outcome
ggplot(exp2_df[exp2_df$Condition == 'Treatment', ],
       aes(x=Response_Count)) +
  geom_histogram(aes(y=..density..), colour="black", fill="white", bind
width=50)+
  geom_density(alpha=.2, fill="#FF6666") +
  ggtitle('Treatment Histogram') +
  theme(panel.border = element_blank(), panel.grid.major = element_blan
k(),
        panel.grid.minor = element_blank(), axis.line = element_line(co
lour = "black"),
        plot.title = element_text(hjust = 0.5))
```

```
## Warning: Ignoring unknown parameters: bindwidth
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(exp2_df[exp2_df$Condition == 'Control', ],
       aes(x=Response_Count)) +
  geom_histogram(aes(y=..density..), colour="black", fill="white", bind
width=50)+
  geom_density(alpha=.2, fill="#FF6666") +
  ggtitle('Control Histogram') +
  theme(panel.border = element_blank(), panel.grid.major = element_blan
k(),
        panel.grid.minor = element_blank(), axis.line = element_line(co
lour = "black"),
        plot.title = element_text(hjust = 0.5))

## Warning: Ignoring unknown parameters: bindwidth

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Here we see in the control group we have non-normal data. Furthermore, with count data, often it is Poisson-distributed, so it may be preferred to relax the normality assumption of the T-test.

```
# A non-parametric test alternative therefore may be preferred.
wilcox.test(exp2_df[exp2_df$Condition == 'Control', 'Response_Count'],
            exp2_df[exp2_df$Condition == 'Treatment', 'Response_Count']
)
```

```
## Warning in wilcox.test.default(exp2_df[exp2_df$Condition == "Control
",
## "Response_Count"], : cannot compute exact p-value with ties
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  exp2_df[exp2_df$Condition == "Control", "Response_Count"] and
exp2_df[exp2_df$Condition == "Treatment", "Response_Count"]
## W = 29, p-value = 0.1119
## alternative hypothesis: true location shift is not equal to 0
```

```
# Results show no significant difference in counts between conditions
# W = 29, p = 0.1119.
```

Poisson Distribution

```
# 1) Statistical Power
pos.mod <- glm(Response_Count ~ Condition, data=exp2_df, family=poisson
)
summary(pos.mod)
```

```
##
## Call:
## glm(formula = Response_Count ~ Condition, family = poisson, data = e
xp2_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7203  -2.0976  -0.2567   0.7548   4.2199
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)           0.7885     0.2132   3.698 0.000217 ***
## ConditionTreatment    0.5199     0.2692   1.931 0.053481 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 67.706  on 19  degrees of freedom
## Residual deviance: 63.850  on 18  degrees of freedom
## AIC: 109.99
##
## Number of Fisher Scoring iterations: 6
```

Caution: power tests here assume a R2 value. Since Poisson doesn't produce R2, we use a pseudo-R2, 1-(Residual Deviance/Null Deviance). Ref: (https://stats.stackexchange.com/questions/11676/pseudo-r-squared-formula-for-glms)

```
pseudo_r2 <- 1-(55.153/56.679)
effect.size <- pseudo_r2/(1-pseudo_r2)
pwr.f2.test(u=1, v=18, f2=effect.size, sig.level = .05)
```

```
##
##      Multiple regression power calculation
##
##               u = 1
##               v = 18
##              f2 = 0.02766849
##       sig.level = 0.05
##           power = 0.1085755
```

Power is 0.1085755.

```r
# 2) Observation from control group
exp2_df %>%
  group_by(Condition) %>%
  summarize(avg = mean(Response_Count),
            sd = sd(Response_Count))
```

```
## # A tibble: 2 x 3
##   Condition   avg       sd
##       <chr> <dbl>    <dbl>
## 1   Control   2.2 3.521363
## 2 Treatment   3.7 2.359378
```

The control group (normal descriptions with typo) had an average response count of 16.2 and standard deviation of 2.66 #### (3) Baseline model In this Poisson regression we have one hypothesis Null hypothesis: there are no significant differences in response count to Craigslist camera lens ads between ads descriptions that have a typo vs don't.

```r
# 4) ATE analysis
exp2_means_df <- exp2_df %>%
  group_by(Condition) %>%
  summarize(avg = mean(Response_Count)) %>%
  as.data.frame()

exp2_ATE <- exp2_means_df[2, 2] - exp2_means_df[1,2]

# ATE for Treatment condition: 2.3

# 6) Regression
pos.mod <- glm(Response_Count ~ Condition, data=exp2_df, family=poisson
)
summary(pos.mod)
```

```
##
## Call:
## glm(formula = Response_Count ~ Condition, family = poisson, data = e
xp2_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7203  -2.0976  -0.2567   0.7548   4.2199
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)          0.7885     0.2132   3.698 0.000217 ***
## ConditionTreatment   0.5199     0.2692   1.931 0.053481 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
```

```
##
##     Null deviance: 67.706  on 19  degrees of freedom
## Residual deviance: 63.850  on 18  degrees of freedom
## AIC: 109.99
##
## Number of Fisher Scoring iterations: 6
```

There are no significant effects in this model. z = 1.234, p = 0.217

```
library(AER)
dispersiontest(pos.mod,trafo=1)

##
##  Overdispersion test
##
## data:  pos.mod
## z = 1.4606, p-value = 0.07206
## alternative hypothesis: true alpha is greater than 0
## sample estimates:
##    alpha
## 2.213391

# There is marginal overdispersion, but not significant
```

(7) Modeling with Condition as a 2-level categorical factor (Control vs Treatment). Response count was our dependent variable.We specified the error distribution to be Poisson, and chose a logarithm as the link function. In other words, the mean of the response is mapped to the linear combination of features via the logarithm function.

```
# 8) Calculate f1 score and robust standard error

# We are using the `vcovHC` function from the library `sandwich`
# to estimate the white heteroskedastic-consistent standard errors
pos.mod.vcovHC <- vcovHC(pos.mod)  # from library(sandwich)

# With these, we can use the `coeftest` function from the `lmtest`
# package to perform hypothesis tests.
# these are the `robust` standard errors.
coeftest(pos.mod, vcov = pos.mod.vcovHC)

##
## z test of coefficients:
##
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)          0.78846    0.53354  1.4778   0.1395
## ConditionTreatment  0.51988    0.57432  0.9052   0.3654

# To print more nicely,  we are taking the square-root of the diagonals
# of this heteroskedastic consistent variance covariance matrix, which
# provides the standard errors for each of the coefficients.
rse1 <- sqrt(diag(pos.mod.vcovHC))
rse1
```

```
##        (Intercept) ConditionTreatment
##          0.5335399          0.5743215

# Compares robust vs non-robust standard errors
r1 <- coeftest(pos.mod, vcov = vcovHC(pos.mod, type = "const"))
r2 <- coeftest(pos.mod, vcov = vcovHC(pos.mod, type = "HC3"))
stargazer(r1, r2, type = "text")

##
## ===============================================
##                         Dependent variable:
##                     ---------------------------
##
##                           (1)            (2)
## -----------------------------------------------
## ConditionTreatment     0.520          0.520
##                       (0.501)        (0.574)
##
## Constant               0.788*         0.788
##                       (0.431)        (0.534)
##
## ===============================================
## ===============================================
## Note:               *p<0.1; **p<0.05; ***p<0.01
```

Experiment 3: OfferUp Camera and Lens

- Control: Bad photo with single line description
- Treatment: Good photo with full description

```
# actual offer up data collected with 2 different products (camera, len
s)
# photoquality determines views, description determines responses
offerup <- read.csv('offerup_data.csv')

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader on
## 'offerup_data.csv'

exp3_df <- as.data.frame(offerup[, c('Condition', 'Response', 'Views')]
)
# exp3_df
exp3_df_lens <- exp3_df[1:2,1:3]
exp3_df_lens

##   Condition Response Views
## 1 Treatment       24   982
## 2   Control        7   338

exp3_df_camera <- exp3_df[3:4,1:3]
exp3_df_camera

##   Condition Response Views
## 3 Treatment       39   972
## 4   Control       17   456

# combined_effect = (total views) / (total responses) for each control
and treatment. this is the number of people view the listing wrt respon
ses we get.
exp3_df$combined_effect <- exp3_df$Views / exp3_df$Response
exp3_df

##   Condition Response Views combined_effect
## 1 Treatment       24   982        40.91667
## 2   Control        7   338        48.28571
## 3 Treatment       39   972        24.92308
## 4   Control       17   456        26.82353
```

T.test

(1) Statistical Power

```
# n = (total views) / (total responses) for each control and treatment.
this is the number of people view the listing wrt responses we get.
# not being used now
# d = (diff in mean) / std_dev
# sig_level (alpha): 0.05 (95% confidence). or lower this if needed
num_control = exp3_df$Views[2] / exp3_df$Response[2]
num_treatment = exp3_df$Views[1] / exp3_df$Response[1]
```

```r
total_effect_num = round(min(num_control,num_treatment))
total_effect_num
```

```
## [1] 41
```

```r
t.test(exp3_df$Response ~ exp3_df$Condition, var.equal = TRUE)
```

```
##
##  Two Sample t-test
##
## data:  exp3_df$Response by exp3_df$Condition
## t = -2.1633, df = 2, p-value = 0.163
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -58.28359  19.28359
## sample estimates:
##   mean in group Control mean in group Treatment
##                    12.0                    31.5
```

```r
t.test(exp3_df$combined_effect ~ exp3_df$Condition, var.equal = TRUE)
```

```
##
##  Two Sample t-test
##
## data:  exp3_df$combined_effect by exp3_df$Condition
## t = 0.34632, df = 2, p-value = 0.7621
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -52.94771  62.21721
## sample estimates:
##   mean in group Control mean in group Treatment
##                37.55462                32.91987
```

```r
# Calculate effect size
cohens_d <- function(x, y) {
  lx <- length(x)- 1
  ly <- length(y)- 1
  md  <- abs(mean(x) - mean(y))        ## mean difference (numerator)
  csd <- lx * var(x) + ly * var(y)
  csd <- csd/(lx + ly)
  csd <- sqrt(csd)                     ## common sd computation

  cd  <- md/csd                        ## cohen's d
}

# Description - Response
exp3_cohens_d <- cohens_d(
  exp3_df[exp3_df$Condition == 'Control', 'Response'],
  exp3_df[exp3_df$Condition == 'Treatment', 'Response']
)
exp3_cohens_d
```

```
## [1] 2.163331
```

```
pwr.t.test(n=2, d = exp3_cohens_d, sig.level = 0.05,
           type = 'two.sample', alternative = 'two.sided')
```

```
##
##      Two-sample t test power calculation
##
##               n = 2
##               d = 2.163331
##       sig.level = 0.05
##           power = 0.2437954
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

```
# Description & View - combined_effect
exp3_cohens_d <- cohens_d(
  exp3_df[exp3_df$Condition == 'Control', 'combined_effect'],
  exp3_df[exp3_df$Condition == 'Treatment', 'combined_effect']
)
exp3_cohens_d
```

```
## [1] 0.3463159
```

```
pwr.t.test(n=2, d = exp3_cohens_d, sig.level = 0.05,
           type = 'two.sample', alternative = 'two.sided')
```

```
##
##      Two-sample t test power calculation
##
##               n = 2
##               d = 0.3463159
##       sig.level = 0.05
##           power = 0.05553827
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

Power is 0.6877998.

```
# 2) Observation from control group

# Description - Response
exp3_df %>%
  group_by(Condition) %>%
  summarize(avg = mean(Response),
            sd = sd(Response))
```

```
## # A tibble: 2 x 3
##   Condition   avg        sd
##      <fctr> <dbl>     <dbl>
```

```
## 1   Control  12.0  7.071068
## 2 Treatment  31.5 10.606602

# Photo - Views
exp3_df %>%
  group_by(Condition) %>%
  summarize(avg = mean(Views),
            sd = sd(Views))

## # A tibble: 2 x 3
##   Condition   avg        sd
##      <fctr> <dbl>     <dbl>
## 1   Control   397 83.438600
## 2 Treatment   977  7.071068

# Description & View - combined_effect
exp3_df %>%
  group_by(Condition) %>%
  summarize(avg = mean(combined_effect),
            sd = sd(combined_effect))

## # A tibble: 2 x 3
##   Condition      avg       sd
##      <fctr>    <dbl>    <dbl>
## 1   Control 37.55462 15.17606
## 2 Treatment 32.91987 11.30918
```

The control group (normal descriptions with typo) had an average response count of 10.5 and standard deviation of 1.96

(3) Baseline model In a two-sample T.test (independent groups) we have one hypothesis Null hypothesis: there are no significant differences in response count to Offerup camera lens ads between low quality ads descriptions:

- (bad photo and single description) vs high quality ad descriptions
- (good photo with full description)

```
# 4) ATE analysis

# Description - Response
exp3_means_df <- exp3_df %>%
  group_by(Condition) %>%
  summarize(avg = mean(Response)) %>%
  as.data.frame()

exp3_ATE <- exp3_means_df[2, 2] - exp3_means_df[1,2]
exp3_ATE

## [1] 19.5

# ATE for Treatment condition: 8
```

```r
# Photo - Views
exp3_means_df <- exp3_df %>%
  group_by(Condition) %>%
  summarize(avg = mean(Views)) %>%
  as.data.frame()

exp3_ATE <- exp3_means_df[2, 2] - exp3_means_df[1,2]
exp3_ATE
```

```
## [1] 580
```

```r
# combined effect
exp3_means_df <- exp3_df %>%
  group_by(Condition) %>%
  summarize(avg = mean(combined_effect)) %>%
  as.data.frame()

exp3_ATE <- exp3_means_df[2, 2] - exp3_means_df[1,2]
exp3_ATE
```

```
## [1] -4.63475
```

```r
# ATE for Treatment condition: 8

# 6) Regression

# CHECK: response is dependent on condition and views? or should be sep
arated
summary(lm(Response ~ Condition + Views, data=exp3_df))
```

```
##
## Call:
## lm(formula = Response ~ Condition + Views, data = exp3_df)
##
## Residuals:
##       1       2       3       4
## -7.8672 -0.6667  7.8672  0.6667
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -17.15787   53.52266  -0.321    0.803
## ConditionTreatment -23.09840   78.14073  -0.296    0.817
## Views                0.07345    0.13334   0.551    0.679
##
## Residual standard error: 11.17 on 1 degrees of freedom
## Multiple R-squared:  0.7703, Adjusted R-squared:  0.3109
## F-statistic: 1.677 on 2 and 1 DF,  p-value: 0.4793
```

```r
# You can model a t.test as a simple linear regression with
# a dummy-coded variable for the condition factor.
```

```r
# This is equivalent to a two-sample T-test
t.test(exp3_df$Response ~ exp3_df$Condition, var.equal = TRUE)

##
##  Two Sample t-test
##
## data:  exp3_df$Response by exp3_df$Condition
## t = -2.1633, df = 2, p-value = 0.163
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -58.28359  19.28359
## sample estimates:
##   mean in group Control mean in group Treatment
##                    12.0                    31.5

# You can relax this assumption by removing the var.equal argument.
t.test(exp3_df$Response ~ exp3_df$Condition)

##
##  Welch Two Sample t-test
##
## data:  exp3_df$Response by exp3_df$Condition
## t = -2.1633, df = 1.7423, p-value = 0.1815
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -64.33531  25.33531
## sample estimates:
##   mean in group Control mean in group Treatment
##                    12.0                    31.5
```

In all cases, results show a significant difference between conditions on the response count. Regression output shows the coefficient for condition is significantly different from 0, t(18) = 6.908, p = 0.018503.

(7) Modeling We used an independent groups T-Test to support the experiment of assigning subjects (cities) to condition. With Condition as a 2-level categorical factor (Control vs Treatment). The control condition had a bad photo and single line description, whereas the treatment condition had a good photo and full description. Response count was our dependent variable.

```r
# 8) Calculate f1 score and robust standard error

# We are using the `vcovHC` function from the library `sandwich`
# to estimate the white heteroskedastic-consistent standard errors
m1 <- lm(Response ~ Condition, data = exp3_df)
m1.vcovHC <- vcovHC(m1)  # from library(sandwich)

# With these, we can use the `coeftest` function from the `lmtest`
# package to perform hypothesis tests.
# these are the `robust` standard errors.
coeftest(m1, vcov = m1.vcovHC)
```

```
##
## t test of coefficients:
##
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          12.0000     7.0711  1.6971   0.2318
## ConditionTreatment   19.5000    12.7475  1.5297   0.2657
```

```
# To print more nicely,  we are taking the square-root of the diagonals
# of this heteroskedastic consistent variance covariance matrix, which
# provides the standard errors for each of the coefficients.
rse1 <- sqrt(diag(m1.vcovHC))
rse1
```

```
##        (Intercept) ConditionTreatment
##           7.071068          12.747549
```

```
# Compares robust vs non-robust standard errors
r1 <- coeftest(m1, vcov = vcovHC(m1, type = "const"))
r2 <- coeftest(m1, vcov = vcovHC(m1, type = "HC3"))
stargazer(r1, r2, type = "text")
```

```
##
## ================================================
##                         Dependent variable:
##                     ---------------------------
##
##                           (1)            (2)
## ------------------------------------------------
## ConditionTreatment      19.500         19.500
##                        (9.014)        (12.748)
##
## Constant                12.000         12.000
##                        (6.374)        (7.071)
##
## ================================================
## ================================================
## Note:                *p<0.1; **p<0.05; ***p<0.01
```

Non-parametric for test on counts

Mann-Whitney U-test tests if counts in one group tend to be higher than counts in another group. Given our sample size is low here (10). We cannot be confident our T statistics calculated with parametric tests are reliable. Furthermore, with count data, often it is Poisson-distributed, so it may be preferred to relax the normality assumption of the T-test.

```
# A non-parametric test alternative therefore may be preferred.
wilcox.test(exp3_df[exp3_df$Condition == 'Control', 'Response'],
            exp3_df[exp3_df$Condition == 'Treatment', 'Response'])
```

```
##
##  Wilcoxon rank sum test
##
## data:  exp3_df[exp3_df$Condition == "Control", "Response"] and exp3_
df[exp3_df$Condition == "Treatment", "Response"]
## W = 0, p-value = 0.3333
## alternative hypothesis: true location shift is not equal to 0
```

```
# Results show a significant difference in counts between conditions
# W = 23.5, p = 0.04868.
```

Poisson

```
# 1) Statistical Power
pos.mod <- glm(Response ~ Condition, data=exp3_df, family=poisson)
summary(pos.mod)
```

```
##
## Call:
## glm(formula = Response ~ Condition, family = poisson, data = exp3_df
)
##
## Deviance Residuals:
##      1       2       3       4
## -1.395  -1.567   1.288   1.357
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)          2.4849     0.2041  12.174  < 2e-16 ***
## ConditionTreatment   0.9651     0.2399   4.023 5.74e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 26.0236  on 3  degrees of freedom
## Residual deviance:  7.9024  on 2  degrees of freedom
## AIC: 30.919
##
## Number of Fisher Scoring iterations: 4
```

Caution: power tests here assume a R2 value.Since Poisson doesn't produce R2, we use a pseudo-R2, 1-(Residual Deviance/Null Deviance). Ref: (https://stats.stackexchange.com/questions/11676/pseudo-r-squared-formula-for-glms)

```
pseudo_r2 <- 1-(54.521/76.879)
effect.size <- pseudo_r2/(1-pseudo_r2)
pwr.f2.test(u=1, v=18, f2=effect.size, sig.level = .05)
```

```
##
##      Multiple regression power calculation
```

```
##
##                u = 1
##                v = 18
##               f2 = 0.4100805
##        sig.level = 0.05
##            power = 0.7728986
```

Power is 0.7728986.

```
# 2) Observation from control group
exp3_df %>%
  group_by(Condition) %>%
  summarize(avg = mean(Response),
            sd = sd(Response))
```

```
## # A tibble: 2 x 3
##   Condition   avg        sd
##      <fctr> <dbl>     <dbl>
## 1   Control  12.0  7.071068
## 2 Treatment  31.5 10.606602
```

The control group (normal descriptions with typo)had an average response count of 10.5 and standard deviation of 1.96.

(3) Baseline model In this Poisson regression we have one hypothesis Null hypothesis: there are no significant differences in response count to Craigslist camera lens ads between ads descriptions that have a typo vs don't.

```
# 4) ATE analysis
exp3_means_df <- exp3_df %>%
  group_by(Condition) %>%
  summarize(avg = mean(Response)) %>%
  as.data.frame()

exp3_ATE <- exp3_means_df[2, 2] - exp3_means_df[1,2]
exp3_ATE
```

```
## [1] 19.5
```

```
# ATE for Treatment condition: 8

# 6) Regression
pos.mod <- glm(Response ~ Condition, data=exp3_df, family=poisson)
summary(pos.mod)
```

```
##
## Call:
## glm(formula = Response ~ Condition, family = poisson, data = exp3_df
)
##
## Deviance Residuals:
##      1       2       3       4
```

```
## -1.395  -1.567   1.288   1.357
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)          2.4849     0.2041  12.174  < 2e-16 ***
## ConditionTreatment   0.9651     0.2399   4.023 5.74e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 26.0236  on 3  degrees of freedom
## Residual deviance:  7.9024  on 2  degrees of freedom
## AIC: 30.919
##
## Number of Fisher Scoring iterations: 4
```

You can model the experiment as a generalized linear model (GLM) with a dummy-coded variables for the condition factor. Here, in R we specify the outcome is Poisson distributed as is a common error distribution for integer count data

```
# There is a significant effect of condition in this model.
# z = 4.636, p = 0.00000356
library(AER)
dispersiontest(pos.mod,trafo=1)

##
##  Overdispersion test
##
## data:  pos.mod
## z = 4.3684, p-value = 6.257e-06
## alternative hypothesis: true alpha is greater than 0
## sample estimates:
##     alpha
## 0.9345238

# There is marginal overdispersion, but not significant.
```

(7)) Modeling With Condition as a 2-level categorical factor (Control vs Treatment). Response count was our dependent variable. We specified the error distribution to be Poisson, and chose a logarithm as the link function. In other words, the mean of the response is mapped to the linear combination of features via the logarithm function.

```
# 8) Calculate f1 score and robust standard error

# We are using the `vcovHC` function from the library `sandwich`
# to estimate the white heteroskedastic-consistent standard errors
pos.mod.vcovHC <- vcovHC(pos.mod)  # from library(sandwich)
```

```r
# With these, we can use the `coeftest` function from the `lmtest`
# package to perform hypothesis tests.
# these are the `robust` standard errors.
coeftest(pos.mod, vcov = pos.mod.vcovHC)
```

```
##
## z test of coefficients:
##
##                     Estimate Std. Error z value  Pr(>|z|)
## (Intercept)          2.48491    0.58926   4.217 2.475e-05 ***
## ConditionTreatment  0.96508    0.67868   1.422     0.155
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# To print more nicely, we are taking the square-root of the diagonals
# of this heteroskedastic consistent variance covariance matrix, which
# provides the standard errors for each of the coefficients.
rse1 <- sqrt(diag(pos.mod.vcovHC))
rse1
```

```
##        (Intercept) ConditionTreatment
##          0.5892557          0.6786758
```

```r
# Compares robust vs non-robust standard errors
r1 <- coeftest(pos.mod, vcov = vcovHC(pos.mod, type = "const"))
r2 <- coeftest(pos.mod, vcov = vcovHC(pos.mod, type = "HC3"))
stargazer(r1, r2, type = "text")
```

```
##
## ================================================
##                         Dependent variable:
##                     ---------------------------
##
##                           (1)            (2)
## ------------------------------------------------
## ConditionTreatment     0.965*         0.965
##                       (0.568)        (0.679)
##
## Constant              2.485***      2.485***
##                       (0.531)        (0.589)
##
## ================================================
## ================================================
## Note:                *p<0.1; **p<0.05; ***p<0.01
```