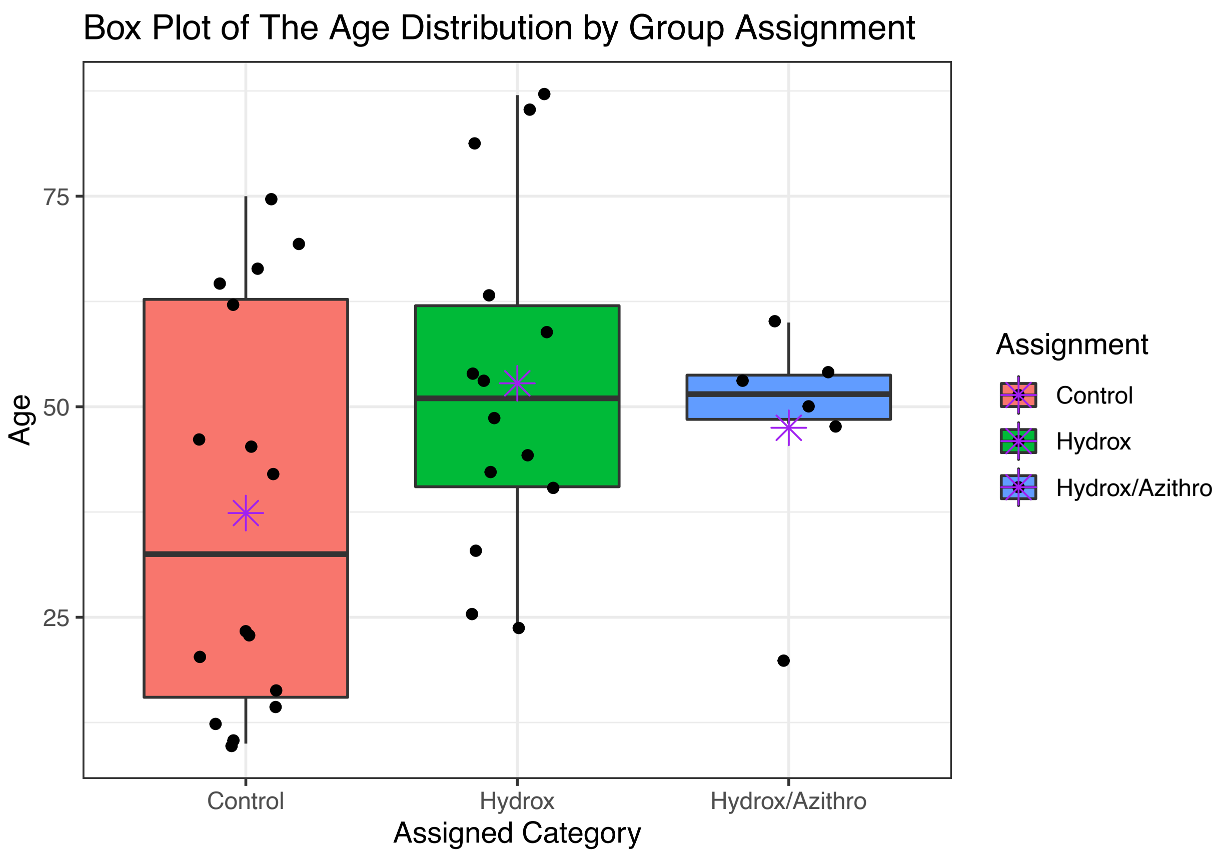
COVID-19: Hydroxychloroquine Experiment Result Expansion

Patrick Poleshuk

8/20/2021

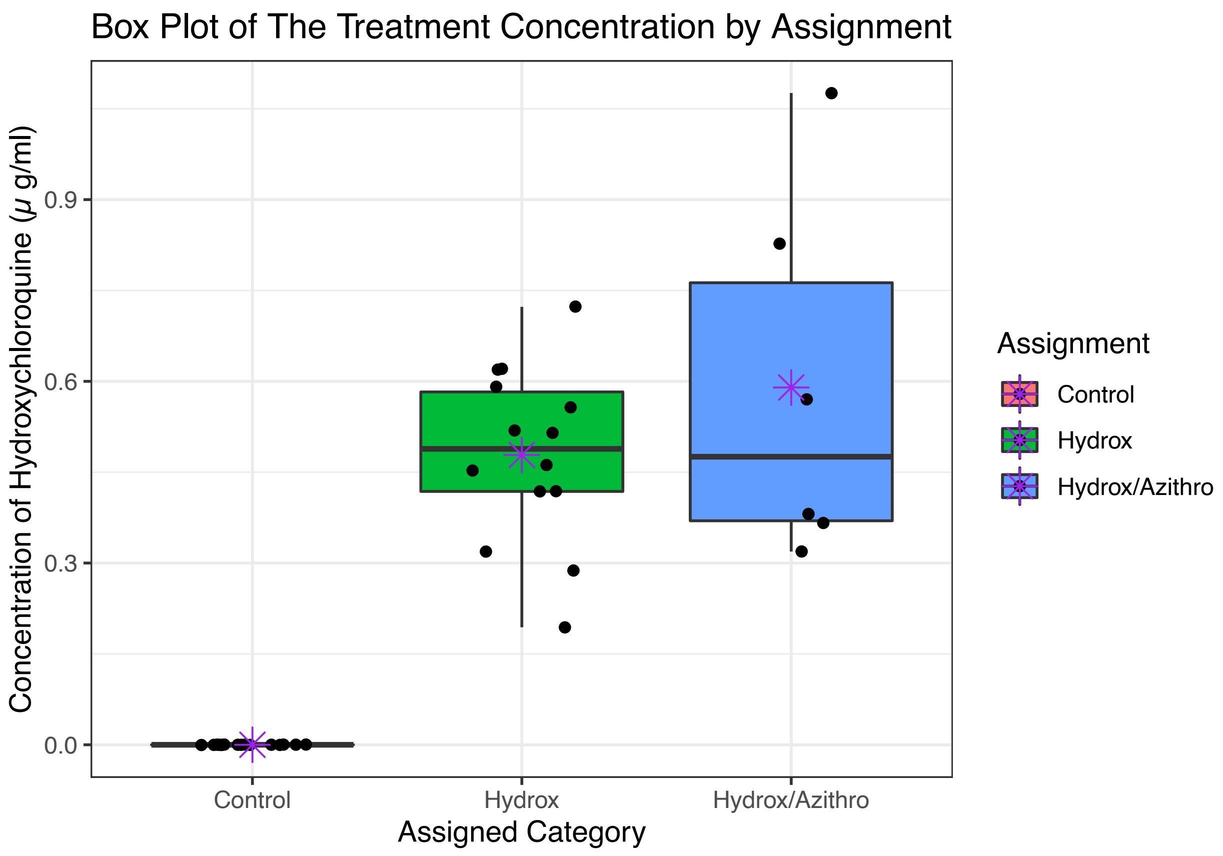
```r
set.seed(101)
# Bayesian regression (using the R package 'rstanarm') uses Markov-chain Monte Carlo
# simulation to supply the parameter estimates. Essentially, to find our regression
# coefficients, we need to find the posterior distribution that they lie on, a probability
# distribution that is proportional to the maximum likelihood estimation of our coefficients
# and our prior beliefs of the probability distribution that our parameter of interest follows.
# The algorithm is complex, but essentially we can construct a random sample from our
# posterior distribution, where each estimate sampled is influenced by the one that came
# before it (called a Markov Chain), Using Markov chain sampling, we can draw samples from
# the posterior distribution without knowing the complete properties of said distribution. Under
# standard Monte Carlo simulation, samples (which follow a distribution) are made independent,
# yet using a Markov chain, in this case, allows us to hone in on the posterior distribution
# where we believe our true coefficient of effect lies.


# 4,000 samples are taken into account, and the median values of these probabilities
# are our best estimate for beta (which is assumed to follow a probability
# distribution in Bayesian statistics, rather than simply being a fixed, point-estimate as in
# the Frequentist case). To reiterate, our estimates derive from generating random values
# from the posterior PDF, where our best estimate beta is stationed,
# so it stands that every time we run a bayesian regression our best estimate will change
# very slightly since our random samples will fluctuate. The change isn't significant; however, we s
#


# Loading up required libraries & the excel data set of the experiment outcomes.
library(rstanarm)
library(dplyr)
library(tidyverse)

gautret <- readxl::read_excel('/Users/patrickpoleshuk/gautretData_2.xlsx')
names(gautret)[1] <- c('age')

# Before conducting any kind of regression analysis, I will like to visualize any
# structural differences between the control and treatment groups, in regards to
# age, sex, and disease classification.

g = gautret %>% mutate(category =  ifelse(ctrtCode == 0, "Control",
ifelse(ctrtCode == 1 & atrtCode == 0,  "Hydrox", "Hydrox/Azithro")))

old <- theme_set(theme_bw())
# Setting a theme for ggplot.
```

```r
dat = as.data.frame(g %>% group_by(category) %>% summarise(avg = mean(age)))
ggplot(data = g, aes(x = category, y = age, fill = category)) +
  geom_boxplot(outlier.shape = NA) + geom_jitter(width = 0.2) +
  geom_point(data = dat, aes(x = category, y = avg), color = "purple", shape = 8,
             size = 4) + labs(x = "Assigned Category",
                              y = "Age",
title = "Box Plot of The Age Distribution by Group Assignment") +
  scale_fill_discrete(name = 'Assignment')
```

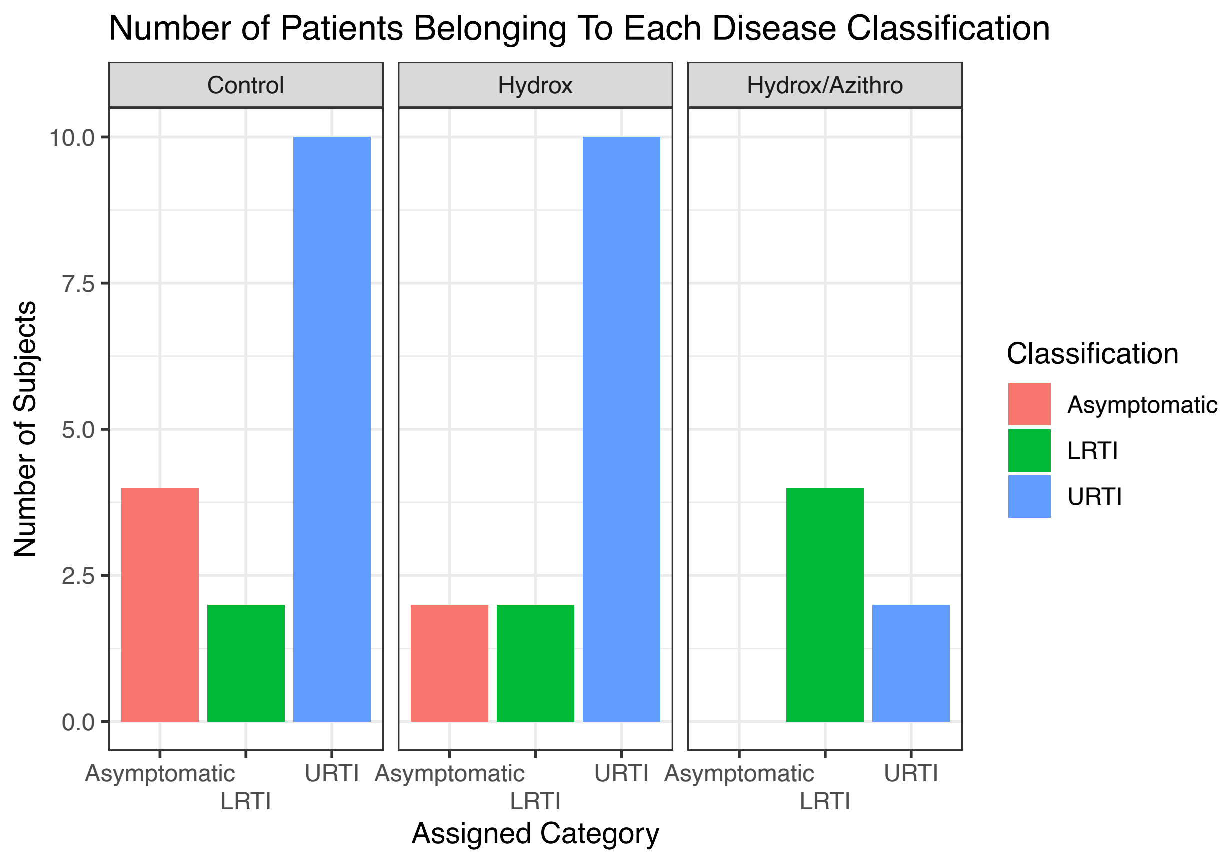


```r
# We can see from the box plot that the median, mean, & deviation from the mean,
# regarding age, differs between the control and treatment groups.

# For a numerical representation:
stat_age = g %>% group_by(category) %>% summarise(Average = mean(age),
                                                    Median = median(age),
                                                    StDev = sd(age),
                                                    IQR = IQR(age))

# Visualizing dose concentration values:
g$conc = as.numeric(g$conc)
g$conc[is.na(g$conc)] = 0

data = g %>% group_by(category)

dat = as.data.frame(g %>% group_by(category) %>% summarise(avg = mean(conc)))
```

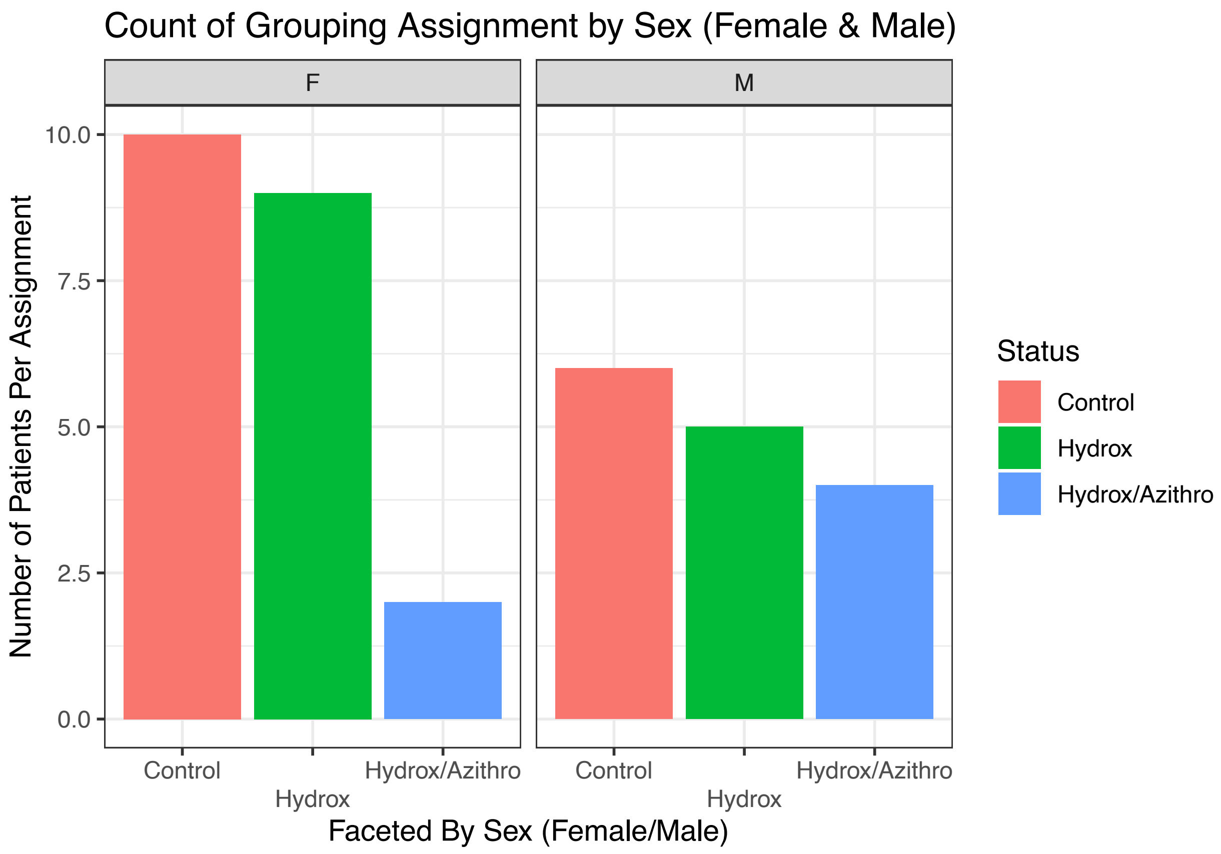
```r
ggplot(data = g, aes(x = category, y = conc, fill = category)) +
  geom_boxplot() + geom_jitter(width = .2) +
  labs(x = "Assigned Category",
       y = "Concentration of Hydroxychloroquine (μ g/ml)",
       title = "Box Plot of The Treatment Concentration by Assignment") +
  scale_fill_discrete(name = 'Assignment') +
  geom_point(data = dat, aes(x = category, y = avg), color = "purple", shape = 8,
             size = 4)
```



```r
stat_conc = g %>% group_by(category)%>% summarise(Average = mean(conc),
                                                  Median = median(conc),
                                                  StDev = sd(conc),
                                                  IQR = IQR(conc))

# For those exposed to either just the hydroxychloroquine treatment
# or hydroxychloroquine paired with the azithromycin, we can get a good
# idea of the concentration of the treatment within the bodies of the
# subjects. We can see that, even though all subjects in the treatment
# were dosed with 600 micrograms per milliliter, a higher dose resonates
# for those who supplemented with azithromycin. Whether this is the result
# of some differing physiological trait that either enhances or hinders the
# the effect of the dosage on bodily drug concentration, a synergistic
# impact of the azithromycin, or simply the result of an experimental
# blunder in distributing the dosages, I am not sure.
# Onto Disease classification differences:
g[31, 3] = 'LRTI'
# Correcting a data entry discrepancy in the excel data spreadsheet.

bar = g %>% group_by(category, status) %>%
  summarise(len = length(status))

library(scales)
ggplot(data = bar, aes(x = status, y = len, fill = as.factor(status))) +
  geom_col() +
  facet_wrap(~category) + scale_x_discrete(guide = guide_axis(n.dodge = 2)) +
  scale_fill_discrete(name = 'Classification') +
  labs(x = 'Assigned Category',
       y = 'Number of Subjects',
       title = "Number of Patients Belonging To Each Disease Classification")
```



```r
# The groups remained relatively balanced in regards to the number of URTI/LRTI/Asymptomatic
# patients contained within them. Still, we see no significant systematic difference that would
# bias the findings in the clinical experiment.


# Sex differences:
table = g %>% group_by(sex, category) %>% summarise(n = n())
ggplot(data = table, aes(x = category, y = n, fill = as.factor(category))) +
  geom_col() +
  facet_wrap(~sex) + scale_x_discrete(guide = guide_axis(n.dodge = 2)) +
  scale_fill_discrete(name = 'Status') + labs(x = 'Faceted By Sex (Female/Male)',
                                      y = 'Number of Patients Per Assignment',
  title = 'Count of Grouping Assignment by Sex (Female & Male)')
```



```
# Roughly an equally balanced amount of females and males in each group, when accounting for
# the fact that there were more female patients studied in this experiment.
```

```
set.seed(101)
# Even though we have observed that selection bias should be minimal, with regards to
# confounding our hydroxychloroquine & azithromycin treatment effect on COVID-19 test
# results, given that all three assignment groups share a similar set of characteristics, we
# can still examine whether these characteristics influence the outcome of interest.

# Regression Models:

# t6 ~ Age

logistic_age = stan_glm(data = g, t6 ~ age, family = binomial(link = 'logit'))
```

```
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000879 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 8.79 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.061015 seconds (Warm-up)
## Chain 1:                0.058248 seconds (Sampling)
## Chain 1:                0.119263 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.8e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.18 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.057672 seconds (Warm-up)
## Chain 2:                0.056904 seconds (Sampling)
## Chain 2:                0.114576 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.4e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.14 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.054374 seconds (Warm-up)
## Chain 3:                0.076701 seconds (Sampling)
## Chain 3:                0.131075 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 2.1e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.21 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.0722 seconds (Warm-up)
## Chain 4:                0.070193 seconds (Sampling)
## Chain 4:                0.142393 seconds (Total)
## Chain 4:
```

```r
print(logistic_age, digits = 4)
```

```
## stan_glm
##  family:       binomial [logit]
##  formula:      t6 ~ age
##  observations: 36
##  predictors:   2
## ------
##             Median MAD_SD
## (Intercept) 0.2261 0.7714
## age         0.0027 0.0155
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```r
# Given that our dependent variable follows a Bernoulli distribution (our random variable
# can take only 2 possible outcomes as either you test negative or positive for COVID-19),
# we can estimate using a logistic regression. This allows us to find the probability that our
# dependent variable is true, given what we observe from our independent variables.

# More specifically, the coefficients we receive will tell us the log-odds that we test
# positive, given a 1 unit increase in our continuous independent variable, age.
# To give a better interpretation, we can standardize our age variable and take the inverse
# logit of the coefficient to convert our log-odds value into a probability. As it is now, our
# intercept value tells us that our model predicts that the log-odds of contracting COVID
# when someone is of age 0 is 0.2381. Additionally, for every one year increase in age, the log
# odds of contracting COVID increase by 0.0024. As we can see, this doesn't make for a very
# useful interpretation.


inv_logit <- function(x) {
  formula = exp(x) / (1 + exp(x))
  return(formula)
}
g$agesc = (g$age - mean(g$age))/(sd(g$age))
logistic_age_standardized = stan_glm(data = g, t6 ~ agesc,
                                     family = binomial(link = 'logit'))
```

```
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.5e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.25 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.054726 seconds (Warm-up)
## Chain 1:                0.057627 seconds (Sampling)
## Chain 1:                0.112353 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 2).
## Chain 2:
```

```
## Chain 2: Gradient evaluation took 1.6e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.16 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.061444 seconds (Warm-up)
## Chain 2:                0.059849 seconds (Sampling)
## Chain 2:                0.121293 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.7e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.17 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.054556 seconds (Warm-up)
## Chain 3:                0.055586 seconds (Sampling)
## Chain 3:                0.110142 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.6e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.16 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
```

```
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.061886 seconds (Warm-up)
## Chain 4:                0.055614 seconds (Sampling)
## Chain 4:                0.1175 seconds (Total)
## Chain 4:
```

```
print(logistic_age_standardized, digits = 4)
```

```
## stan_glm
##  family:       binomial [logit]
##  formula:      t6 ~ agesc
##  observations: 36
##  predictors:   2
## ------
##             Median MAD_SD
## (Intercept) 0.3574 0.3485
## agesc       0.0486 0.3468
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```
print(inv_logit(logistic_age_standardized$coefficients))
```

```
## (Intercept)       agesc
##   0.5884221   0.5121460
```

```
# Here we can see that our intercept value tells us the probability of one testing
# positive for COVID, given that they are of average age: 0.5884221. Our beta 1 coefficient
# gives us insight into the probability of testing positive for COVID for every 1 standard
# deviation we move away from the mean age observed. To give an example, where someone is 1
# standard deviation away from the mean age:

int = logistic_age_standardized$coefficients[1]
beta1 = logistic_age_standardized$coefficients[2]
inv_logit(int + 1*beta1)
```

```
## (Intercept)
##   0.6001378
```

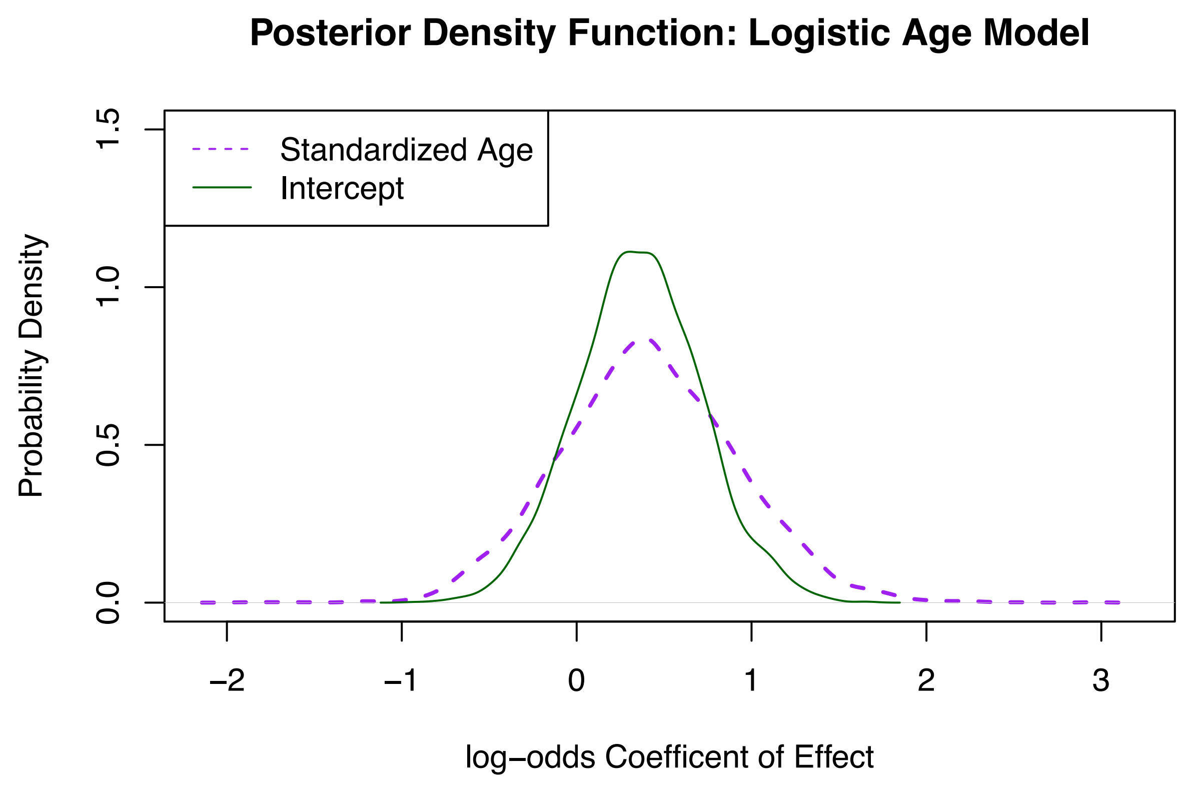
```r
# We see that someone who fits this description is predicted to have a 0.6001378 probability
# of testing positive.

inv_logit(int + 2.5*beta1)
```

```
## (Intercept)
##   0.6174951
```

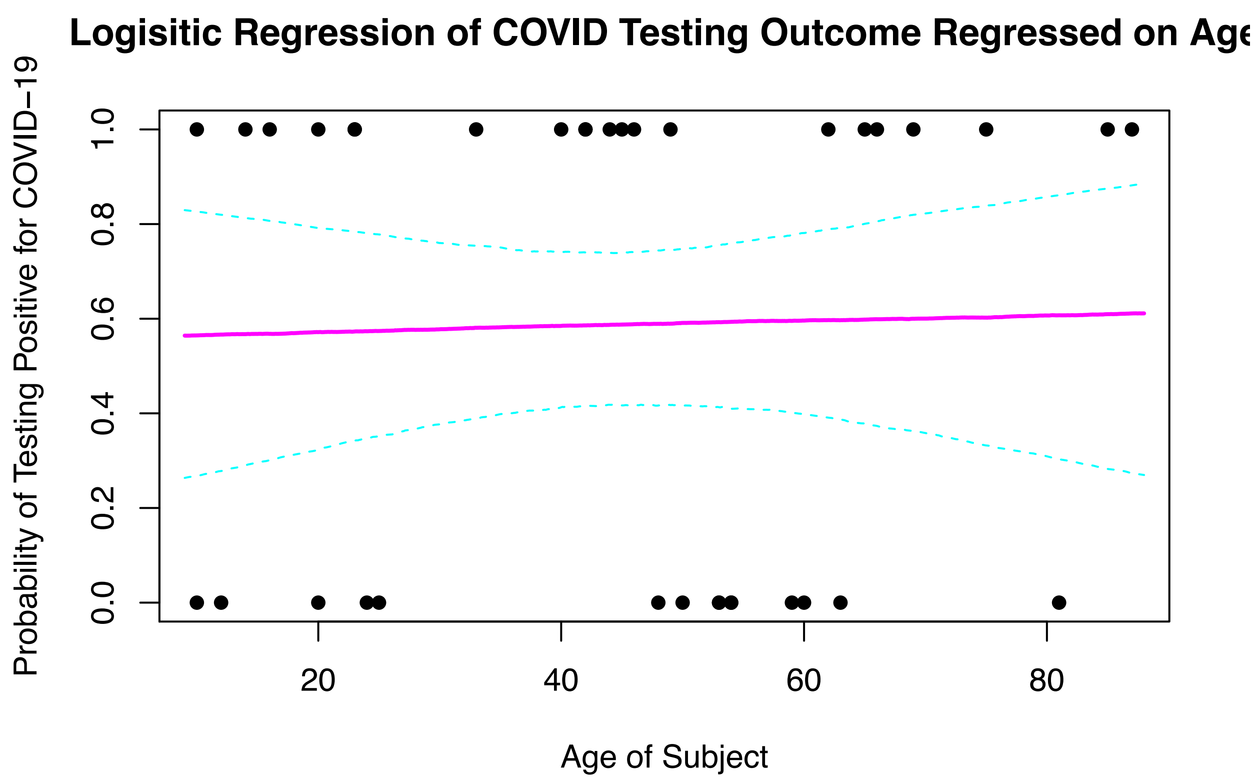
```r
# Meanwhile, someone who is, say, 2.5 standard deviations away from the mean age is predicted
# to have a 0.6174951 probability of testing positive for COVID.

# In other words, we can see that older people are more susceptible to test positive for COVID,
# given our observed data, however, the effect is negligible.

posterior_interval(logistic_age_standardized)
```

```
##                     5%       95%
## (Intercept) -0.2166741 0.9551938
## agesc       -0.5115072 0.6569168
```

```r
# Unfortunately and predictably, our Bayesian credible interval tells us that our estimator
# is inefficient, given that 0 is a plausible value for both these estimators. Our
# interpretation for our beta 1 estimate, thus, is that there is a 95% probability that the
# log-odds effect, for every 1 standard deviation shift away from the average mean observed,
# on testing positive is between -0.5115072 & 0.6569168.

# To illustrate this with the posterior distribution of our parameter estimates: we can see
# that the distribution of our parameters vary way too much for us to determine that this
# effect of age, in this study, provided a meaningful effect on the susceptibility of testing
# positive for COVID.

sims = as.matrix(logistic_age_standardized)
int = sims[, 1]
agesc = int + sims[, 2]

plot(density(agesc), col = 'purple', lty = 2, lwd = 2,
     xlab = "log-odds Coefficent of Effect",
     main = "Posterior Density Function: Logistic Age Model",
     y = "Probability Density", ylim = c(0, 1.5))
lines(density(int), lwd = 1, col = 'darkgreen')
legend('topleft', legend = c("Standardized Age", "Intercept"), col = c("purple", "darkgreen"), lty =
```



```
sequence = seq(min(g$age)-1, max(g$age)+1, .01)

sims = as.matrix(logistic_age)

int = sims[, 1]
age = sims[, 2]

lower = c()
higher = c()
med = c()

for (i in 1:length(sequence)) {
  ypred = int + age*sequence[i]
  lower[i] = quantile(ypred, prob = .025)
  higher[i] = quantile(ypred, prob = .975)
  med[i] = median(ypred)
}

plot(g$age, g$t6, pch = 16, xlab = "Age of Subject",
     ylab = "Probability of Testing Positive for COVID-19",
     main = "Logisitic Regression of COVID Testing Outcome Regressed on Age")
lines(sequence, inv_logit(lower), lty = 2, col = "cyan")
lines(sequence, inv_logit(higher), lty = 2, col = "cyan")
lines(sequence, inv_logit(med), lwd = 2, col = "magenta")
```



```
# As we can see visually, given that age lacks the significance of determining a robust
# probabilistic outcome for COVID-19 test results, our logit model doesn't even remotely
# resemble a sigmoid curve. I should reiterate that this is desirable, in this case, as
# age could very well have been a meaningful confounder. Thankfully, however, that is not
# shown to be the case here.


# For variables like this, plotting the logistic regression would be ineffective, and it would be
# better to just visualize the posterior density function shown above.

# t6 ~ Sex
logistic_sex = stan_glm(data = g, t6 ~ sexCode, family = binomial(link = 'logit'))
```

```
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.2 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.054064 seconds (Warm-up)
## Chain 1:                0.052196 seconds (Sampling)
## Chain 1:                0.10626 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 2.3e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.23 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.051393 seconds (Warm-up)
## Chain 2:                0.057781 seconds (Sampling)
## Chain 2:                0.109174 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.4e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.14 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.04998 seconds (Warm-up)
## Chain 3:                0.057465 seconds (Sampling)
## Chain 3:                0.107445 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.8e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.18 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.047397 seconds (Warm-up)
## Chain 4:                0.054426 seconds (Sampling)
## Chain 4:                0.101823 seconds (Total)
## Chain 4:
```

```
inv_logit(logistic_sex$coefficients)
```

```
## (Intercept)     sexCode
##   0.4719344   0.6884330
```

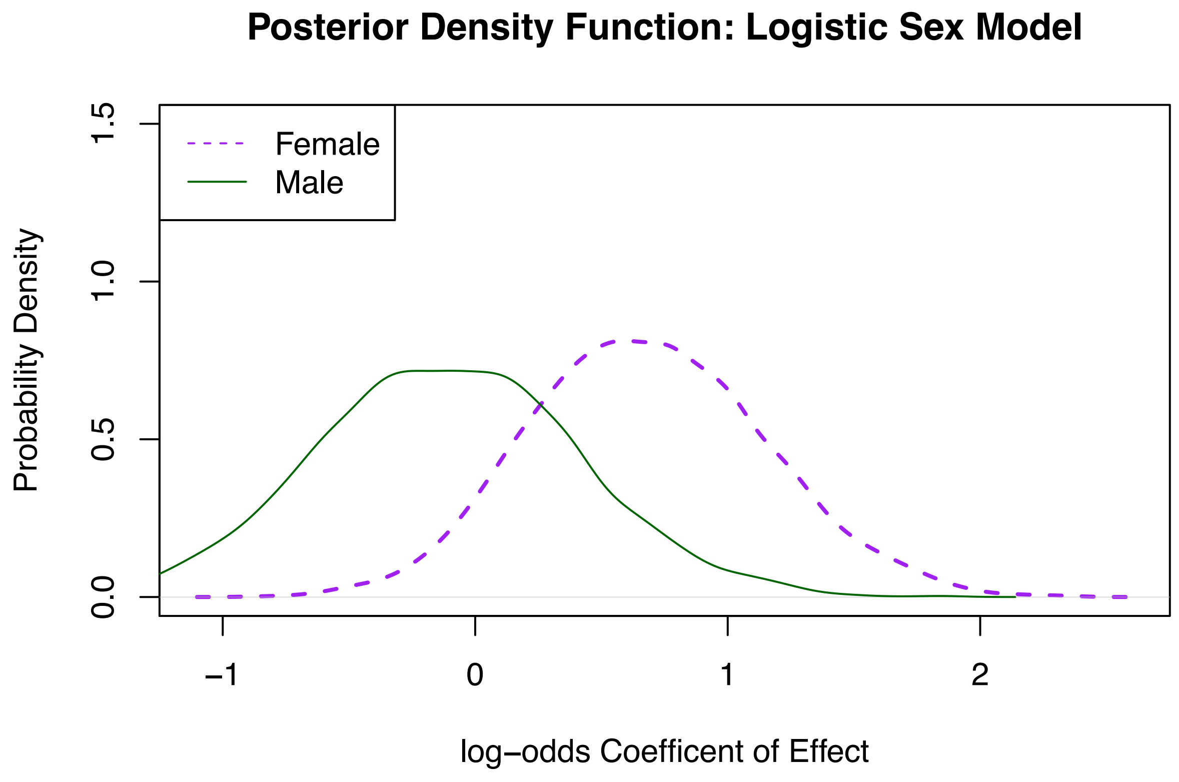
```
# If one is male, the probability that they test positive is 0.4733507. If one is female,
# the probability increases to 0.6903189.

# log-odds Estimates:
print(logistic_sex, digits = 4)
```

```
## stan_glm
##  family:       binomial [logit]
##  formula:      t6 ~ sexCode
##  observations: 36
##  predictors:   2
## ------
##             Median  MAD_SD
## (Intercept) -0.1124  0.5214
## sexCode      0.7928  0.7001
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```
posterior_interval(logistic_sex)
```

```
##                      5%       95%
## (Intercept) -0.9768360 0.7567365
## sexCode     -0.3335057 1.9450223
```

```
# As we can see, the log-odds coefficients, from which the probabilities derive, are
# insignificant as 0 is a plausible value here.

# Visually we can this here:
sims = as.matrix(logistic_sex)
int = sims[, 1]
female = int + sims[, 2]
plot(density(female), col = 'purple', lty = 2, lwd = 2,
     xlab = "log-odds Coefficent of Effect",
     main = "Posterior Density Function: Logistic Sex Model",
     y = "Probability Density", ylim = c(0, 1.5))
lines(density(int), lwd = 1, col = 'darkgreen')
legend('topleft', legend = c("Female", "Male"), col = c("purple", "darkgreen"), lty = 2:1)
```



```
# We can see that the posterior distributions have very wide tails, indicating that
# the credible intervals are very large for our estimates. We can't, therefore, be sure
# in our estimated log effect of sex on testing positive for COVID. This is, again, good
# news for the prospect of absent confounding.
```

```r
# t6 ~ LRTI/URTI/Control (Disease Classification)
g$statCode = ifelse(g$status == 'Asymptomatic', 0, ifelse(g$status == 'LRTI', 1, 2))
logistic_test = stan_glm(data = g, t6 ~ statCode, family = binomial(link = 'logit'))
```

```
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.9e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.19 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.054286 seconds (Warm-up)
## Chain 1:                0.057757 seconds (Sampling)
## Chain 1:                0.112043 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.6e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.16 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.054774 seconds (Warm-up)
## Chain 2:                0.060115 seconds (Sampling)
## Chain 2:                0.114889 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.4e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.14 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.05649 seconds (Warm-up)
## Chain 3:                0.056794 seconds (Sampling)
## Chain 3:                0.113284 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 2.1e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.21 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.049898 seconds (Warm-up)
## Chain 4:                0.054794 seconds (Sampling)
## Chain 4:                0.104692 seconds (Total)
## Chain 4:
```

```r
print(logistic_test, digits = 4)
```

```
## stan_glm
##  family:       binomial [logit]
##  formula:      t6 ~ statCode
##  observations: 36
##  predictors:   2
## ------
##             Median  MAD_SD
## (Intercept) -1.0795  0.7922
## statCode     0.9895  0.4948
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```
posterior_interval(logistic_test)
```

```
##                      5%       95%
## (Intercept) -2.5239562 0.1615144
## statCode     0.2178896 1.8861495
```

```
# Given that there is a 95% probability that the effect of testing positive for COVID-19
# increases, depending on whether someone is of classification URTI / LRTI, I will continue
# this analysis after controlling for dose concentration value.

logistic_char = stan_glm(data = g, t6 ~ statCode + conc,
                         family = binomial(link = 'logit'))
```

```
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.2 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.066513 seconds (Warm-up)
## Chain 1:                0.068595 seconds (Sampling)
## Chain 1:                0.135108 seconds (Total)
## Chain 1:
##
```

```
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.5e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.15 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.072704 seconds (Warm-up)
## Chain 2:                0.071046 seconds (Sampling)
## Chain 2:                0.14375 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.5e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.15 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.069716 seconds (Warm-up)
## Chain 3:                0.065555 seconds (Sampling)
## Chain 3:                0.135271 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.5e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.15 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.064802 seconds (Warm-up)
## Chain 4:                0.073831 seconds (Sampling)
## Chain 4:                0.138633 seconds (Total)
## Chain 4:
```

```r
print(logistic_char, digits = 4)
```

```
## stan_glm
##  family:       binomial [logit]
##  formula:      t6 ~ statCode + conc
##  observations: 36
##  predictors:   3
## ------
##             Median  MAD_SD
## (Intercept) -0.0266  0.8798
## statCode     1.5708  0.6416
## conc        -5.9226  2.0820
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```r
posterior_interval(logistic_char)
```

```
##                      5%       95%
## (Intercept)  -1.5321288  1.428489
## statCode      0.5758119  2.761981
## conc        -10.0377189 -2.960194
```

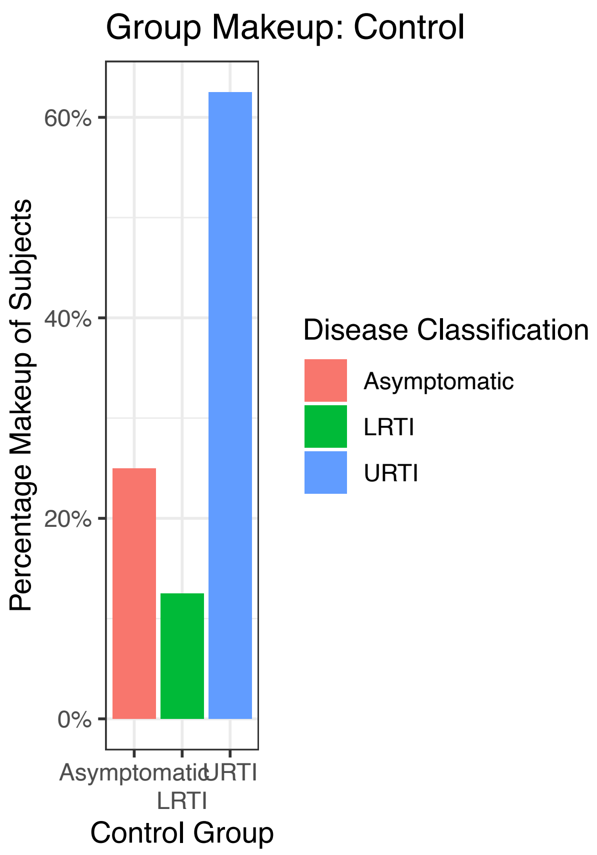
```r
# Interestingly, we still observe that there is a 95% probability that 0 is not a plausible
# value for the effect of disease classification on testing positive for COVID-19.

inv_logit(logistic_char$coefficients)
```

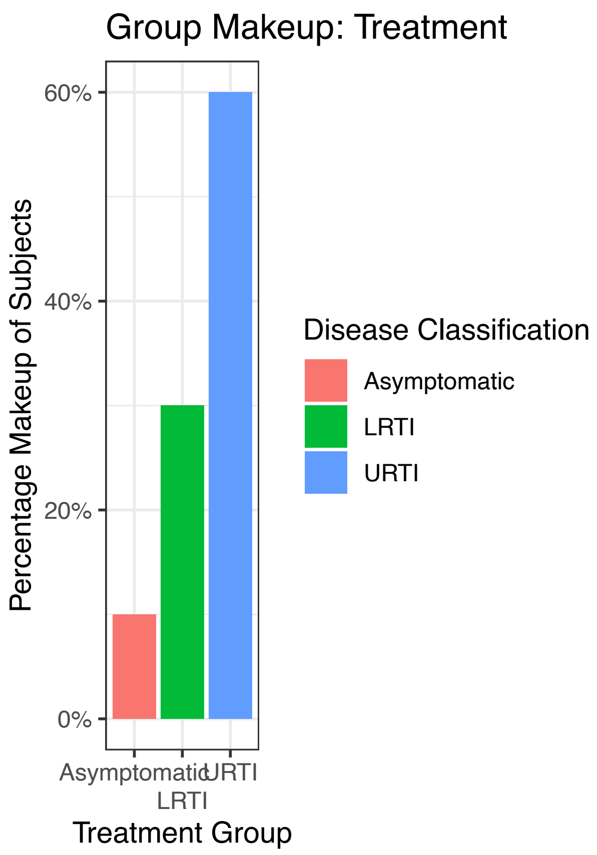
```
## (Intercept)    statCode        conc
## 0.493362024 0.827895800 0.002671007
```

```
# We can see that if someone is asymptomatic, the probability that they test positive is
# 0.496434705. This estimate is not significant, but our beta 1 value is. If someone suffers
# from LRTI, their probability of testing positive for COVID is predicted to be roughly
# 0.825825933.

inv_logit(2*logistic_char$coefficients[2])
```

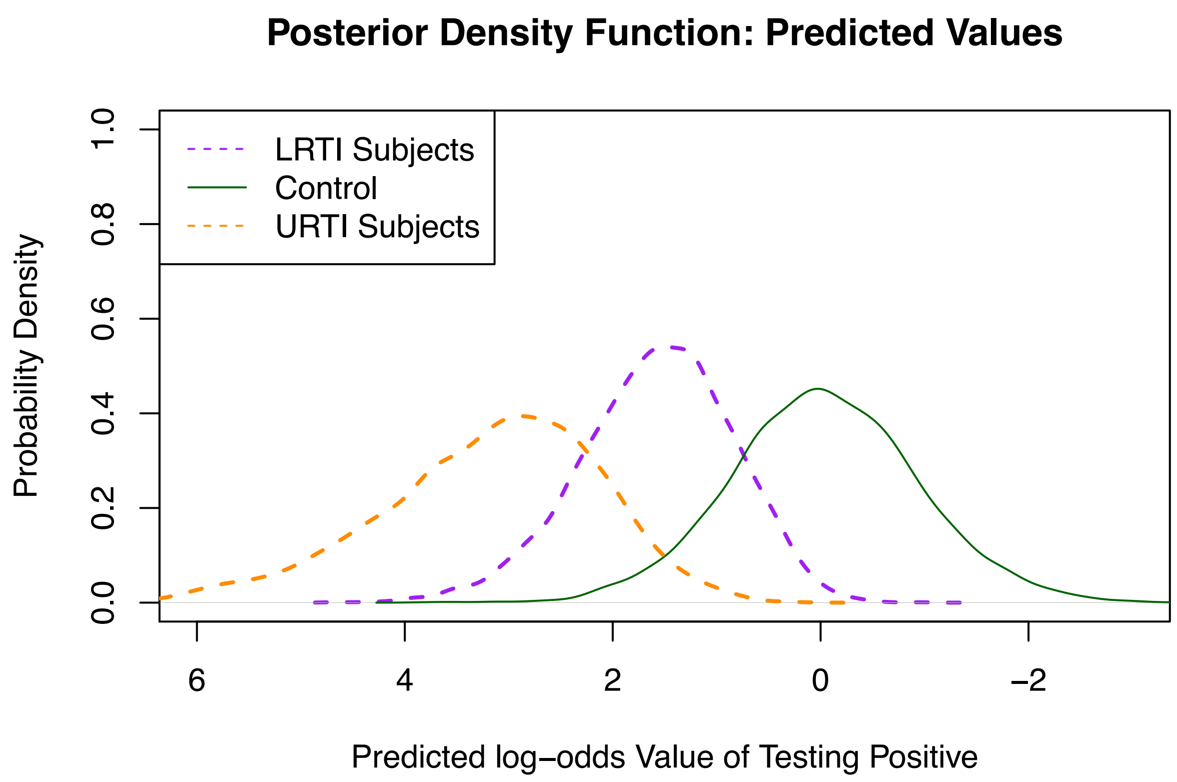
```
##  statCode
## 0.9585754
```

```
# Finally, if someone suffers from URTI, their probability of testing positive for COVID
# in this experiment will be the highest at 0.9574119.


# It is interesting to see that if you are an URTI patient, your probability of testing
# positive will be greater than if you were an LRTI patient, after controlling for the dose
# concentration differences the two between subjects.

# This provides some interesting insight to the question of whether
# hydroxychloroquine & azithromycin are more effective towards LRTI patients.

# To check one more time, and determine whether there was a significant difference
# between the number of LTRI and HTRI patients in the treatment and control groups,
# we can visualize a bar chart of the counts.

g$treatment = ifelse(g$conc > 0, 'Treatment', "Control")
df = tibble(g %>% group_by(treatment, status) %>%
  summarise(len = length(status)))

data1 = df %>% filter(treatment == 'Control') %>% mutate(prop = len/16)
data2 = df %>% filter(treatment == 'Treatment') %>% mutate(prop = len/20)

library(cowplot)
g1 = ggplot(data = data1, aes(x = status, y = prop, fill = as.factor(status))) +
  geom_col() + scale_y_continuous(labels = percent) +
  labs(x = 'Control Group',
       y = "Percentage Makeup of Subjects",
       title = "Group Makeup: Control")  +
  scale_fill_discrete(name = 'Disease Classification') +
  scale_x_discrete(guide = guide_axis(n.dodge = 2))
g2 = ggplot(data =data2, aes(x = status, y = prop, fill = as.factor(status))) +
  geom_col() + scale_y_continuous(labels = percent) +
    scale_fill_discrete(name = 'Disease Classification') +
  labs(x = 'Treatment Group',
       y = "Percentage Makeup of Subjects",
       title = "Group Makeup: Treatment") +
  scale_x_discrete(guide = guide_axis(n.dodge = 2))

plot_grid(g1, g2)
```





```
# There doesn't seem to be significantly more URTI/LRTI patients in the Treatment
# group than the Control group, scaled by their respective populations,
# however, this is an interesting question to ponder: whether hydroxychloroquine
# affects URTI and LRTI patients differently.

# Finally, let's create a posterior density function of our parameter estimates:
# * Remember this is after controlling for concentration effect.

sims = as.matrix(logistic_char)
control = sims[, 1]
LRTI = sims[, 2] + control
URTI = 2*sims[, 2] + control
plot(density(LRTI), col = 'purple', lty = 2, lwd = 2,
     xlab = "Predicted log-odds Value of Testing Positive",
     main = "Posterior Density Function: Predicted Values",
     y = "Probability Density", ylim = c(0, 1), xlim = c(6, -3))

lines(density(control), lwd = 1, col = 'darkgreen')
lines(density(URTI), lwd = 2, col = 'darkorange', lty = 2)
legend('topleft', legend = c("LRTI Subjects", "Control", 'URTI Subjects'),
       col = c("purple", "darkgreen", 'darkorange'),
       lty = 2:1)
```



```r
# Given that 0 is not a major plausible value for the predicted log-odds value of LRTI &
# URTI patients, these predictions seem robust (but vary wildly in effect).


# t6 ~ conc
logistic_conc = stan_glm(data = g, t6 ~ conc, family = binomial(link = 'logit'))
```

```
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.2 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.054692 seconds (Warm-up)
## Chain 1:                0.059875 seconds (Sampling)
## Chain 1:                0.114567 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.8e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.18 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.053842 seconds (Warm-up)
## Chain 2:                0.058586 seconds (Sampling)
## Chain 2:                0.112428 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 3.1e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.31 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.05562 seconds (Warm-up)
## Chain 3:                0.055722 seconds (Sampling)
## Chain 3:                0.111342 seconds (Total)
## Chain 3:
##
```

```
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 2.1e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.21 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.055226 seconds (Warm-up)
## Chain 4:                0.059854 seconds (Sampling)
## Chain 4:                0.11508 seconds (Total)
## Chain 4:
```

```
print(logistic_conc, digits = 4)
```

```
## stan_glm
##  family:       binomial [logit]
##  formula:      t6 ~ conc
##  observations: 36
##  predictors:   2
## ------
##             Median  MAD_SD
## (Intercept)  1.6195  0.6180
## conc        -4.2418  1.5470
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```
posterior_interval(logistic_conc)
```

```
##                     5%       95%
## (Intercept)  0.6768372  2.788618
## conc        -7.1112006 -1.838666
```

```
# There is a 95% probability that the true log-odds effect of hydroxychloroquine
# concentration on testing positive for COVID-19 is between -6.9077346 & -1.939355
# *This is a significant finding, and is far larger than anything we have observed from
# the other variables.
```

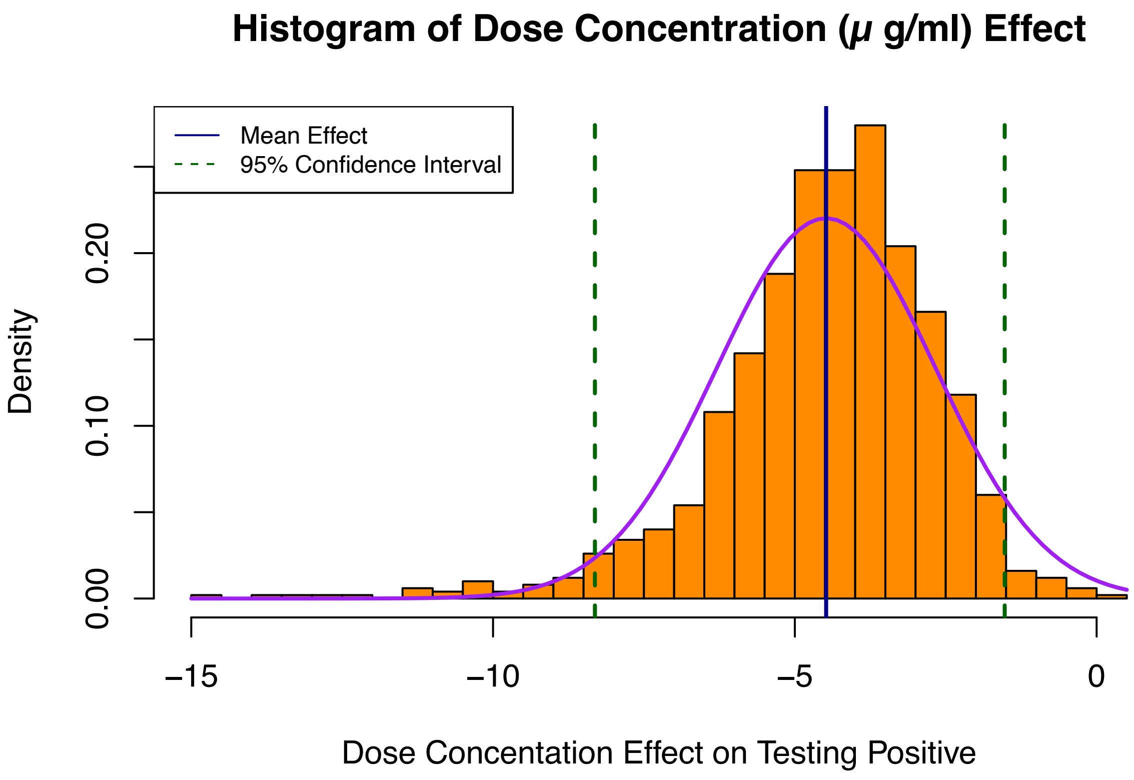
```r
inv_logit(logistic_conc$coefficients[1])
```

```
## (Intercept)
##   0.8347258
```

```r
# For a probabilistic interpretation: someone without the dose concentration has a 0.8333466
# probability of testing positive for COVID.

inv_logit(logistic_conc$coefficients[1] + .5*logistic_conc$coefficients[2])
```

```
## (Intercept)
##   0.3772145
```

```r
# If someone increases their dose by .5 (μ g/ml), their probability of testing positive is
# 0.3789761. As we can see, this is significantly lower.


# Onto an additional method:

# Although this analysis has rested on Bayesian methods, I feel that it's useful, given the
# relatively small sample size of subjects in this clinical experiment, to conduct frequentist
# bootstrap analysis.



# Using this method, we can sample from our sample (with replacement) using our original
# sample data. Given that we assume our sample is our best estimate of what our population
# distribution looks like, sampling from it is akin to running a continuous amount of
# homogenous clinical trials, all outcomes in some way representing what our population
# results would be.


# It is important to note that this bootstrap method outlined is based on the Frequentist
# school of thought. I've decided to briefly switch over from Bayesian methods, as I don't
# have any real experience with bayesian bootstrapping, and I feel as if I wouldn't be
# accurately able to explain what is going on behind the code. My understanding is our
# current described method wouldn't make intuitive sense to a Bayesian, as the data is
# presumed fixed, and repeatedly sampling from it would acknowledge that it comes from a
# larger population distribution and is not "fixed." In this sense, a bayesian bootstrap
# would strive to sample from a series of posterior distributions containing regression
# coefficients, instead of the observed data sample.

set.seed(101)
library(mosaic)
lower = c()
higher = c()
coef = c()
sims = 1000

for (i in 1:sims) {
  new = resample(g)
  m = glm(data = new, t6 ~ conc, family = binomial(link = 'logit'))
  coef[i] = m$coefficients[2]
  lower[i] = suppressMessages(confint(m))[2]
  higher[i] = suppressMessages(confint(m))[4]
}

hist(coef, 30, col = 'darkorange', prob = T,
     xlab = "Dose Concentation Effect on Testing Positive",
     main = 'Histogram of Dose Concentration (μ g/ml) Effect')
curve(dnorm(x, mean = mean(coef), sd = sd(coef)), add = T, col = 'purple', lwd = 2)
abline(v = mean(coef), col = 'darkblue', lwd = 2)
abline(v = mean(lower), col = 'darkgreen', lwd = 2, lty = 2)
abline(v = mean(higher), col = 'darkgreen', lwd = 2, lty = 2)
legend('topleft', legend = c("Mean Effect", "95% Confidence Interval"),
       col = c("darkblue", "darkgreen"), lty = 1:2, cex= .75)
```



```
bootstrap_CI = c(mean(lower), mean(higher))
bootstrap_CI
```

```
## [1] -8.312200 -1.522801
```

```
# 95% Frequentist Confidence Interval from bootstrap sampling:
# We are 95% certain that the true, log-odds effect of hydroxychloroquine concentration on
# testing positive for COVID lies between (-8.312200, -1.522801).

# Rather than being based on the observed sample data, which is assumed to be "fixed" from
# our above observations, this result is based on hypothesized repeated experiments.
# This is an important distinction, as we assume that our data is 'random' in that our
```

```r
# observations are sampled from a larger, unknown population distribution. We then try to
# estimate a point estimate from modeling our regression estimators after our sample data, in
# contrast to the bayesian assumption that our parameter estimates follow a distribution and
# we can assume a prior distribution before attempting to maximize the likelihood of observing
# the data given our model parameters. This is the reason why we are allowed to estimate a
# posterior distribution (a compromise between our prior distribution assumptions and the
# evidence of what our effect looks like after having seen our fixed data sample) of the
# coefficients above.



# In this frequentist interpretation, we can say that if we were to repeat this experiment
# an infinite amount of times and draw confidence intervals each of the time for dose effect,
# 95% of those intervals will contain the true, parameter effect of dose concentration value
# on testing positive for COVID.


# * Remember, these effects resemble log-odds, as we can't gauge whether 0 is a plausible
# value if we convert the log-odds values into probabilities. This is because the interval
# can never encompass 0 with both negative and positive values, as non-negativity is a
# self-evident truth of probability, written into Kolmogorov's first axiom that P(A) >= 0.

# While we are using a Frequentist method, let's analyze statistical significance through the
# p-value:
z = (0 - mean(coef)) / sd(coef)
p = pnorm(z, lower.tail = T)
qnorm(p)
```

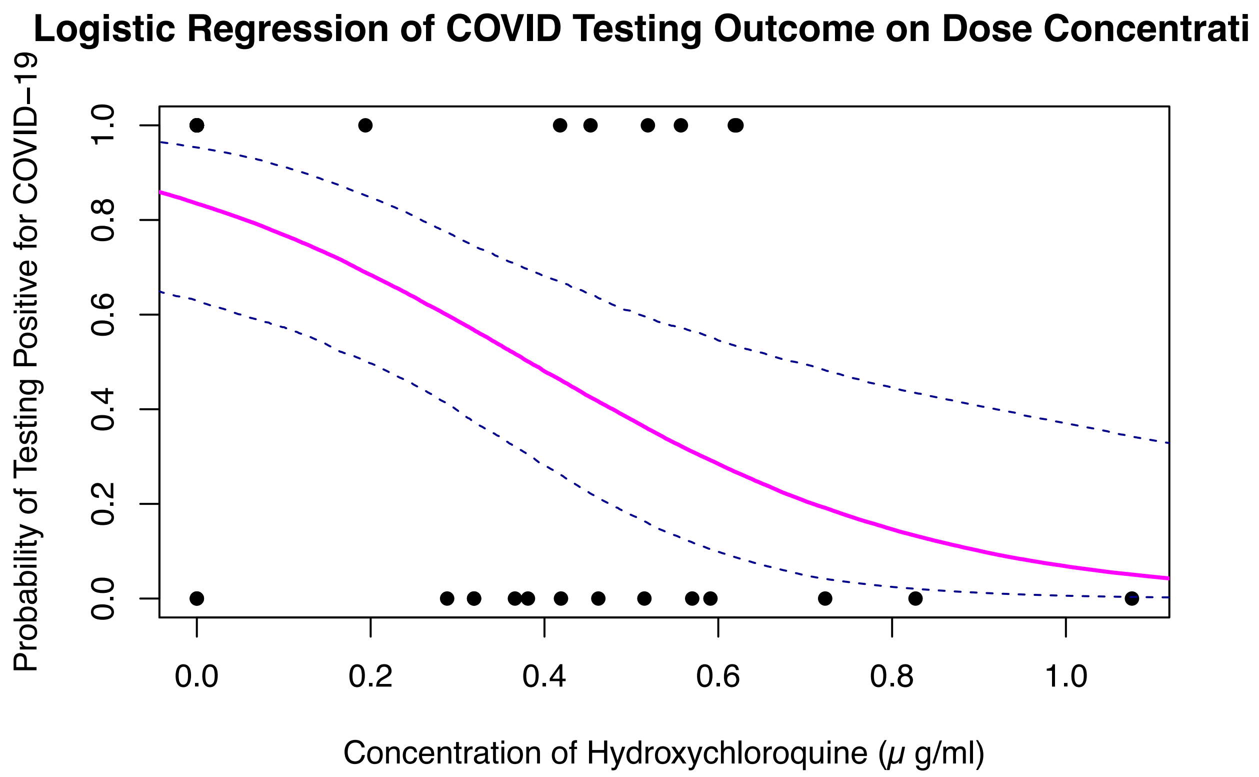
```
## [1] 2.473545
```

```r
sum(coef >= 0) / sims
```

```
## [1] 0.001
```

```r
# This is our p-value, which tells us the probability that we have observed what we have
# observed, given that our null hypothesis is true. More specifically, it is the probability
# that we observe an equal or greater value than our test-statistic of roughly ~ 2.5,
# assuming that the beta effect is 0.


# As we can see, the probability of observing this value, under the assumption that our
# null hypothesis is true, is extremely low. We can, thus, reject the null hypothesis of a 0
# beta value, for the dose concentration, log effect of testing positive for COVID,
# at the 5% & 1% significance level.


# For a visual representation of the logistic regression:
sims = as.matrix(logistic_conc)
int = sims[, 1]
conc = sims[, 2]

lower = c()
```

```r
higher = c()
med = c()
sequence = seq(min(g$conc)-.5, max(g$conc)+.5, .001)

for (i in 1:length(sequence)) {
  ypred = int + conc*sequence[i]
  lower[i] = quantile(ypred, prob = .025)
  higher[i] = quantile(ypred, prob = .975)
  med[i] = median(ypred)
}

plot(g$conc, g$t6, pch = 16, xlab = "Concentration of Hydroxychloroquine (µ g/ml)",
     ylab = "Probability of Testing Positive for COVID-19",
     main = "Logistic Regression of COVID Testing Outcome on Dose Concentration")
lines(sequence, inv_logit(lower), lty = 2, col = "darkblue")
lines(sequence, inv_logit(higher), lty = 2, col = "darkblue")
lines(sequence, inv_logit(med), lwd = 2, col = "magenta")
```



```r
# It is nice to see that our logistic regression now follows a sigmoid curve, thus indicating
# that hydroxychloroquine dose concentration serves as a robust probabilistic indicator for
# COVID-19 test results
```

```r
# This chunk of code will be attributed to result recreation in the original
# experiment.

library(readxl)
library(sqldf)
virus = read_excel("/Users/patrickpoleshuk/gautretData_(1).xlsx")
```

```r
virus$yes = ifelse(virus$chloroquine == "Yes", 1, 0)

outlist = c()
fn <- function(x){
  outlist <- c(outlist, mean(x), sd(x))
}
results = tapply(virus$age, virus$yes, fn)
#results_2 = as.data.frame(tapply(virus$age, virus$yes, fn))
results = as.data.frame(do.call(rbind, results))
results[1] = formatC(results$V1, digits = 2)
results[2] = formatC(results$V2, digits = 2)

names(results) <- c("Age_Mean", "Age_SD")
info = sqldf("SELECT Age_Mean || ' +/- ' || Age_SD AS concat FROM results")
de <- as.data.frame(sqldf("SELECT ROUND(avg(age),2) || ' +/- ' ||
                          ROUND(stdev(age), 2) AS concat FROM virus"))
info = rbind(info, de)

with(virus, t.test(age[yes == 0], age[yes == 1]))
```

```
##
##  Welch Two Sample t-test
##
## data:  age[yes == 0] and age[yes == 1]
## t = -1.893, df = 27.9, p-value = 0.06878
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -28.787767   1.137767
## sample estimates:
## mean of x mean of y
##    37.375    51.200
```

```r
# p-value = 0.06878
# t-value = -1.893
info = cbind(info, "-1.9", ".06")
info[1, 2] = ""; info[3, 2] =  ""
info[1, 3] = ""; info[3, 3] =  ""

extra = as.data.frame(as.matrix(c("Control(N=16)",
"Hydroxychloroquine(N=20)", "All(36)")))
info = cbind(extra, info)

gen = sqldf("SELECT COUNT(sex) FROM virus WHERE sex = 'M'
                          GROUP BY chloroquine")

library(qpcR)
info = qpcR:::cbind.na(info, gen)
info[3, NCOL(info)] <- sum(gen$`COUNT(sex)`)
with(virus, t.test(sexCode[yes == 0], sexCode[yes == 1]))
```

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

```
##
## data:  sexCode[yes == 0] and sexCode[yes == 1]
## t = 0.44309, df = 32.566, p-value = 0.6606
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.2695505  0.4195505
## sample estimates:
## mean of x mean of y
##     0.625     0.550
```

```
# p-value = 0.6606

info = cbind(info, ".66")
info[1, 6] = ""; info[3, 6] =  ""

clin_1 = sqldf("
      SELECT COUNT(status) FROM virus
      WHERE status = 'Asymptomatic'
      GROUP BY chloroquine
      ")

clin_2 = sqldf("
      SELECT COUNT(status) st2 FROM virus
      WHERE status = 'URTI'
      GROUP BY chloroquine
      ")

clin_3 = sqldf("
      SELECT COUNT(status) st3 FROM virus
      WHERE status = 'LRTI'
      GROUP BY chloroquine
      ")

info = qpcR:::cbind.na(info, clin_1, clin_2, clin_3)
info[3, 7] = sum(clin_1$`COUNT(status)`)
info[3, 8] = sum(clin_2$st2)
info[3, 9] = sum(clin_3$st3)

v = sqldf("SELECT * FROM virus WHERE status IS NOT NULL")
v$status = as.factor(v$status)
v$status = as.numeric(v$status)
s = v %>% filter(status != 1)
with(s, t.test(status[yes == 0], status[yes == 1]))
```

```
##
##  Welch Two Sample t-test
##
## data:  status[yes == 0] and status[yes == 1]
## t = 1.0397, df = 26.906, p-value = 0.3077
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.1623087  0.4956420
## sample estimates:
## mean of x mean of y
##  2.833333  2.666667
```

```r
# p-value = 0.3077
# We found that this p-value is calculated by filtering out the
# asymptomatic classifications, and comparing if LRTI & URTI classes
# are disproportionately assigned to the treatment/control group.
# We see, overall, from the table that this is not true in any of the
# cases, except for age which the study has already established is
# different, with the treatment group being older on average.

info = cbind(info, ".31")
info[1, 10] = ""; info[3, 10] =  ""


names(virus)[4] <- c("inclusion_time")

inc = sqldf("SELECT ROUND(avg(inclusion_time), 2),
ROUND(stdev(inclusion_time), 2) FROM virus
      GROUP BY chloroquine ")

v = sqldf("SELECT inclusion_time, yes FROM virus WHERE
          inclusion_time IS NOT NULL AND yes IS NOT NULL")
v$inclusion_time = as.numeric(v$inclusion_time)

ans = sqldf("SELECT ROUND(avg(inclusion_time), 2), ROUND(stdev(inclusion_time), 2) FROM v
      GROUP BY yes")

with(v, t.test(inclusion_time[yes == 0], inclusion_time[yes == 1]))
```

```
##
##  Welch Two Sample t-test
##
## data:  inclusion_time[yes == 0] and inclusion_time[yes == 1]
## t = -0.14381, df = 17.651, p-value = 0.8873
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.431224  2.120113
## sample estimates:
## mean of x mean of y
##  3.900000  4.055556
```

```r
# t-value = -0.14381
# p-value = 0.8873
info = qpcR:::cbind.na(info, ans)

names(info)[11:12] <- c("mi", "si")
v = sqldf("SELECT * FROM v WHERE inclusion_time IS NOT NULL")
info[3, 11] <- formatC(mean(v$inclusion_time), digits = 2)
info[3, 12] <- formatC(sd(v$inclusion_time), digits = 2)

concat2 = sqldf("SELECT mi || ' +/- ' || si AS concat2 FROM info")
info = cbind(info, concat2)
info = info %>% dplyr::select(-mi, -si)
info = cbind(info, "-.14", ".88")
info[1, 12] = ""; info[3, 12] = ""
info[1, 13] = ""; info[3, 13] = ""

names(info) <- c("", "M+-SD[AGE]", "t",
                 "p", "Male-count", "p",
                 "Asymp", "URTI", "LRTI",
                 "p", "M+-S[INCLU]", "t",
                 "p")
library(xtable)
print.xtable(xtable(info), file = "./Downloads/info_covid_2.txt")

agrestiCaffo <- function(x1,n1,x2,n2,alpha=.05) {
  #
  # The Agresti-Caffo proportion estimate with alpha = .05
  #

  zCrit       <- qnorm(1-alpha/2);

  pHat1       <- (x1)/(n1);
  pPrime1     <- (x1+1)/(n1+2);
  pPrimeV1    <-  (pPrime1*(1-pPrime1)/(n1+2));
  pHat2       <- (x2)/(n2);
  pPrime2     <- (x2+1)/(n2+2);
  pPrimeV2    <- (pPrime2*(1-pPrime2)/(n2+2));
  pPrimeSD    <- sqrt(pPrimeV1+pPrimeV2);
  lCI         <- pPrime1-pPrime2 - zCrit*pPrimeSD;
  uCI         <- pPrime1-pPrime2 + zCrit*pPrimeSD;
  zScore      <- (pPrime1-pPrime2)/pPrimeSD;

  pValue      <- (1-pnorm(abs(zScore)))*2;
  pValueGgtP  <- (1-pnorm(zScore))

  myParms          <- list()

  myParms$zCrit    <- zCrit;
  myParms$pHat1    <- pHat1;
  myParms$pPrime1  <- pPrime1;
  myParms$pHat2    <- pHat2;
  myParms$pPrime2  <- pPrime2;
  myParms$deltaP   <- pHat1-pHat2;
  myParms$lower    <- lCI;
  myParms$upper    <- uCI;
  myParms$pValue   <- pValue;
  myParms$pValueGgtP  <- pValueGgtP;
  return(myParms)
}

# DAY 3:
sqldf("SELECT COUNT(D3) FROM virus WHERE chloroquine = 'Yes' AND
      D3 = 'NEG' ")
```

```
##   COUNT(D3)
## 1        10
```

```r
# 10
sqldf("SELECT COUNT(D3) FROM virus WHERE chloroquine = 'Yes'")
```

```
##   COUNT(D3)
## 1        20
```

```r
# 20

sqldf("SELECT COUNT(D3) FROM virus WHERE chloroquine = 'No' AND
      D3 = 'NEG' ")
```

```
##   COUNT(D3)
## 1         1
```

```r
# 1
sqldf("SELECT COUNT(D3) FROM virus WHERE chloroquine = 'No'")
```

```
##   COUNT(D3)
## 1        16
```

```r
# 16

# DAY 4:
sqldf("SELECT COUNT(D4) FROM virus WHERE chloroquine = 'Yes' AND
      D4 = 'NEG' ")
```

```
##   COUNT(D4)
## 1        12
```

```r
# 12
sqldf("SELECT COUNT(D4) FROM virus WHERE chloroquine = 'Yes'")
```

```
##   COUNT(D4)
## 1        20
```

```r
# 20

sqldf("SELECT COUNT(D4) FROM virus WHERE chloroquine = 'No' AND
      D4 = 'NEG'")
```

```
##   COUNT(D4)
## 1         4
```

```r
# 4
sqldf("SELECT COUNT(D4) FROM virus WHERE chloroquine = 'No'")
```

```
##   COUNT(D4)
## 1        16
```

```
# 16


# NOTE: We count ND for the negative treatment count on day 5 & day 6,
# because even though the subject hasn't been tested that day, it was already
# shown in the previous day that the subject tested negative.


# DAY 5:
sqldf("SELECT COUNT(D5) FROM virus WHERE chloroquine = 'Yes' AND
      (D5 = 'ND' OR D5 = 'NEG')")
```

```
##   COUNT(D5)
## 1        13
```

```
# 13
sqldf("SELECT COUNT(D5) FROM virus WHERE chloroquine = 'Yes'")
```

```
##   COUNT(D5)
## 1        20
```

```
# 20

sqldf("SELECT COUNT(D5) FROM virus WHERE chloroquine = 'No' AND
      D5 = 'NEG'")
```

```
##   COUNT(D5)
## 1         3
```

```
# 3
sqldf("SELECT COUNT(D5) FROM virus WHERE chloroquine = 'No'")
```

```
##   COUNT(D5)
## 1        16
```

```
# 16

# DAY 6:
sqldf("SELECT COUNT(D6) FROM virus WHERE chloroquine = 'Yes' AND
      (D6 = 'NEG' OR D6 = 'ND')")
```

```
##   COUNT(D6)
## 1        14
```

```
# 14
sqldf("SELECT COUNT(D6) FROM virus WHERE chloroquine = 'Yes'")
```

```
##   COUNT(D6)
## 1        20
```

```r
# 20

sqldf("SELECT COUNT(D6) FROM virus WHERE chloroquine = 'No' AND
      D6 = 'NEG'")
```

```
##   COUNT(D6)
## 1         2
```

```r
# 2
sqldf("SELECT COUNT(D6) FROM virus WHERE chloroquine = 'No'")
```

```
##   COUNT(D6)
## 1        16
```

```r
# 16

agrestiCaffo(10, 20, 1, 16)
```

```
## $zCrit
## [1] 1.959964
##
## $pHat1
## [1] 0.5
##
## $pPrime1
## [1] 0.5
##
## $pHat2
## [1] 0.0625
##
## $pPrime2
## [1] 0.1111111
##
## $deltaP
## [1] 0.4375
##
## $lower
## [1] 0.1344662
##
## $upper
## [1] 0.6433116
##
## $pValue
## [1] 0.002736951
##
## $pValueGgtP
## [1] 0.001368476
```

```r
# 0.002736951
agrestiCaffo(12, 20, 4, 16)
```

```
## $zCrit
## [1] 1.959964
##
## $pHat1
## [1] 0.6
##
## $pPrime1
## [1] 0.5909091
##
## $pHat2
## [1] 0.25
##
## $pPrime2
## [1] 0.2777778
##
## $deltaP
## [1] 0.35
##
## $lower
## [1] 0.02154174
##
## $upper
## [1] 0.6047209
##
## $pValue
## [1] 0.0353122
##
## $pValueGgtP
## [1] 0.0176561
```

```
# 0.0353122
agrestiCaffo(13, 20, 3, 16)
```

```
## $zCrit
## [1] 1.959964
##
## $pHat1
## [1] 0.65
##
## $pPrime1
## [1] 0.6363636
##
## $pHat2
## [1] 0.1875
##
## $pPrime2
## [1] 0.2222222
##
## $deltaP
## [1] 0.4625
##
## $lower
## [1] 0.1361262
##
```

```
## $upper
## [1] 0.6921566
##
## $pValue
## [1] 0.003504445
##
## $pValueGgtP
## [1] 0.001752222
```

```
# 0.003504445
agrestiCaffo(14, 20, 2, 16)
```

```
## $zCrit
## [1] 1.959964
##
## $pHat1
## [1] 0.7
##
## $pPrime1
## [1] 0.6818182
##
## $pHat2
## [1] 0.125
##
## $pPrime2
## [1] 0.1666667
##
## $deltaP
## [1] 0.575
##
## $lower
## [1] 0.2553024
##
## $upper
## [1] 0.7750006
##
## $pValue
## [1] 0.0001020631
##
## $pValueGgtP
## [1] 5.103155e-05
```

```
# .0001020631

library(tidyverse)
table2 = tibble(
  x = c("Hydroxychloroquine(N=20)",
"Control(N=16)"), Prop_Day3 = c('10/20',
                                             '1/16'), Percent_Day3 =
  c('50%', '6.3%'), p3 = c("", ".003"), Prop_Day4 = c('12/20',
                                                          '4/16'),
Percent_Day4 = c('60%', '25%'), p4 = c("", "0.04"), Prop_Day5 =
  c('13/20', '3/16'), Percent_Day5 = c("56%", "18.8%"), p5 = c(
    "", ".004"), Prop_Day6 = c("14/20", "2/16"), Percent_Day6 = c(
      "70%", "12.5%"), p6 = c("", ".0001")

)

names(table2) <- c("", "Neg/Pos3", "%", "p3", "Neg/Pos4", "%", "p4",
                   "Neg/Pos5", "%", "p5", "Neg/Pos6", "%", "p6")

print.xtable(xtable(table2), file = "./table2_covid_data.txt")

# With the Azithromycin counts, we don't need to do new counts for
# the control group and chloroquine tests. Since only the subjects who
# were dosed with Azithromycin were in the treatment group, the negative
# counts can just be subtracted from the base hydroxychloroquine negative
# counts.

# To find the statistical significance of the differences between
# three proportions, I just pooled the Azithromycin & Hydroxychloroquine
# proportions against the control proportions.

# DAY 3:
sqldf("SELECT COUNT(D3) FROM virus WHERE azithromycin = 'Yes' AND
      D3 = 'NEG' ")
```

```
##   COUNT(D3)
## 1         5
```

```
# 5
sqldf("SELECT COUNT(D3) FROM virus WHERE azithromycin = 'Yes'")
```

```
##   COUNT(D3)
## 1         6
```

```
# 6

# DAY 4:
sqldf("SELECT COUNT(D4) FROM virus WHERE azithromycin = 'Yes' AND
      D4 = 'NEG' ")
```

```
##   COUNT(D4)
## 1         5
```

```
# 5
sqldf("SELECT COUNT(D4) FROM virus WHERE azithromycin = 'Yes'")
```

```
##   COUNT(D4)
## 1         6
```

```
# 6

# DAY 5:
sqldf("SELECT COUNT(D5) FROM virus WHERE azithromycin = 'Yes' AND
      D5 = 'NEG' ")
```

```
##   COUNT(D5)
## 1         6
```

```
# 6
sqldf("SELECT COUNT(D5) FROM virus WHERE azithromycin = 'Yes'")
```

```
##   COUNT(D5)
## 1         6
```

```
# DAY 6:
sqldf("SELECT COUNT(D6) FROM virus WHERE azithromycin = 'Yes' AND
      D6 = 'NEG' ")
```

```
##   COUNT(D6)
## 1         6
```

```
# 6
sqldf("SELECT COUNT(D6) FROM virus WHERE azithromycin = 'Yes'")
```

```
##   COUNT(D6)
## 1         6
```

```
# 6

# Pooled 5/14 + 5/6 = 10/20
agrestiCaffo(1, 16, 10, 20)
```

```
## $zCrit
## [1] 1.959964
##
## $pHat1
## [1] 0.0625
##
## $pPrime1
## [1] 0.1111111
##
## $pHat2
## [1] 0.5
##
## $pPrime2
## [1] 0.5
##
## $deltaP
## [1] -0.4375
##
## $lower
## [1] -0.6433116
##
## $upper
## [1] -0.1344662
##
```

```
## $pValue
## [1] 0.002736951
##
## $pValueGgtP
## [1] 0.9986315
```

```
# 0.002736951

# Pooled 7/14 + 5/6 = 12/20
agrestiCaffo(4, 16, 12, 20)
```

```
## $zCrit
## [1] 1.959964
##
## $pHat1
## [1] 0.25
##
## $pPrime1
## [1] 0.2777778
##
## $pHat2
## [1] 0.6
##
## $pPrime2
## [1] 0.5909091
##
## $deltaP
## [1] -0.35
##
## $lower
## [1] -0.6047209
##
## $upper
## [1] -0.02154174
##
## $pValue
## [1] 0.0353122
##
## $pValueGgtP
## [1] 0.9823439
```

```
# 0.0353122

# Pooled 7/14 + 6/6 = 13/20
agrestiCaffo(3, 16, 13, 20)
```

```
## $zCrit
## [1] 1.959964
##
## $pHat1
## [1] 0.1875
##
## $pPrime1
## [1] 0.2222222
##
## $pHat2
## [1] 0.65
##
## $pPrime2
## [1] 0.6363636
##
## $deltaP
## [1] -0.4625
##
## $lower
## [1] -0.6921566
##
## $upper
## [1] -0.1361262
##
## $pValue
## [1] 0.003504445
##
## $pValueGgtP
## [1] 0.9982478
```

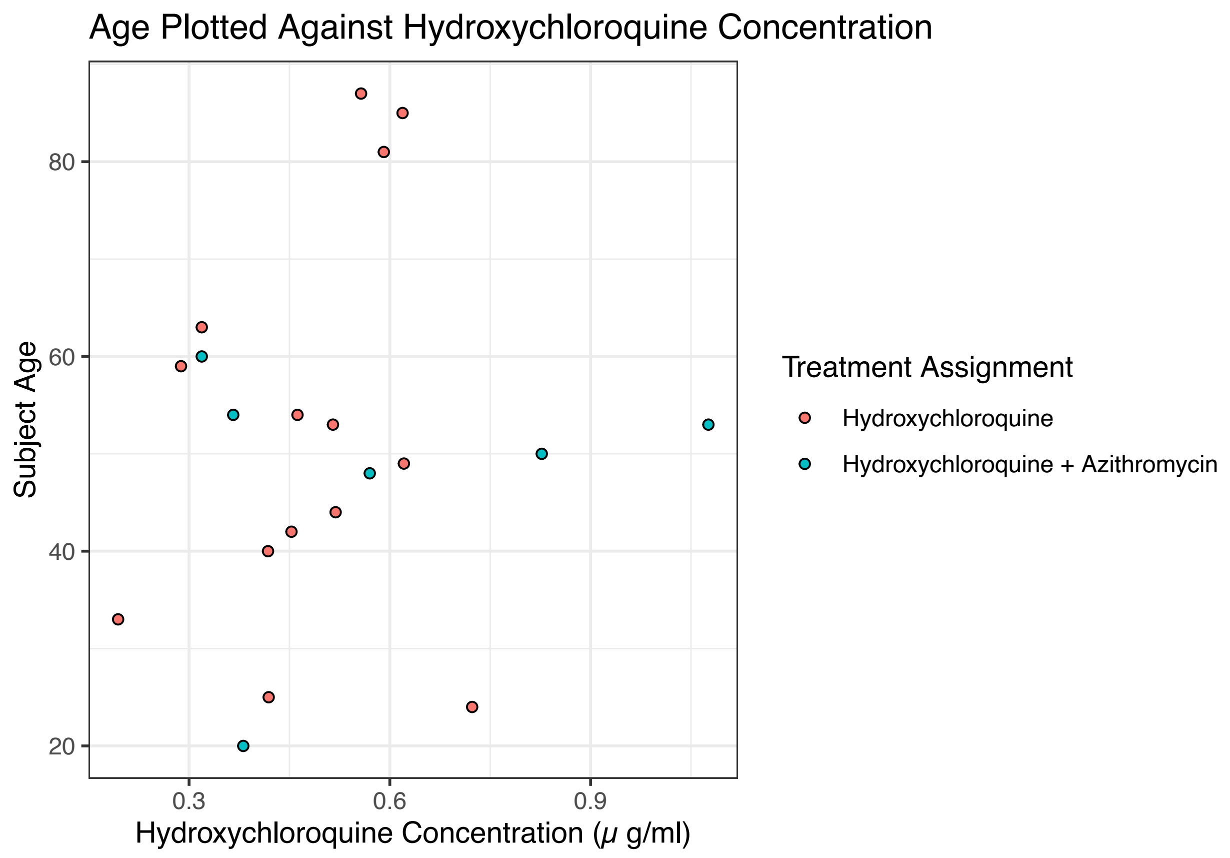
```r
# 0.003504445

# Pooled 8/14 + 6/6 = 14/20
agrestiCaffo(2, 16, 14, 20)
```

```
## $zCrit
## [1] 1.959964
##
## $pHat1
## [1] 0.125
##
## $pPrime1
## [1] 0.1666667
##
## $pHat2
## [1] 0.7
##
## $pPrime2
## [1] 0.6818182
##
## $deltaP
## [1] -0.575
##
## $lower
## [1] -0.7750006
##
## $upper
## [1] -0.2553024
##
## $pValue
## [1] 0.0001020631
```

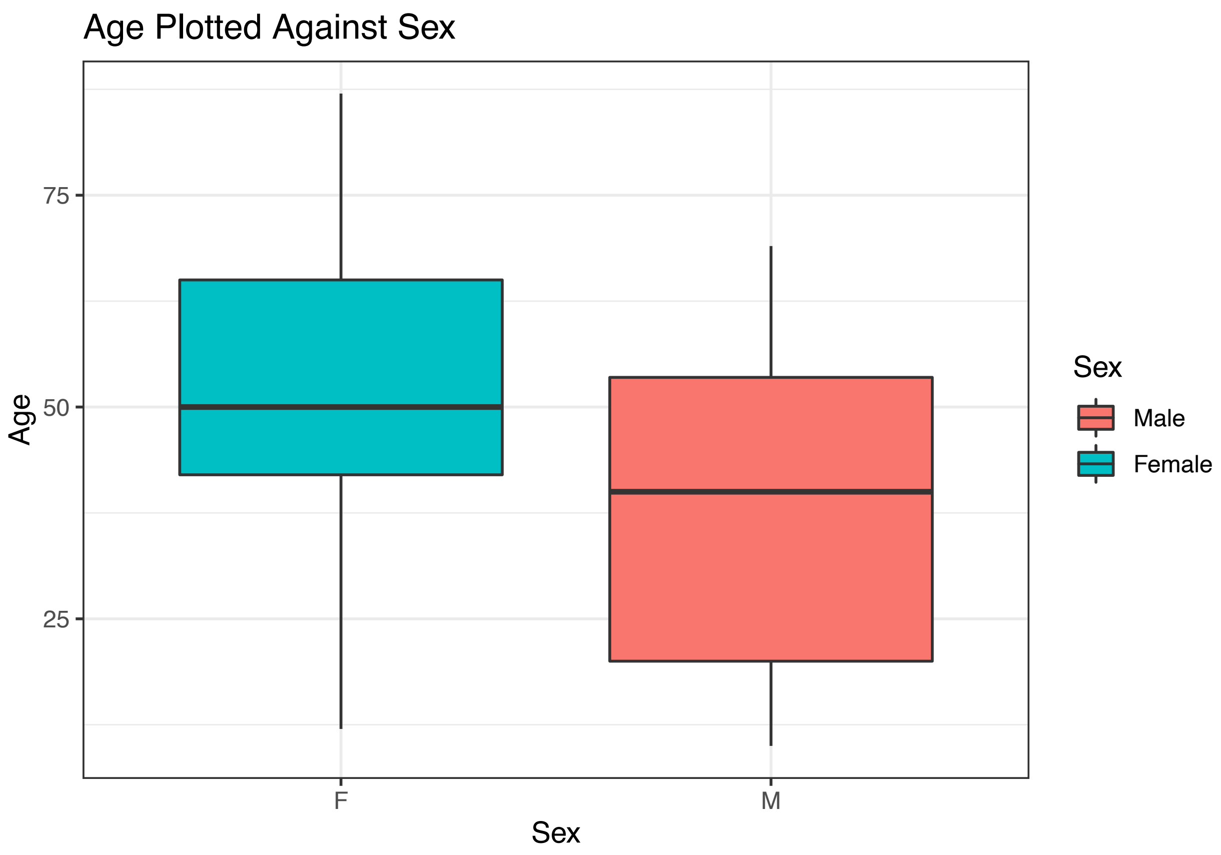
```
##
## $pValueGgtP
## [1] 0.999949
```

```
# 0.0001020631


table3 = tibble(
  x = c("Control(N=16)",
"HydC.(N=14)", "Azithromycin(N=6)"), Prop_Day3 = c('1/16',
                                             '5/14', "5/6"),
Percent_Day3 =
  c('6.3%', '35.7%', '83.3%'), p3 = c("", ".003", ""),
Prop_Day4 = c('4/16', '7/14', '5/6'),
Percent_Day4 = c('25%', '50%', '83.3%'), p4 = c("", "0.04", ""),
Prop_Day5 =
  c('3/16', '7/14', '6/6'), Percent_Day5 = c("18.8%", "50%", "100%"),
p5 = c(
    "", ".004", ""), Prop_Day6 = c("2/16", "8/14", "6/6"),
Percent_Day6 = c(
      "12.5%", "57.1%", "100%"), p6 = c("", ".0001", "")
)


names(table3) <- c("", "Neg/Pos3", "%", "p3", "Neg/Pos4", "%", "p4",
                   "Neg/Pos5", "%", "p5", "Neg/Pos6", "%", "p6")
print.xtable(xtable(table3), file = "./table3_covid_data.txt")
```

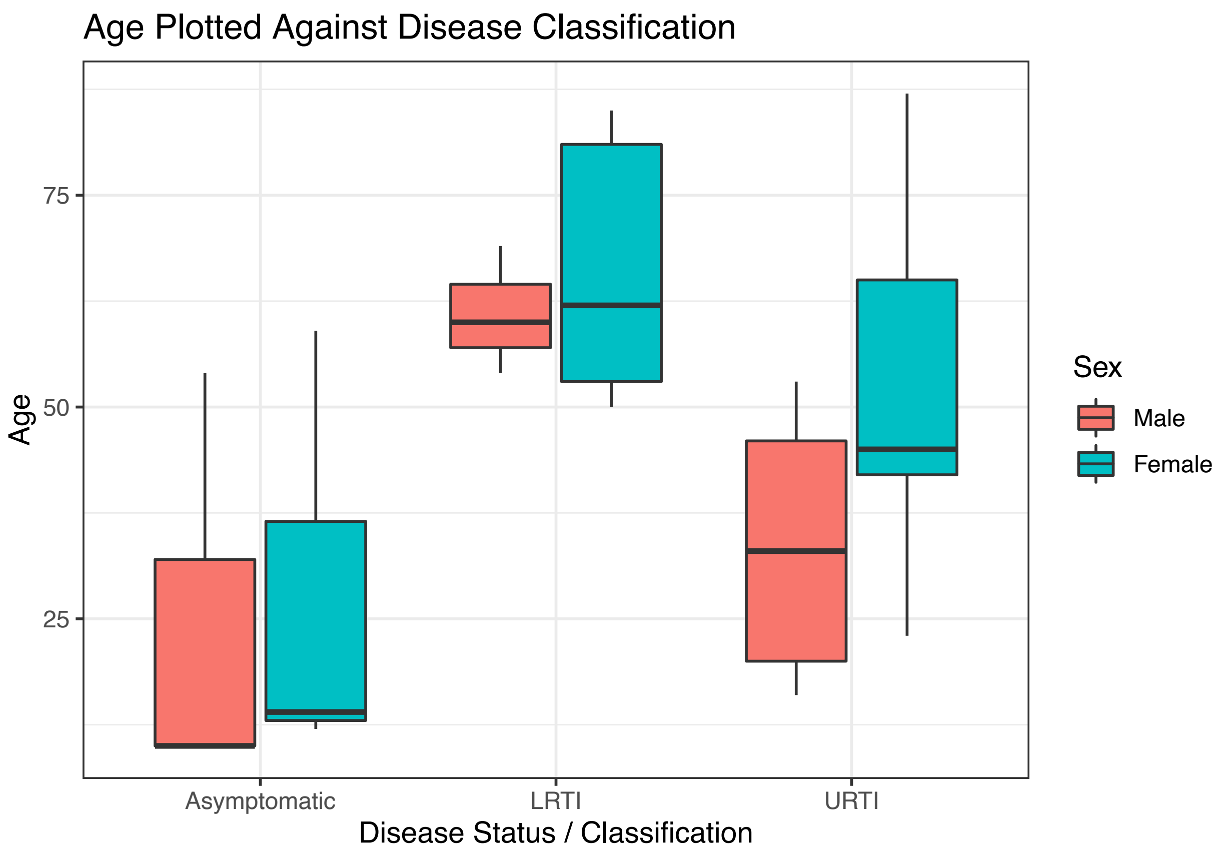
```
# Additional Graphics for Potential Variable Relationships.

gg = as.data.frame(g %>% filter(conc > 0))
gg$code = ifelse(gg$category == "Control", 0, ifelse(gg$category == "Hydrox", 1, 2))
ggplot(data = gg, aes(y = age, x = conc, fill = as.factor(as.numeric(code)))) +
  geom_point(pch = 21) +
  labs(title = "Age Plotted Against Hydroxychloroquine Concentration",
       x = "Hydroxychloroquine Concentration (µ g/ml)",
       y = "Subject Age") + scale_fill_discrete(name = 'Treatment Assignment',
                                                labels = c("Hydroxychloroquine",
                                 "Hydroxychloroquine + Azithromycin"))
```



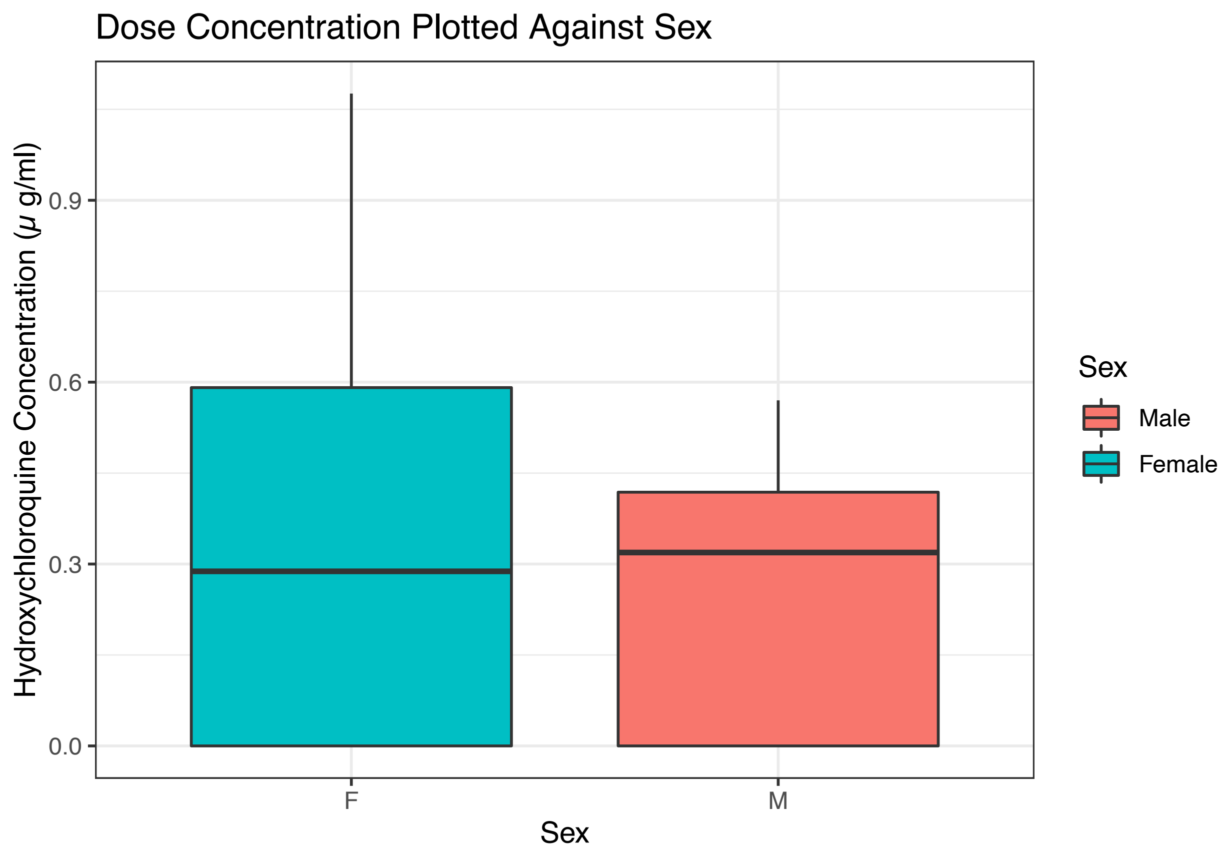
```
ggplot(data = g, aes(y = age, x = sex, fill = as.factor(sexCode))) +
  geom_boxplot() + labs(x = "Sex", y = "Age",
                        title = "Age Plotted Against Sex") +
  scale_fill_discrete(name = 'Sex', labels = c("Male", "Female"))
```



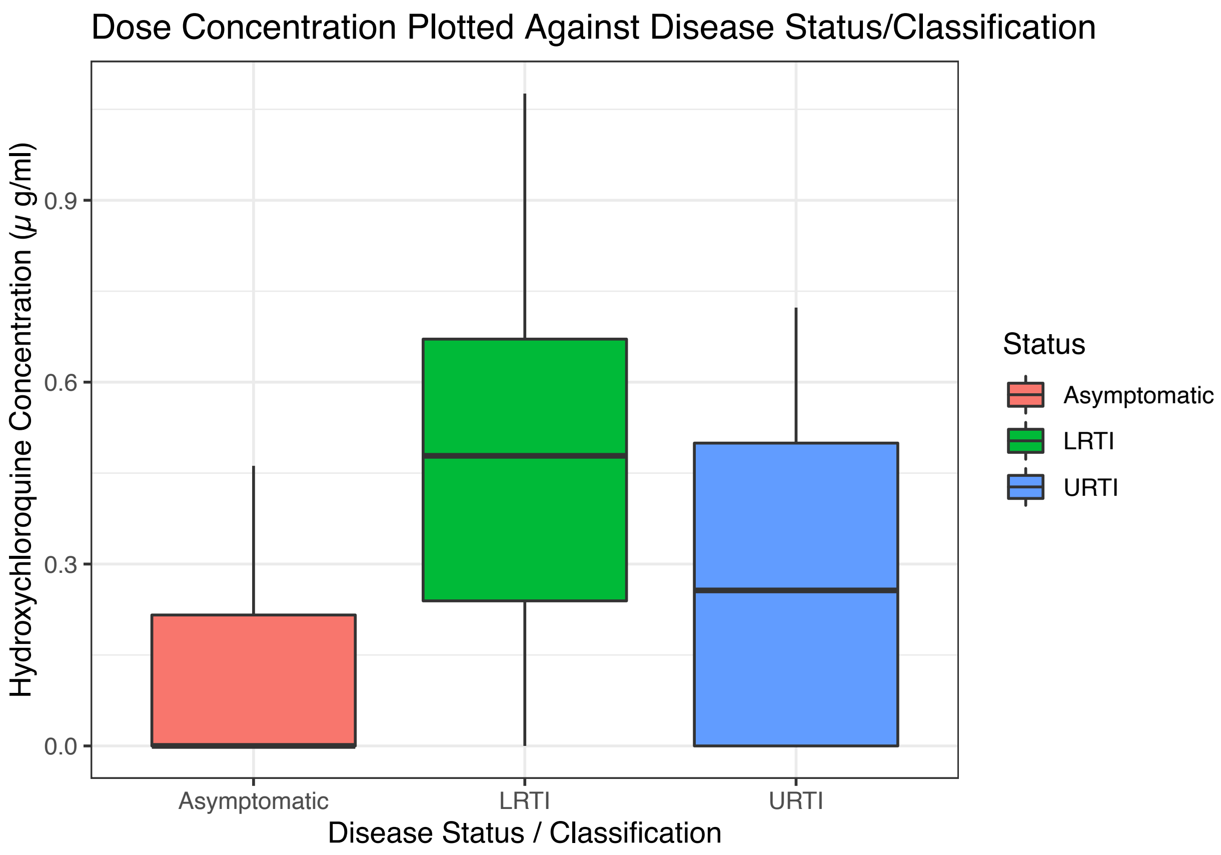
```r
ggplot(data = g, aes(y = age, x = status, fill = as.factor(sexCode))) +
  geom_boxplot() + labs(x = "Disease Status / Classification", y = "Age",
                        title = "Age Plotted Against Disease Classification") +
  scale_fill_discrete(name = 'Sex', labels = c('Male', "Female"))
```



```
ggplot(data = g, aes(y = conc, x = sex, fill = as.factor(sexCode))) +
  geom_boxplot() + labs(x = "Sex",
                        y = "Hydroxychloroquine Concentration (μ g/ml)",
                        title = "Dose Concentration Plotted Against Sex") +
   scale_fill_discrete(name = 'Sex', labels = c('Male', "Female"))
```



```r
ggplot(data = g, aes(y = conc, x = status, fill = as.factor(statCode))) +
  geom_boxplot() + labs(x = "Disease Status / Classification",
                        y = "Hydroxychloroquine Concentration (μ g/ml)",
  title = "Dose Concentration Plotted Against Disease Status/Classification") +
   scale_fill_discrete(name = 'Status', labels = c("Asymptomatic",
                                                   "LRTI",
                                                   "URTI"))
```


Dose Concentration Plotted Against Disease Status/Classification
Hydroxychloroquine Concentration (μ g/ml)
0.9
0.6
0.3
0.0
Asymptomatic
LRTI
URTI
Disease Status / Classification
Status
Asymptomatic
LRTI
URTI