8.3 simulating from the fitted model

Chris Parrish

July 3, 2016

Contents

8.3 simulating from the fitted model

reference:
- ARM chapter 08, github

```
library(arm)            # for sim
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
library(ggplot2)
library(reshape2)       # for melt
```

speed of light (Simon Newcomb, 1882)

algorithm for model checking :
compare distribution of data y to distributions of simulated $\tilde{y}$

1. data : y
2. use data to fit model : $y \sim 1$ with parameters β and σ
   find beta and sigma such that $y = X\beta + \epsilon$
3. use model to generate n hypothetical values of $\tilde{y}$
   $\tilde{y} = X\beta + \epsilon$

4. replicate step 3 $n.sims$ times
   result is a matrix with dims $n.sims, n$
5. compare the distribution of the real data y with the distributions of the simulated $\tilde{y}$

check the model : if the distributions of the simulated $\tilde{y}$ do not correspond to the distribution of the original data y, then the model is suspect

simulate data, fit the model, and check the coverage of the conf intervals

```
source("lightspeed.data.R", echo = TRUE)
```

```
##
## > "y" <- c(28, 26, 33, 24, 34, -44, 27, 16, 40, -2,
## +     29, 22, 24, 21, 25, 30, 23, 29, 31, 19, 24, 20, 36, 32, 36,
## +     28, 25, 21, 28, 29, 37,  .... [TRUNCATED]
##
## > "N" <- 66
```

```
str(y)
```

```
##  num [1:66] 28 26 33 24 34 -44 27 16 40 -2 ...
```

model

lightspeed.stan

```
data {
  int<lower=0> N;
  vector[N] y;
}
parameters {
  vector[1] beta;
  real<lower=0> sigma;
}
model {
  y ~ normal(beta[1], sigma);
}
```

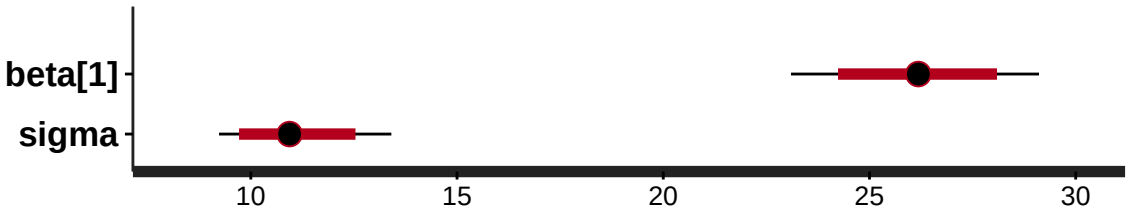
fit

```
## Model fit (lightspeed.stan)
## lm (y ~ 1)
dataList.1 <- c("N","y")
lightspeed.sf1 <- stan(file='lightspeed.stan', data=dataList.1,
                       iter=1000, chains=4)
```

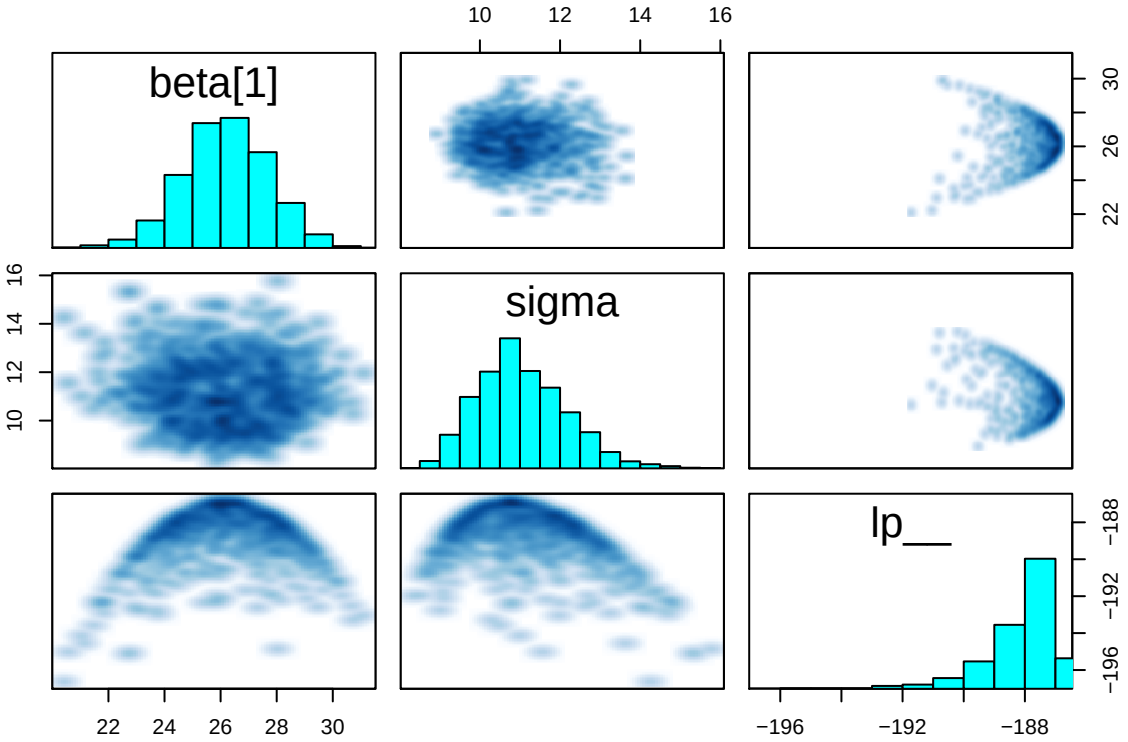
```
plot(lightspeed.sf1)
```

```
## ci_level: 0.8 (80% intervals)
```

```
## outer_level: 0.95 (95% intervals)
```




```
pairs(lightspeed.sf1)
```




```
print(lightspeed.sf1)
```

```
## Inference for Stan model: lightspeed.
## 4 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500, total post-warmup draws=2000.
##
##           mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff
## beta[1]   26.19    0.04 1.52   23.10   25.20   26.19   27.23   29.11  1298
## sigma     11.05    0.03 1.11    9.23   10.24   10.94   11.78   13.41  1305
## lp__    -188.09    0.04 1.21 -191.22 -188.59 -187.75 -187.22 -186.87  1073
##         Rhat
## beta[1]    1
## sigma      1
## lp__       1
##
## Samples were drawn using NUTS(diag_e) at Fri Jul  8 13:49:23 2016.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
##  The estimated Bayesian Fraction of Missing Information is a measure of
##  the efficiency of the sampler with values close to 1 being ideal.
##  For each chain, these estimates are
```

```
##  0.8 0.8 0.7 1.1
```

post

```
post <- extract(lightspeed.sf1)
str(post)
```

```
## List of 3
##  $ beta : num [1:2000, 1] 25.8 27 27.7 28.2 27.2 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ iterations: NULL
##   .. ..$           : NULL
##  $ sigma: num [1:2000(1d)] 12.5 11.1 10.4 10.6 12 ...
##   ..- attr(*, "dimnames")=List of 1
##   .. ..$ iterations: NULL
##  $ lp__ : num [1:2000(1d)] -188 -187 -188 -188 -188 ...
##   ..- attr(*, "dimnames")=List of 1
##   .. ..$ iterations: NULL
```

create the replicated data

```
## Create the replicated data
n.sims <- 1000
```

create fake data

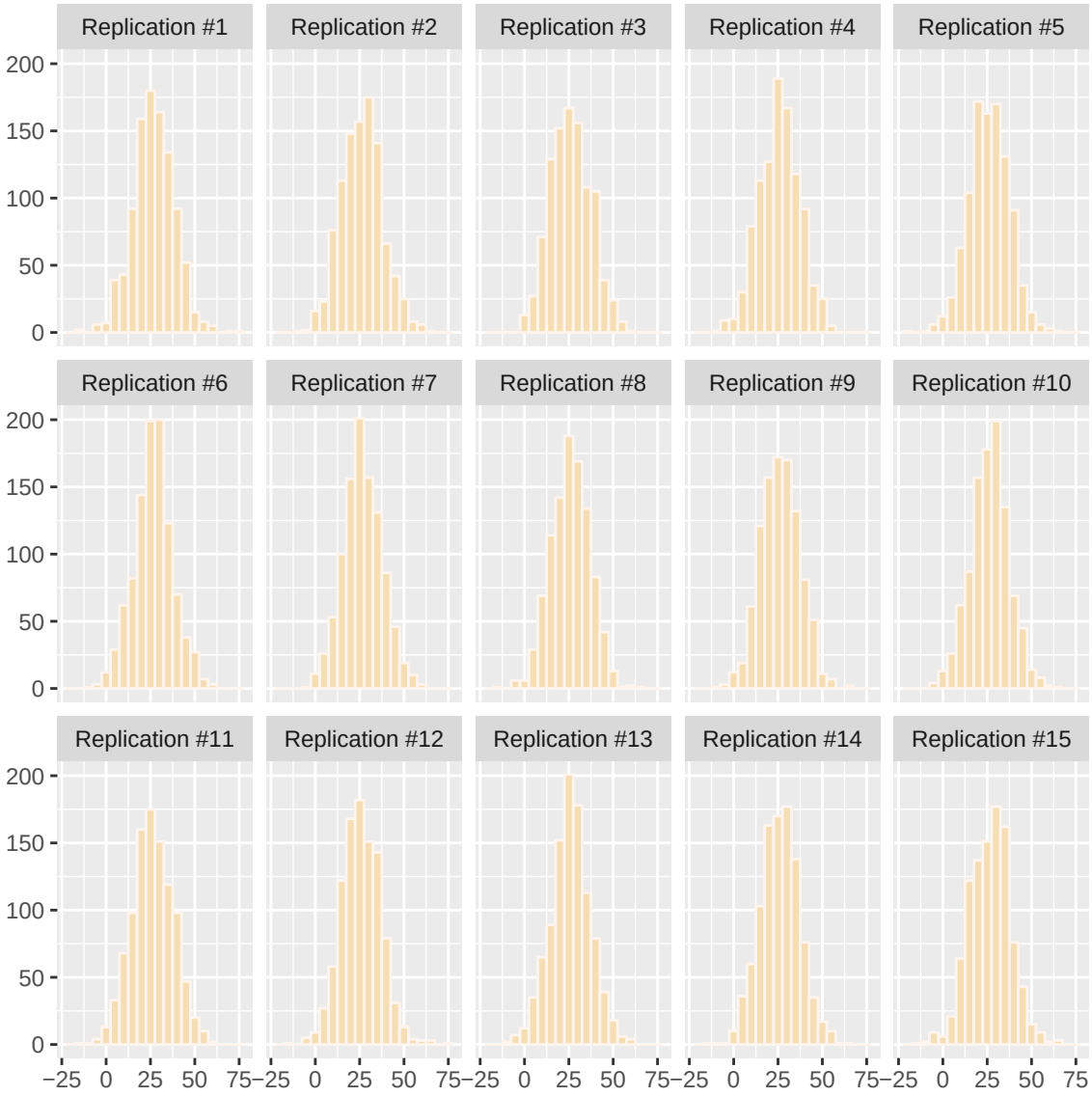
```
## Create fake data
n <- 15
y.rep <- array (NA, c(n.sims, n))
for (s in 1:n.sims){
  y.rep[s,] <- rnorm (n, post$beta[s], post$sigma[s])
}
str(y.rep)
```

```
##  num [1:1000, 1:15] 35.31 9.44 33.4 21.33 28.24 ...
```

```
## Histogram of replicated data (Figure 8.4)
y.new <- melt(y.rep)
y.new$Var2 <- factor(y.new$Var2,
                     levels=c('1','2','3','4','5','6','7','8','9','10','11','12','13','14','15'),
                     labels=c('Replication #1','Replication #2','Replication #3','Replication #4',
                     'Replication #5','Replication #6','Replication #7','Replication #8',
                     'Replication #9','Replication #10','Replication #11','Replication #12',
                     'Replication #13','Replication #14','Replication #15'))
str(y.new)
```

```
## 'data.frame':    15000 obs. of  3 variables:
##  $ Var1 : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Var2 : Factor w/ 15 levels "Replication #1",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ value: num  35.31 9.44 33.4 21.33 28.24 ...
```

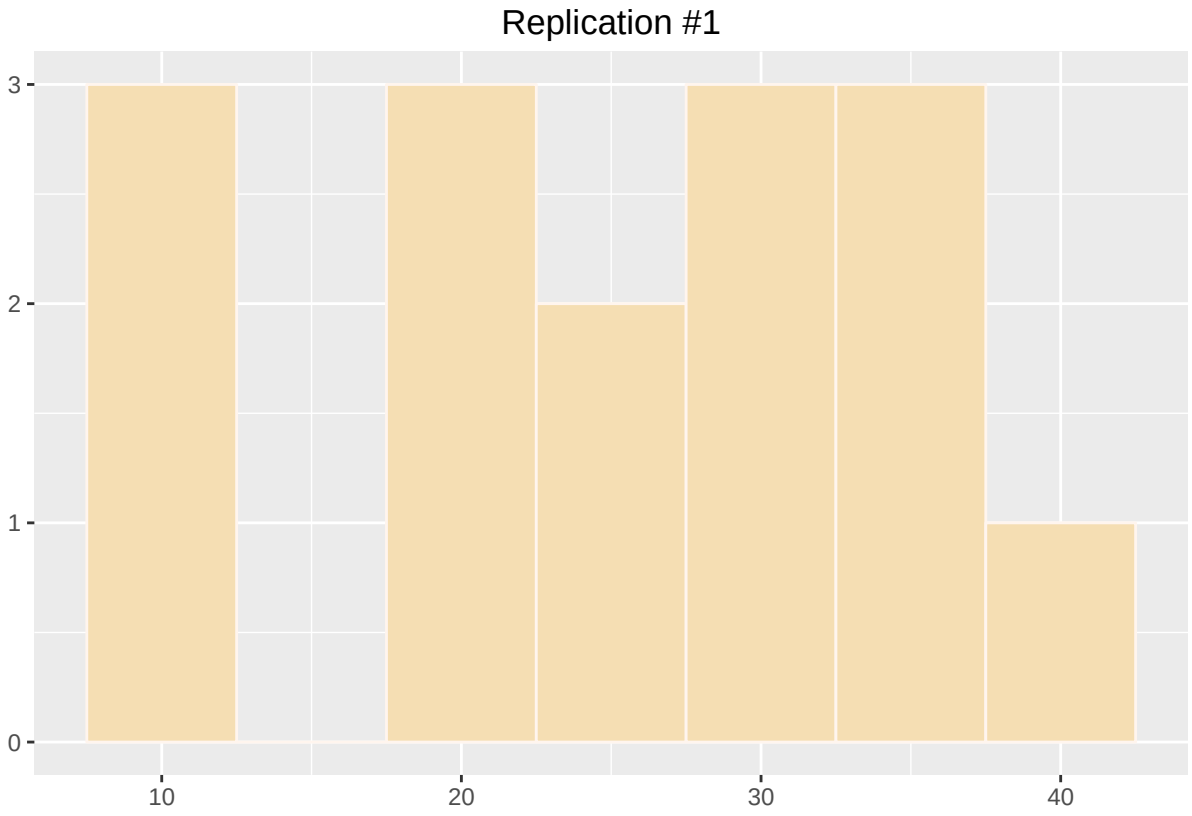
```
p1 <- ggplot(y.new, aes(value)) +
      geom_histogram(colour = "seashell", fill = "wheat", binwidth=5) +
      theme_gray() +
      facet_wrap( ~ Var2, ncol=5) +
      theme(axis.title.y = element_blank(), axis.title.x=element_blank())
print(p1)
```

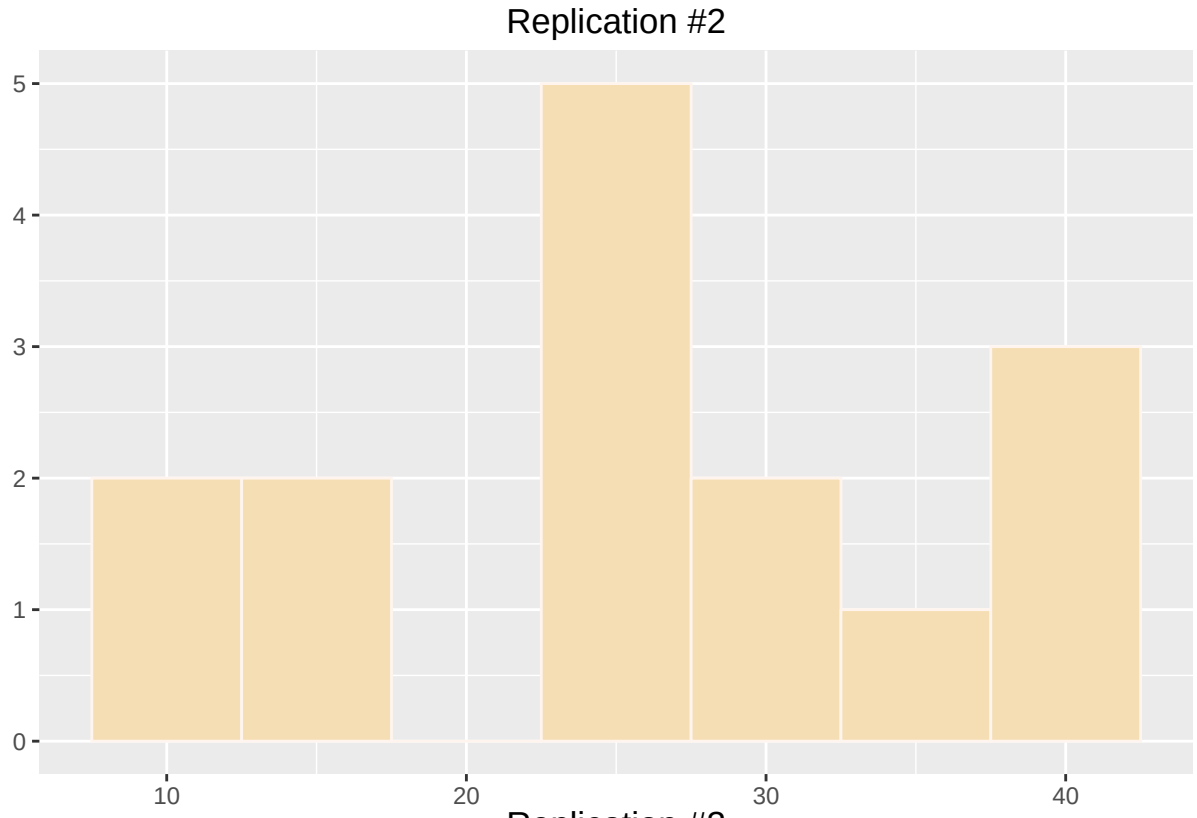


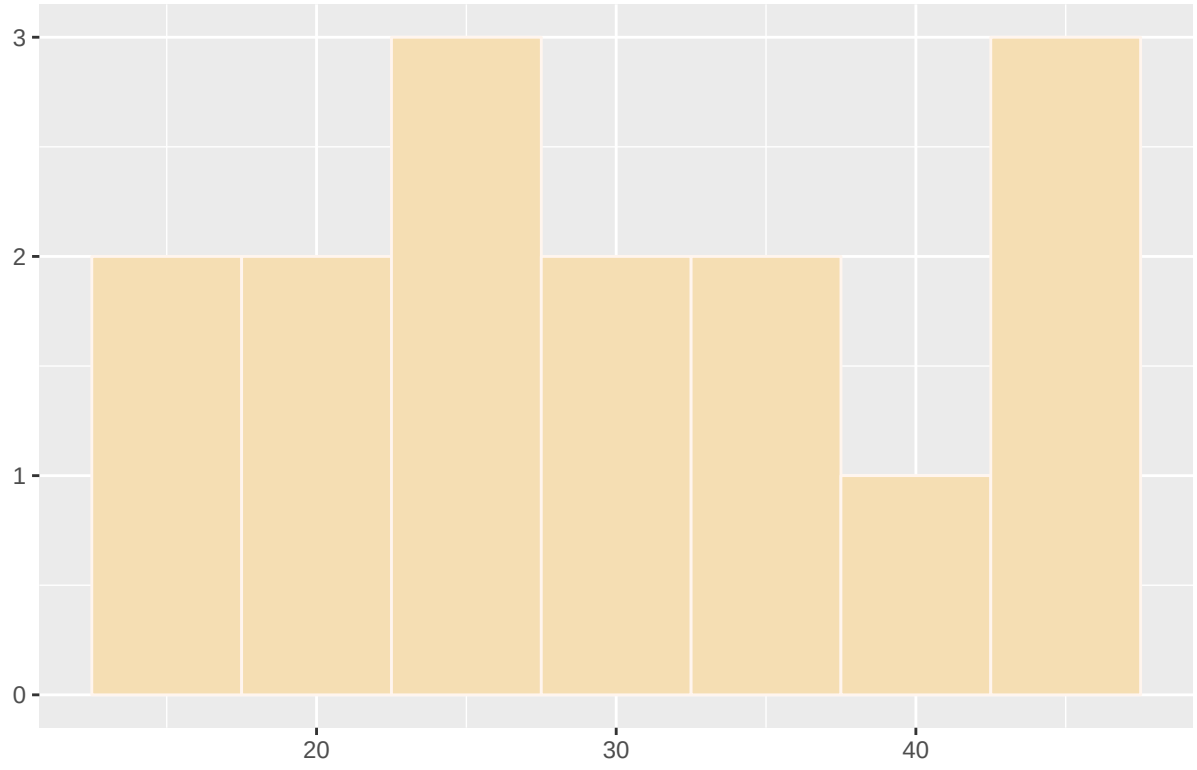

```
## Write a function to make histograms with specified bin widths and ranges
Hist.preset <- function (a, width, xtitle,ytitle,maintitle){
#  dev.new()
  a.hi <- max (a, na.rm=TRUE)
  a.lo <- min (a, na.rm=TRUE)
  if (is.null(width)) width <- min (sqrt(a.hi-a.lo), 1e-5)
  bin.hi <- width*ceiling(a.hi/width)
  bin.lo <- width*floor(a.lo/width)
  frame1 = data.frame(x1=a)
  p2 <- ggplot(frame1,aes(x=x1)) +
        geom_histogram(colour = "seashell", fill = "wheat", binwidth=width) +
```

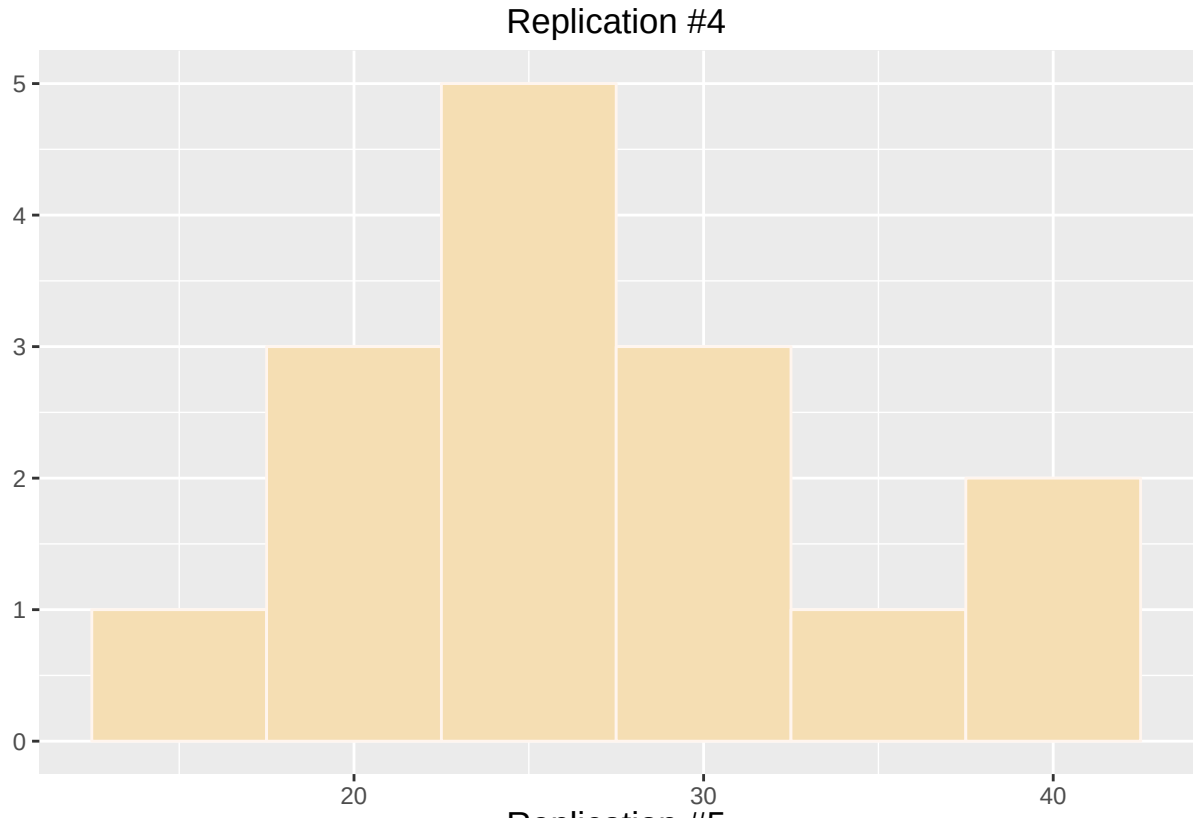
```
        theme_gray() +
        scale_x_continuous(xtitle) +
        scale_y_continuous(ytitle) +
        labs(title=maintitle)
  print(p2)
}
```

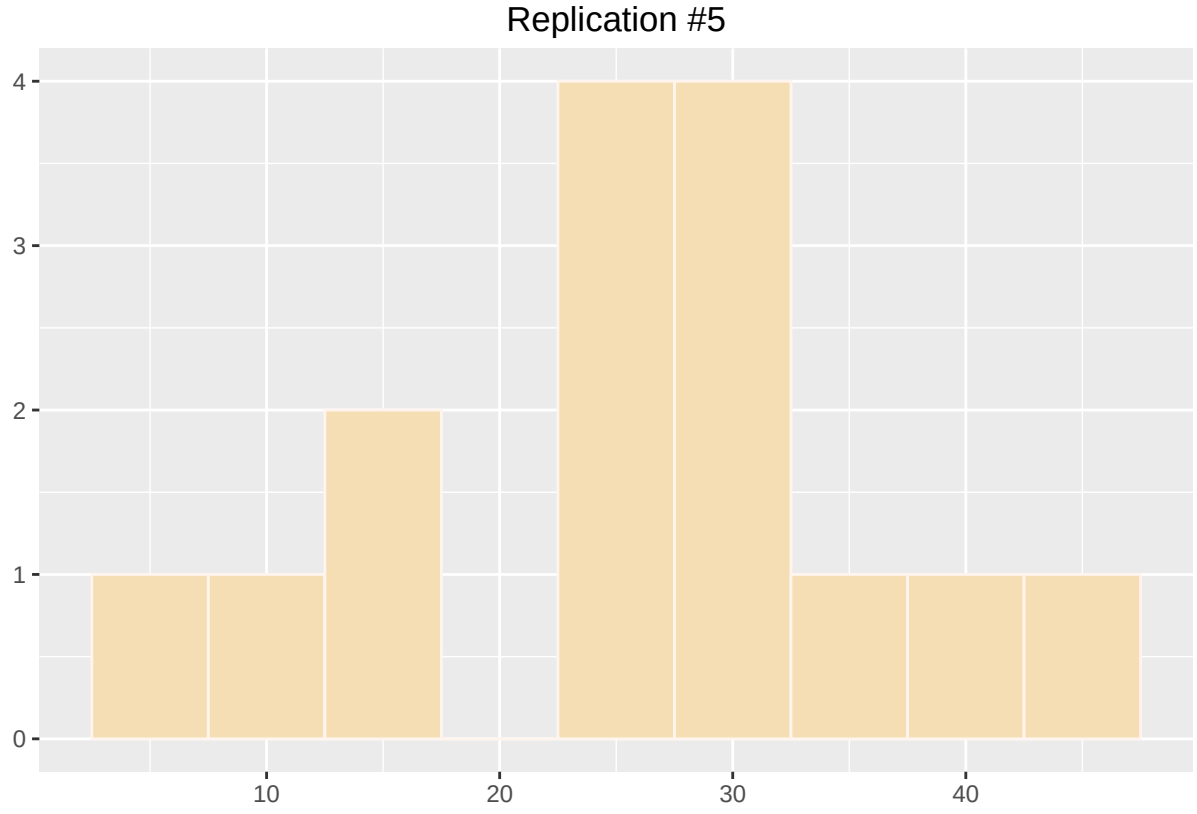
```
## Run the function
for (s in 1:20){
  Hist.preset (y.rep[s,], width=5, "","",paste("Replication #",s,sep=""))
}
```

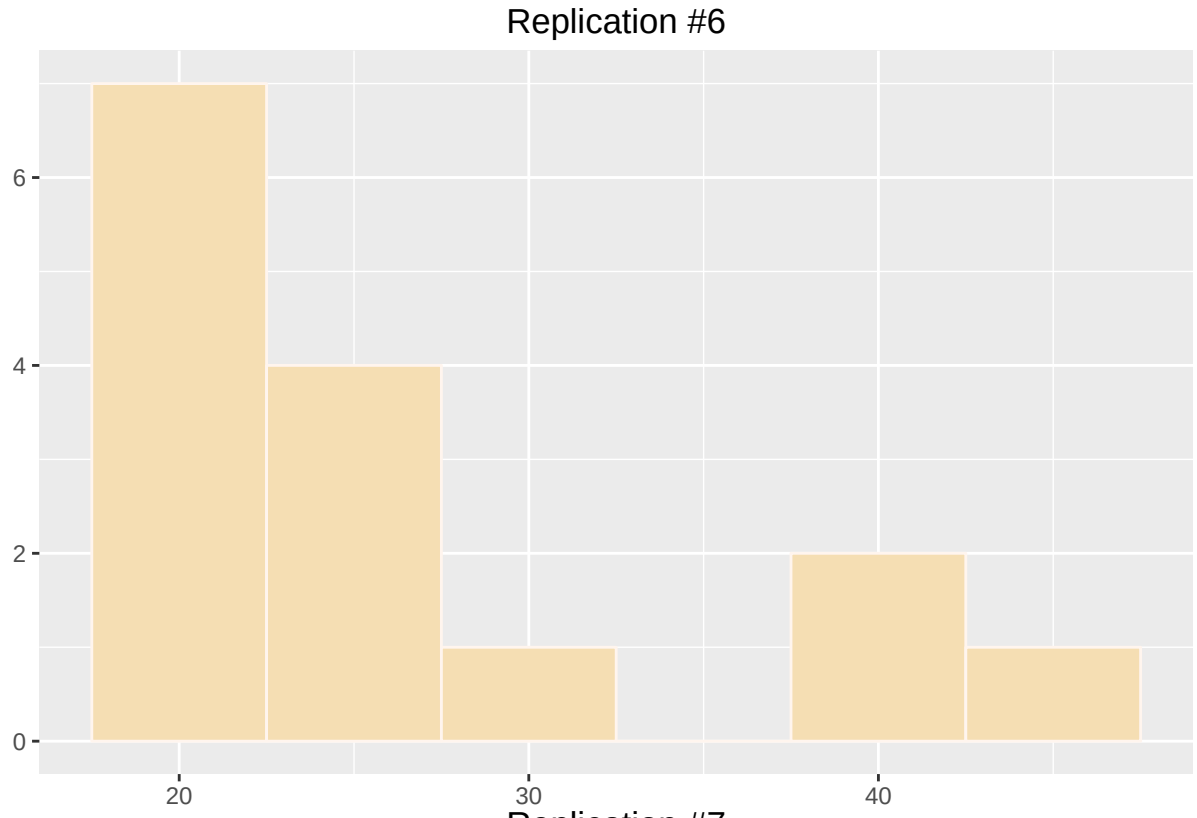


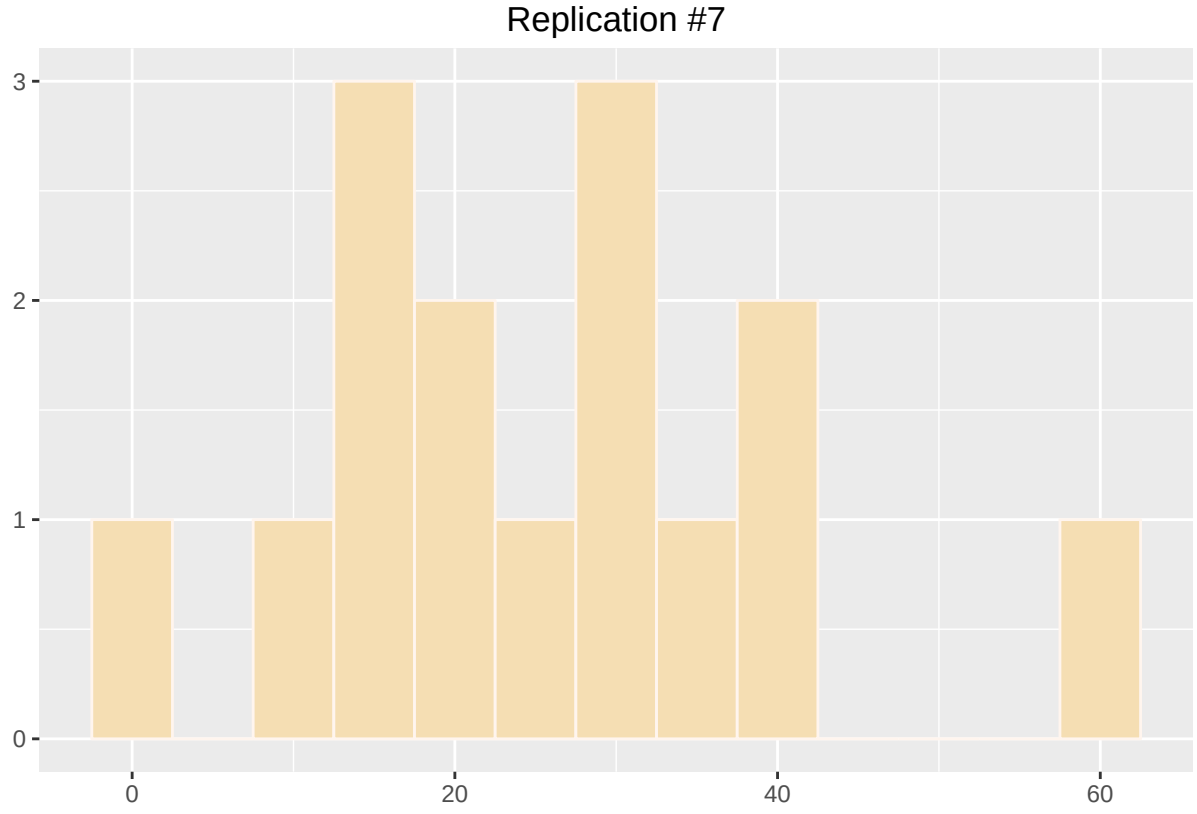

Replication #2
5
4
3
2
1
0
10
20
30
40

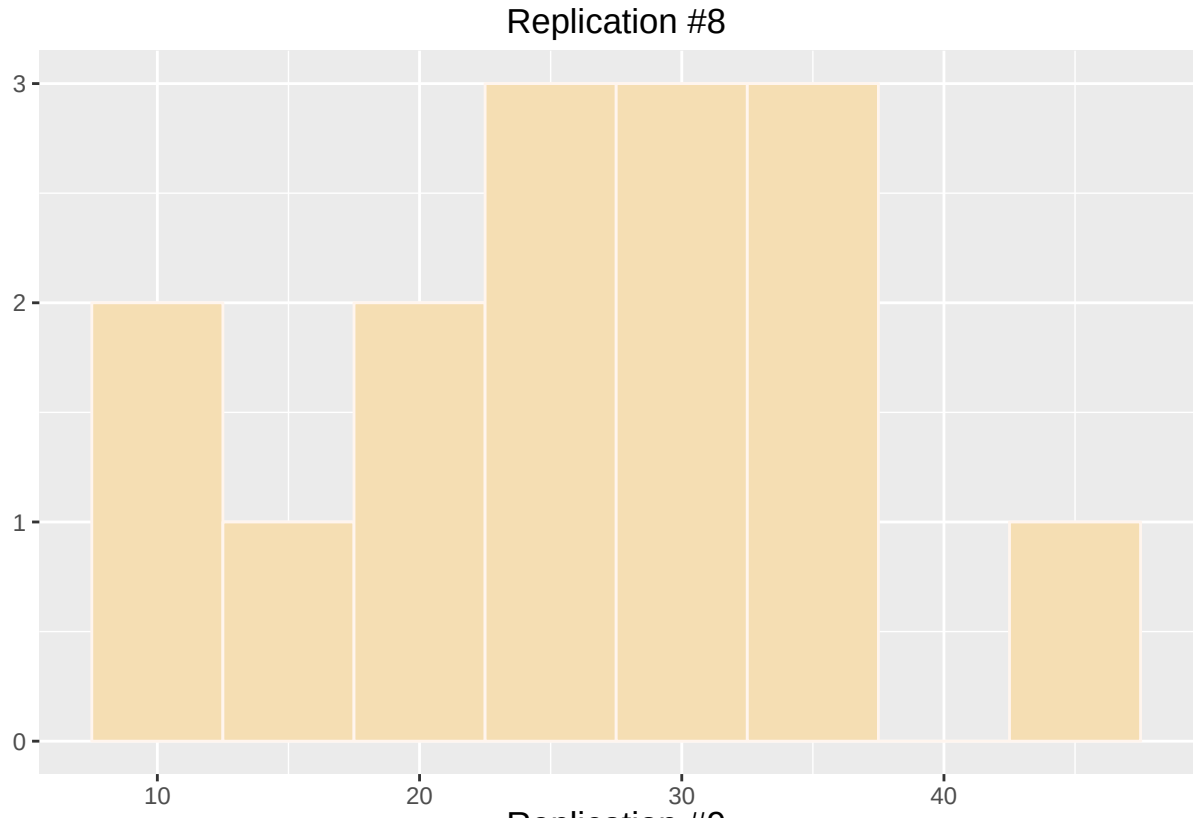

Replication #3
3
2
1
0
20
30
40

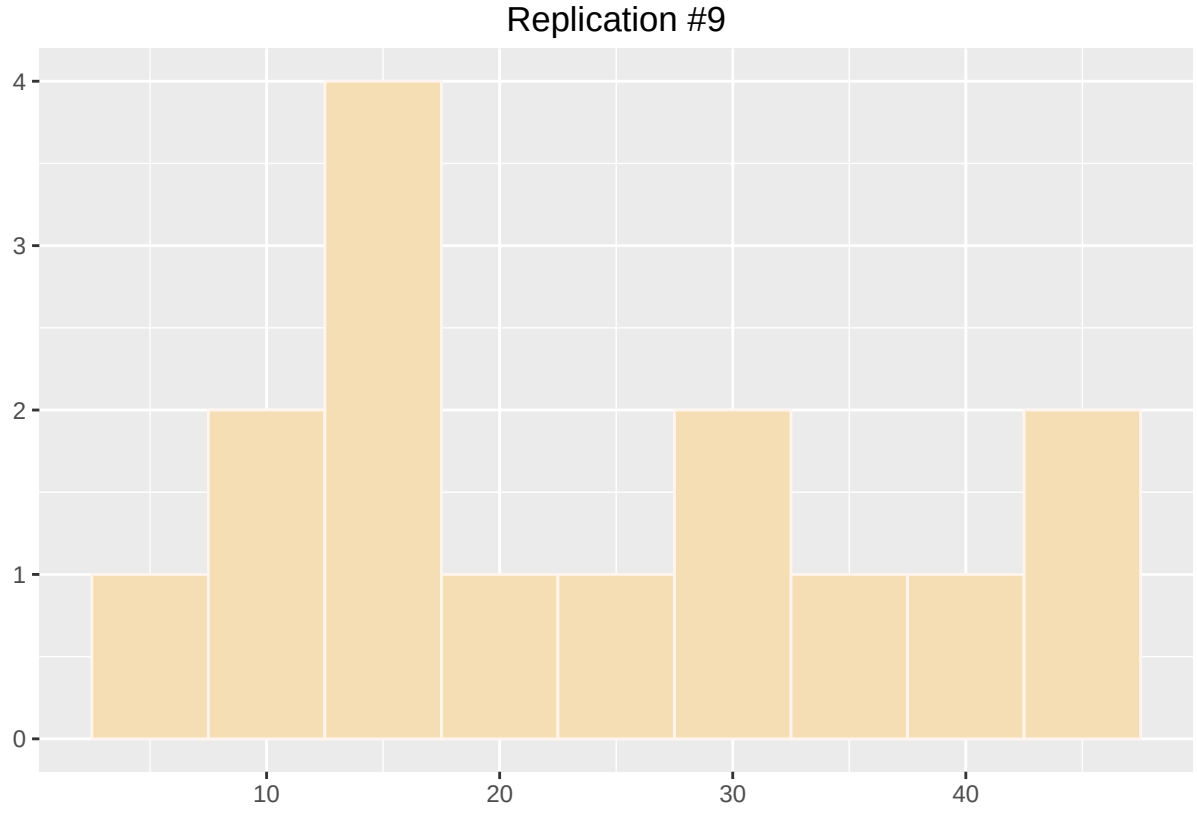

Replication #4
5
4
3
2
1
0
20
30
40

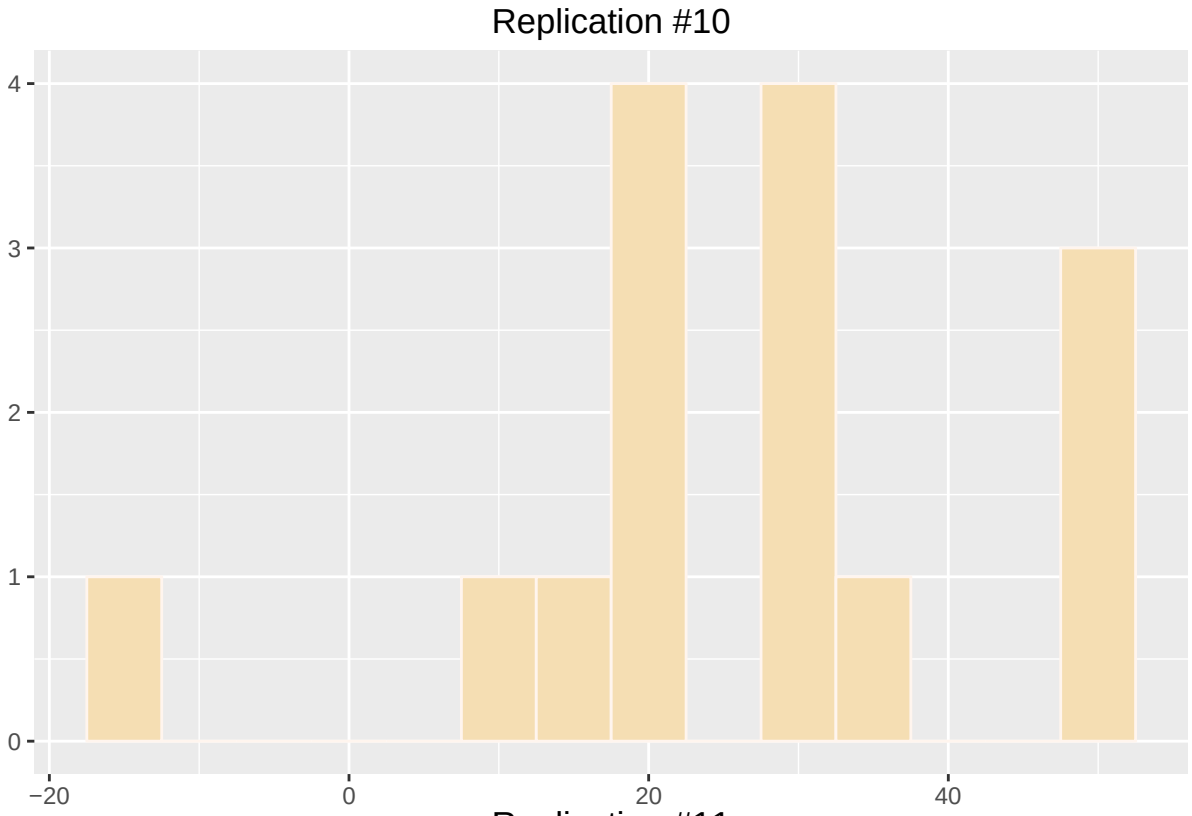

Replication #5
4
3
2
1
0
10
20
30
40

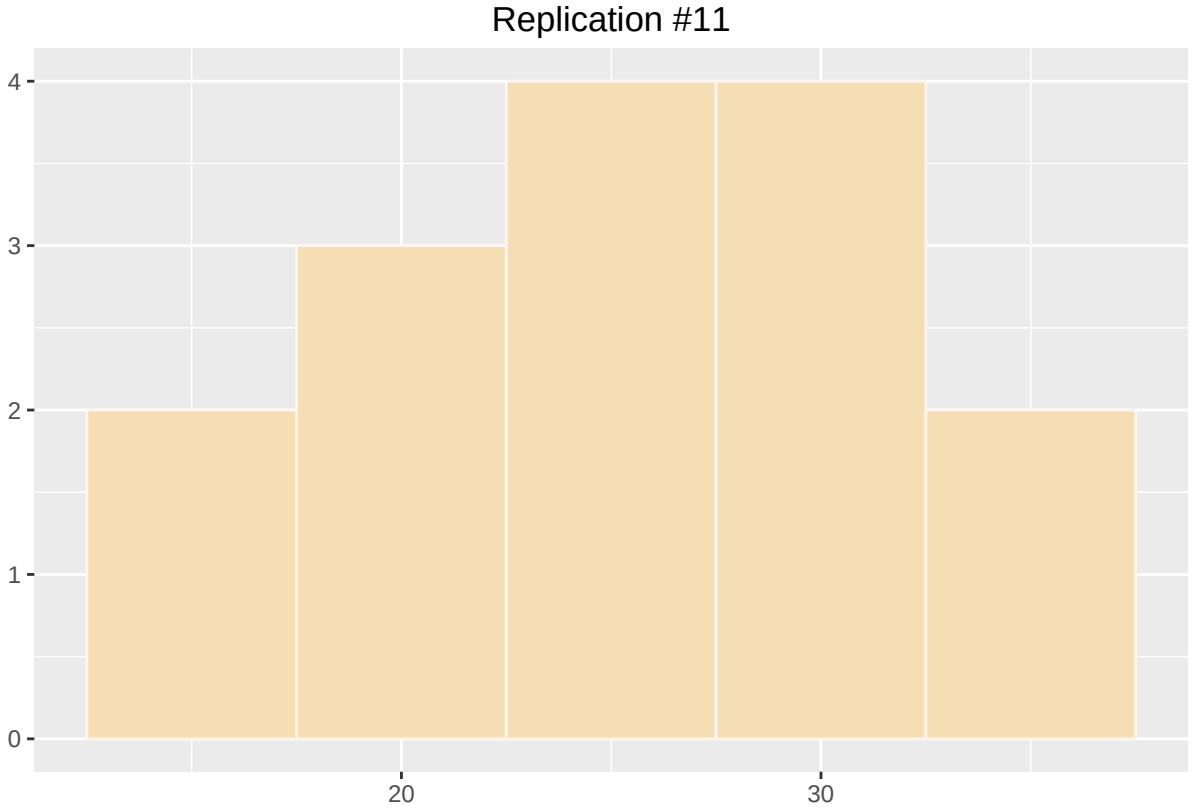

Replication #6
6
4
2
0
20
30
40

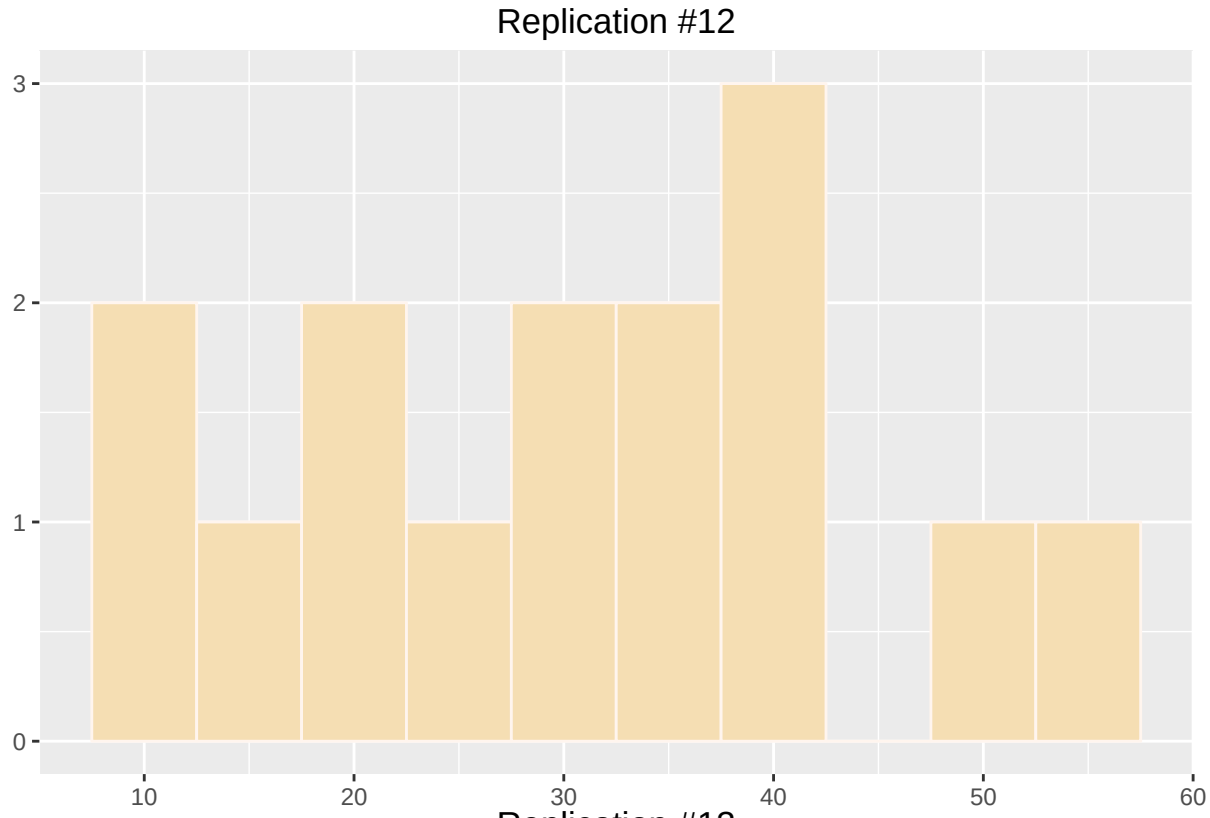

Replication #7
3
2
1
0
0
20
40
60

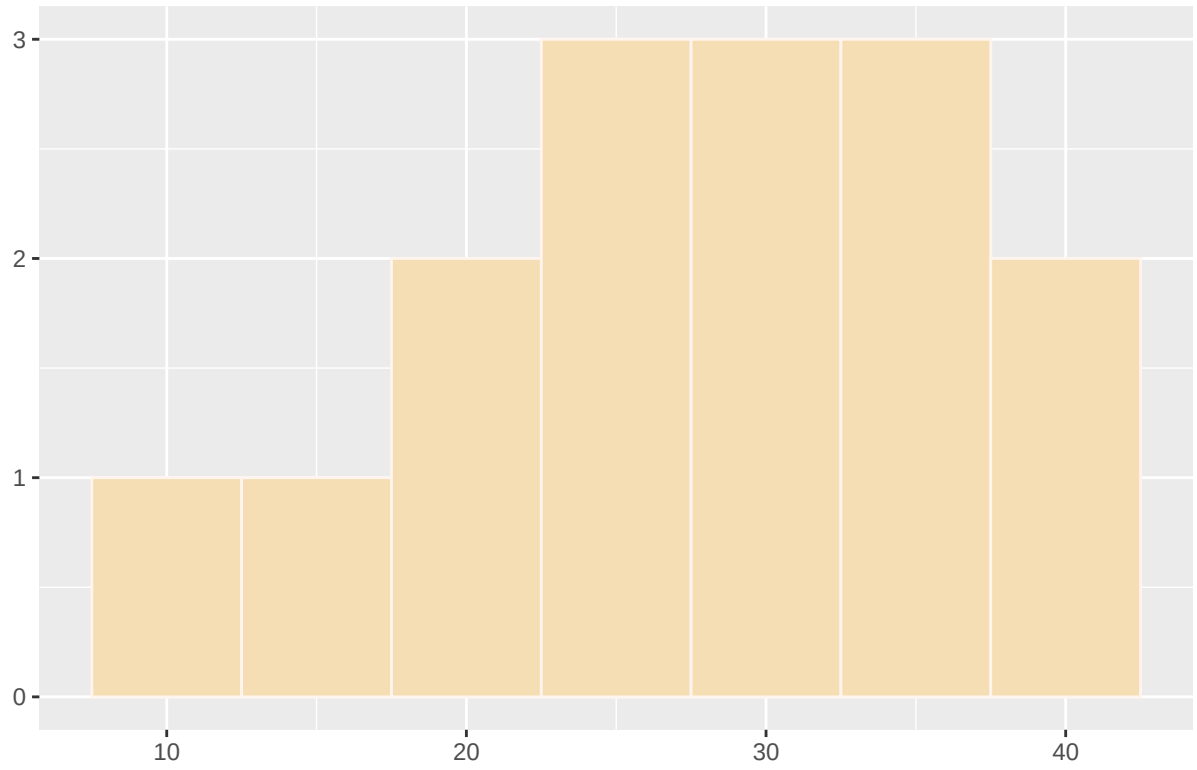

Replication #8
3
2
1
0
10
20
30
40

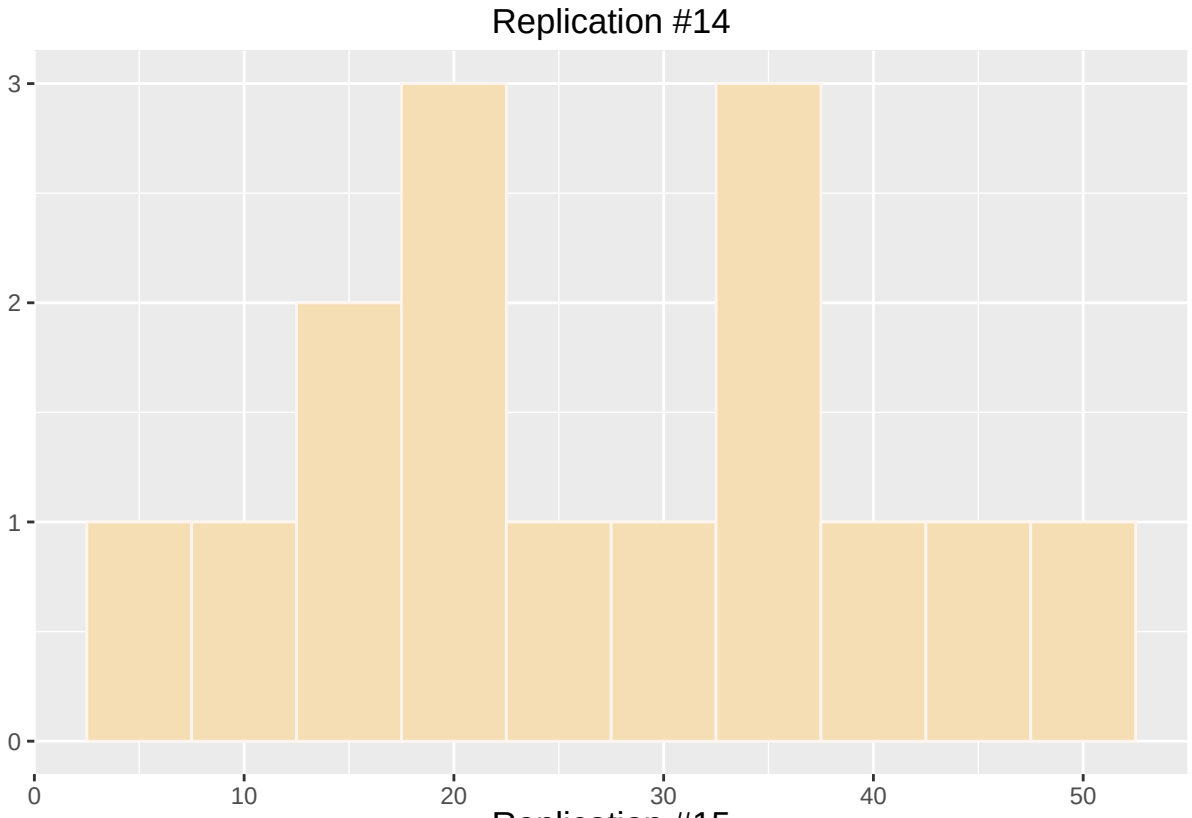

Replication #9
4
3
2
1
0
10
20
30
40

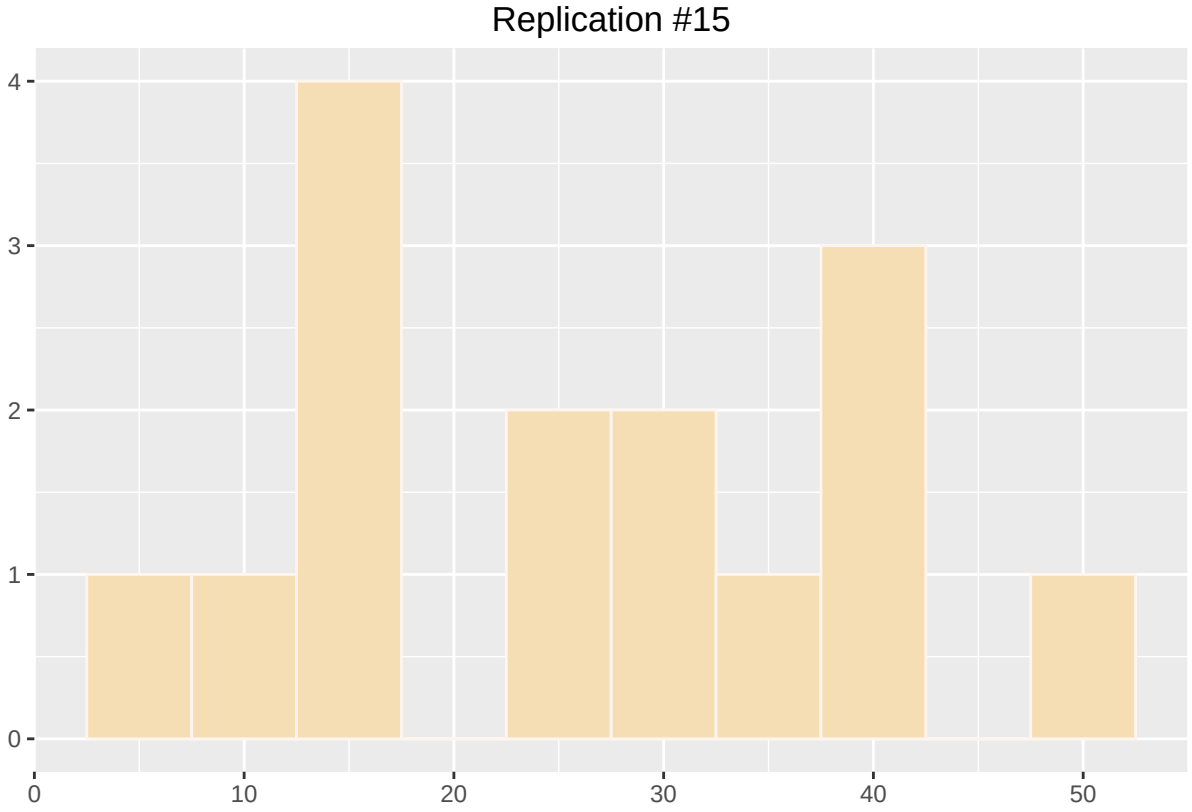

Replication #10
4
3
2
1
0
−20
0
20
40

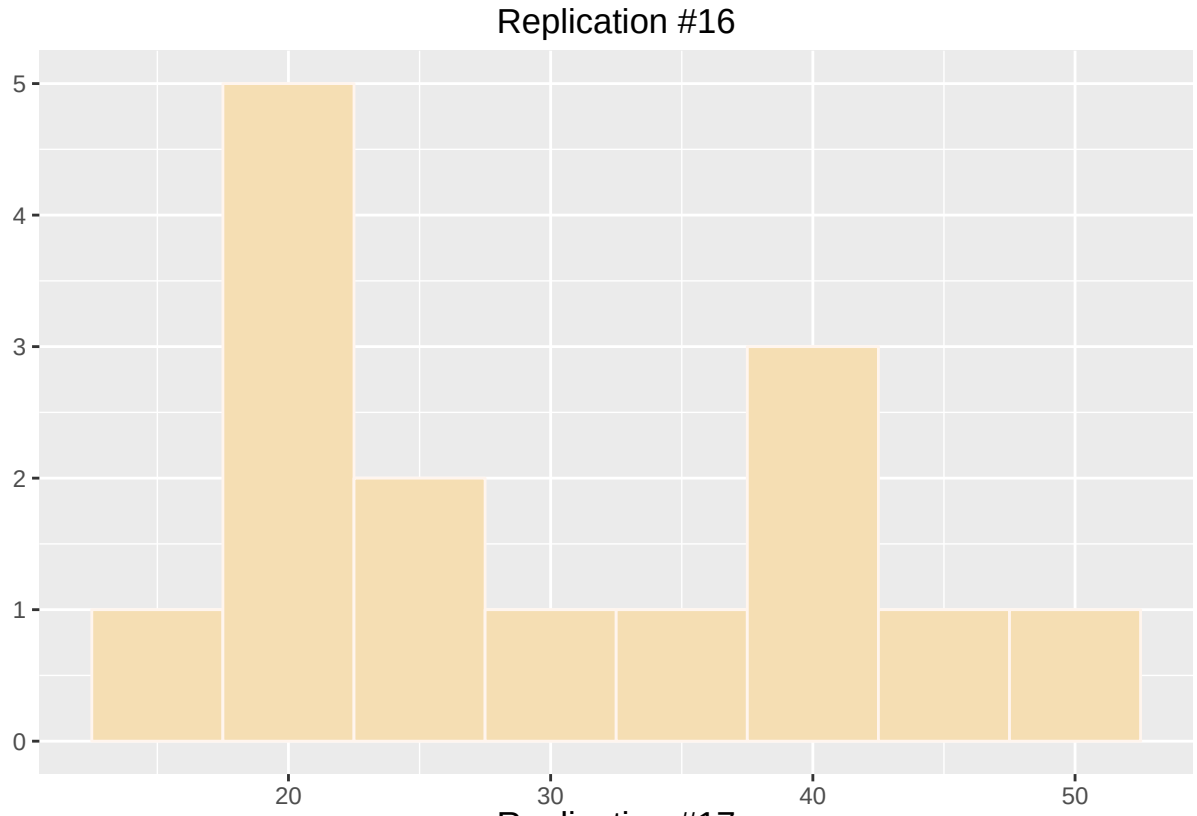

Replication #11
4
3
2
1
0
20
30

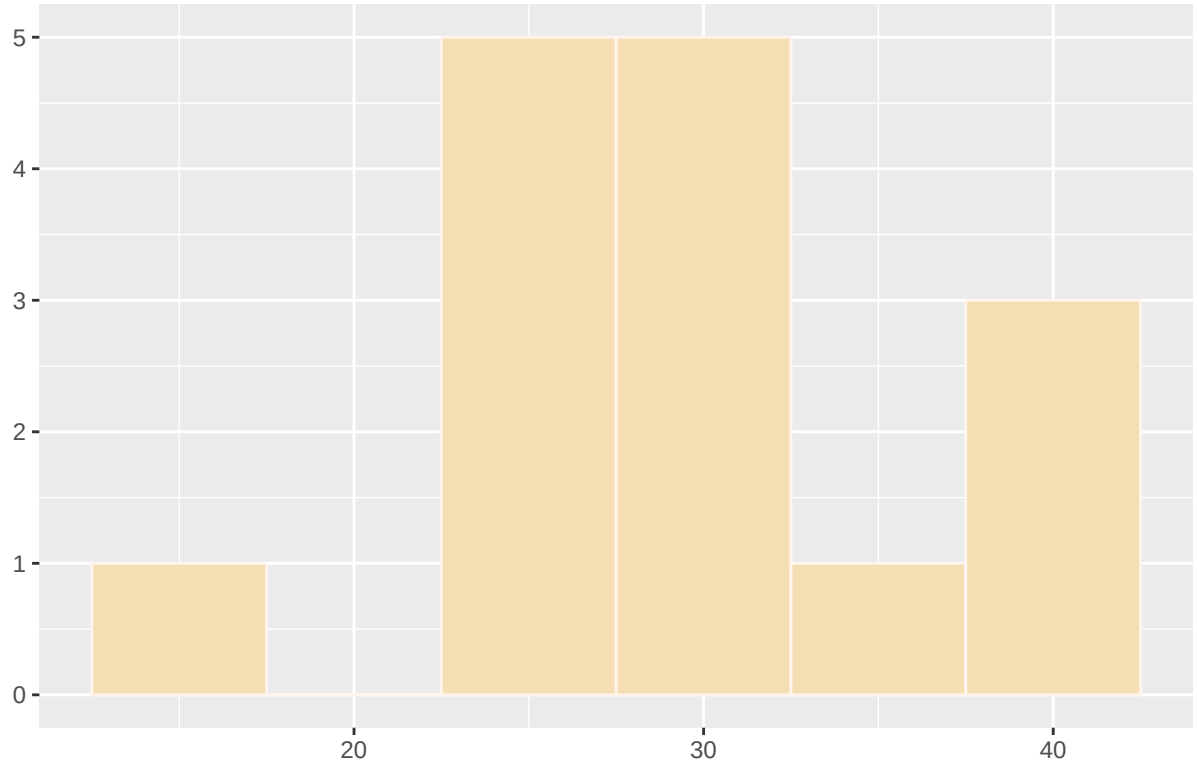

Replication #12
3
2
1
0
10
20
30
40
50
60

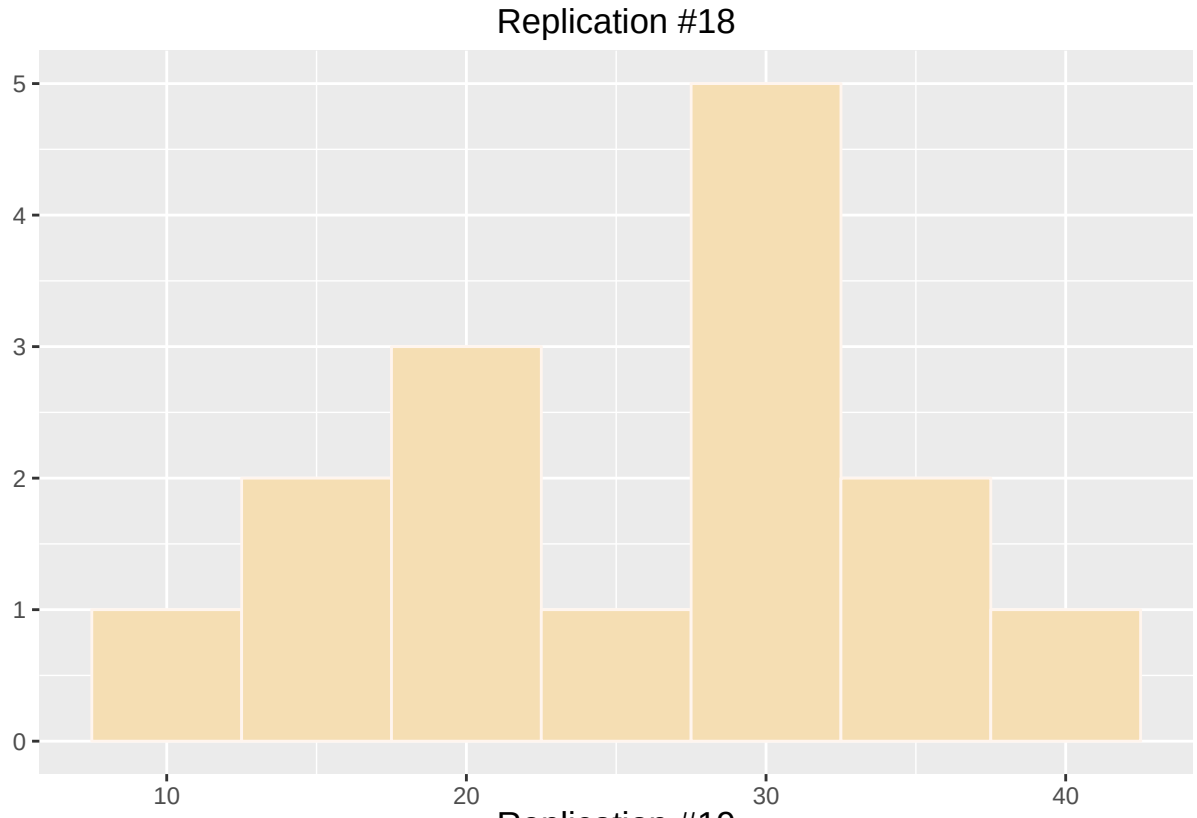

Replication #13
3
2
1
0
10
20
30
40

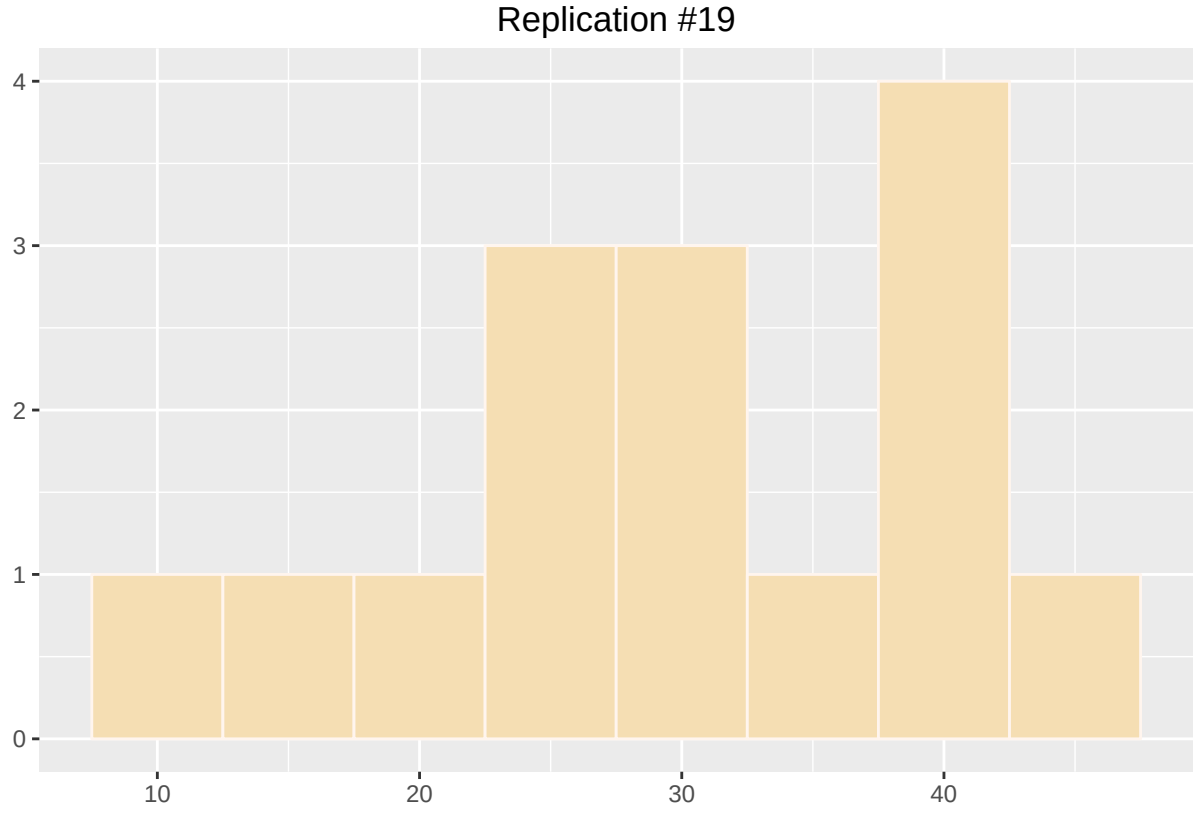

Replication #14
3
2
1
0
0
10
20
30
40
50

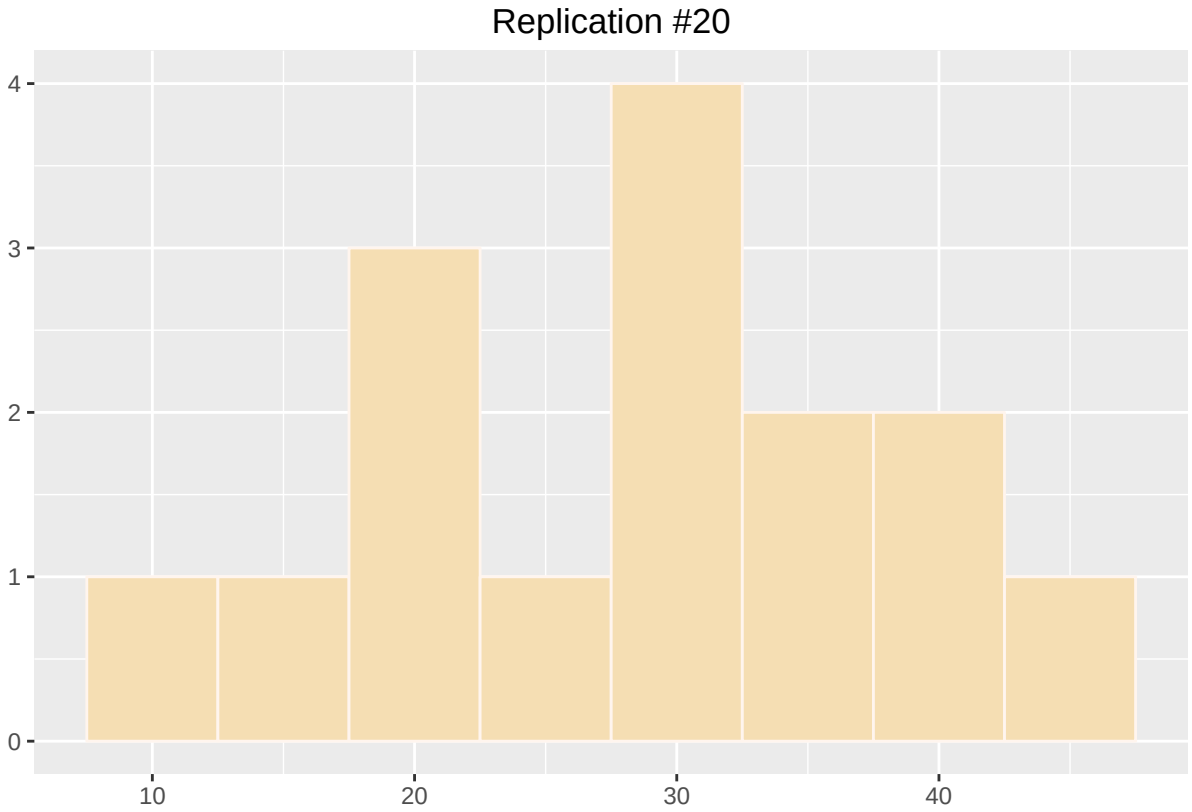

Replication #15
4
3
2
1
0
0
10
20
30
40
50


Replication #16
5
4
3
2
1
0
20
30
40
50


Replication #17
5
4
3
2
1
0
20
30
40


Replication #18
5
4
3
2
1
0
10
20
30
40


Replication #19
4
3
2
1
0
10
20
30
40

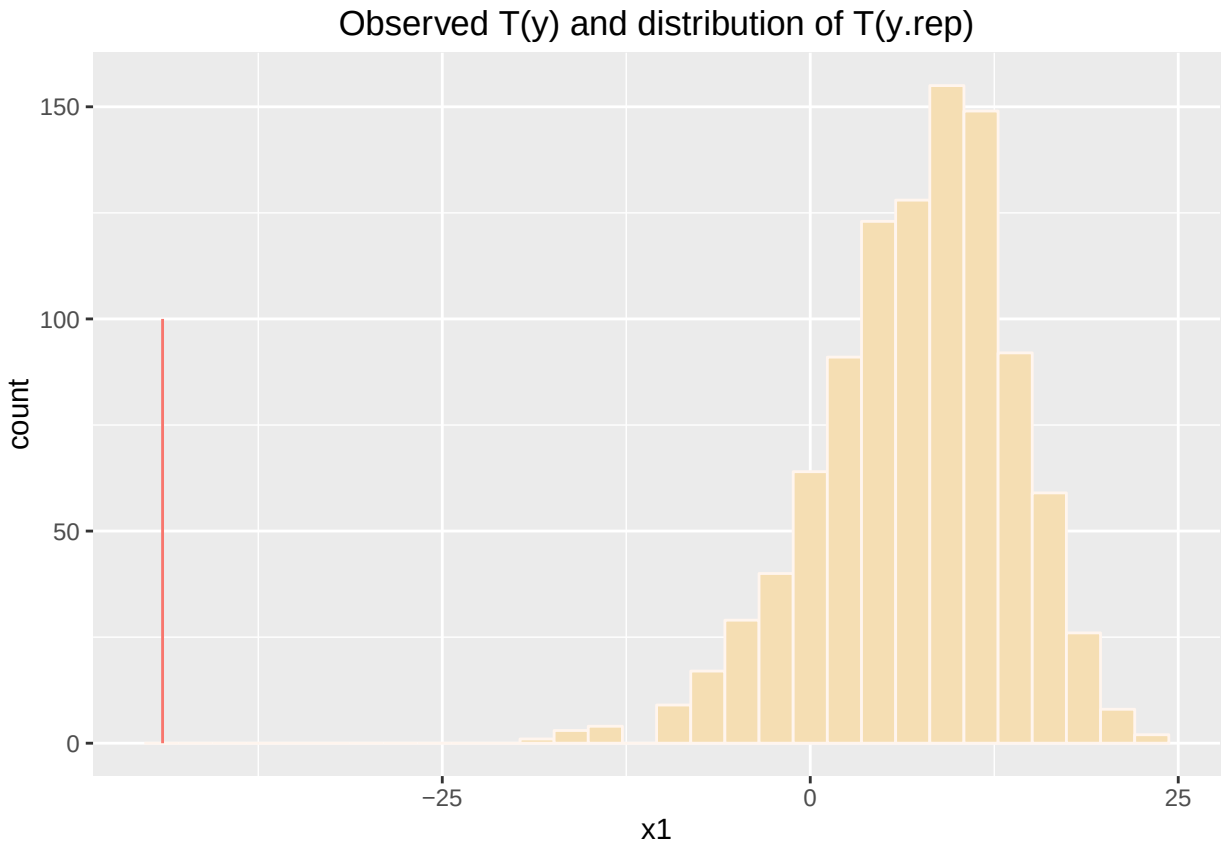



```
## Numerical test
Test <- function (y){
  min (y)
}
test.rep <- rep (NA, n.sims)
for (s in 1:n.sims){
  test.rep[s] <- Test (y.rep[s,])
}
str(test.rep)
```

```
##  num [1:1000] 8.55 9.44 13.83 12.63 4.73 ...
```

figure 8.5

```
## Histogram Figure 8.5
# dev.new()
  frame2 = data.frame(x1 = test.rep)
  frame3 <- data.frame(x2 = Test(y))
  p3 <- ggplot(frame2, aes(x = x1)) +
        geom_histogram(colour = "seashell", fill = "wheat") +
        geom_segment(aes(x = x2, y = 0, xend = x2, yend = 100,
                         color = "saddlebrown"), data = frame3) +
        theme_gray() +
        theme(legend.position="none") +
        labs(title="Observed T(y) and distribution of T(y.rep)")
  print(p3)
```



roaches

data

```
################################################################################
## Read the cleaned data
# All data are at http://www.stat.columbia.edu/~gelman/arm/examples/roaches
# if bad initial values, this model fails
# NOTE: can't find same exact data set as ARM book uses..
roachdata <- read.csv ("roachdata.csv")
str(roachdata)
```

```
## 'data.frame':    262 obs. of  6 variables:
##  $ X        : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ y        : int  153 127 7 7 0 0 73 24 2 2 ...
##  $ roach1   : num  308 331.25 1.67 3 2 ...
##  $ treatment: int  1 1 1 1 1 1 1 1 0 0 ...
##  $ senior   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ exposure2: num  0.8 0.6 1 1 1.14 ...
```

```
attach(roachdata)
```

```
## The following object is masked _by_ .GlobalEnv:
##
##     y
```

model

roaches.stan

```
data {
  int<lower=0> N;
  vector[N] exposure2;
  vector[N] roach1;
  vector[N] senior;
  vector[N] treatment;
  int y[N];
}
transformed data {
  vector[N] log_expo;
  log_expo = log(exposure2);
}
parameters {
  vector[4] beta;
}
model {
  y ~ poisson_log(log_expo +
                  beta[1] +
                  beta[2] * roach1 +
                  beta[3] * treatment +
                  beta[4] * senior);
}
```

fit

```
dataList.1 <- list(N=length(roachdata$y), y=roachdata$y,roach1=roachdata$roach1,
                   treatment=roachdata$treatment,exposure2=roachdata$exposure2,
                   senior=roachdata$senior)
roaches.sf1 <- stan(file='roaches.stan', data=dataList.1, iter=5000, chains=4)
print(roaches.sf1)
```

```
## Inference for Stan model: roaches.
## 4 chains, each with iter=5000; warmup=2500; thin=1;
## post-warmup draws per chain=2500, total post-warmup draws=10000.
##
##             mean se_mean     sd     2.5%      25%      50%      75%
## beta[1]     3.06    0.03   0.17     2.93     3.07     3.09     3.10
## beta[2]     0.01    0.00   0.00     0.01     0.01     0.01     0.01
## beta[3]    -0.49    0.03   0.19    -0.56    -0.53    -0.52    -0.50
## beta[4]    -0.36    0.02   0.12    -0.45    -0.40    -0.38    -0.35
## lp__    17564.33   49.35 323.36 17555.19 17603.04 17604.20 17604.95
##            97.5% n_eff Rhat
## beta[1]     3.13    31 1.09
## beta[2]     0.01    42 1.07
## beta[3]    -0.30    31 1.10
## beta[4]    -0.26    28 1.12
## lp__    17605.70    43 1.06
##
## Samples were drawn using NUTS(diag_e) at Fri Jul  8 13:50:36 2016.
```

```
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
##  The estimated Bayesian Fraction of Missing Information is a measure of
##  the efficiency of the sampler with values close to 1 being ideal.
##  For each chain, these estimates are
##  0 1 1 0.9
```

post

```
post <- extract(roaches.sf1)
```

```
## Comparing the data to a replicated dataset
n <- length(roachdata$y)
X <- cbind (rep(1,n), roach1, treatment, senior)
y.hat <- exposure2 * exp (X %*% colMeans(post$beta))
y.rep <- rpois (n, y.hat)

print (mean (roachdata$y==0))
```

```
## [1] 0.3587786
```

```
print (mean (y.rep==0))
```

```
## [1] 0
```

```
## Comparing the data to 1000 replicated datasets
n.sims <- 1000
y.rep <- array (NA, c(n.sims, n))
for (s in 1:n.sims){
  y.hat <- exposure2 * exp (X %*% post$beta[s,])
  y.rep[s,] <- rpois (n, y.hat)
}
```

```
 # test statistic
Test <- function (y){
  mean (y==0)
}
test.rep <- rep (NA, n.sims)
for (s in 1:n.sims){
  test.rep[s] <- Test (y.rep[s,])
}
```

```
# p-value
print (mean (test.rep > Test(roachdata$y)))
```

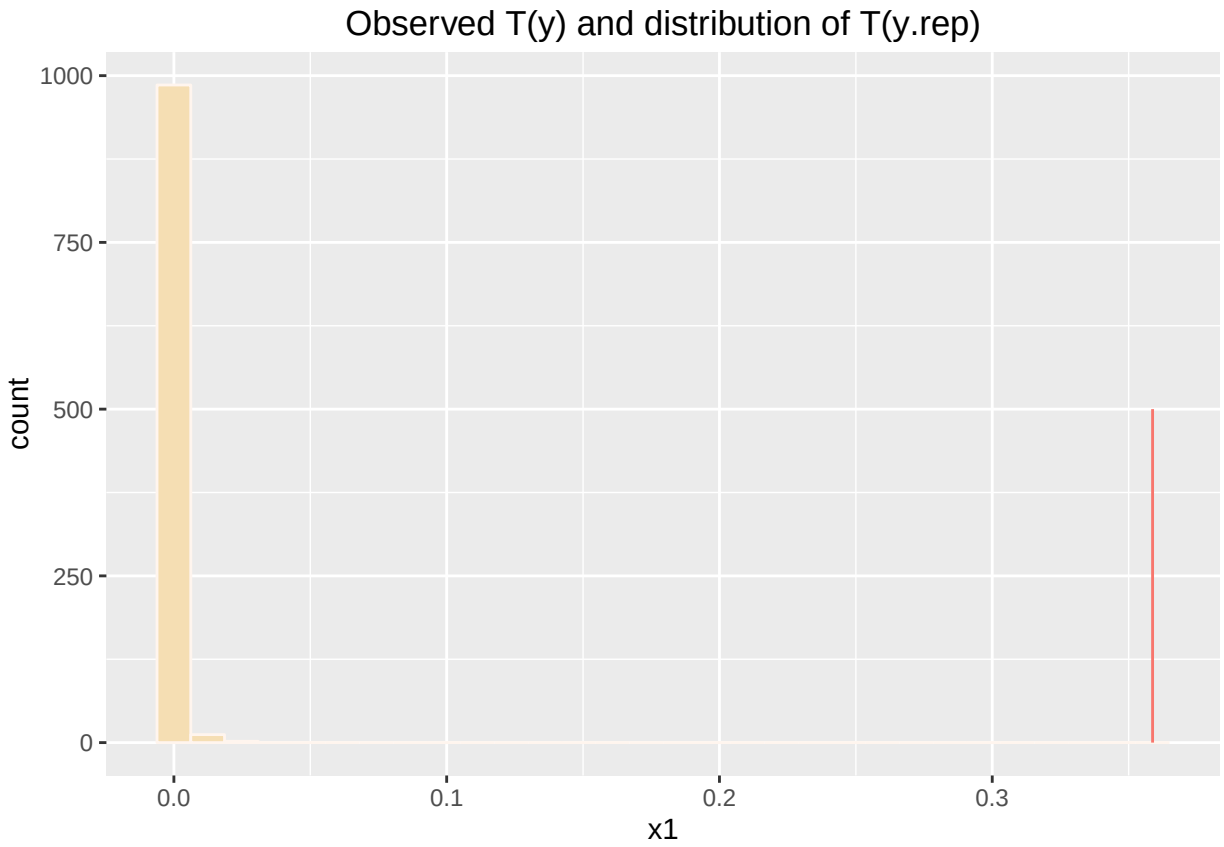
```
## [1] 0
```

figure

```
## Histogram Figure
# dev.new()
  frame4 = data.frame(x1 = test.rep)
  frame5 = data.frame(x2 = Test(roachdata$y))
```

```
  p4 <- ggplot(frame4, aes(x=x1)) +
        geom_histogram(colour = "seashell", fill = "wheat") +
        geom_segment(aes(x = x2, y = 0, xend = x2, yend = 500,
                     color = "saddlebrown"), data = frame5) +
        theme_gray() +
        theme(legend.position="none") +
       labs(title="Observed T(y) and distribution of T(y.rep)")
  print(p4)
```

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



T(y) = 0.36, but all the values of test.rep are much smaller.

```
summary(test.rep)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000000 0.0000000 0.0000000 0.0007939 0.0000000 0.0229000
```

model

roaches_overdispersion.stan

```
data {
  int<lower=0> N;
  vector[N] exposure2;
  vector[N] roach1;
  vector[N] senior;
  vector[N] treatment;
```

```
  int y[N];
}
transformed data {
  vector[N] log_expo;
  log_expo = log(exposure2);
}
parameters {
  vector[4] beta;
  vector[N] lambda;
  real<lower=0> tau;
}
transformed parameters {
  real<lower=0> sigma;
  sigma = 1.0 / sqrt(tau);
}
model {
  tau ~ gamma(0.001, 0.001);
  for (i in 1:N) {
    lambda[i] ~ normal(0, sigma);
    y[i] ~ poisson_log(lambda[i] +
                       log_expo[i] +
                       beta[1] +
                       beta[2]*roach1[i] +
                       beta[3]*senior[i] +
                       beta[4]*treatment[i]);
  }
}
```

fit

```
## Checking the overdispersed model
# NOTE: can't find same exact data set as ARM book uses..
roaches_overdispersion.sf1 <- stan(file='roaches_overdispersion.stan',
                                   data=dataList.1,
                                   iter=1000, chains=4)
# print(roaches_overdispersion.sf1)
```

post

```
post <- extract(roaches_overdispersion.sf1)
```

switch to glm

```
glm.2 <- glm(y ~ roach1 + treatment + senior, data = roachdata,
             family=quasipoisson, offset=log(exposure2))
sim.2 <- sim(glm.2, n.sims)
```

```
 # 1000 replicated datasets
y.rep <- array (NA, c(n.sims, n))
overdisp <- summary(glm.2)$dispersion
```

```
for (s in 1:n.sims){
  y.hat <- exposure2 * exp (X %*% sim.2@coef[s,])
  a <- y.hat/(overdisp-1)              # using R's parametrization for the
  y.rep[s,] <- rnegbin (n, y.hat, a)   # negative binomial distribution
}

test.rep <- rep (NA, n.sims)
for (s in 1:n.sims){
  test.rep[s] <- Test (y.rep[s,])
}
```

compare each value of test.rep with the number Test(roachdata$y)

```
# p-value
summary(test.rep)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1565  0.2824  0.3168  0.3165  0.3473  0.4695
```

```
print (mean (test.rep > Test(roachdata$y)))
```

```
## [1] 0.168
```

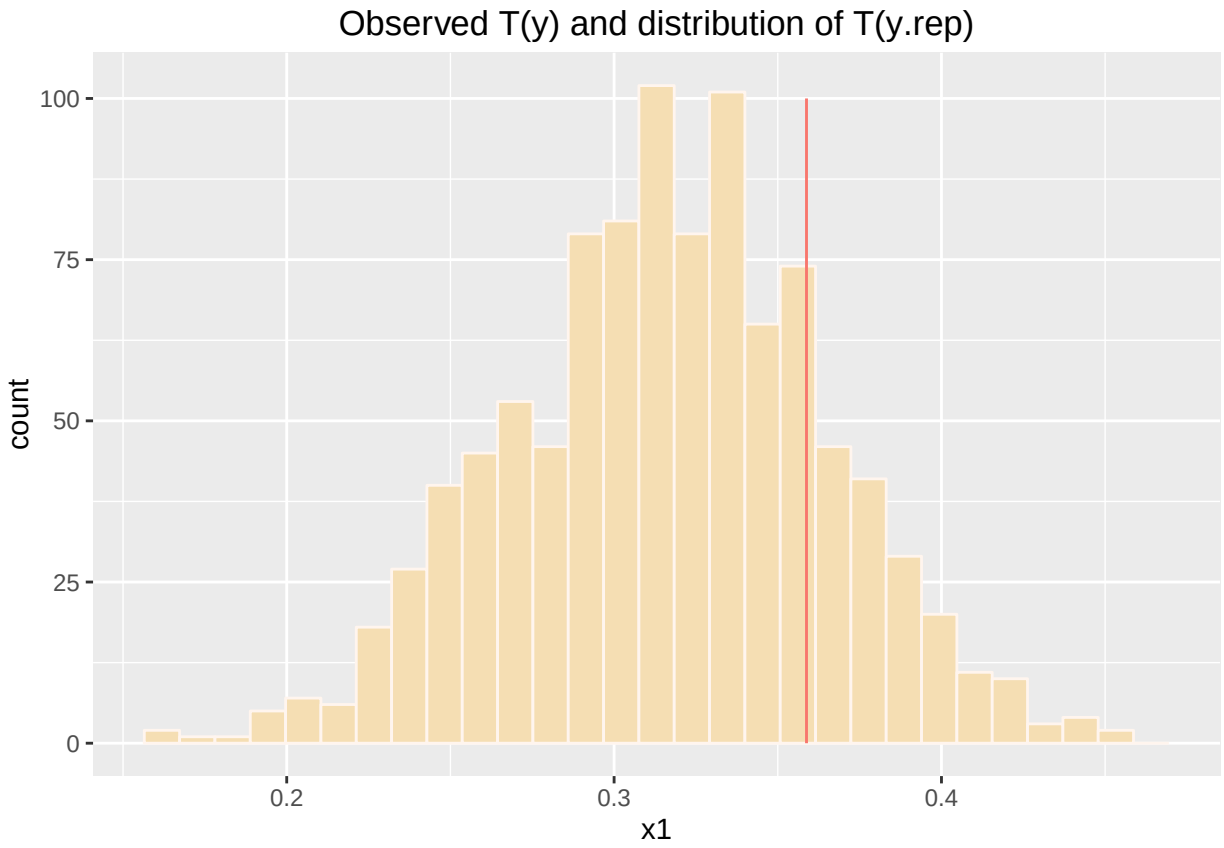
```
Test(roachdata$y)
```

```
## [1] 0.3587786
```

figure

```
## Histogram Figure
# dev.new()
  frame4 = data.frame(x1 = test.rep)
  frame5 = data.frame(x2 = Test(roachdata$y))
  p5 <- ggplot(frame4, aes(x=x1)) +
        geom_histogram(colour = "seashell", fill = "wheat") +
        geom_segment(aes(x = x2, y = 0, xend = x2, yend = 100,
                     color = "saddlebrown"), data = frame5) +
        theme_gray() +
        theme(legend.position="none") +
       labs(title="Observed T(y) and distribution of T(y.rep)")
  print(p5)
```

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


Observed T(y) and distribution of T(y.rep)
count
100
75
50
25
0
0.2
0.3
0.4
x1