XDAS Week 7 Recitation

Chris Jackson

10/17/2019

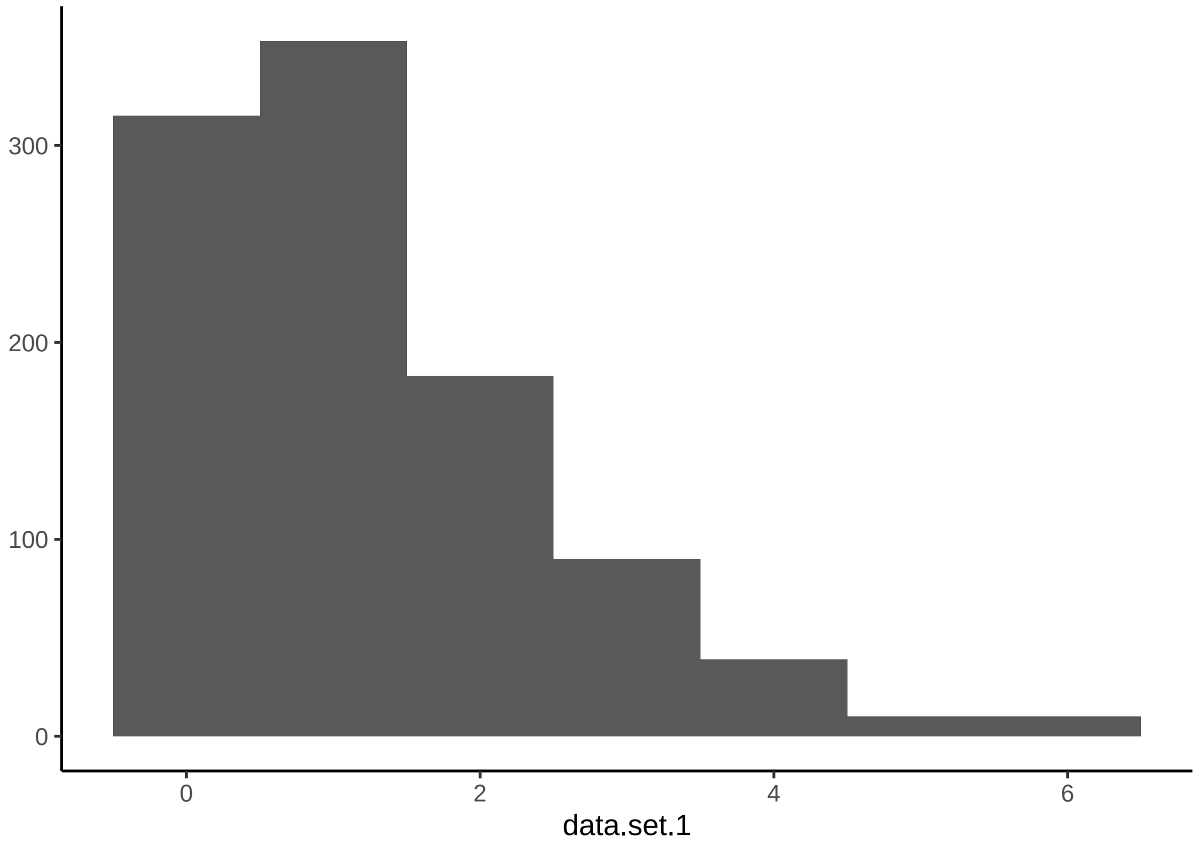
Hypothesis Testing with random sampling

Let's go ahead and simulate two data sets using `rnbinom`.

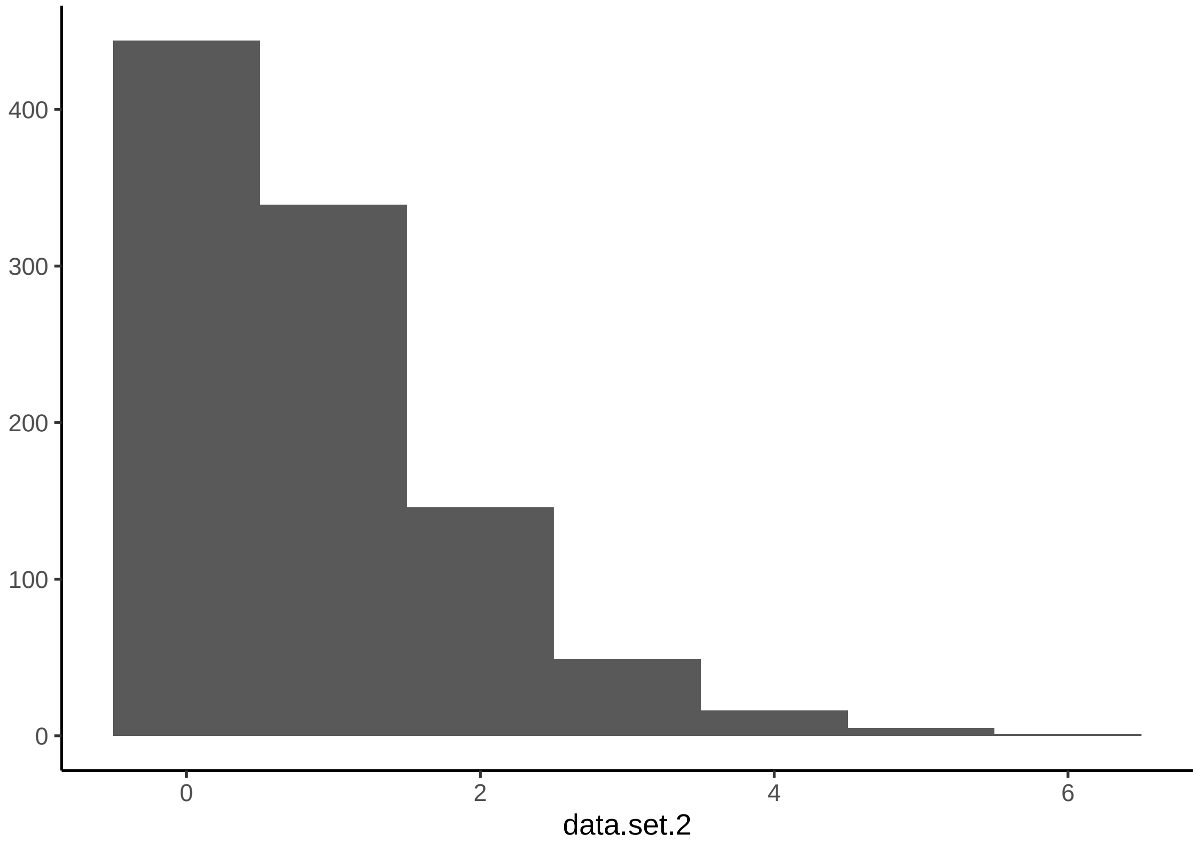
```
data.set.1 <- rnbinom(1000, prob = 0.8, size = 5)
data.set.2 <- rnbinom(1000, prob = 0.85, size = 5)
```

What does a negative binomial distribution look like again?

```
qplot(data.set.1, geom="histogram", binwidth=1)
```



```
qplot(data.set.2, geom="histogram", binwidth=1)
```



Are the means of these data sets different by t-test?

```
t.test(data.set.1, data.set.2)
```

```
##
##  Welch Two Sample t-test
##
## data:  data.set.1 and data.set.2
## t = 7.5273, df = 1915.2, p-value = 7.927e-14
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.2824717 0.4815283
## sample estimates:
## mean of x mean of y
##     1.255     0.873
```

Awesome - the thing we defined as being different is in fact, different. But wait - what does this actually mean?

Even though it's technically *significant*, is this test interpretable?

Well that sucks. What can we do that IS interpretable?

We know that this is negative binomial data, what about fitting the distribution to the data?

```
fitter.function <- function(data.set, dist.name = "negative binomial", add.prob=T) {
  fit.params <- MASS::fitdistr(data.set, dist.name)
  if (add.prob) {fit.params$estimate['p'] <- calc.prob(fit.params)}
  return(fit.params)
}
```

```
nbinom.fit <- list()
nbinom.fit$data.1 <- fitter.function(data.set.1)
nbinom.fit$data.2 <- fitter.function(data.set.2)
print(nbinom.fit$data.1)
```

```
## Warning in rbind(x$estimate, x$sd): number of columns of result is not a
## multiple of vector length (arg 2)
```

```
##       size          mu           p
##   5.40997655   1.25499975   0.81170229
##  (1.35443887) (0.03932086) (1.35443887)
```

```
print(nbinom.fit$data.2)
```

```
## Warning in rbind(x$estimate, x$sd): number of columns of result is not a
## multiple of vector length (arg 2)
```

```
##       size          mu           p
##   5.27073536   0.87300046   0.85790397
##  (1.77606542) (0.03189976) (1.77606542)
```

Well that's certainly different values for p, but how do we test it?

We've (I've) managed to screw up propagating the estimate of standard error, so that doesn't help very much - we need to actually do something ourselves.

Lets try something a little different this time - the T-test requires certain conditions to be met. What conditions do we need to meet for a nonparametric test (lets try Mann-Whitney-Wilcoxon!)

```
test.vals <- wilcox.test(data.set.1, data.set.2)
test.vals
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  data.set.1 and data.set.2
## W = 587372, p-value = 9.635e-13
## alternative hypothesis: true location shift is not equal to 0
```

That's pretty good - how well does this hold up with small n though?

```
data.set.1 <- rnbinom(10, prob = 0.8, size = 5)
data.set.2 <- rnbinom(10, prob = 0.85, size = 5)
```

```
test.vals <- wilcox.test(data.set.1, data.set.2)
```

```
## Warning in wilcox.test.default(data.set.1, data.set.2): cannot compute
## exact p-value with ties
```

```
test.vals
```

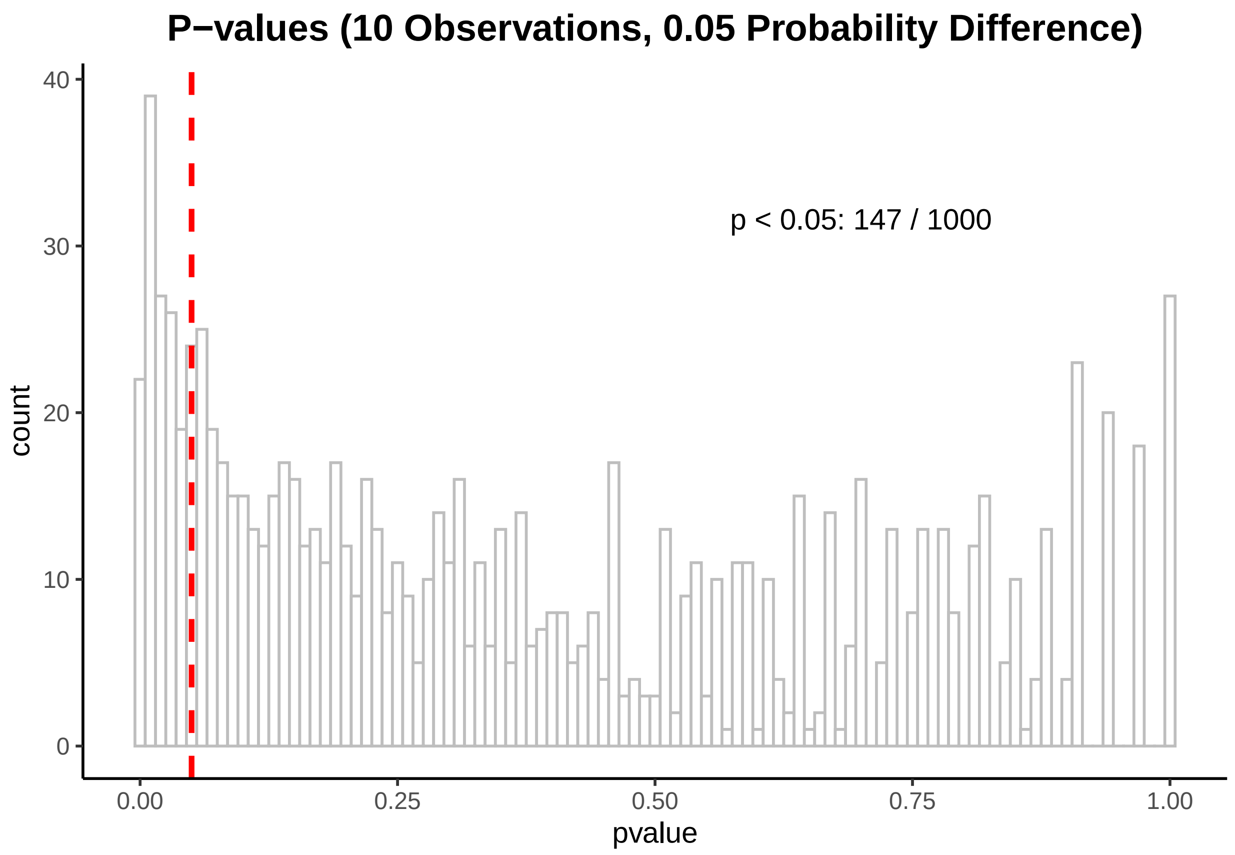
```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  data.set.1 and data.set.2
## W = 65, p-value = 0.2458
## alternative hypothesis: true location shift is not equal to 0
```

Well that's not nearly as awesome. Even though we KNOW that the means of these two populations is different (we defined it that way!), the test has failed to demonstrate a statistically significant difference (in fact, it's been pretty awful). How often does that occur?

```
# Make a function that tests normal data num.t times
# The number of observations per sampling is num.O
# The means of these normal distributions are different
# The standard deviation is the same though

run.wilcox.function <- function(num.t, num.o, prob.1=0.8, prob.2=0.85, size=10) {
  print(paste0("Probability 1: ", prob.1, "; Probability 2: ", prob.2))

  test.vals <- NULL
  for (i in 1:num.t) {
    loop.set.1 <- rnbinom(num.o, size=size, prob = prob.1)
    loop.set.2 <- rnbinom(num.o, size=size, prob = prob.2)
    loop.vals <- suppressWarnings(wilcox.test(loop.set.1, loop.set.2))
    test.vals <- rbind(test.vals,
                       data.frame(pvalue=loop.vals$p.value,
                                  diff=0.5, n=10))
  }
  return(test.vals)
}

n.tests <- 1000
n.observations <- 10
prob.diff <- 0.05

n10.diff0.5 <- run.wilcox.function(n.tests, n.observations, prob.1 = 0.8, prob.2 = 0.85)
```

```
## [1] "Probability 1: 0.8; Probability 2: 0.85"
```

```
plot.title <- paste0("P-values (", toString(n.observations),
                     " Observations, ", toString(prob.diff),
                     " Probability Difference)")
sig.by.test <- sum(n10.diff0.5$pvalue < 0.05)

ggplot(n10.diff0.5, aes(x=pvalue)) +
  theme_classic() +
  labs(title=plot.title) +
  theme(plot.title = element_text(hjust=0.5, size=14, face="bold")) +
  geom_histogram(binwidth = 0.01, color="grey", fill="white") +
  geom_vline(xintercept = 0.05, color = "red", linetype = 'dashed', size=1) +
  annotate("text", label=paste("p < 0.05:", toString(sig.by.test), "/",
                               toString(n.tests)), x=0.7, y=sqrt(n.tests))
```

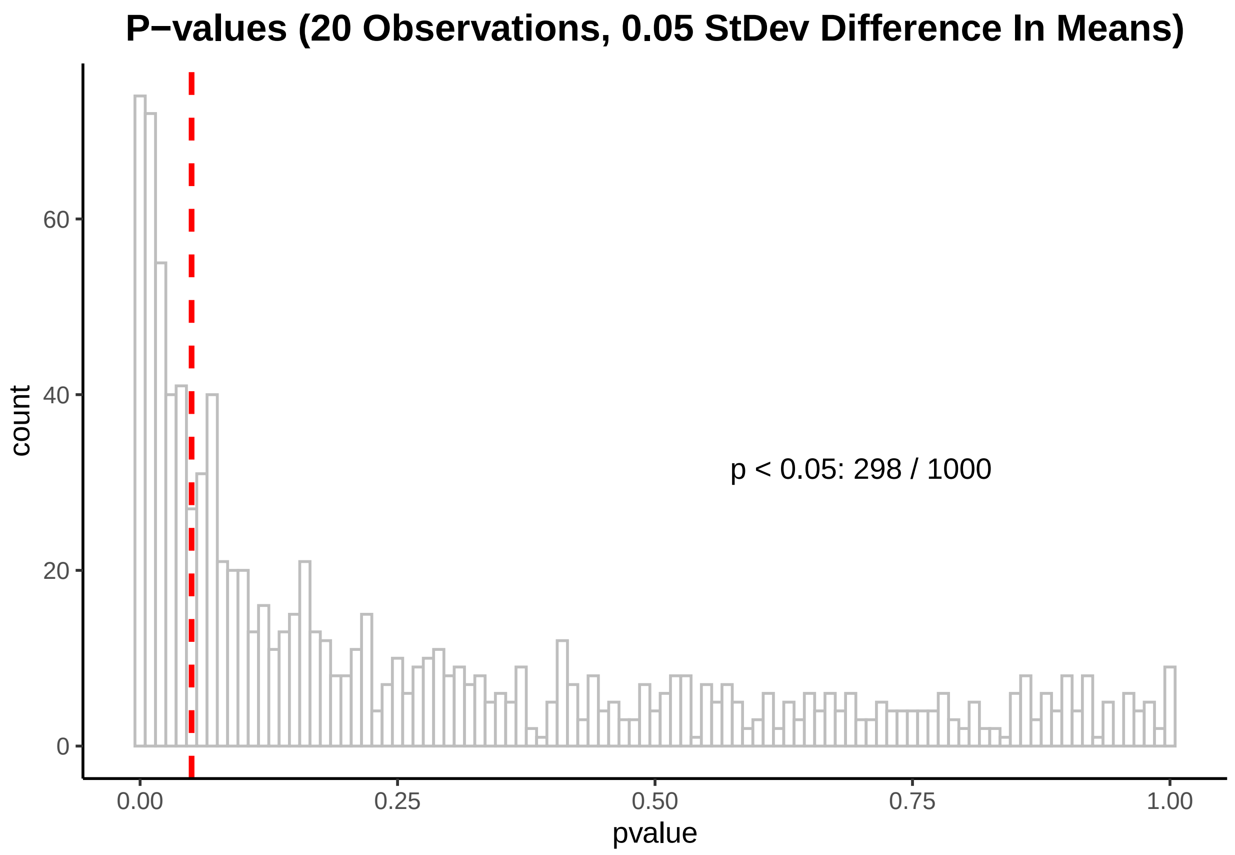


Interesting. About 15% of the time we decide on statistical significance for when n=10 observations, r = 10, and p is either 0.8 or 0.85. How does this change with increasing the number of observations?

```
n.observations <- 20
n20.diff0.5 <- run.wilcox.function(n.tests, n.observations,  prob.1 = 0.8, prob.2 = 0.85)
```

```
## [1] "Probability 1: 0.8; Probability 2: 0.85"
```

```
plot.title <- paste0("P-values (", toString(n.observations),
                     " Observations, ", toString(prob.diff),
                     " StDev Difference In Means)")
sig.by.test <- sum(n20.diff0.5$pvalue < 0.05)

ggplot(n20.diff0.5, aes(x=pvalue)) +
  theme_classic() +
  labs(title=plot.title) +
  theme(plot.title = element_text(hjust=0.5, size=14, face="bold")) +
  geom_histogram(binwidth = 0.01, color="grey", fill="white") +
  geom_vline(xintercept = 0.05, color = "red", linetype = 'dashed', size=1) +
  annotate("text", label=paste("p < 0.05:", toString(sig.by.test), "/",
                               toString(n.tests)), x=0.7, y=sqrt(n.tests))
```



Doubling the number of observations brings us from 15% to 25% the number of 'successful' tests. How well does that hold up?