Linguistic Data: Quantitative Analysis and Visualisation

Lab 3. Student`s t-test. Binomial test. R: Simulating data, matrices, boxplots

Student's t-test

Aspiration and vowel duration in Icelandic

This set is based on (Coretta 2017, link (https://goo.gl/NrfgJm). This dissertation dealt with the relation between vowel duration and aspiration in consonants. Author carried out a data collection with 5 natives speakers of Icelandic. Then he extracted the duration of vowels followed by aspirated versus non-aspirated consonants. Check out whether vowels before aspirated consonants (like in Icelandic takka ???key??? [t??a??ka]) are signi???cantly shorter than vowels followed by non-aspirated consonants (like in kagga ??? barrel??? [k??akka]). Link (https://raw.githubusercontent.com/LingData2019/LingData2020/master/data/icelandic.csv) to the dataset.

```
library(readr)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------- tidyverse 1.2.1 --
```

```
## <U+221A> ggplot2 2.2.1     <U+221A> purrr   0.2.4
## <U+221A> tibble  1.4.2     <U+221A> dplyr   0.7.4
## <U+221A> tidyr   0.7.2     <U+221A> stringr 1.2.0
## <U+221A> ggplot2 2.2.1     <U+221A> forcats 0.2.0
```

```
## -- Conflicts ------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
df <- read_csv("https://raw.githubusercontent.com/LingData2019/LingData2020/master/data/icelandic.csv")
```

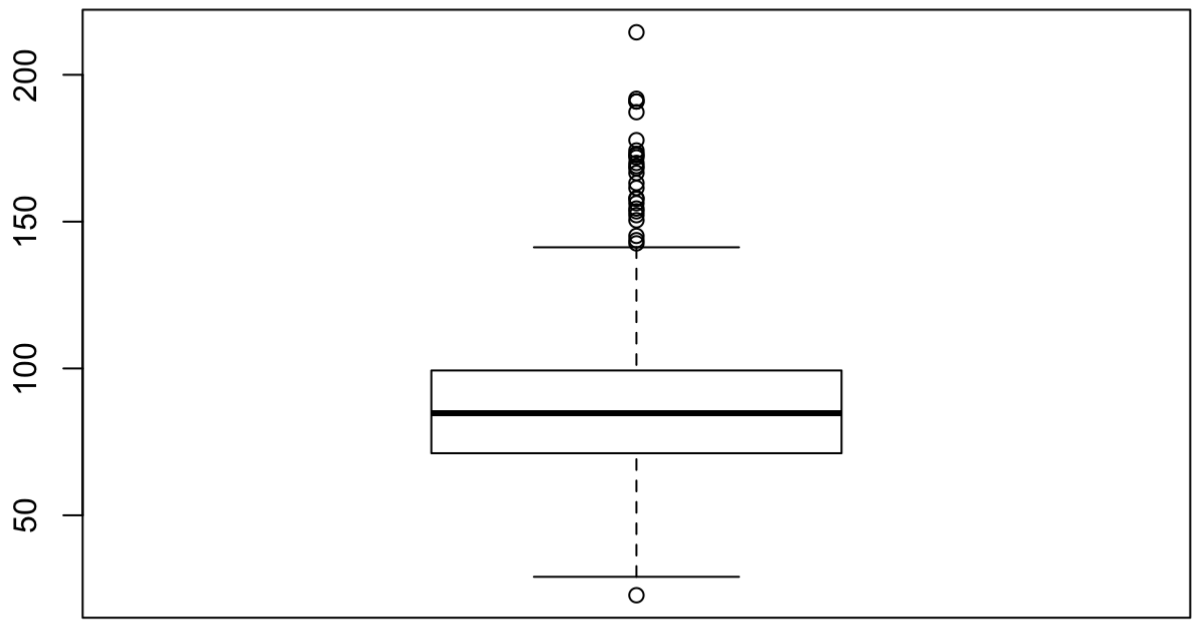
```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   index = col_integer(),
##   time = col_double(),
##   word.dur = col_double(),
##   voicing.dur = col_double(),
##   vowel.dur = col_double(),
##   cluster.dur = col_double(),
##   spreading.dur = col_double(),
##   sonorant.dur = col_double(),
##   closure.dur = col_double(),
##   vor = col_double(),
##   voffr = col_double(),
##   mor = col_double(),
##   cond_no = col_integer()
## )
```

```
## See spec(...) for full column specifications.
```

Descriptive statistics

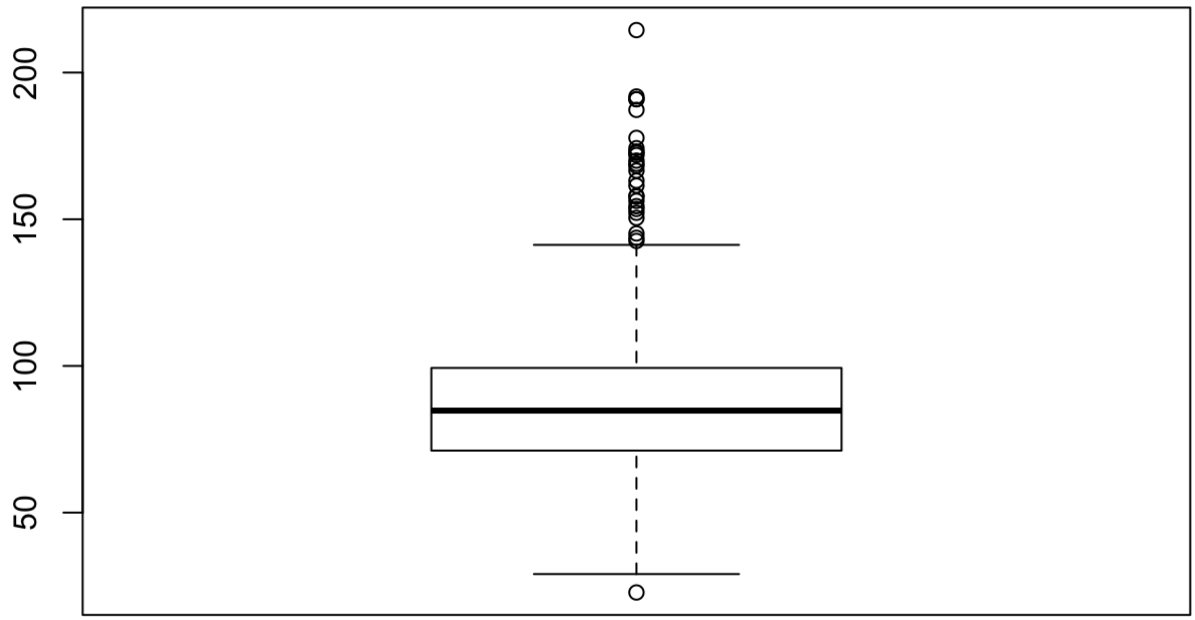
A general boxplot:

```
boxplot(df$vowel.dur)
```



Get the number of outliers:

```
length(boxplot(df$vowel.dur)$out)
```




```
## [1] 27
```

Look at number of observations by groups (aspirated and non-aspirated cases):

```
table(df$aspriration)
```

```
## Warning: Unknown or uninitialised column: 'aspriration'.
```

```
## < table of extent 0 >
```

Choose two subsamples, one for words where vowels are followed by aspirated consonants and another for non-aspirated consonants.

```
asp <- df[df$aspiration == 'yes',]
nasp <- df[df$aspiration == 'no',]
```

Summary for aspirated and non-aspirated cases:

```
summary(asp$vowel.dur)
```

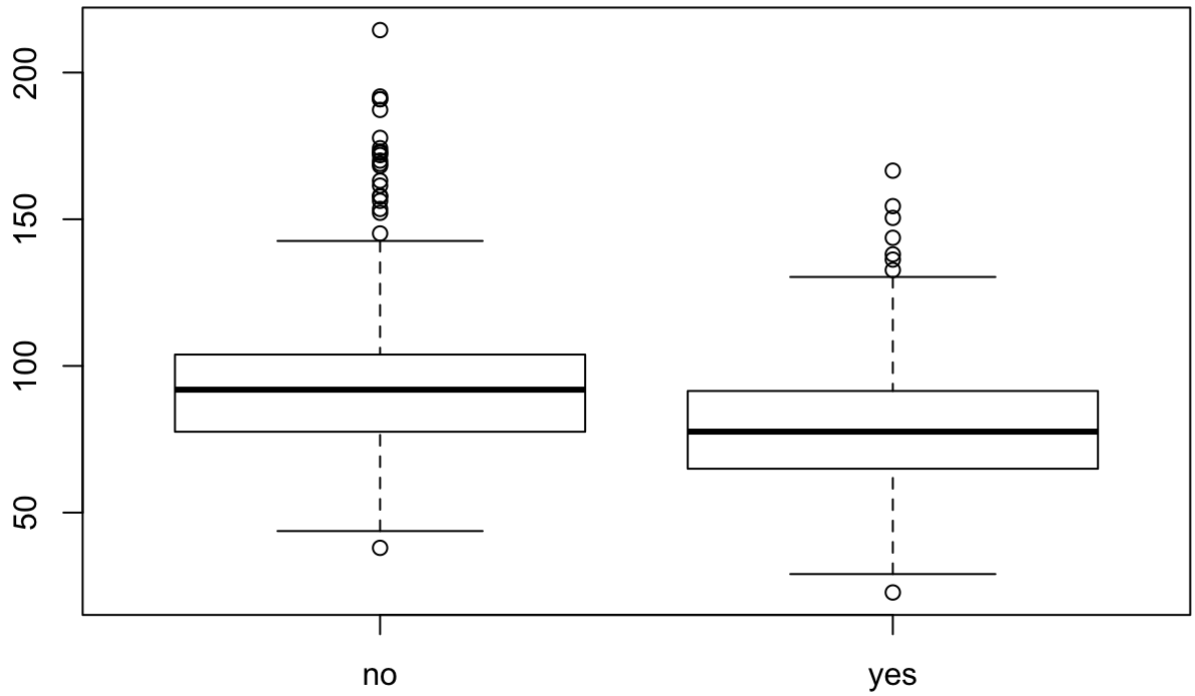
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   22.78   64.96   77.60   78.76   91.46  166.60
```

```
summary(nasp$vowel.dur)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   37.98   77.56   91.91   94.69  103.90  214.50
```

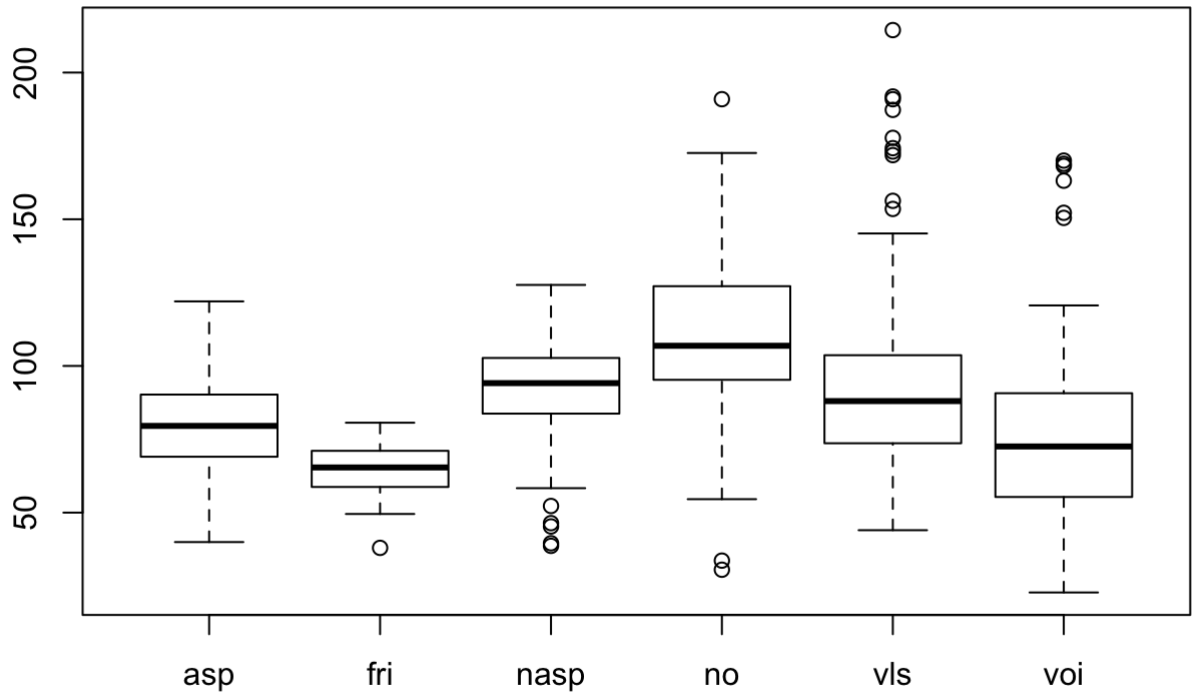
Boxplot by groups:

```
boxplot(df$vowel.dur ~ df$aspiration)
```



More interesting - let us create a boxplot by all groups (see the field `cons1`):

```
boxplot(df$vowel.dur ~ df$cons1)
```



You can compare distribution of `vowel.dur` in asp(irated), fri(cative), nasp(non-aspirated), voi(ced), etc.

We can limit our data to just one type of vowels, say, middle vowels. Therefore, we will work with the same type of a consonant:

```
asp <- df[df$aspiration == 'yes' & df$height == 'mid', ]
nasp <- df[df$aspiration == 'no' & df$height == 'mid', ]
```

Again, here is a summary for a corrected case:

```
summary(asp$vowel.dur)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   38.67   71.41   81.92   82.65   95.19  150.50
```

```
summary(nasp$vowel.dur)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   37.98   80.90   97.97   98.73  110.50  190.90
```

```
nrow(asp)
```

```
## [1] 156
```

```
nrow(nasp)
```

```
## [1] 174
```

T-test

Let us formulate the null hypothesis, the alternative hypotesis, and apply t-test to our dataset.

```
t.test(asp$vowel.dur, nasp$vowel.dur)
```

```
##
##  Welch Two Sample t-test
##
## data:  asp$vowel.dur and nasp$vowel.dur
## t = -6.4869, df = 317.72, p-value = 3.356e-10
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -20.94772 -11.19801
## sample estimates:
## mean of x mean of y
##  82.65371  98.72657
```

By default, R calculates t.test with regard to the bi-directional alternative hypothesis, such as $\mu_1 \neq \mu_2$.

Unidirectional t-test

H1: $\mu_{asp} < \mu_{nasp}$

```
t.test(asp$vowel.dur, nasp$vowel.dur, alternative = "less")
```

```
##
##  Welch Two Sample t-test
##
## data:  asp$vowel.dur and nasp$vowel.dur
## t = -6.4869, df = 317.72, p-value = 1.678e-10
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
##       -Inf -11.98542
## sample estimates:
## mean of x mean of y
##  82.65371  98.72657
```

Density plots

```
require(tidyverse)
require(dplyr)
```

Let's get a descriptive summary of our data in a dplyr style.

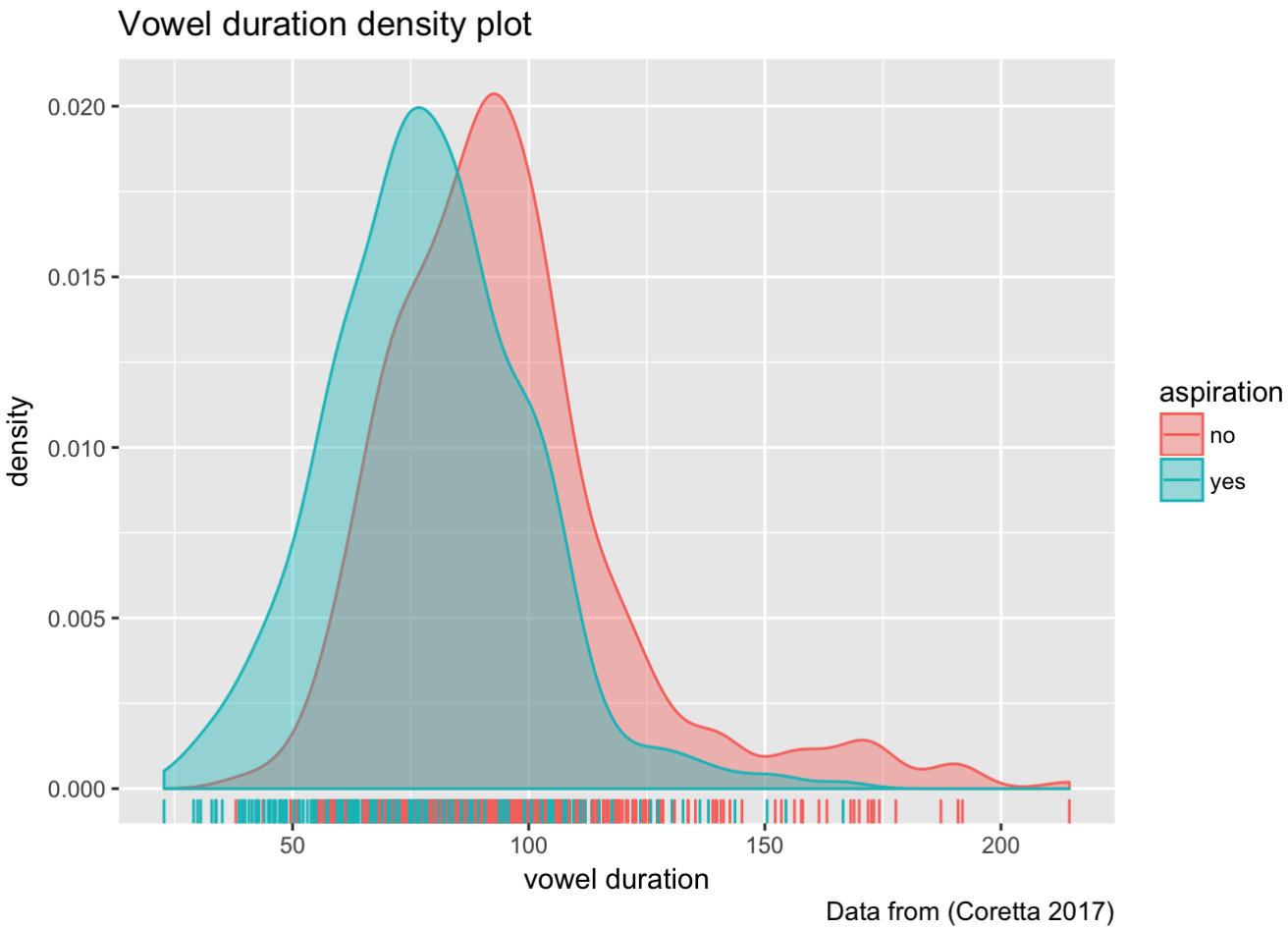
```
df %>%
  group_by(aspiration) %>%
  summarise(mean = mean(vowel.dur),
            st.dev = sd(vowel.dur))
```

aspiration <chr>	mean <dbl>	st.dev <dbl>
no	94.69124	25.85561
yes	78.75772	21.24177

2 rows

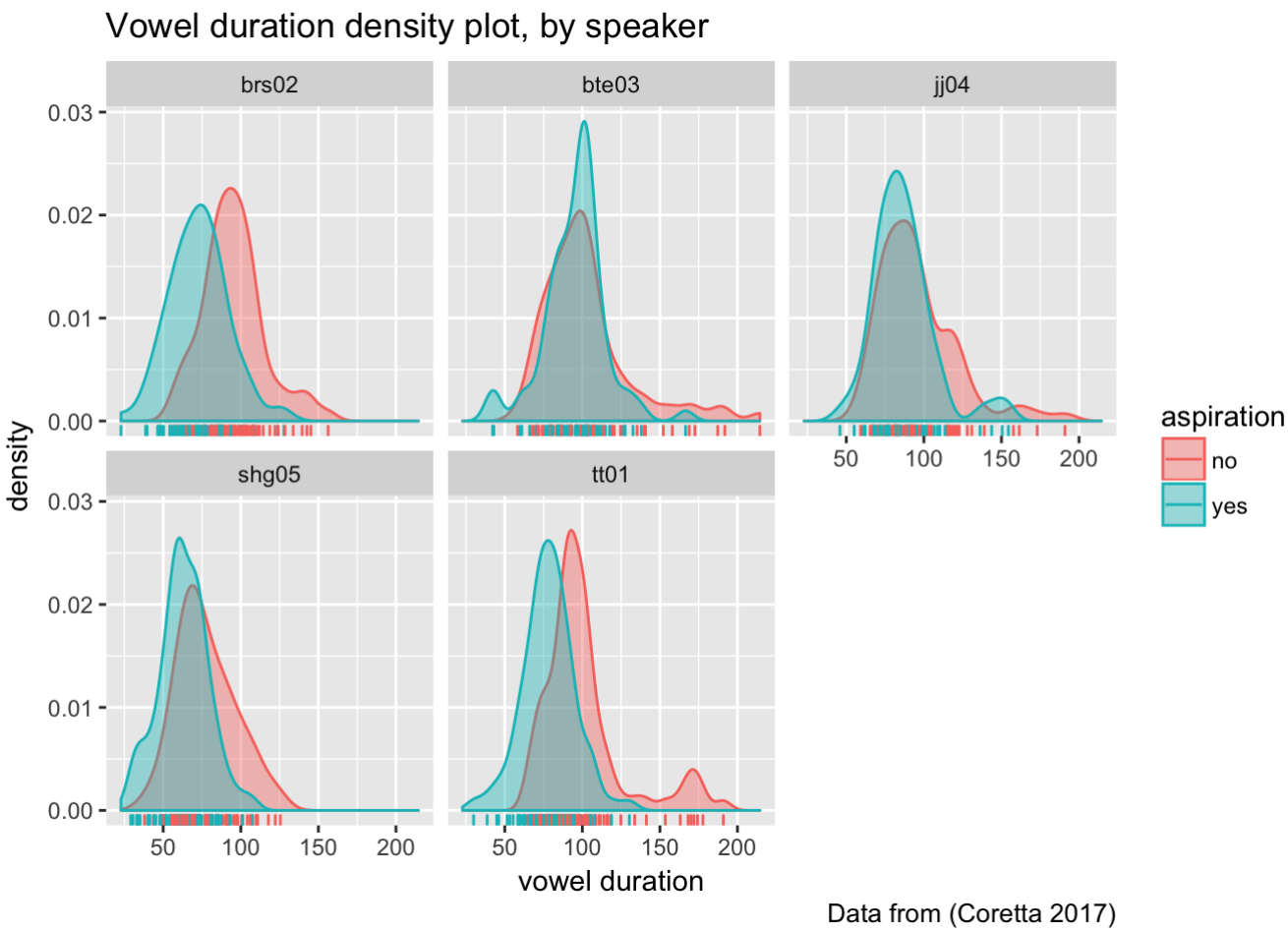
Density plots can be thought of as plots of smoothed histograms.

```
library(ggplot2)
df %>%
  ggplot(aes(vowel.dur, fill = aspiration, color = aspiration))+
  geom_density(alpha = 0.4)+
  geom_rug()+
  labs(title = "Vowel duration density plot",
       caption = "Data from (Coretta 2017)",
       x = "vowel duration")
```



Density plot by speaker:

```
df %>%
  ggplot(aes(vowel.dur, fill = aspiration, color = aspiration))+
  geom_density(alpha = 0.4)+
  geom_rug()+
  facet_wrap(~speaker)+
  labs(title = "Vowel duration density plot, by speaker",
       caption = "Data from (Coretta 2017)",
       x = "vowel duration")
```



and descriptive statistics:

```
df %>%
  group_by(aspiration, speaker) %>%
  summarise(mean = mean(vowel.dur),
            st.dev = sd(vowel.dur))
```

aspiration <chr>	speaker <chr>	mean <dbl>	st.dev <dbl>
no	brs02	95.27593	19.81246
no	bte03	102.75791	29.38451
no	jj04	95.72652	25.13620
no	shg05	77.65635	18.86992
no	tt01	100.59767	26.77305
yes	brs02	72.87988	18.85812
yes	bte03	95.40109	20.03149
yes	jj04	86.80463	20.06335
yes	shg05	63.27287	15.72727
yes	tt01	78.25853	16.74632

1-10 of 10 rows

Simulating data

Create a matrix 2 * 3 consisting of 0:

```
matrix(0, nrow=2, ncol=3)
```

```
##      [,1] [,2] [,3]
## [1,]    0    0    0
## [2,]    0    0    0
```

Arrange a vector of 12 values into matrix 3 * 4 arrange by rows:

```
v <- 1:12
m <- matrix(v, n=3, ncol=4, byrow = TRUE)
m
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8
## [3,]    9   10   11   12
```

Sum of every row in a matrix:

```
rowSums(m)
```

```
## [1] 10 26 42
```

Create a sample of 0 and 1 of size 10.

```
sample(c(0, 1), 10, replace = TRUE)
```

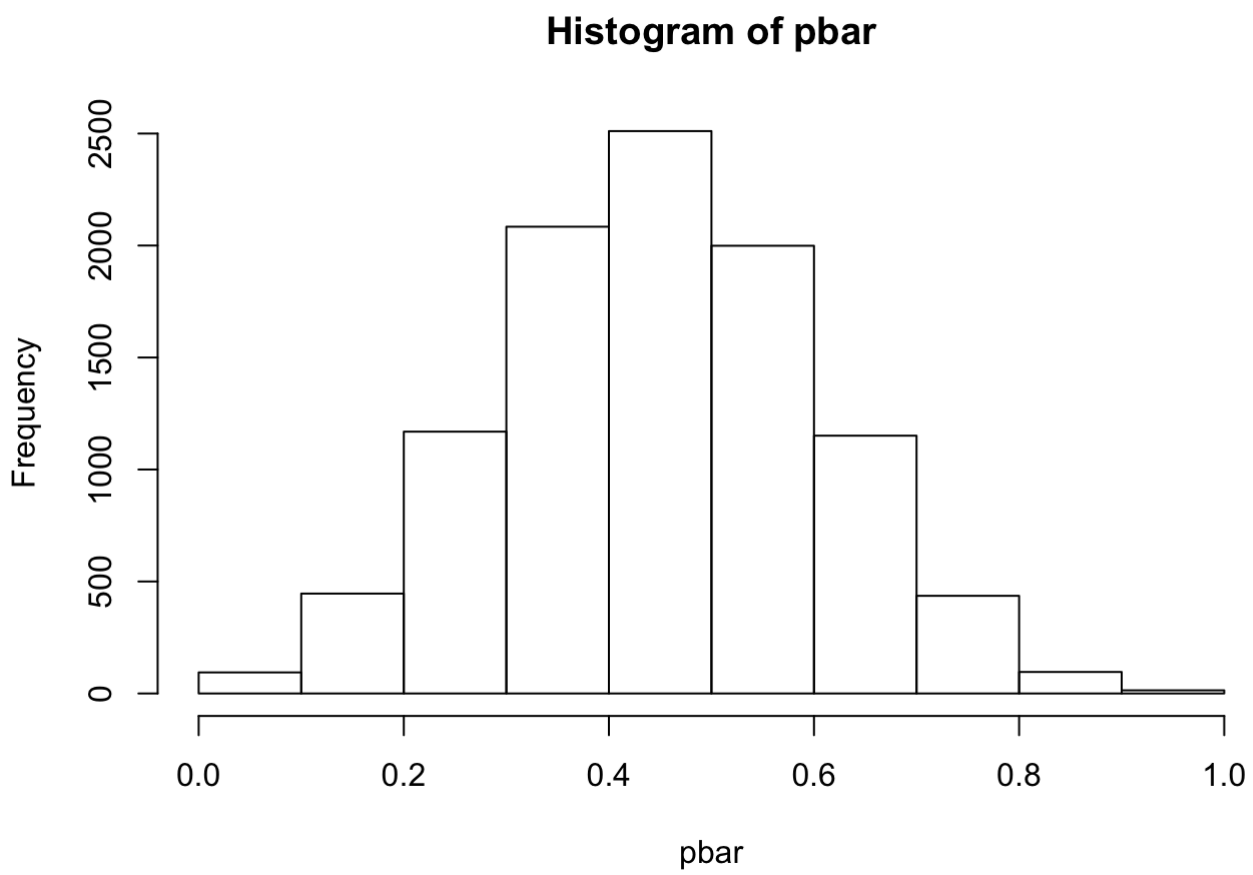
```
##  [1] 0 1 0 0 1 1 0 1 0 0
```

Experiment: toss a coin 10 times and repeat this sequence 10000 times:

```
tosses <- 10
samples <- 10000
dat <- matrix(sample(c(0, 1), tosses * samples, replace=TRUE), ncol=tosses, byrow=TRU
E)
```

Calculate `phats` - proportions of heads in each experiment:

```
pbar <- rowSums(dat) / tosses
hist(pbar, breaks=tosses, xlim=c(0, 1))
```



Test $H_0 : p = 0.5$:

```
binom.test(3, 10, p=0.5) # 3 out of 10 - a fair coin?
```

```
##
##  Exact binomial test
##
## data:  3 and 10
## number of successes = 3, number of trials = 10, p-value = 0.3438
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.06673951 0.65245285
## sample estimates:
## probability of success
##                    0.3
```

```
binom.test(2, 10)  # 2 out of 10 - a fair coin?
```

```
##
##  Exact binomial test
##
## data:  2 and 10
## number of successes = 2, number of trials = 10, p-value = 0.1094
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.02521073 0.55609546
## sample estimates:
## probability of success
##                    0.2
```

```
binom.test(1, 10)  # 1 out of 10 - a fair coin?
```

```
##
##  Exact binomial test
##
## data:  1 and 10
## number of successes = 1, number of trials = 10, p-value = 0.02148
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.002528579 0.445016117
## sample estimates:
## probability of success
##                    0.1
```

Load some dataset and check some null hypothesis with binomial test.

```
df <- read.csv("https://raw.githubusercontent.com/LingData2019/LingData2020/master/da
ta/poetry_last_in_lines.csv", sep = "\t")
```

Suggest your hypotheses about p of nouns. Look at frequencies:

```
table(df$UPoS)
```

```
##
##  ADJ  ADP  ADV  DET INTJ NOUN  NUM PART PRON VERB    X
##   52    1   16    6    1  222    3    2    6   54    1
```

```
table(df$UPoS)/sum(table(df$UPoS))
```

```
##
##         ADJ         ADP         ADV         DET        INTJ        NOUN
## 0.142857143 0.002747253 0.043956044 0.016483516 0.002747253 0.609890110
##         NUM        PART        PRON        VERB           X
## 0.008241758 0.005494505 0.016483516 0.148351648 0.002747253
```

Is it enough to make conclusions? No, proceed to formal tests:

```
# select lines with nouns
nouns <- df[df$UPoS=='NOUN',]
total <- nrow(df) # number of trials
nnouns <- nrow(nouns) # number of successes
```

```
# H0: p = 0.6
```

```
# H0: p = 0.4
```

```
# choose lines with one-syllable words at the end
```

```
# you can test on your own for every number of syllables
```

Supplementary R code

This code generates a dataset that consists of Utterances (strings of letters) and Responces corresponding to each utterance (either 0 or 1)

```
# require(stringi)
n <- 1000 # the number of datapoins
df <- cbind.data.frame(Utterance = stringi::stri_rand_strings(10, 5), # generate a ra
ndom string
                       Responce = rbinom(n, 1, 0.2))  # generate an answer (either 0
 ot 1) randomly with p(1) = 0.2
```

This code run binom.test n times with forward-pipes:

```
require(dplyr)
require(broom)
m <- 5 # sample size in each run
n <- 10 # the number of experiments
dat <- replicate(n=n, expr = sample(0:1, size=m, replace=TRUE)) %>%
           t() %>%                    # transpose row and columns
           as.data.frame() %>%
           mutate(ID=row_number(), sum=rowSums(.), m=m) # add ID, row sums, sample si
ze
dat2 <- dat %>%
            group_by(ID) %>%
            do(tidy(binom.test(.$sum, .$m, alternative = "two.sided"))) %>%
            select(ID, p.value)
```