Announcements

- Make sure to sign in on the google form (I send a list of what section questions are useful for what pset questions afterwards)
- Pset 2 due Friday 2/10

Alphabet Soup

The following question deals with the set of four letter words in the Scrabble dictionary available here.

In this question, we will be modeling the letters in four letter words as draws from a multinomial distribution: $\vec{Y} \sim \text{Mult}_k(4, \vec{p})$. Recall that the Multinomial PMF is

$$P(Y_1 = n_1, ..., Y_k = n_k) = \frac{n!}{\prod_{i=1}^k n_i!} \prod_{i=1}^k p_i^{n_i}$$

1. What should k be?

26 for 26 letters

2. Suppose we generated a draw from the distribution. What additional step would we have to perform to construct a word from the draw?

We would need to put the draw in a particular order to make word. The draw itself is just a list of letter frequencies.

3. What assumption of the Multinomial distribution is obviously violated here?

The Multinomial distribution assumes that the n draws are independent of each other. However, that is clearly false; for example, knowing the first three letters are 'joy' perfectly determines that the word must be 'joys.' Relatedly, the assumption of equal probabilities $\vec{p}$ on each of the n draws is wrong as seen below. However, we will still model this problem as a Multinomial because it provides a nice example.

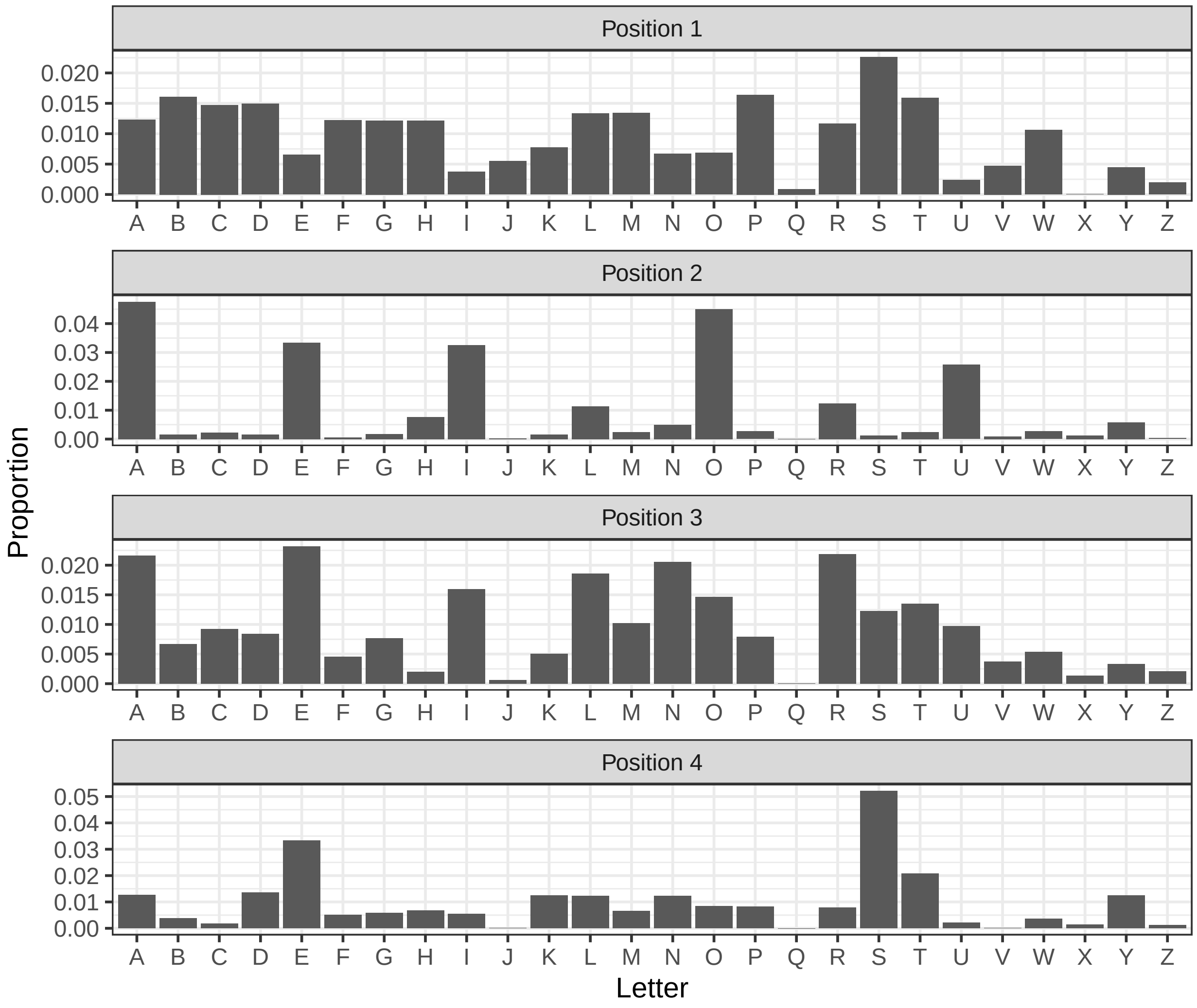
```r
# Split words into letters
letters_out <- strsplit(dictionary$words, "")

df <- data.frame(matrix(ncol = 2, nrow = 0))
colnames(df) <- c("letter", "position")

# Make a dataframe of letters and their positions
for (letter in letters_out) {
  df <- rbind(df, data.frame("letter" = letter, "position" = 1:4))
}

# Turn this into a table mapping letters to frequencies
df <- data.frame(table(df))
df$Freq <- df$Freq / sum(df$Freq)
df$position <- paste0("Position ", df$position)

ggplot(df, aes(x = toupper(letter), y = Freq)) +
  geom_bar(stat="identity") +
  xlab("Letter") +
  ylab("Proportion") +
  theme_bw() +
  facet_wrap(as.factor(df$position), nrow = 4, scales = "free")
```



4. Find the likelihood function $L(\vec{p};\mathbf{Y})$ where $\mathbf{Y}$ is a $n\times k$ matrix with one row for each word and one column for each letter. (E.g. "face" would become a row $1,0,1,0,1,1,0,...,0$.) Then, find the log likelihood function. What constants can we drop?

$$L(\vec{p};\vec{y}) = \prod_{i=1}^{n}\frac{4!}{\prod_{j=1}^{k}\mathbf{Y}_{i,j}!}\prod_{j=1}^{k}p_j^{\mathbf{Y}_{i,j}} \propto \prod_{i=1}^{n}\prod_{j=1}^{k}p_j^{\mathbf{Y}_{i,j}}$$

Taking the log gives:

$$\ell(\vec{p};\vec{y}) = \sum_{i=1}^{n}\sum_{j=1}^{k}\mathbf{Y}_{i,j}\log(p_j) = \sum_{j=1}^{k}\log(p_j)\left(\sum_{i=1}^{n}\mathbf{Y}_{i,j}\right)$$

and $\sum_{i=1}^{n}\mathbf{Y}_{i,j}$ is just the sum of the j^{th} column of $\mathbf{Y}$ (how many times letter number j appeared in all the words).

5. If we maximized the log likelihood as it stands now, we would end up with $p_j=\infty$ for all p_j. To prevent this, we'll restrict the sum of the p_j with a Lagrangian constraint. Specifically, find the gradient of $\ell(\vec{p};\vec{y})+\lambda(1-\sum_{j=1}^{k}p_j)$ (the derivative with respect to each p_j), set it equal to 0, solve for p_j, and then use the fact that $\sum_{j=1}^{k}p_j=1$ to solve for λ. Explain in words what this MLE is.

Taking the derivative with respect to p_j gives:

$$\frac{(\sum_{i=1}^n \mathbf{Y}_{i,j})}{\hat{p_j}} - \lambda = 0 \implies \hat{p_j} = \frac{(\sum_{i=1}^n \mathbf{Y}_{i,j})}{\lambda}$$

Using the fact that $\sum_{j=1}^k \hat{p_j} = 1$,

$$1 = \frac{1}{\lambda}\sum_{j=1}^k \left(\sum_{i=1}^n \mathbf{Y}_{i,j}\right) \implies \lambda = 4n$$

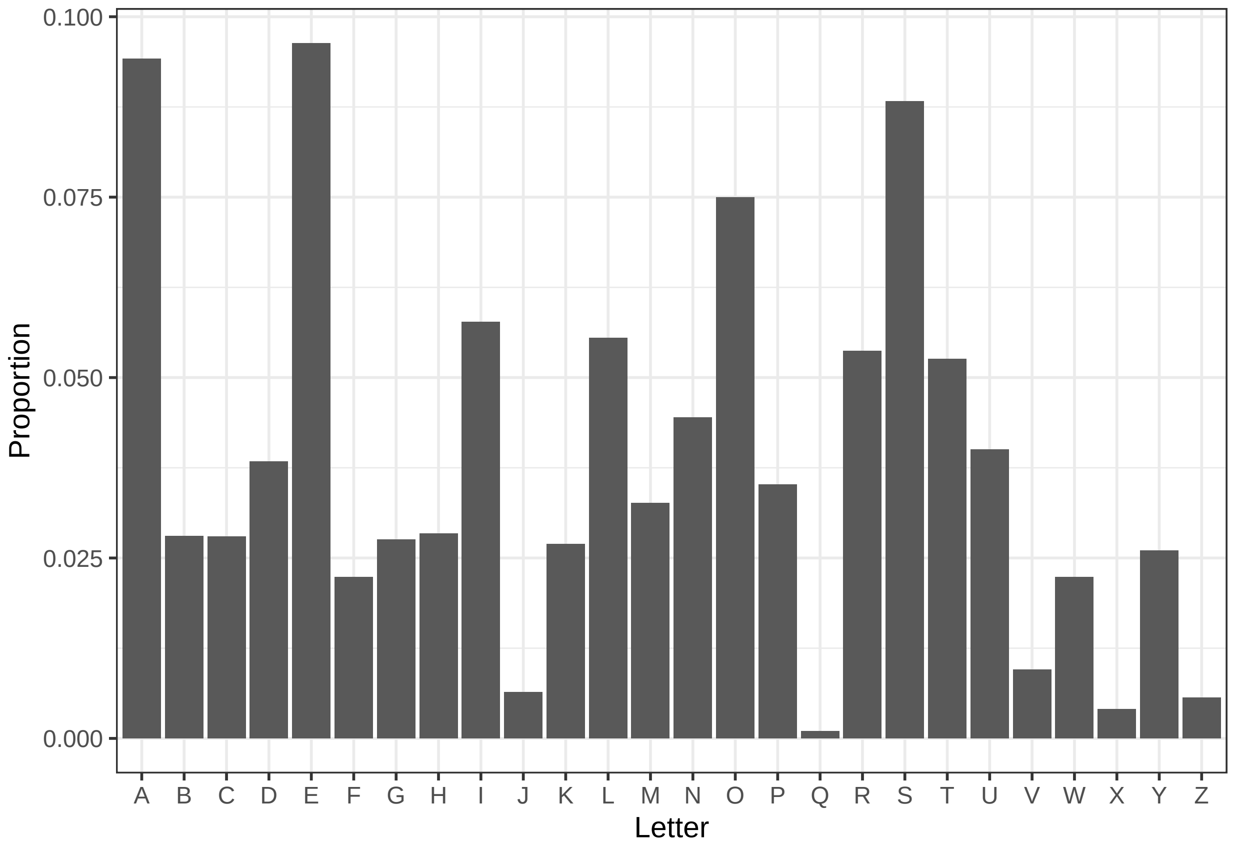
since we are summing over the whole $\mathbf{Y}$ matrix. Thus, $\hat{p_j} = \frac{\sum_{i=1}^n \mathbf{Y}_{i,j}}{4n}$, which is just the proportion of letter number j in the list.

6. Find the $\hat{p_j}$ from the data and display them visually.

```r
# Split words into letters and create one long vector of all the letters
letters_out <- unlist(strsplit(dictionary$words, ""))

# Make a dataframe with the proportion of each letter
df <- data.frame(table(letters_out) / length(letters_out))
colnames(df) <- c("letters", "proportions")

ggplot(df, aes(x = toupper(letters), y = proportions)) +
  geom_bar(stat="identity") +
  xlab("Letter") +
  ylab("Proportion") +
  theme_bw()
```



7. Suppose you generate a set of 4 letters from the multinomial distribution above and put the 4 letters in a random order. What is the probability of producing the word “stat”? Find this in two ways: Find this in two ways: first, condition on the multinomial. Second, use counting.

First, the probability of drawing the letters s, t, a, t from the multinomial distribution is $\frac{4!}{2!}p_1p_{19}^2p_{20}$. The probability of randomly arranging these into the correct order is then $2/4!$. Thus, the probability is $\frac{4!}{2!}p_1p_{19}^2p_{20} \cdot 2/4! = p_1p_{19}^2p_{20}$. By direct counting, we have a p_{19} chance of drawing an “S” first, a p_{20} chance of drawing a “T” second, and so on, which gives a probability of $p_1p_{19}^2p_{20}$. In our dataset, this probability is 0.000023.

```
df$proportions[df$letters == "s"] *
  df$proportions[df$letters == "t"] *
  df$proportions[df$letters == "a"] *
  df$proportions[df$letters == "t"]
```

```
## [1] 2.298363e-05
```

8. With the magic of seed setting, we can make this probability 1! (Interestingly, the probability above means the expected number of seeds until generating “stat” is 43478, and the actual seed was 31083.)

```
set.seed(31083)

# Make a single draw from Mult(4, p)
draw <- as.vector(rmultinom(1, 4, df$proportions))

# Turn this draw into a vector of letters
bag_of_letters <- vector()
for (i in 1:length(letters)) {
  bag_of_letters <- c(bag_of_letters, rep(letters[i], draw[i]))
}

# Put these letters in a random order
word <- paste0(sample(bag_of_letters, 4), collapse = "")

print(word)
```

```
## [1] "stat"
```

9. Find a maximum likelihood estimator for $\text{Cov}(\vec{Y}_{[1]}, \vec{Y}_{[2]})$, the covariance between the number of As and the number of Bs.

By the invariance property,

$$\widehat{\text{Cov}(\vec{Y}_{[1]}, \vec{Y}_{[2]})} = -4\hat{p_1}\hat{p_2} = -4\frac{\sum_{i=1}^n \mathbf{Y}_{i,1}}{4n}\frac{\sum_{i=1}^n \mathbf{Y}_{i,2}}{4n}$$

10. Find a method of moments estimator for the covariance between the number of As and the number of Bs.

By the methods of moments principle (changing estimands to estimators and expectations to sample moments),

$$\text{Cov}(\vec{Y}_{[1]}, \vec{Y}_{[2]}) = E(\vec{Y}_{[1]}\vec{Y}_{[2]}) - E(\vec{Y}_{[1]})E(\vec{Y}_{[2]}) \implies$$

$$\widehat{\text{Cov}(\vec{Y}_{[1]}, \vec{Y}_{[2]})} = \frac{1}{n}\sum_{i=1}^n \mathbf{Y}_{i,1}\mathbf{Y}_{i,2} - \left(\frac{1}{n}\sum_{i=1}^n \mathbf{Y}_{i,1}\right)\left(\frac{1}{n}\sum_{i=1}^n \mathbf{Y}_{i,2}\right)$$

11. Compare the MSEs of these two estimators through simulation with $n = 30$ “words” in each draw.

```r
nsims <- 10000
n <- 30

single_sim <- function() {
  # Make n draws from a multinomial
  y <- rmultinom(n, 4, df$proportions)

  # Calculate the MLE
  mle <- -4 * rowSums(y)[1] / (4 * n) * rowSums(y)[2] / (4 * n)

  # Calculate the MOM estimator
  mom <- mean(y[1, ] * y[2, ]) - mean(y[1, ]) * mean(y[2, ])

  return (c(mle, mom))
}

# Run this nsims times, storing results in a matrix
estimates <- replicate(nsims, single_sim())

# Get the true covariance
true_cov <- - 4 * df$proportions[1] * df$proportions[2]

# MLE MSE
print(mean((estimates[1,] - true_cov)^2))
```

```
## [1] 4.096464e-05
```

```r
# MOM MSE
print(mean((estimates[2,] - true_cov)^2))
```

```
## [1] 0.0009540737
```

On MSE, the MLE is about 20 times better.

Logistic Logic

The Logistic(s) distribution is defined to be the distribution of $s\log\left(\frac{U}{1-U}\right)$ where $U\sim \text{Unif}(0,1)$.

1. Find the CDF $F(y)$ of the Logistic distribution. (Hint: Let Y have the Logistic distribution and let U have the Uniform distribution. Then, write Y in terms of U, isolate U, and use the Uniform PDF.)

$$\begin{aligned}P(Y<y)&=P\left(s\log\left(\frac{U}{1-U}\right)<y\right)\\&=P\left(\frac{U}{1-U}<e^{y/s}\right)\\&=P\left(U<\frac{e^{y/s}}{1+e^{y/s}}\right)\\&=\frac{e^{y/s}}{1+e^{y/s}}\\&=\frac{1}{e^{-y/s}+1}\end{aligned}$$

2. Find the PDF of the Logistic distribution.

Taking the derivative of the expression above with respect to y gives:

$$F'(y) = \frac{e^{-y/s}}{s(e^{-y/s}+1)^2}$$

3. Let Y have the Logistic distribution. Find $E(Y)$ and $\text{Var}(Y)$. You may use the facts that $\int_0^1 \ln\left(\frac{x}{1-x}\right) dx = 0$ and $\int_0^1 \ln^2\left(\frac{x}{1-x}\right) dx = \pi^2/3$.

Using the representation of the Logistic distribution involving Uniform random variables,

$$E(Y) = \int_0^1 s \log\left(\frac{x}{1-x}\right) dx = 0$$

and

$$\text{Var}(Y) = s^2\text{Var}\left(\log\left(\frac{U}{1-U}\right)\right) = s^2\text{E}\left(\log^2\left(\frac{U}{1-U}\right)\right) = s^2\int_0^1 \log^2\left(\frac{x}{1-x}\right) dx = \frac{s^2\pi^2}{3}$$

4. Suppose we are measuring vehicle velocities on a congested highway which are distributed Logistic(s). (It makes sense for the distribution to be symmetric around 0 since the vehicles are equally likely to be going either direction). However, our instrument can only measure velocities in the range of $[-c, c]$ (for any physicists in the room, c does not represent the speed of light). We want to estimate s despite this limitation. Find the likelihood function for s given n_1 observed velocities $Y_1, ..., Y_{n_1}$, n_2 velocities less than $-c$ and n_3 velocities more than c.

We can write the likelihood function using the Logistic PDF for observed points and the CDF for unobserved points. Note that the probability of a car going above c is the same as the probability of it going below $-c$:

$$P(Y_i > c) = 1 - P(Y_i < c) = 1 - \frac{1}{e^{-c/s}+1} = \frac{e^{-c/s}}{e^{-c/s}+1} = \frac{1}{1+e^{c/s}} = P(Y_i < -c)$$

Thus we can group all the speeds below $-c$ or above c:

$$L(s;\vec{y}) = \left(\prod_{i=1}^{n_1} \frac{e^{-y_i/s}}{s(e^{-y_i/s}+1)^2}\right)\left(\frac{1}{e^{c/s}+1}\right)^{n_2+n_3}$$

5. A closed form solution for the maximum likelihood estimator of s does not exist (or at least I wasn't able to find it after an hour of number pushing, and Google doesn't seem to have it either). Instead, consider the following method of moments estimator for s:

$$\hat{s} = \sqrt{\frac{3}{n_1\pi^2}\sum_{i=1}^{n_1} Y_i^2}$$

Describe the logic behind this estimator.

Using the method of moments with $Y \sim \text{Logistic}(s)$, we have

$$\text{Var}(Y) = E(Y^2) = \frac{s^2\pi^2}{3} \implies s = \sqrt{3E(Y^2)/\pi^2}$$

If we keep only the Y_i for which we have exact values and plug in sample means, we get $\hat{s} = \sqrt{\frac{3}{n_1\pi^2}\sum_{i=1}^{n_1} Y_i^2}$.

6. Find the sign of the bias of $\hat{s}$ for s with an argument about how $E\left(\frac{3}{n_1\pi^2}\sum_{i=1}^{n_1} Y_i^2\right)$ compares to s^2 and Jensen's inequality. (Since the square root function is concave, Jensen's inequality says $E(\sqrt{X}) < \sqrt{E(X)}$.)

Since observations below $-c$ and above c are thrown out, $E(Y_i^2)$ for $i \in \{1, ..., n_1\}$ is less than $E(Y^2)$, so

$$E\left(\frac{3}{n_1\pi^2}\sum_{i=1}^{n_1} Y_i^2\right) < s^2$$

Also, by Jensen's inequality,

$$E(\hat{s}) = E\left(\sqrt{\frac{3}{n_1\pi^2}\sum_{i=1}^{n_1} Y_i^2}\right) < \sqrt{E\left(\frac{3}{n_1\pi^2}\sum_{i=1}^{n_1} Y_i^2\right)}$$

Putting these together, we get

$$E(\hat{s}) < \sqrt{s^2} = s \implies E(\hat{s}) - s < 0$$

so the bias is negative, and our estimator underestimates s. The parameter s is a scale parameter in the Logistic distribution, so it makes sense that dropping extreme values causes us to underestimate s.