Chapter 1 examples

Brooke Anderson

2/6/2020

```
library(tidyverse)
library(purrr)
library(forcats)
```

Example: Mutations in HIV genome with replication

Calculate the chance of three mutations under a Poisson distribution with $\lambda = 5$:

```
# Using d* function in R
dpois(x = 3, lambda = 5)
```

```
## [1] 0.1403739
```

```
# Using equation for Poisson distribution
5 ^ 3 * exp(-5) / factorial(3)
```

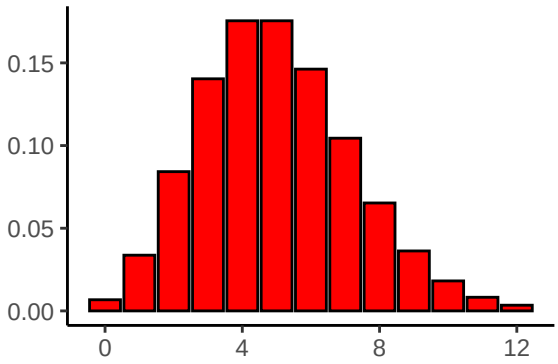
```
## [1] 0.1403739
```

Make a probability mass distribution plot for Poisson(5) model (Figure 1.1 in text, but with `tidyverse`):

```
# Tidyverse version
pois5_pmd <- tibble(x = 0:12) %>%
  mutate(prob_x = dpois(x, lambda = 0.0005 * 10000))
pois5_pmd
```

```
## # A tibble: 13 x 2
##        x  prob_x
##    <int>   <dbl>
##  1     0 0.00674
##  2     1 0.0337
##  3     2 0.0842
##  4     3 0.140
##  5     4 0.175
##  6     5 0.175
##  7     6 0.146
##  8     7 0.104
##  9     8 0.0653
## 10     9 0.0363
## 11    10 0.0181
## 12    11 0.00824
## 13    12 0.00343
```

```
ggplot(pois5_pmd, aes(x = x, y = prob_x)) +
  geom_col(fill = "red", color = "black") +
  theme_classic() +
  theme(axis.title = element_blank())
```

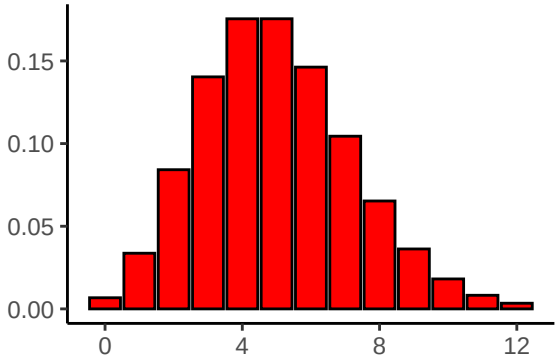


Here, the Poisson is a reasonable way to approximate a

```
# Tidyverse version
binom_pmd <- tibble(x = 0:12) %>%
  mutate(prob_x = dbinom(x, prob = 0.0005, size = 10000))
binom_pmd
```

```
## # A tibble: 13 x 2
##        x  prob_x
##    <int>   <dbl>
##  1     0 0.00673
##  2     1 0.0337
##  3     2 0.0842
##  4     3 0.140
##  5     4 0.175
##  6     5 0.176
##  7     6 0.146
##  8     7 0.104
##  9     8 0.0653
## 10     9 0.0363
## 11    10 0.0181
## 12    11 0.00823
## 13    12 0.00343
```

```
ggplot(binom_pmd, aes(x = x, y = prob_x)) +
  geom_col(fill = "red", color = "black") +
  theme_classic() +
  theme(axis.title = element_blank())
```



Another way to think through the HIV mutations example:

```
hiv_mutations <- tibble(nucleotide_index = 1:10000,
                        mutation = sample(c("no mutation", "mutation"),
                                          size = 10000,
                                          replace = TRUE,
                                          prob = c(1 - 0.0005, 0.0005)))
head(hiv_mutations)
```

```
## # A tibble: 6 x 2
##   nucleotide_index mutation
##              <int> <chr>
## 1                1 no mutation
## 2                2 no mutation
## 3                3 no mutation
## 4                4 no mutation
## 5                5 no mutation
## 6                6 no mutation
```

```
hiv_mutations %>%
  group_by(mutation) %>%
  count()
```

```
## # A tibble: 2 x 2
## # Groups:   mutation [2]
##   mutation        n
##   <chr>       <int>
## 1 mutation        8
## 2 no mutation  9992
```

Here's an alternative using `rbinom` and letting "0" stand for "no mutation" and "1" for "mutation":

```
hiv_mutations <- tibble(nucleotide_index = 1:10000,
                        mutation = rbinom(10000, prob = 0.0005, size = 1))
head(hiv_mutations)
```

```
## # A tibble: 6 x 2
##   nucleotide_index mutation
##              <int>    <int>
## 1                1        0
## 2                2        0
## 3                3        0
## 4                4        0
## 5                5        0
## 6                6        1
```

```
hiv_mutations %>%
  group_by(mutation) %>%
  count()
```

```
## # A tibble: 2 x 2
## # Groups:   mutation [2]
##   mutation     n
##      <int> <int>
## 1        0  9992
## 2        1     8
```

If order doesn't matter, you can increase the `size` parameter (number of trials) and just get out the count of successes across all those trials:

```
rbinom(1, prob = 0.0005, size = 10000)
```

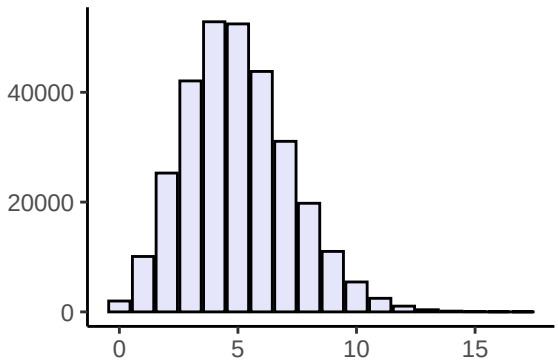
```
## [1] 2
```

Simulate this process lots of times—what are "typical" numbers of mutations with each replication cycle across the HIV genome?

```
hiv_simul <- tibble(sim_index = 1:300000,
                    n_of_mutations = rbinom(300000, prob = 0.0005, size = 10000))
hiv_simul %>%
  group_by(n_of_mutations) %>%
  count()
```

```
## # A tibble: 18 x 2
## # Groups:   n_of_mutations [18]
##    n_of_mutations     n
##             <int> <int>
##  1              0  1982
##  2              1 10094
##  3              2 25290
##  4              3 42075
##  5              4 52851
##  6              5 52462
##  7              6 43804
##  8              7 31071
##  9              8 19787
## 10              9 11020
## 11             10  5449
## 12             11  2474
## 13             12  1049
## 14             13   388
## 15             14   137
## 16             15    45
## 17             16    16
## 18             17     6
```

```
hiv_simul %>%
  group_by(n_of_mutations) %>%
  count() %>%
  ggplot(aes(x = n_of_mutations, y = n)) +
  geom_col(fill = "lavender", color = "black") +
  theme_classic() +
  theme(axis.title = element_blank())
```

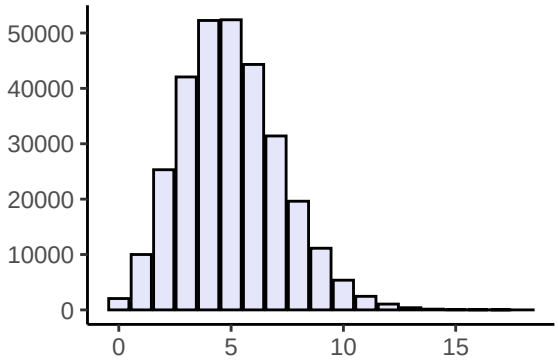


Same, but using a Poisson distribution to approximate, since the probability of success is very low:

```
hiv_simul <- tibble(sim_index = 1:300000,
                    n_of_mutations = rpois(300000, lambda = 0.0005 * 10000))
hiv_simul %>%
  group_by(n_of_mutations) %>%
  count()
```

```
## # A tibble: 19 x 2
## # Groups:   n_of_mutations [19]
##    n_of_mutations     n
##             <int> <int>
##  1              0  2053
##  2              1 10000
##  3              2 25310
##  4              3 42062
##  5              4 52263
##  6              5 52387
##  7              6 44320
##  8              7 31408
##  9              8 19621
## 10              9 11131
## 11             10  5365
## 12             11  2449
## 13             12  1052
## 14             13   392
## 15             14   134
## 16             15    35
## 17             16    13
## 18             17     4
## 19             18     1
```

```
hiv_simul %>%
  group_by(n_of_mutations) %>%
  count() %>%
  ggplot(aes(x = n_of_mutations, y = n)) +
  geom_col(fill = "lavender", color = "black") +
  theme_classic() +
  theme(axis.title = element_blank())
```



Example—epitopes with ELISA

Actual data:

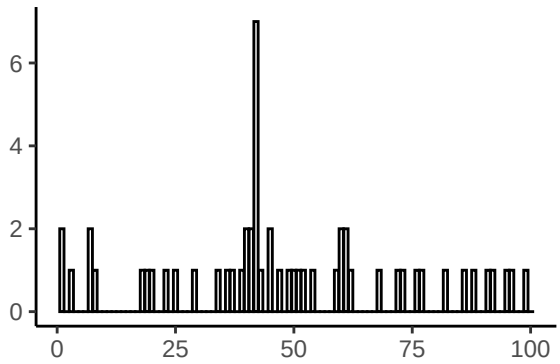
```
load("data/e100.RData")
e100
```

```
##   [1] 2 0 1 0 0 0 2 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 0 1 0 0 0 1 0 0 0 0 1 0 1 1
##  [38] 0 1 2 2 7 1 0 2 0 1 0 1 1 1 1 0 1 0 0 0 0 1 2 2 1 0 0 0 0 0 1 0 0 0 1 1 0
##  [75] 0 1 1 0 0 0 0 1 0 0 0 1 0 1 0 0 1 1 0 0 1 1 0 0 1 0
```

```
e100_tidy <- tibble(protein_position = 1:100,
                    n_pos_epitope_tests = e100)
head(e100_tidy)
```

```
## # A tibble: 6 x 2
##   protein_position n_pos_epitope_tests
##              <int>               <dbl>
## 1                1                   2
## 2                2                   0
## 3                3                   1
## 4                4                   0
## 5                5                   0
## 6                6                   0
```

```
e100_tidy %>%
  ggplot(aes(x = protein_position, y = n_pos_epitope_tests)) +
  geom_col(fill = "white", color = "black") +
  theme_classic() +
  theme(axis.title = element_blank())
```



Simulate data for 50 patient samples under the null distribution:

```
elisa_sim <- tibble(patient_id = 1:50) %>%
  mutate(test_results = map(patient_id, ~ rbinom(100, prob = 0.01, size = 1)))
elisa_sim %>%
  slice(1:3)
```

```
## # A tibble: 3 x 2
##   patient_id test_results
##        <int> <list>
## 1          1 <int [100]>
## 2          2 <int [100]>
## 3          3 <int [100]>
```

```
elisa_sim$test_results[[1]]
```

```
##   [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
##  [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Unnest these results and arrange by position number along the protein:

```
elisa_sim <- elisa_sim %>%
  unnest(test_results) %>%
  mutate(protein_position = rep(1:100, times = 50))
elisa_sim %>%
  slice(1:10)
```

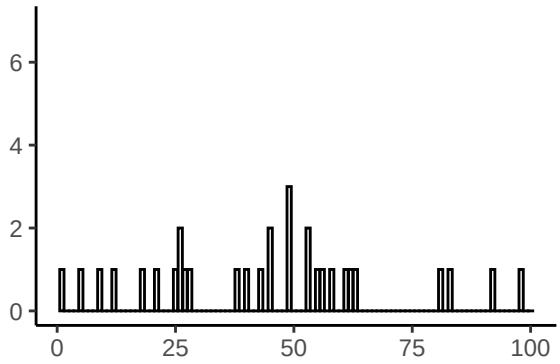
```
## # A tibble: 10 x 3
##    patient_id test_results protein_position
##         <int>        <int>            <int>
##  1          1            0                1
##  2          1            0                2
##  3          1            0                3
##  4          1            0                4
##  5          1            0                5
##  6          1            0                6
##  7          1            0                7
##  8          1            0                8
##  9          1            0                9
## 10          1            0               10
```

Get the positive test counts by position and plot a bar chart with those numbers:

```
positives_by_position <- elisa_sim %>%
  group_by(protein_position) %>%
  summarize(n = sum(test_results))
positives_by_position %>%
  slice(1:10)
```

```
## # A tibble: 10 x 2
##    protein_position     n
##               <int> <int>
##  1                1     1
##  2                2     0
##  3                3     0
##  4                4     0
##  5                5     1
##  6                6     0
##  7                7     0
##  8                8     0
##  9                9     1
## 10               10     0
```

```
ggplot(positives_by_position, aes(x = protein_position, y = n)) +
  geom_col(fill = "white", color = "black") +
  theme_classic() +
  theme(axis.title = element_blank()) +
  ylim(c(0, 7))
```

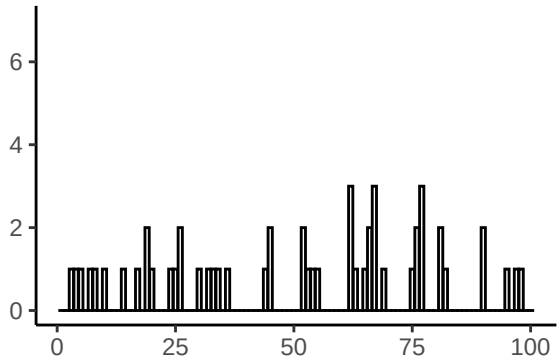


An even easier approach:

```
positives_by_position <- tibble(protein_position = 1:100,
                                n_positive = rbinom(100, prob = 0.01, size = 50))
positives_by_position %>%
  slice(1:10)
```

```
## # A tibble: 10 x 2
##    protein_position n_positive
##               <int>      <int>
##  1                1          0
##  2                2          0
##  3                3          1
##  4                4          1
##  5                5          1
##  6                6          0
##  7                7          1
##  8                8          1
##  9                9          0
## 10               10          1
```

```
ggplot(positives_by_position, aes(x = protein_position, y = n_positive)) +
  geom_col(fill = "white", color = "black") +
  theme_classic() +
  theme(axis.title = element_blank()) +
  ylim(c(0, 7))
```



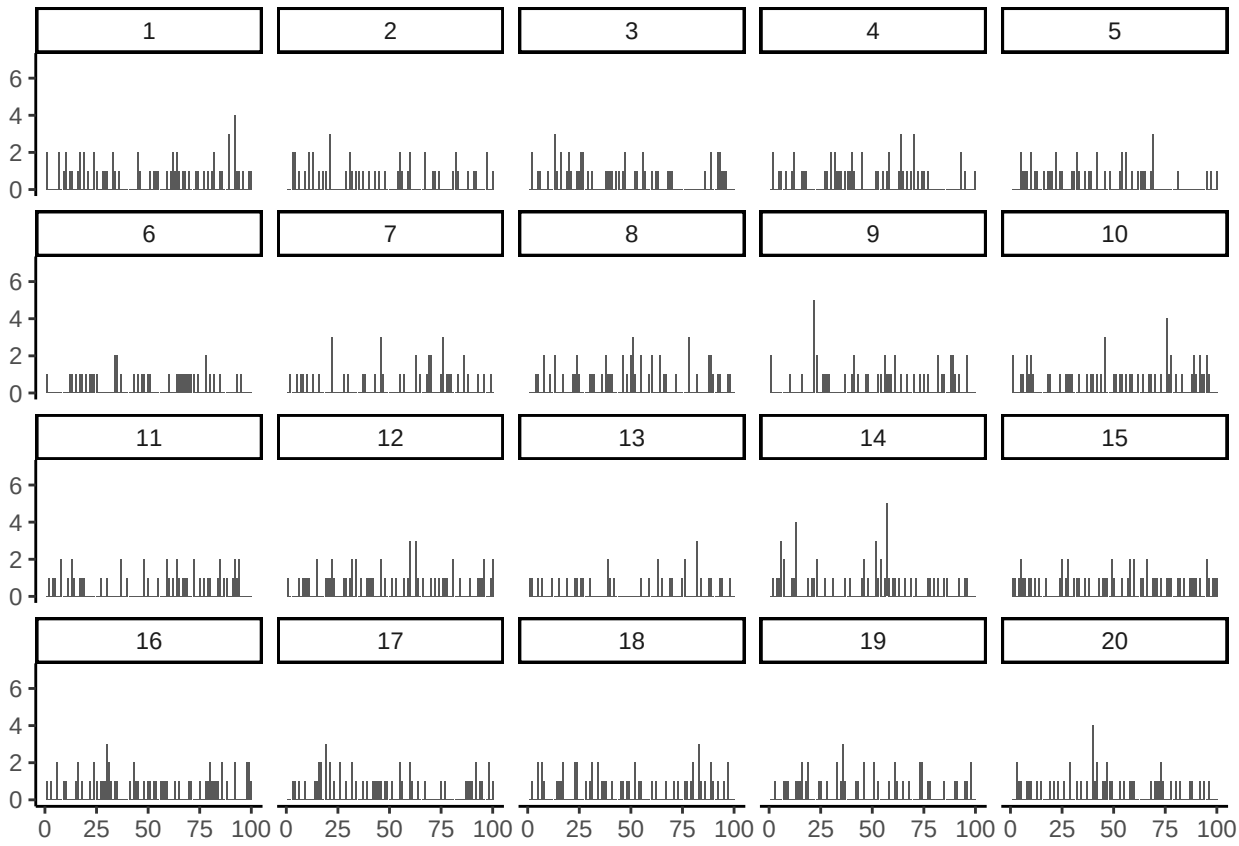
Lots of simulations:

```
many_elisa_sims <- tibble(sim_index = 1:20) %>%
  mutate(n_positive_by_position = map(sim_index,
                                      ~ rbinom(100, prob = 0.01, size = 50))) %>%
  unnest(n_positive_by_position) %>%
```

```
  mutate(protein_position = rep(1:100, times = 20))
many_elisa_sims %>%
  slice(1:10)
```

```
## # A tibble: 10 x 3
##    sim_index n_positive_by_position protein_position
##        <int>                  <int>            <int>
##  1         1                      2                1
##  2         1                      0                2
##  3         1                      0                3
##  4         1                      0                4
##  5         1                      0                5
##  6         1                      0                6
##  7         1                      2                7
##  8         1                      0                8
##  9         1                      1                9
## 10         1                      2               10
```

```
ggplot(many_elisa_sims, aes(x = protein_position, y = n_positive_by_position)) +
  geom_col() +
  facet_wrap(~ sim_index) +
  theme_classic() +
  ylim(c(0, 7)) +
  theme(axis.title = element_blank())
```

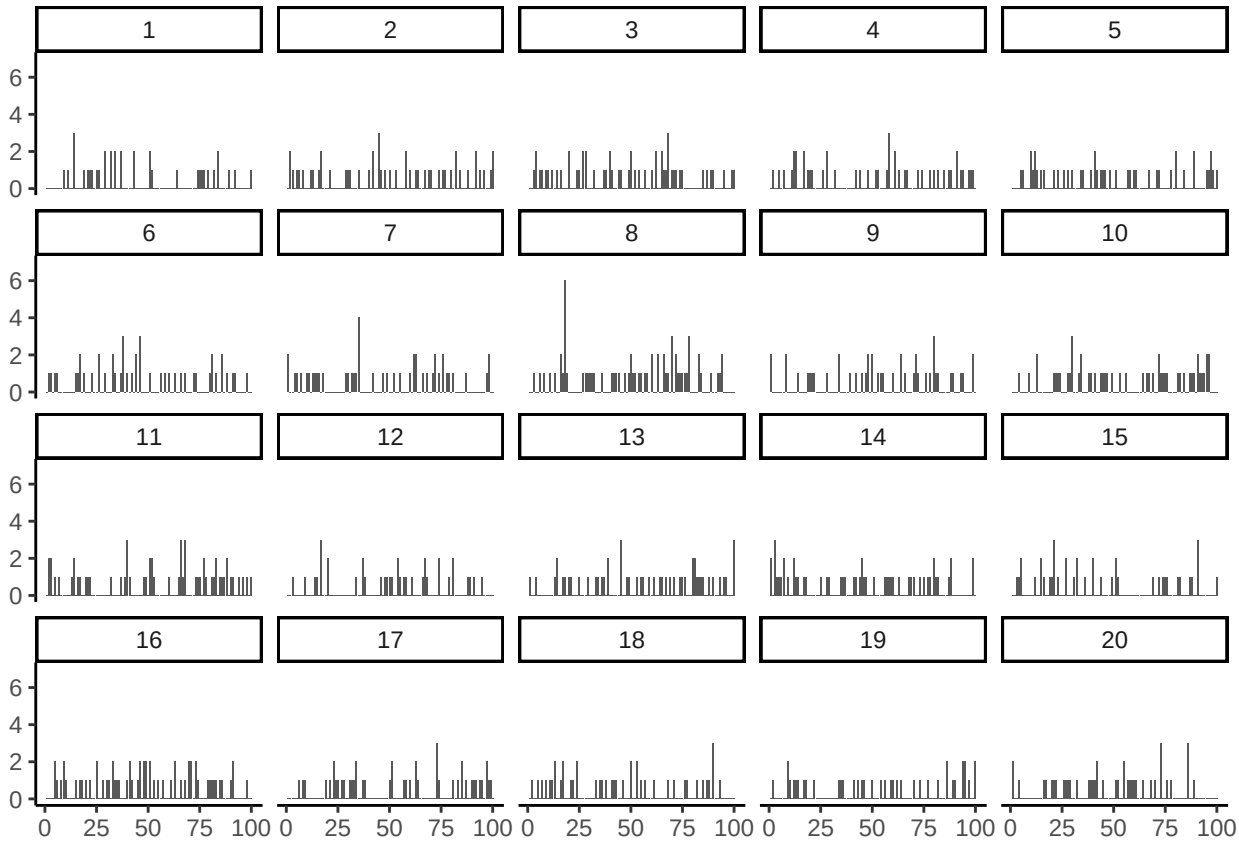


Same but with Poisson to simulate random values:

```r
many_elisa_sims <- tibble(sim_index = 1:20) %>%
  mutate(n_positive_by_position = map(sim_index,
                                      ~ rpois(100, lambda = 0.01 * 50))) %>%
  unnest(n_positive_by_position) %>%
  mutate(protein_position = rep(1:100, times = 20))
many_elisa_sims %>%
  slice(1:10)
```

```
## # A tibble: 10 x 3
##    sim_index n_positive_by_position protein_position
##        <int>                  <int>            <int>
##  1         1                      0                1
##  2         1                      0                2
##  3         1                      0                3
##  4         1                      0                4
##  5         1                      0                5
##  6         1                      0                6
##  7         1                      0                7
##  8         1                      0                8
##  9         1                      1                9
## 10         1                      0               10
```

```r
ggplot(many_elisa_sims, aes(x = protein_position, y = n_positive_by_position)) +
  geom_col() +
  facet_wrap(~ sim_index) +
  theme_classic() +
  ylim(c(0, 7)) +
  theme(axis.title = element_blank())
```



`forcats` alternatives

For factors with more than two levels, the `forcats` package has some nice alternatives to base R.

```
genotypes <- tibble(genotype = c("AA", "AO", "BB", "AO", "OO",
                                 "AO", "AA", "BO", "BO", "AO",
                                 "BB", "AO", "BO", "AB", "OO",
                                 "AB", "BB", "AO", "AO")) %>%
  mutate(genotype = as_factor(genotype))

genotypes %>%
  group_by(genotype) %>%
  count()
```

```
## # A tibble: 6 x 2
## # Groups:   genotype [6]
##   genotype     n
##   <fct>    <int>
## 1 AA           2
## 2 AO           7
## 3 BB           3
## 4 OO           2
## 5 BO           3
## 6 AB           2
```