Code

AMMS Practical: Drug Resistance

Andres Aranda-Diaz, Lucy Okell, Bob Verity

2022-08-04

Dependencies for Practical

Please copy and paste the below code chunk in it's entirety to your console to download R package libraries needed for this practical. If you are having trouble installing any of the R packages, please ask an instructor for a pre-loaded flash drive.

Hide

```
if (!("tidyverse" %in% installed.packages())) {
  install.packages("tidyverse", dependencies = TRUE)
}
```

Now load all of those libraries into this session using the code chunk below. Please copy and paste it in its entirety.

Hide

```
library(tidyverse)
```

Intro to drug resistance

Drug resistance is one of the primary use-cases of malaria molecular surveillance. By quantifying the frequency of mutations that are known to confer resistance to commonly used drugs, we obtain an effective early warning system that can be followed up with therapeutic efficacy studies (TES) to establish risk of clinical failure. Drug resistance data can be quite simple, consisting of a numerator and a denominator that can used to estimate prevalence, but there are also some complexities with combinations of mutations and drug resistant haplotypes that we also need to explore.

Overview of Data

In this practical we will use a real-world dataset consisting of a large molecular inversion probe (MIP) dataset taken from DRC and surrounding countries, previously analysed by Verity et al. 2020 (https://pubmed.ncbi.nlm.nih.gov/32355199/). We will be using a simplified version of this dataset focusing on just three mutations in the *dhps* gene.

Practical Goals

By the end of this practical, you should be able to:

- Import data to RStudio
- Describe a genetic data set
- Apply tidyverse functions to summarize data
- Calculate prevalence from sample-based data
- Visualize prevalence data
- Construct a 95% confidence interval
- Visualize haplotype data

Loading and summarizing genetic data

Overview of data

Genetic data can be aggregated or on a per-sample basis. Information contained in a simple per-sample data set should include a sample identifier, locus information (position or some other identifier, reference and alternate sequence) and the abundance of reference and alternate sequences in that locus. Metadata can also be included. For samples we may want to know when and were they were collected, as well as any other relevant information. For loci, we may want to know, for example, the amino acid sequences that the codons code for.

In this practical, we will use a data set collected in the Democratic Republic of the Congo in 20?. We have simplified the original data set for instructional purposes. We will be looking at resistance-related mutations in the dhps gene.

Getting familiar with data

Let’s load the data set:

Hide

```
dr.data <- readRDS("data/DRC_DR2.rds")
```

Can you see dr.data in your Environment?

To get familiar with the data set, let’s answer the following questions:

Q1. How many rows and columns are there in dr.data? What are the column names?

Click For Answer

A1.

Hide

```
ncol(dr.data)
```

```
## [1] 13
```

Hide

```
colnames(dr.data)
```

```
##  [1] "SAMPLE_ID" "REGION"    "CHROM"     "POS"       "REF"       "ALT"
##  [7] "GENE_NAME" "CODON_NUM" "CODON"     "CODON_POS" "REF_AA"    "ALT_AA"
## [13] "REF_WSAF"
```

Hide

```
nrow(dr.data)
```

```
## [1] 1617
```

We can see that we have 13 columns and 1617 rows. The columns contains a sample identifier (SAMPLE_ID) and sample metadata (REGION), as well as locus information (the rest of the columns). The last column is called REF_WSAF and it describes the proportion of reference sequences in a given locus and sample.

Now let’s take a quick look at dr.data. Because dr.data has 1617 rows, it is too long to display, let’s only display the first 10 rows

Hide

```
dr.data[1:10,]
```

```
##    SAMPLE_ID REGION CHROM    POS REF ALT GENE_NAME CODON_NUM CODON CODON_POS
## 1  1025N6S5R   EAST     8 549685   G   C      dhps       437   GGT         1
## 2  1025N6S5R   EAST     8 549993   A   G      dhps       540   AAA         0
## 3  1025N6S5R   EAST     8 550117   C   G      dhps       581   GCG         1
## 4  1032V2A9H   EAST     8 549685   G   C      dhps       437   GGT         1
## 5  1032V2A9H   EAST     8 549993   A   G      dhps       540   AAA         0
## 6  1032V2A9H   EAST     8 550117   C   G      dhps       581   GCG         1
## 7  1145J7T1L   WEST     8 549685   G   C      dhps       437   GGT         1
## 8  1145J7T1L   WEST     8 549993   A   G      dhps       540   AAA         0
## 9  1145J7T1L   WEST     8 550117   C   G      dhps       581   GCG         1
## 10 1152C2M8Y   EAST     8 549685   G   C      dhps       437   GGT         1
##    REF_AA ALT_AA  REF_WSAF
## 1       A      G 0.0000000
## 2       K      E 1.0000000
## 3       A      G 1.0000000
## 4       A      G 0.6666667
## 5       K      E        NA
## 6       A      G        NA
## 7       A      G 0.2058824
## 8       K      E 1.0000000
## 9       A      G 1.0000000
## 10      A      G 1.0000000
```

As you may be able to tell, each sample has multiple rows. This likely means that our data is in a long format, where the information for multiple loci for a sample is contained in different rows, instead of columns. Also, there are some rows that show NA values in REF_WSAF. That means that in that sample, there is no information in that locus.

Q2. Why would we want to store data in a long format?

Click For Answer

A2. Long format is useful when storing irregular or missing values, and is generally considered more efficient. One major advantage for our purposes is that it allows us to use `ggplot2` functions, which expect data in long format as input.

Now, let's get more insights on what's contained in dr.data by answering the following questions:

How many codons are we looking at? How many unique samples are there? How many samples per region are there?

To summarize data, tools from the tidyverse library are very useful. Some of those tools include the `distinct`, `unique`, `select`, `group_by` and `summarise` functions. To get more information on a function, access the documentation using ?function. Copy `?distinct` in your console.

Let's use the `unique` and `length` functions to identify how many samples there are in dr.data

Hide

```
# get unique IDs
samples <- unique(dr.data$SAMPLE_ID)

# calculate number of such IDs
length(samples)
```

```
## [1] 539
```

Let's use the distinct function to identify how many and which codons are contained in dr.data.

Hide

```
# First, let's select relevant columns for this question: GENE_NAME, CODON_NUM
dr.data.loci <- dr.data[,c("GENE_NAME","CODON_NUM")]

# Next, lets select distinct rows
dr.data.loci <- distinct(dr.data.loci)

# let's display the result
dr.data.loci
```

```
##   GENE_NAME CODON_NUM
## 1      dhps       437
## 2      dhps       540
## 3      dhps       581
```

We can see that there are 3 unique codons for gene *dhps* in the data set.

We performed two actions on dr.data: (1) we selected the relevant columns and (2) we selected only distinct rows from the data frame. We can also use pipes to perform sequential actions on an object. The following code performs exactly the same operations:

Hide

```
dr.data.loci <- dr.data %>%
  select(GENE_NAME,CODON_NUM) %>%
  distinct()

dr.data.loci
```

```
##   GENE_NAME CODON_NUM
## 1      dhps       437
## 2      dhps       540
## 3      dhps       581
```

Now let's use pipes and the `select`, `distinct`, `group_by` and `summarise` functions to get a count of samples per REGION

Hide

```
dr.data.per.region <- dr.data %>%
  select(SAMPLE_ID,REGION) %>%
  distinct() %>%
  group_by(REGION) %>%
  summarise(n = n())

dr.data.per.region
```

```
## # A tibble: 2 × 2
##   REGION     n
##   <chr>  <int>
## 1 EAST     296
## 2 WEST     243
```

Estimating prevalence

In the next sections, you will be comparing results with other members of your group. Each of you will focus on one codon: Participant 1: 437 Participant 2: 540 Participant 3: 581

For drug resistance, the prevalence of a mutant is defined as the proportion of observations in a population that contain a mutation. Because malaria infections can be polyclonal, an individual can be infected with a mixture of strains that have the wild type or mutant alleles. Thus, we need to define how we will count observations.

One way of doing this is by counting the prevalence of wild type, mutant and mixed infections. Alternatively, we can count each parasite as an observation (meaning that a polyclonal infection will count as multiple observations).

We will use the latter definition. Let's first calculate the overall prevalence of drug resistant mutants in the country as a whole.

As we explained above, each sample has one row for each locus, and the `REF_WSAF` variable describes the proportion of reference (wild type) observations within a sample. `REF_WSAF = 1` means that the sample is exclusively wild type and `REF_WSAF = 0` means that it's exclusively mutant. Any value between 0 and 1 indicates that there is a mixture of both alleles in the sample. Thus, we can count the number of samples containing a wild type observation as those with `REF_WSAF > 0`, and those containing a mutant observation with `REF_WSAF < 1`.

First, let's filter the data to contain the rows corresponding to your assigned codon and create 2 variables that denote whether the wild type and mutant alleles are present in the sample. In the following code, please change the assigned_codon to your corresponding codon

Hide

```
assigned_codon <- 540
dr.data.filtered <- dr.data %>%
  filter(CODON_NUM == assigned_codon) %>%
  mutate(REF_ALLELE = REF_WSAF > 0) %>%
  mutate(ALT_ALLELE = REF_WSAF < 1)
```

Let's calculate the total number of wild type observations

Hide

```
sum(dr.data.filtered$REF_ALLELE)
```

```
## [1] NA
```

The value is `NA` because, as we explained above, some of the samples have no information for this locus, and most operations will return an `NA` if at least 1 value is `NA`. We then, have to specify that we want to remove NAs:

Hide

```
sum(dr.data.filtered$REF_ALLELE, na.rm = TRUE)
```

```
## [1] 335
```

Similarly, we can calculate the number of mutant observations:

Hide

```
sum(dr.data.filtered$ALT_ALLELE, na.rm = TRUE)
```

```
## [1] 185
```

Q3. What is the prevalence of mutants in your codon? How does it compare to the other markers within your group?

Click For Answer

A3.

Hide

```
total_ALT <- sum(dr.data.filtered$ALT_ALLELE, na.rm = TRUE)
total_REF <- sum(dr.data.filtered$REF_ALLELE, na.rm = TRUE)

PREVALENCE <- total_ALT/(total_ALT + total_REF)
PREVALENCE
```

```
## [1] 0.3557692
```

Now we will perform a regional analysis.

Q4. What is the prevalence of mutants in your codon in the two regions? How does it compare to the other markers within your group?

Click For Answer

A4.

Hide

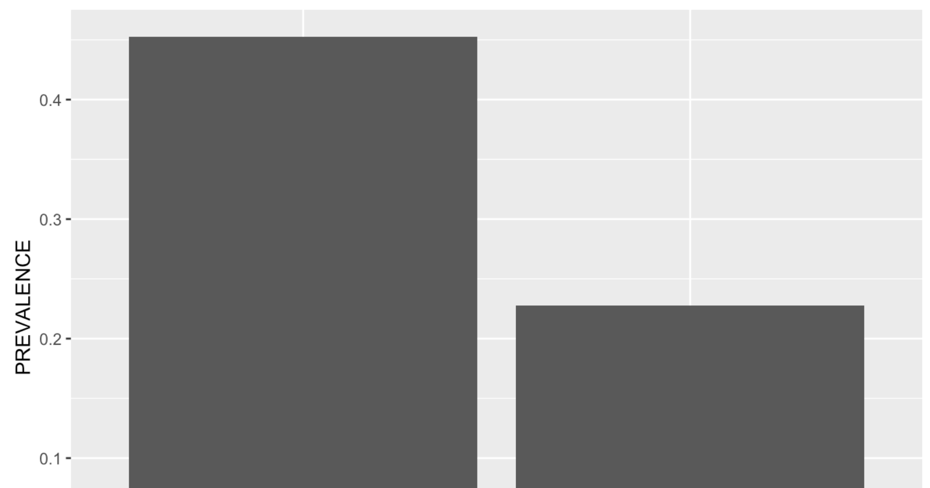
```
dr.prevalence.region <- dr.data.filtered %>%
  group_by(REGION) %>%
  summarise(total_ALT = sum(ALT_ALLELE, na.rm = TRUE),
            total_REF = sum(REF_ALLELE, na.rm = TRUE),
            PREVALENCE = total_ALT / (total_ALT + total_REF))
dr.prevalence.region
```

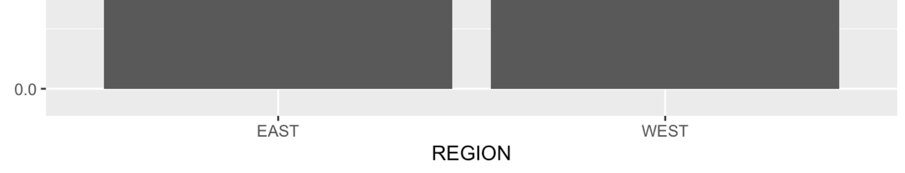
```
## # A tibble: 2 × 4
##   REGION total_ALT total_REF PREVALENCE
##   <chr>      <int>     <int>      <dbl>
## 1 EAST         134       162      0.453
## 2 WEST          51       173      0.228
```

Now, let's also make a bargraph of your data:

Hide

```
ggplot(data = dr.prevalence.region,
       aes(x = REGION, y = PREVALENCE))+
  geom_bar(stat="identity",position="dodge")
```





Q5. Where is prevalence higher, east or west?

Click For Answer

A5. This may vary depending on your codon, but based on the bargraph above (for codon 540) it appears that prevalence is highest in the east.

Calculate confidence intervals

Every prevalence estimate has some uncertainty. Here, you will calculate the confidence interval using Wald's interval formula.

The lower bound is calculated as: $p - z\sqrt{\frac{p(1-p)}{n}}$

The upper bound is calculated as: $p + z\sqrt{\frac{p(1-p)}{n}}$

Thus, we need to know the prevalence (p), sample size (n) and the level of confidence we want. For this practical, we will calculate confidence levels at 95%, which means that $z = 1.96$.

Let's first calculate the confidence interval for the country as a whole:

Hide

```
total_ALT <- sum(dr.data.filtered$ALT_ALLELE, na.rm = TRUE)
total_REF <- sum(dr.data.filtered$REF_ALLELE, na.rm = TRUE)

N <- total_ALT + total_REF
PREVALENCE <- total_ALT / N

CI.LOWER <- PREVALENCE - 1.96 * sqrt(PREVALENCE * (1 - PREVALENCE) / N)
CI.UPPER <- PREVALENCE + 1.96 * sqrt(PREVALENCE * (1 - PREVALENCE) / N)

c(PREVALENCE, CI.LOWER, CI.UPPER)
```

```
## [1] 0.3557692 0.3146202 0.3969182
```

Q6. Does your response to the question "What codon has a higher prevalence?" change now that you have seen confident intervals?

Click For Answer

A6. We find that CIs for codon 540 and codon 437 are overlapping. This means it is difficult to say with confidence which of these is at higher prevalence. Codon 581 is lower and non-overlapping, so we can be confidence that this is indeed at the lowest prevalence.

Now let's calculate the confidence intervals for the prevalence in each region, which will be easier with tidyverse tools:

Hide

```
dr.prevalence.region <- dr.data.filtered %>%
  group_by(REGION) %>%
  summarise(total_ALT = sum(ALT_ALLELE, na.rm = TRUE),
            total_REF = sum(REF_ALLELE, na.rm = TRUE),
            N = total_ALT + total_REF,
            PREVALENCE = total_ALT / N) %>%
  mutate(ci.lower = PREVALENCE - 1.96 * sqrt(PREVALENCE * (1 - PREVALENCE) / N),
         ci.upper = PREVALENCE + 1.96 * sqrt(PREVALENCE * (1 - PREVALENCE) / N))

dr.prevalence.region
```

```
## # A tibble: 2 × 7
##   REGION total_ALT total_REF     N PREVALENCE ci.lower ci.upper
##   <chr>      <int>     <int> <int>      <dbl>    <dbl>    <dbl>
## 1 EAST         134       162   296      0.453    0.396    0.509
## 2 WEST          51       173   224      0.228    0.173    0.283
```

Q7. Does your response to the question “In which region is there higher prevalence?” change now that you have seen confidence intervals? What about the other codons in your group?

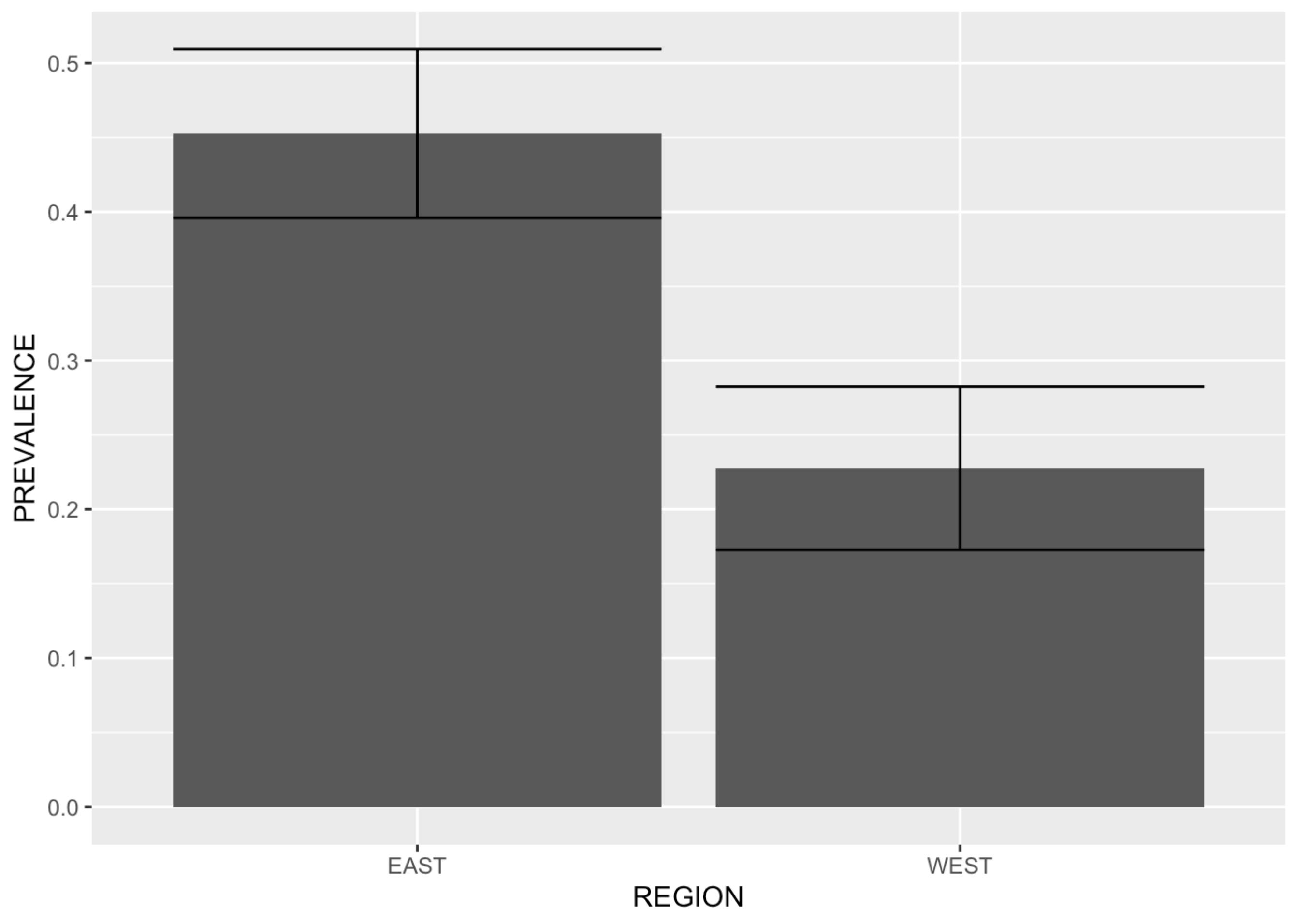
Click For Answer

A7. This may vary depending on your codon, but based on the bargraph above (for codon 540) it appears that confidence intervals are non-overlapping. This means it is likely that prevalence is really higher in the east, and this is not just a sampling effect.

Now let’s add that new information to our visualization:

Hide

```
ggplot(data = dr.prevalence.region,
       aes(x = REGION, y = PREVALENCE, ymin = ci.lower, ymax = ci.upper)) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_errorbar()
```



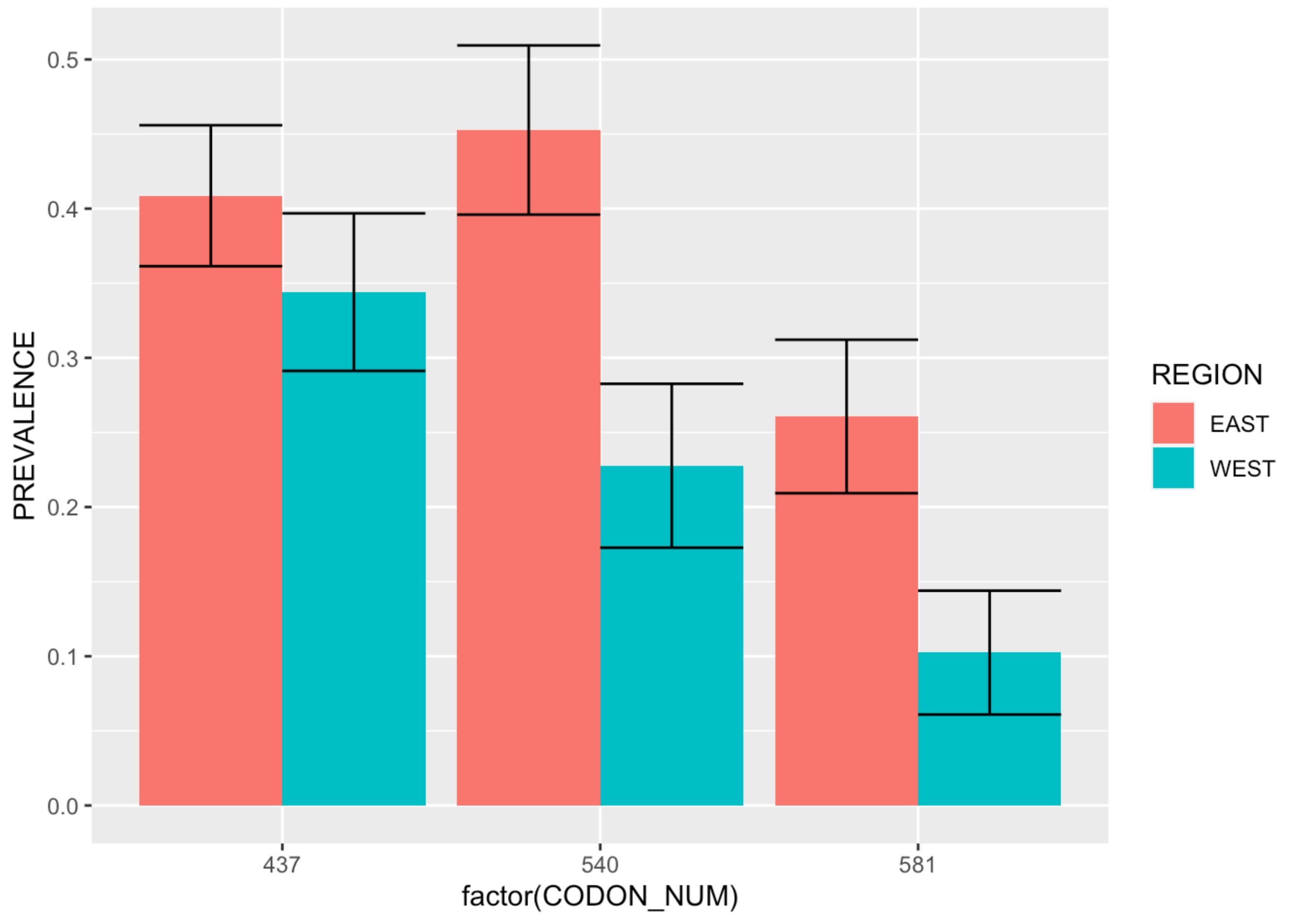
Integrating data from multiple codons

Now that each of you has worked on a codon independently, it’s time to put all of them together in one figure:

Hide

```
dr.data.grouped.wald <- dr.data %>%
  group_by(CODON_NUM, REGION) %>%
  summarise(total_ALT = sum(REF_WSAF < 1, na.rm = TRUE),
            total_REF = sum(REF_WSAF > 0, na.rm = TRUE),
            N = total_ALT + total_REF,
            PREVALENCE = total_ALT / N) %>%
  mutate(ci.lower = PREVALENCE - 1.96 * sqrt(PREVALENCE * (1 - PREVALENCE) / N),
         ci.upper = PREVALENCE + 1.96 * sqrt(PREVALENCE * (1 - PREVALENCE) / N))

ggplot(data = dr.data.grouped.wald,
       aes (fill = REGION,
            y = PREVALENCE,
            x = factor(CODON_NUM),
            ymin = ci.lower,
            ymax = ci.upper)) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_errorbar(position = "dodge")
```



Visualizing haplotypes

As we discussed in the lecture, for some drugs the combination of alleles (haplotypes) is the most relevant information. Next, we will use a packge, `UpSetR`, to visualize combinations of mutations.Documentation on the package can be found here (https://cran.r-project.org/web/packages/UpSetR/UpSetR.pdf).

Polyclonal data presents a challenge for constructing haplotypes. For example, if a sample contains 2 alleles at each of 2 loci, how can we know which allele goes with which in each of the parasites in that infection? This issue is called **phasing**, and we are often working with **unphased** data.

To simplify our data set, we will remove samples with `NAs` and consider only the most abundant allele in each sample:

Hide

```
# identify NAs
dr.data.NA <- dr.data %>%
  group_by(SAMPLE_ID) %>%
  summarise(na = !any(is.na(REF_WSAF)))

# simplify by rounding within-sample allele frequencies to 0 or 1
dr.data.simplified <- dr.data %>%
  mutate(REF_WSAF_ROUND = round(REF_WSAF)) %>%
  filter(SAMPLE_ID %in% (dr.data.NA %>% filter(na))$SAMPLE_ID)

# count the number of each haplotype
dr.data.simplified %>%
  select(SAMPLE_ID, CODON_NUM, REF_WSAF_ROUND) %>%
  pivot_wider(names_from = CODON_NUM , values_from = REF_WSAF_ROUND) %>%
  mutate(haplotype_437_540_581 = paste(`437`,`540`,`581`)) %>%
  group_by(haplotype_437_540_581) %>%
  summarize(n = n())
```

```
## # A tibble: 6 × 2
##   haplotype_437_540_581     n
##   <chr>                 <int>
## 1 0 0 0                     1
## 2 0 0 1                     1
## 3 0 1 1                   126
## 4 1 0 0                    44
## 5 1 0 1                    76
## 6 1 1 1                   162
```

We can see that some combinations of haplotypes are more common than others. For example, the triple mutant is very common, as is the double 540+581 mutant.