STAT 145 Examples, Unit 1 Day 2

T.Scofield

Sampling using software

We mentioned two unbiased ways of sampling from a population:

- simple random sample (SRS)
- independent and identically distributed (iid) sample

Sampling without replacement (SRS)

Say you intend to call home on two randomly-selected days of the week. You can model this on the idea that you have each day written on a slip of paper, and you mix them up in a hat before drawing two of them out, presumably without replacement. This would be an SRS with sample size $n = 2$, simulated using the `sample()` command:

```
hatContents <- c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")
sample(hatContents, size=2)
```

```
[1] "Sunday"   "Saturday"
```

You might try out these commands, too, so as to be able to make sense of variants.

```
sample(hatContents, size=5)
sample(hatContents, size=15)
sample(hatContents)     # what happens when there is no size specification?
```

Sampling with replacement (iid sample)

A fair die is a sort of population, one that can produce the numbers 1 through 6 with equal likelihood. Say I want to sample 24 times *with replacement* from this population. That is, I wish to take an iid random sample of size $n = 24$. That is achieved via commands like this. Note that `resample()` behaves very much like sample, except it *replaces* each *draw* before *drawing* again.

```
die = 1:6
resample(die, size=24)
```

```
[1] 6 4 3 5 2 6 4 2 6 4 6 3 6 2 3 5 2 5 4 3 4 4 3 5
```

You might imagine doing similar steps to simulate 10 flips of a coin. To demonstrate another feature, suppose you want your coin to behave not like a fair coin, but one that is twice as likely to come up "heads" as it is "tails". Here is one way to make that happen:

```
coin = c( rep("H", 2), "T" )
resample(coin, size=10)
```

```
[1] "H" "H" "H" "H" "T" "T" "H" "H" "H" "T"
```

Or, this also achieves the same sort of behavior:

```
resample( c("H", "T"), prob=c(2/3, 1/3), size=10 )
```

```
[1] "H" "H" "H" "H" "T" "T" "H" "H" "H" "T"
```

Looking for the appearance of an association between variables

Two categorical variables

A two-way table is a bivariate display, breaking down frequencies (or relative frequencies) simultaneously in two categorical variables. The `HELPrct` data frame in the `mosaicData` contains data on clinical trials from inpatients at a detoxification unit. Here are several commands that construct a two way table on variables `sex` and `substance`, a possible first step in investigating whether there is an association between the variables. Try each of them.

```
tally(~ substance | sex, data=HELPrct)
```

Since the number of women in the data set is quite different than the number of men, it might suit us better to display **relative frequencies**. Note the above command has relative frequencies on each of the two rows that add to 100 percent.

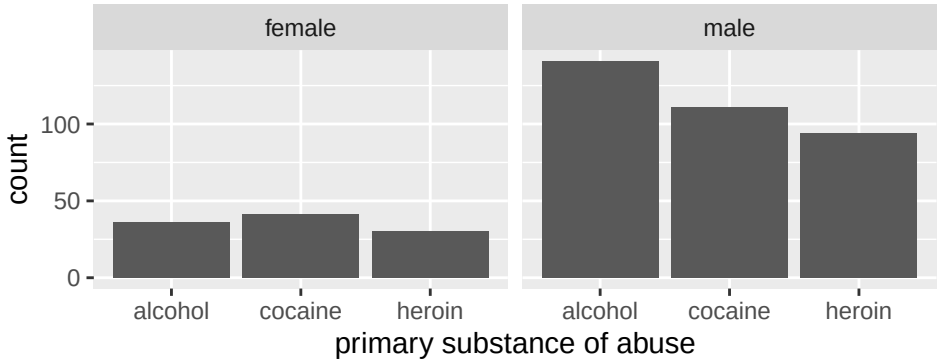
```
props(~ substance | sex, data=HELPrct)
```

```
   response    sex prop_alcohol prop_cocaine prop_heroin
1 substance female    0.3364486    0.3831776   0.2803738
2 substance   male    0.4075145    0.3208092   0.2716763
```

It seems there is an apparent association between `sex` and `substance`, as a smaller percentage (about 33.6%) of women abuse alcohol than the percentage of men (about 40.8%); women also seem more likely to abuse cocaine than men.

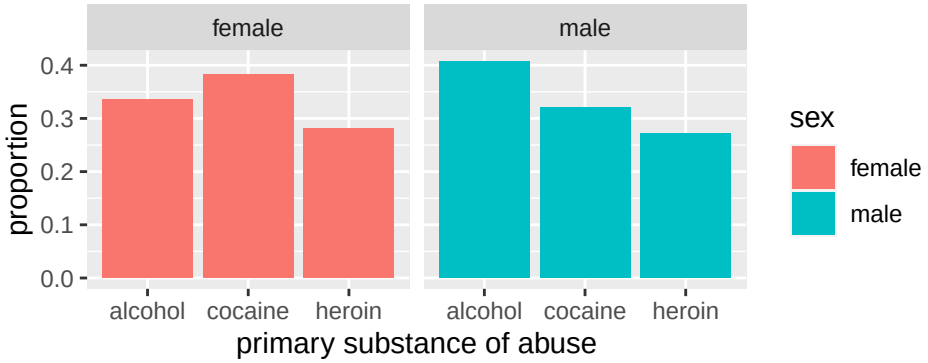
Next, I replace `tally()` with `gf_bar()`, using the same syntax as above.

```
gf_bar(~ substance | sex, data=HELPrct)
```



As before, frequencies (what was provided in the plot above) do not reveal as much as relative frequencies:

```
gf_props(~ substance | sex, data=HELPrct, fill=~sex)
```



One quantitative variable, one categorical

Loading the `palmerpenguins` package gives access to the `penguins` data frame.

```
require(palmerpenguins)
```

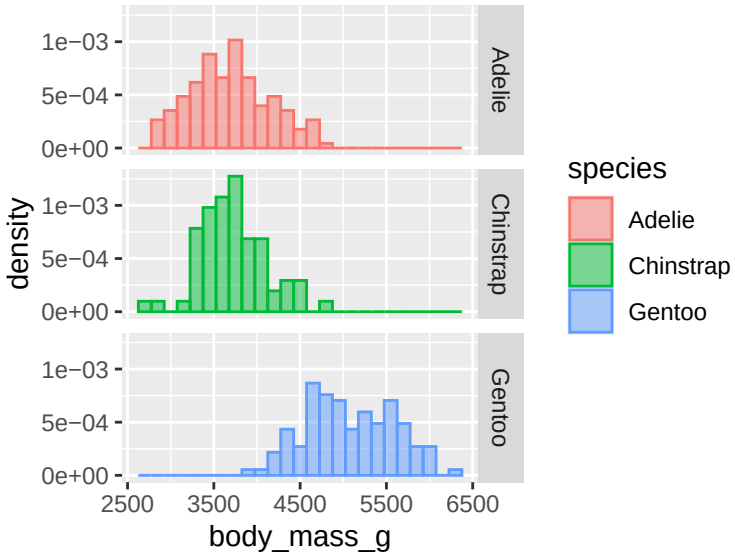
```
Loading required package: palmerpenguins
```

```
names(penguins)
```

```
[1] "species"           "island"            "bill_length_mm"
[4] "bill_depth_mm"     "flipper_length_mm" "body_mass_g"
[7] "sex"               "year"
```

A density histogram of `body_mass_g` might be more interesting if broken down by `species`.

```
gf_dhistogram(~body_mass_g | species~., color=~species, fill=~species, data=penguins)
```



Question: Why is a density histogram better than a mere histogram?

Two remarks:

- The label `body_mass_m` on the above histograms is somewhat opaque. Improve the labeling for plots from the `ggformula` suite of tools by piping your plot to `gf_labs()`. (The code here, which I have not executed, gives one example.)

```
gf_dhistogram(~body_mass_g | species~., color=~species, fill=~species, data=penguins) |>
  gf_labs(x = 'body mass (in grams)')
```

- Histograms are one of several ways we have learned to plot quantitative data. You might try `gf_boxplot()`, `gf_density()` or `gf_dotplot()` with a conditioning categorical variable to see how they adapt.