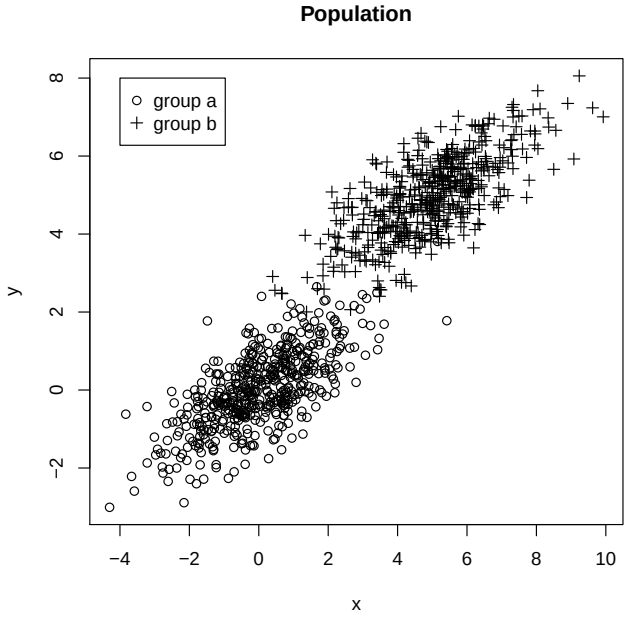
The survey Package in R

The `survey` package was written and is maintained by Thomas Lumley. It can do most of the things covered in 5201 and a lot more. The package website contains tutorials and is located at http://faculty.washington.edu/tlumley/survey/. Lumley has also written a book called **Complex Surveys: A guide to analysis using R.**, based on the `survey` package. UofM students have full online access through the library's website.

This handout will provide examples analyzing sample data from a simulated population using both simple random sampling and stratified random sampling. The simulated population is created in R with the following code.

```
> set.seed(1)
> y <- c(rnorm(500),rnorm(500,mean=5))
> x <- y + rnorm(1000)
> group <- c(rep('a',500),rep('b',500))
> pop <- data.frame(group,x,y)
```

In this population, there are 1000 units which have values for `y` `x`, and `group`. `y` is the characteristic of interest whose population mean and total we will estimate. It consists of 500 observations that have mean zero and correspond to `group a`, and another 500 observations that have mean five and correspond to `group b`. `group` will define strata when we use stratified random sampling. `x` is a fuzzed version of `y` that we will use to illustrate ratio estimation. Finally, here's a plot of the population.



Now, we will draw a simple random sample and a stratified random sample, both of size 100, by choosing some rows using the `sample` function in R.

```
> srs.rows <- sample(1000,100)
> srs <- pop[srs.rows,]
> str.rows <- c(sample(500,50),sample(501:1000,50))
> str <- pop[str.rows,]
```

Now, if we were using real (not simulated) data, the only information available would be one of these samples, either `srs` or `str`, and some auxiliary information like the population size, number of units in each stratum, the population total for `x`, etc. So, we'll finally load the `survey` package and illustrate it's use.

Simple random sampling

First, we will assume that `srs` is our sample, so we ignore `group`. Before we calculate any estimates, we have to give R information about the sample. In the `survey` package, this is done by creating an `svydesign` object.

```
> library(survey)
> fpc.srs <- rep(1000,100)
> des.srs <- svydesign(id=~1,strata=NULL,data=srs,fpc=fpc.srs)
```

Here, I have loaded the package, created a new variable called `fpc`srs., and created a `svydesign` object. Ignore `fpc`srs. for a moment and focus on the `svydesign` function. The first argument, `id`, is meant to tell the function how sampled units are grouped if cluster sampling is used. Since cluster sampling was not used here, we just have to put `~1` in as a place-holder. In the `strata` argument, we just tell R that there are no strata. The `data` argument says where to find the sample data. The `fpc` argument gives the function the necessary information to use a finite population correction in estimation. The supplied argument I have created, `fpc`srs., is to be a vector with the same number of rows as the sample data. The first element in `fpc`srs. corresponds to the first row (observation) in the sample data, and tells the function how many units were in the set from which this observation was drawn. In this case, each observation was drawn from the entire population of 1000 units, so `fpc`srs. is just a vector the repeats the number 1000.

Now that the `svydesign` object is created, we can produce estimates of the population mean or total of `y` based on it. We can also produce Normal-based confidence intervals.

```
> svytotal(~y, design=des.srs)

   total     SE
y 2601.4 254.82

> svymean(~y, design=des.srs)
```

```
    mean     SE
y 2.6014 0.2548
```

```
> confint(svytotal(~y,design=des.srs))
```

```
    2.5 %   97.5 %
y 2102.01 3100.876
```

The only arguments needed to these functions are the variable of interest and the `svydesign` object.

If we also know the population total of `x`, we can use ratio estimation to get an improved estimate of the population total of `y`. First, we have to estimate the ratio of `y`'s total to `x`'s total, and then we can predict `y`'s total based on this estimate.

```
> known.x.total <- sum(pop$x)
> (srs.ratio <- svyratio(numerator=~y,denominator=~x,design=des.srs))
```

```
Ratio estimator: svyratio.survey.design2(numerator = ~y, denominator = ~x, design = des.srs)
Ratios=
          x
y 0.9482768
SEs=
          x
y 0.0294105
```

```
> predict(srs.ratio,total=known.x.total)
```

```
$total
         x
y 2344.225

$se
         x
y 72.70539
```

The arguments to the `svyratio` function are fairly self-explanatory. The `predict` function, when given a `svyratio` object, also requires an argument for total of `x`. Note that, if we provided the population mean of `x` instead of the population total, we would get a ratio estimator of the population mean of `y`.

Stratified random sampling

Now, we will use the stratified random sample that was drawn, `str`, and construct a `svydesign` object for it.

```
> fpc.str <- rep(500,100)
> des.str <- svydesign(id=~1,strata=~group,data=str,fpc=fpc.str)
```

The `id` and `data` arguments are the same as before. Now, however, we recognize that `group` supplies the stratum membership of each observation, and the `fpc` argument has changed. In this case, each observation was selected from a stratum of size 500, so the `fpc` vector is simply the number 500 repeated. Estimation can proceed as before. Note that ratio estimation no longer produces a relative gain in precision.

```
> svytotal(~y,des.str)

   total     SE
y 2426.9 94.438

> predict(svyratio(numerator=~y,denominator=~x,design=des.str),
+         total=known.x.total)

$total
         x
y 2613.394

$se
         x
y 107.5571
```