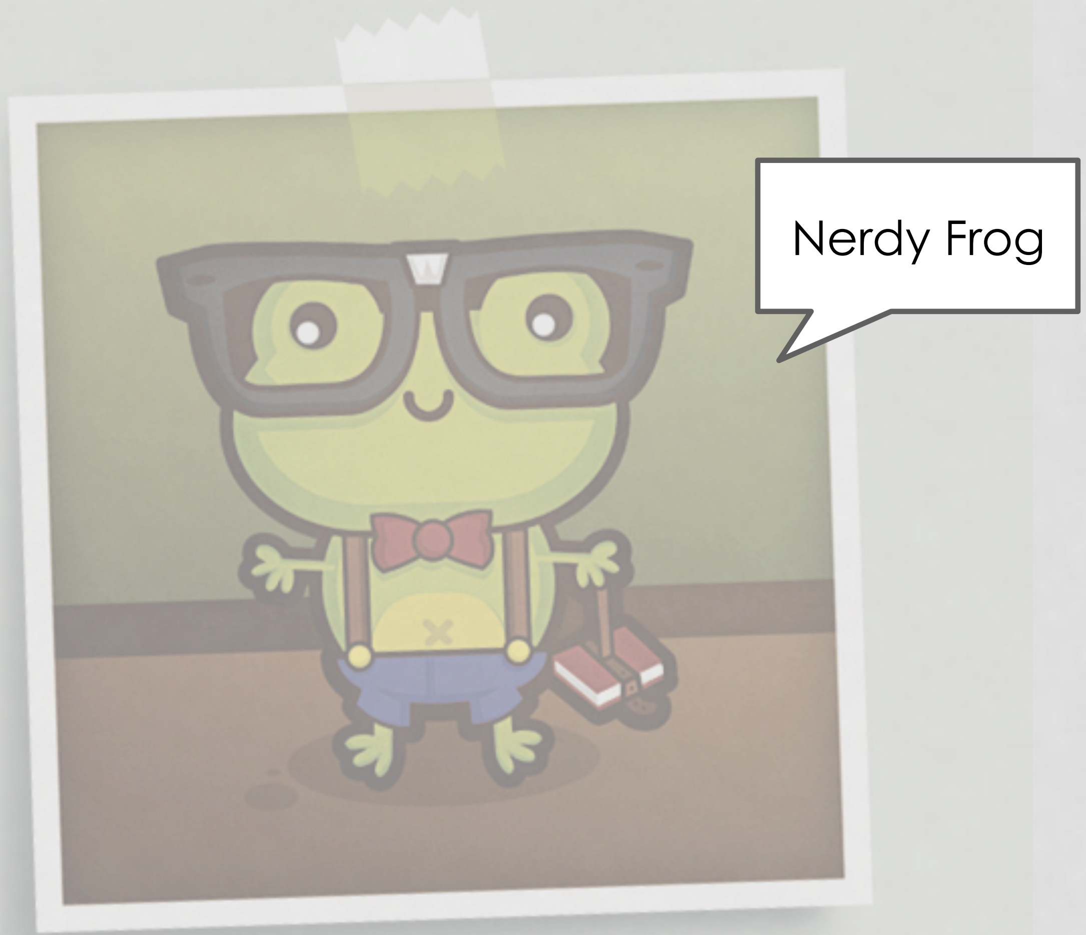
STRATIFIED SAMPLING



Team Frog
marion@lyatiss.com
sebastien@lyatiss.com

WHAT IS STRATIFIED SAMPLING?

Goal

- Sampling a population of N elements with a sampling rate R
- We'll end up with S = R*N samples

Random Sampling VS Stratified Sampling

1) Simple Random Sampling
Randomly pick your S elements

2) Stratified Sampling:
Pick your S elements according to the statistical distribution of the population

SIMPLE RANDOM SAMPLING (VIDEO)

Upside

- Fast & simple

Downside

- Fails to sample infrequent elements

- If the samples are used as a training set to build a classifier => poor classifier (infrequent elements / classes will be missing)

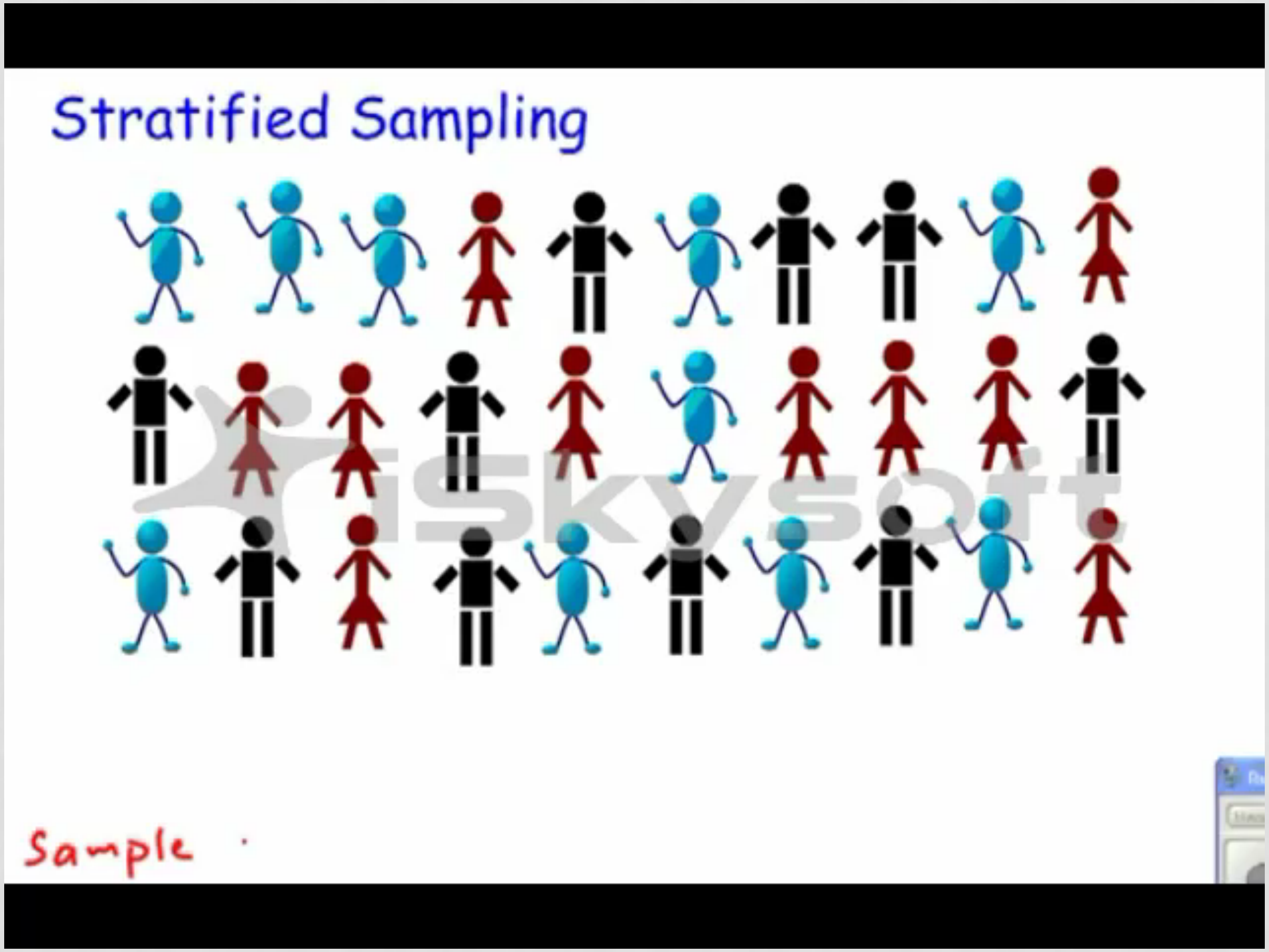
STRATIFIED SAMPLING (VIDEO)

Steps

- Create the Strata

- A strata is made of elements belonging to the same class

- Apply Simple Random sampling to each Stratum



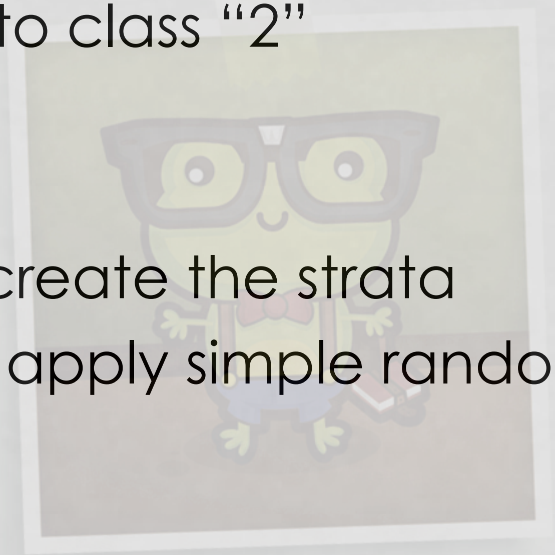
CODE

dataGen.py

- Generate a data set
- 99% of the instances belong to class "1"
- 1% of the instances belong to class "2"
- Stratified sampling is required otherwise we are likely to end up with no samples belonging to class "2"

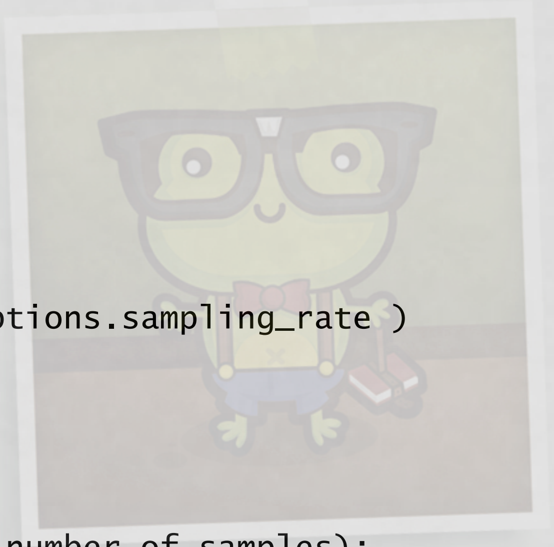
mrSratified.py:

- 1st set of *mapper* & *mapper_final* create the strata
- 2nd set of *reducer* & *reducer_final* apply simple random sampling to each stratum



CODE: CREATE THE STRATA

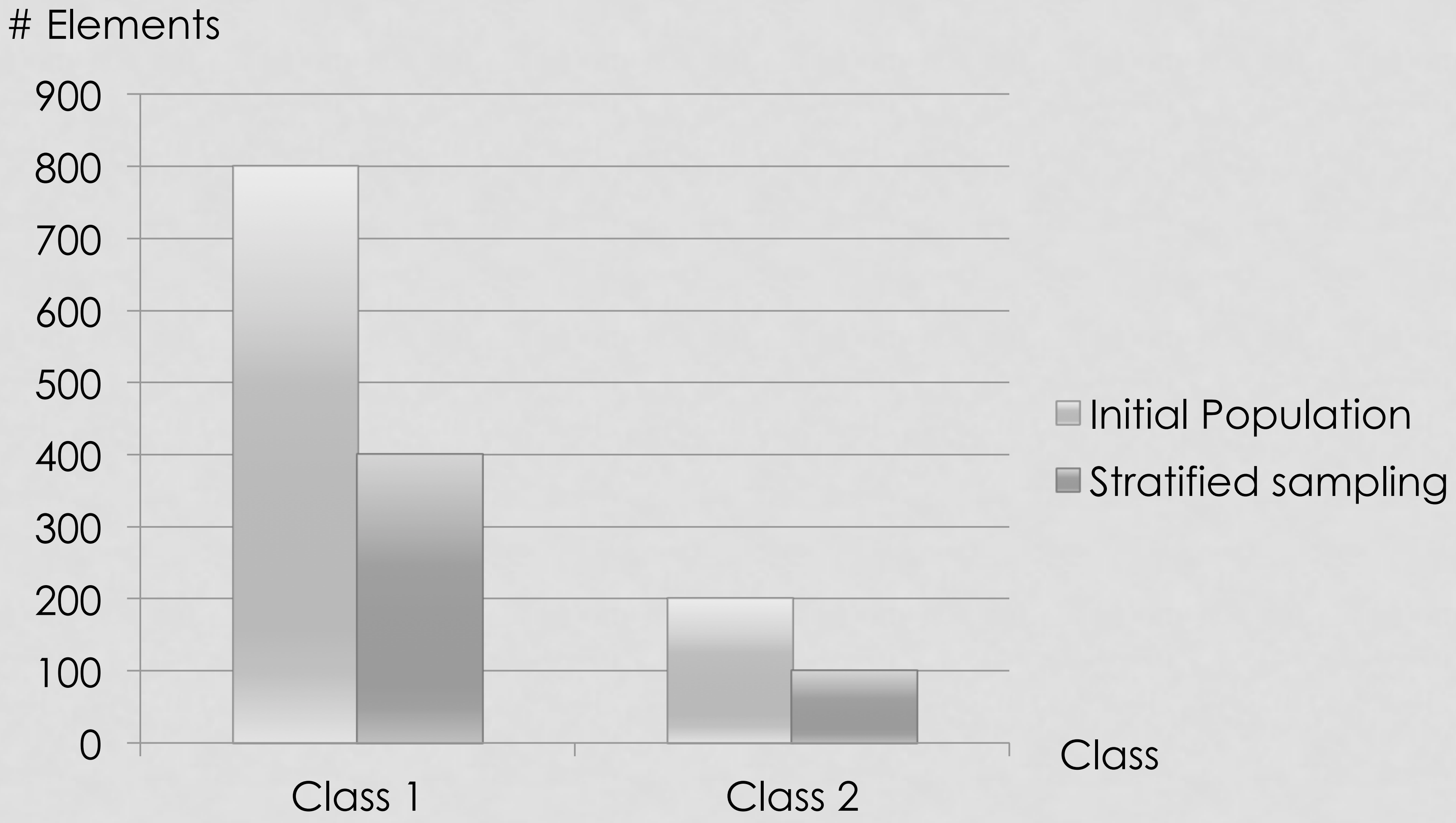
```
def mapper(self, key, line):

       # parse
       instance = json.loads(line)
       label = instance['class']

       try:
           self.strata[label].append(instance)
       except:
           self.strata[label] = []
           self.strata[label].append(instance)

   def mapper_final(self):

       for label in self.strata:
           stratum = self.strata[label]
           number_of_samples = int( len(stratum) * self.options.sampling_rate )

           if not stratum: # stratum should not be empty
     pass
           else:
              for random_sample in random.sample(stratum, number_of_samples):
                   yield label, random_sample
```



CODE: SAMPLE EACH STRATUM

```
def reducer(self, label, samples):

       try:
               self.output[label].extend(samples)
       except:
               self.output[label] = []
             self.output[label].extend(samples)

   def reducer_final(self):

       for label in self.output:
           stratum_samples = self.output[label]
           yield label, (len(stratum_samples), stratum_samples)
```

OUTPUT (R = 50%)



# Elements
900
800
700
600
500
400
300
200
100
0
Class 1
Class 2
Class
Initial Population
Stratified sampling

Q&A

Team Frog

Marion Le Borgne
marion@lyatiss.com

Sebastien Soudan
sebastien@lyatiss.com