COMP 5970/6970-004
Computational Biology: Genomics and Transcriptomics
Lecture notes 3: 1/20/2022

Haynes Heaton

Spring, 2022

Lecture Objectives

- Data analysis through graphing in ggplot2
- Review of geometric distribution
- Binomial distribution
- Poisson distribution

Data analysis and graphing with ggplot2

Data analysis and graphing is a core competency across the fields of science and engineering. Graphing data in different ways is a good way to get a better understanding of it and to find peculiarities that may be caused by either errors or something of interest in the data (true biological effect in our case).

I use many different tools for visualization, but for regular charts, ggplot2 is the best. There are implementations of ggplot in other popular languages, but the original package is written in R. For many applications, I do not recommend R due to its slow speed, but for interactive data analysis and graphing, it is pretty good.

Install R at https://cran.rstudio.com and Rstudio at https://www.rstudio.com.

Install and load ggplot2 package. This can be done from the Rstudio RUI in the lower right panel (install and then check the box next to the package to load) or via the command

```
if(!require(ggplot2)) {
    install.packages("ggplot2", repos = "http://cran.us.r-project.org")
    library(ggplot2)
}

## Loading required package:  ggplot2
```

Download the clusters.csv file at
https://drive.google.com/file/d/1t6bskHdji-wpVKaf3xvksoiVI3LfMPD7/view?usp=sharing. You can then load this dataset via the Rstudio RUI with "Import Dataset" in the top right panel or with the command

```
clusters <- read.csv("~/Downloads/clusters.csv")
```

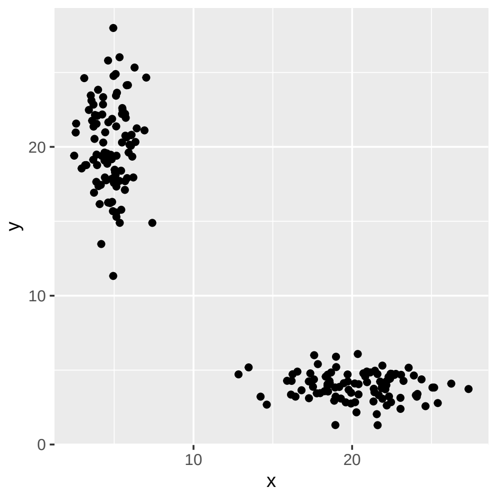
First, let us look at what variables we have access to by printing the first few lines of the data.

```
head(clusters)

##   X        x        y  cluster
## 1 1 3.890076 19.48149 cluster1
## 2 2 5.511740 22.60872 cluster1
## 3 3 5.674010 17.10917 cluster1
## 4 4 6.285058 25.33035 cluster1
## 5 5 2.573625 20.96575 cluster1
## 6 6 5.137913 17.33480 cluster1
```

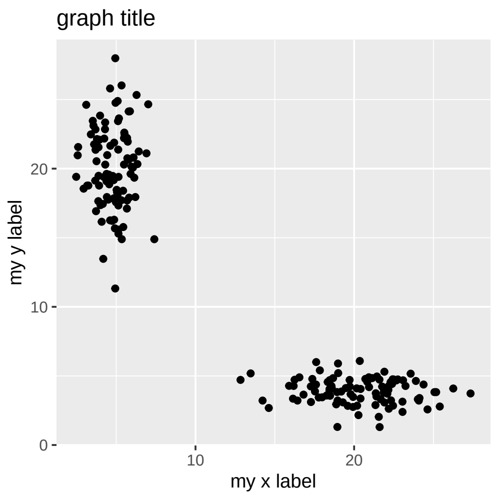
Now we can make our first plot.

```
ggplot(clusters)+geom_point(aes(x=x,y=y))
```



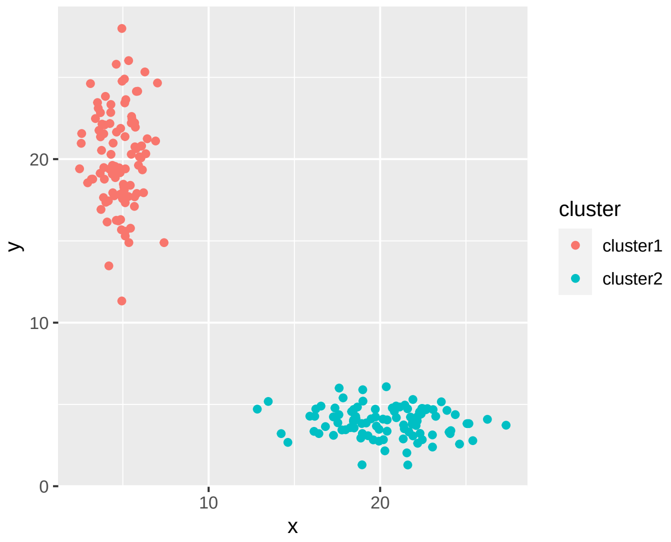
And we can give x and y axis labels and a plot title.

```
ggplot(clusters)+geom_point(aes(x=x,y=y))+xlab("my x label")+ylab("my y label")+ggtitle("graph title")
```



So this is creating a ggplot object, giving it the clusters data frame. Then we add a layer consisting of a geometry. Geometries in ggplot are the different types of graphs such as a scatter plot, a line plot, histogram, etc. Geom_point is the scatter plot geometry. Then we map variables in our dataset (x and y) to visual aspects of our graph. We do this with the aesthetic command (which is aes for short). Each geometry has its particular aesthetic variables it requires and which ones are optional, but these tend to follow a consistent pattern. The geom_point requires an x and a y aesthetic. Other optional aesthetics are things like size, shape, color, etc. Next we will map the cluster categorical variable to the color aesthetic.

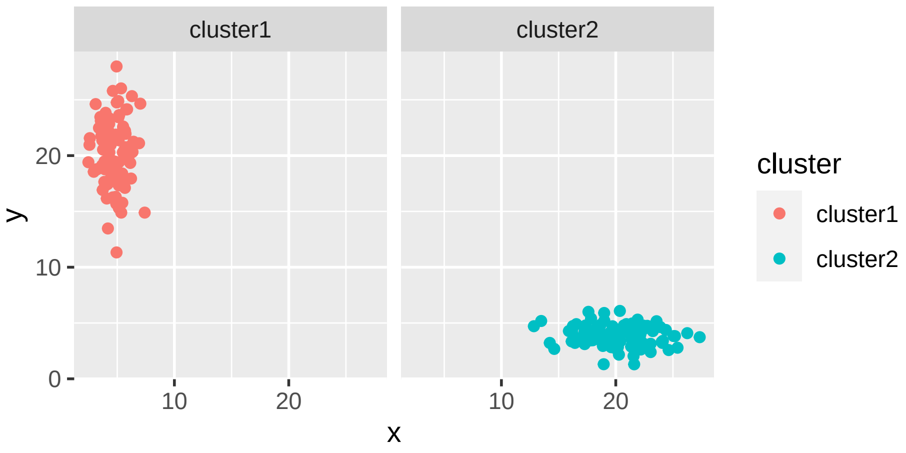
```
ggplot(clusters)+geom_point(aes(x=x,y=y,color=cluster))
```



Mapping some optional aesthetics will automatically generate a legend as you can see.

Another thing that can be useful when you have complex data which can be separated out by categorical variables (different experimental conditions, etc) is to split these into multiple subgraphs. This is termed faceting. There are two types of facets in ggplot, facet_wrap and facet_grid. facet_grid splits the data on two categorical variables and displays them in a grid where one categorical variable is split on the horizontal and the other on the vertical. facet_wrap splits the graph on a single variable and can line wrap if needed.

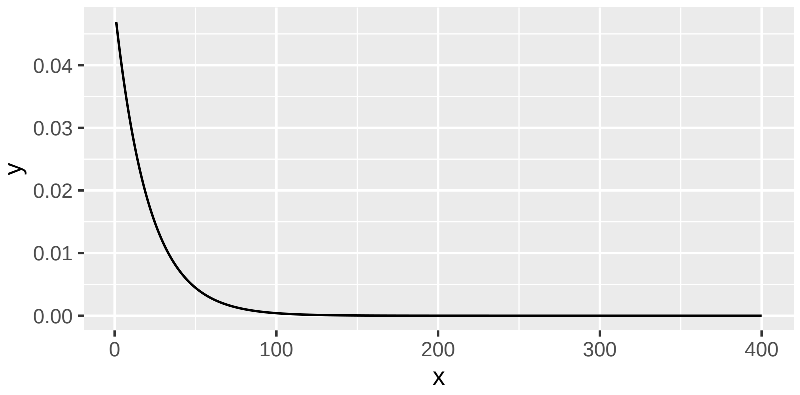
```
ggplot(clusters)+geom_point(aes(x=x,y=y,color=cluster))+facet_wrap(~cluster)
```



Geometric distribution review

Recall that the geometric distribution is the length between positive events, or the number of 0's you see before seeing a 1. The positive event has a probability p at every discrete time point. So the likelihood of seeing k 0's before seeing a 1 is $(1-p)^{(k-1)}p$. Think about what value of k maximizes this likelihood. It is perhaps unintuitive that the single most likely length to the first positive event is 1 no matter what the probability is, but that is the case.

```
x = seq(1,400,by=.1)
p = 3/64
y = (1-p)^(x-1)*p
ggplot(data.frame(x=x,y=y))+geom_line(aes(x=x,y=y))
```



1 Binomial distribution

The binomial distribution is the number of positive events out of n discrete possible trials. Intuitively the likelihood would be

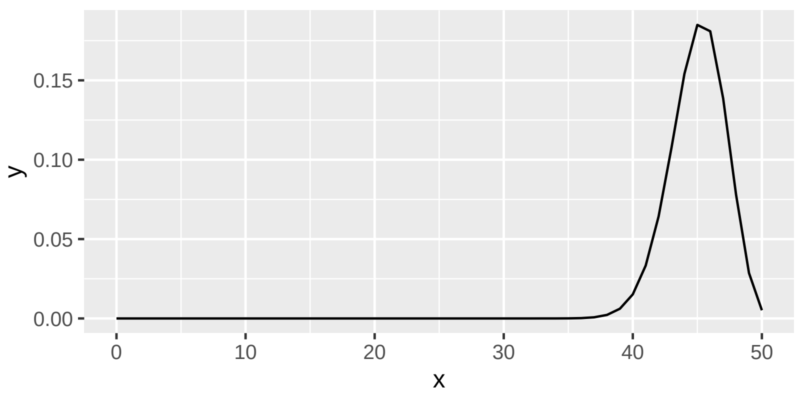
$$p^k(1-p)^{(1-p)} \tag{1}$$

but this is ignoring the different possible orderings we can see these events. So the actual probability density function is

$$\binom{(n}{k)p^k(1-p)^{(1-p)}} \tag{2}$$

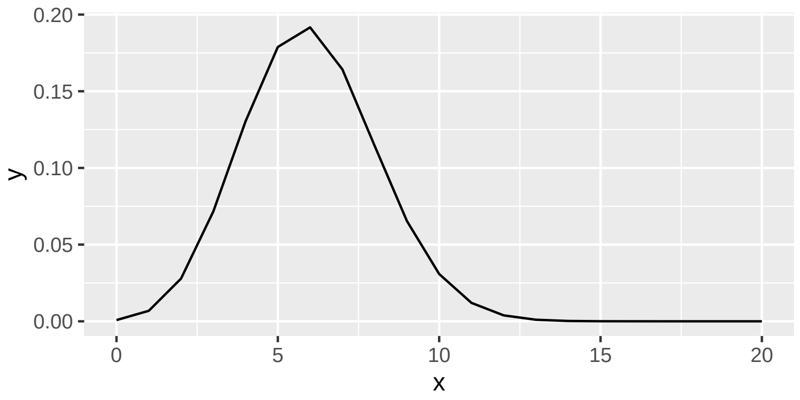
Clearly, the random variable k has support from $[0, n]$ with a mean of np.
Now let's plot this to get a sense of the shape of the distribution.

```r
x = seq(0,50)
y = dbinom(x,50,0.9)
df = data.frame(x=x,y=y)
ggplot(df)+geom_line(aes(x=x,y=y))
```



and with some different parameters

```r
x = seq(0,20)
y = dbinom(x,20,0.3)
df = data.frame(x=x,y=y)
ggplot(df)+geom_line(aes(x=x,y=y))
```



One example of a binomially distributed processes are how many CGs are in a sequence. The probability of CGs in background genomic sequence is <1%, but the probability of CGs in promoter CpG island regions is much higher. See the first example for bayes rule in lecture notes 2 for an example on CpG islands using the binomial distribution.

Poisson distribution

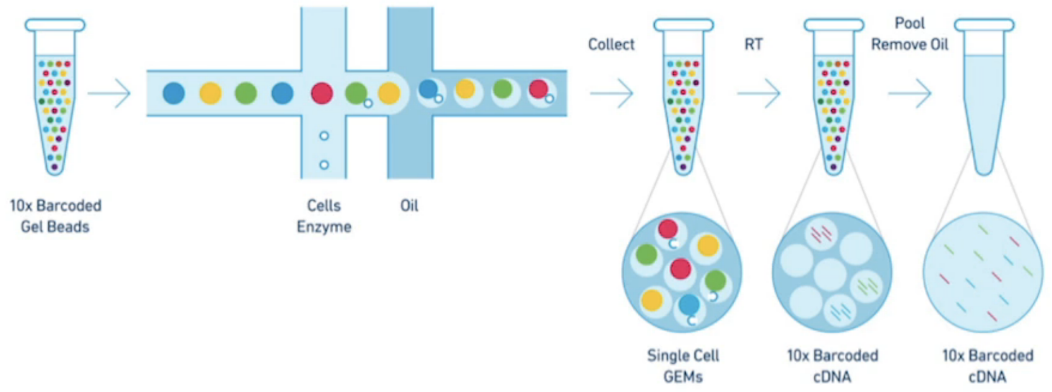
Poisson sampling occurs when some positive event happens at some continuous rate. The number of positive events given a rate and a fixed unit of time or space is distributed as a Poisson distribution. The rate is λ and the random variable of the number events is k. The probability density function is a bit trickier to derive than previous distributions and is

$$\frac{\lambda^k e^{-\lambda}}{k!} \tag{3}$$

This is different from the binomial because it uses a continuous rate instead of a probability for a discrete number of trials. This means that the support for the random variable k is $[0, \infty)$. When trying to determine what distribution a given process produces, it is often useful to narrow it down by identifying the observed/random variable, make a description of it, ask whether it is continuous or discrete, and ask what values it can take on—what its support is. From those questions, it often becomes clear what distribution your data will follow.

One example of a poisson distributed random variable is how many DNA reads overlap a given point in the genome. Because reads can come from anywhere in the genome randomly, and the total number of reads, while finite, is much much larger than the average number of reads at any given location, the maximum possible number can be thought to be infinite. So the rate is $\lambda = \frac{\text{reads}*\text{read\_length}}{\text{genome length}}$.

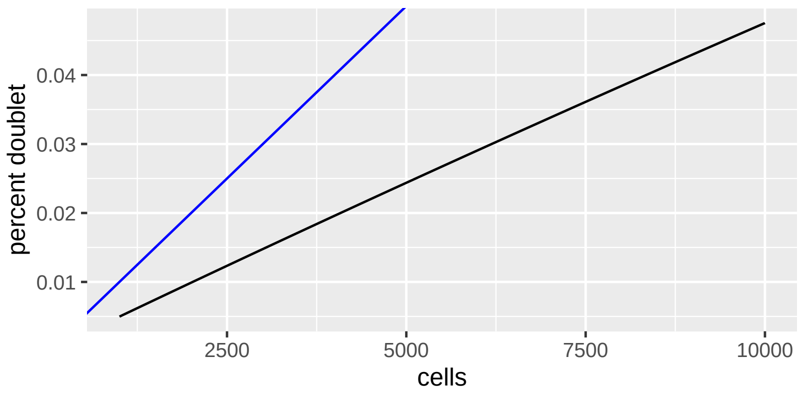
Another process that produces a poisson distribution comes up in single cell sequencing. Single cell sequencing attempts to sequence the RNA from each cell and attach a barcode sequence to each read according to which cell it came from. Because in the central dogma of molecular biology, information passes from DNA to RNA to protein and proteins are responsible for the majority of the function of the cell, sequencing protein would get us the best information on cell function. However, we do not have good technologies for sequencing proteins and we do have good technologies for sequencing RNA (or really RNA turned back into DNA). The cell-barcoding is accomplished by a microfluidic system that creates droplets of aquaous solution containing a gel bead with millions of copies of the same barcode DNA and zero, one, or more cells surrounded by oil. From there, reads are generated from the cell's RNA and the barcode DNA is attached



to them.
The loading of cells into these droplets is a poisson process. The goal is for these droplets to contain either 0 or 1 cell with high probability. Droplets with more than 1 cell are termed doublets or multiplets and cause noise and difficulty in downstream processing. So to avoid these as much as possible while still getting a reasonable number of droplets with 1 cell, we control the rate of this poisson process by loading fewer or more cells. The company states that the rate of doublets is roughly 1% per thousand cells used in the experiment. But of course this is not true as the poisson distribution is not a linear function. So let's find out what the doublet/multiplet number and rate we should actually expect. In order to determine the rate λ, we will use the number of cells input and an unknown constant. We will find which constant best fits the 1% per 1k input cells in the normal range the company recommends, 1000-10k cells.

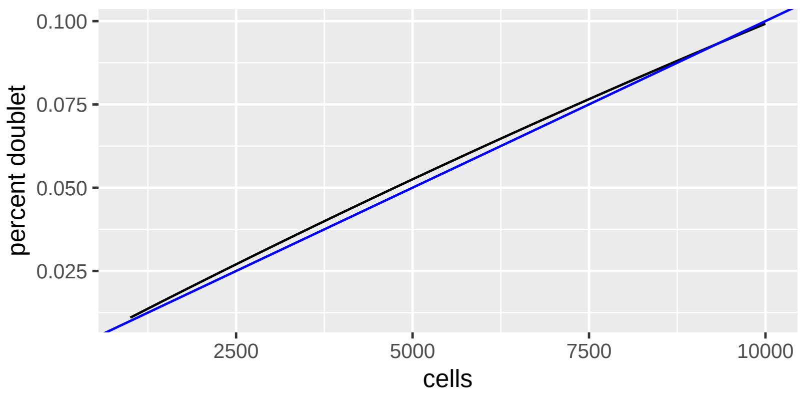
```
cells = seq(1,200000)
constant = 100000
singlet = dpois(1,cells/constant)
doublet = dpois(2,cells/constant)
```

```
triplet = dpois(3,cells/constant)
quad = dpois(4,cells/constant)
pent = dpois(5,cells/constant)
total = singlet + doublet + triplet + quad + pent
data = data.frame(x=cells,y=doublet/total)
# subset to 1000-10000 cells for doublet % comparison
d = subset(data, x < 10000 & x > 1000)
ggplot(d)+geom_line(aes(x=x,y=y))+geom_abline(intercept=0,slope=0.01/1000,color="blue")+ylab("percent do
```



So clearly this constant is predicting more doublets in this cell range than the company says. We could use some optimization strategy, but I already know a good constant to use.

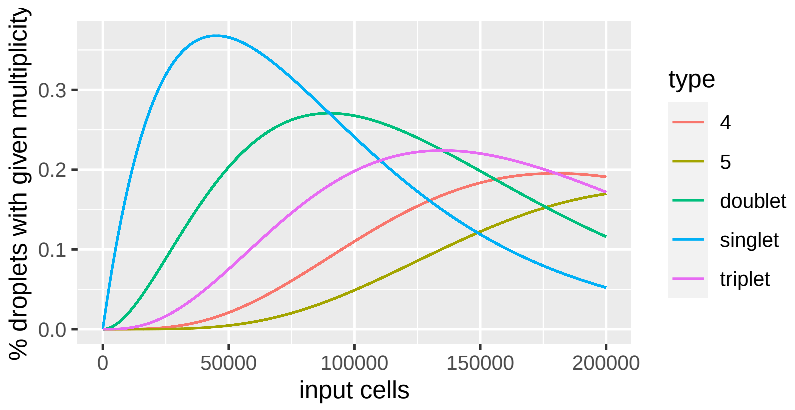
```
cells = seq(1,200000)
constant = 45000
singlet = dpois(1,cells/constant)
doublet = dpois(2,cells/constant)
triplet = dpois(3,cells/constant)
quad = dpois(4,cells/constant)
pent = dpois(5,cells/constant)
total = singlet + doublet + triplet + quad + pent
data = data.frame(x=cells,y=doublet/total)
# subset to 1000-10000 cells for doublet % comparison
d = subset(data, x < 10000 & x > 1000)
ggplot(d)+geom_line(aes(x=x,y=y))+
  geom_abline(intercept=0,slope=0.01/1000,color="blue")+
  ylab("percent doublet")+xlab("cells")
```



Ok, now that we know the rate, we can look at the actual numbers of each different category versus input cells. In practice, this would allow us to accurately choose what trade-off between how many singlets we get versus multiplets.

```r
data = data.frame(x=cells,y=singlet, type="singlet")
data = rbind(data, data.frame(x=cells, y=doublet, type="doublet"))
data = rbind(data, data.frame(x=cells, y=triplet, type="triplet"))
data = rbind(data, data.frame(x=cells, y=quad, type="4"))
data = rbind(data, data.frame(x=cells, y=pent, type="5"))

ggplot(data)+geom_line(aes(x=x,y=y,color=type))+
  ylab("% droplets with given multiplicity")+
  xlab("input cells")
```



So this is an example of using knowledge of the physical process and some data (the 1% per 1k cells in the low end) to extrapolate out and make predictions of what will happen under other circumstances. With an industry partner, we have done experiments to confirm these predictions.