**Chapter1**

**The *Bernoulli* distribution** a single binary trial such as a coin flip

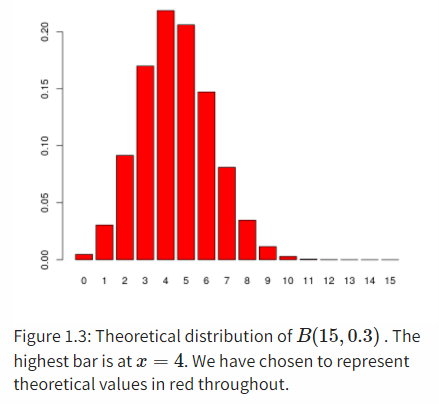
**The *binomial* distribution** n binary trial

**The *Poisson* distribution** when p is small (the 1s are rare). It has only one parameter λ, and the Poisson distribution for λ=np

**The *multinomial* distribution** discrete events that have more than two possible outcomes or **levels**

**Stimulation**

**The *binomial* distribution** n binary trial



probabilities = **dbinom**(0**:**15, prob = 0.3, size = 15)

Theoretical distribution of B(15,0.3)B(15,0.3) . The highest bar is at x=4

**The *Poisson* distribution** when p is small (the 1s are rare). It has only one parameter λ, and the Poisson distribution for λ=np

ELIZA example

the false positive rate is 1%: p(+|-)=0.01

50 patients: n=50

protein is tested at 100 different positions: 100 Independent trials

maxes = **replicate**(100000, {

**max**(**rpois**(100, 0.5))

})

**table**(maxes)

## 1 2 3 4 5 6 7 9

## 7 23028 60840 14364 1604 141 15 1

**mean**( maxes **>=** 7 )

## [1] 0.00016

P(x=7|lambda=0.5)

**Monte Carlo** method:

computer simulation

finds the probabilities of the events we’re interested in based on our generative model

**The *multinomial* distribution** discrete events that have more than two possible outcomes or **levels**

DNA example

how much data we need to collect if we want to test whether a multinomial model with equal probabilities is consistent with the data?

H0: pA=pC=pG=pT=1/4

sequence of length n=20

We’ll propose this as our critical value for testing data and will reject the hypothesis that the data come from a fair process, with equally likely nucleotides

pvec = **rep**(1**/**4, 4)

obsunder0 = **rmultinom**(1000, prob = pvec, size = 20)

exptd = 20 **\*** pvec

stat = **function**(obsvd, exptd) {

**sum**((obsvd **-** exptd)**^**2 **/** exptd)

}

S0 = **apply**(obsunder0, 2, stat)

**quantile**(S0, probs = 0.95)-> 7.6

>95%: 1000 obsvd, sum((obsvd - exptd)^2 / exptd)<7.6

Data: pA=1/8,pC=3/8,pG=3/8,pT=1/8

pvecA = **c**(3**/**8, 1**/**4, 3**/**12, 1**/**8)

observed = **rmultinom**(1000, prob = pvecA, size = 20)

S1 = **apply**(observed, 2, stat)

power = **mean**(S1 **>** 7.6)

[1] 0.199

Run across 1000 simulations, the test identified 199 as coming from an alternative distribution. We’ve thus computed that the probability P(reject H0|HA)P(reject H0|HA) is 0.199.

With a sequence length of n=20, we have a power of about 20% to detect the difference between the fair generating process and our **alternative**.

**Chapter 2**

**“Probability” vs “Statistics”**

**fitting** We used different visualizations and showed how to run simulation experiments to test whether our data could be fit by a fair four-box multinomial model. We encountered the chi-square statistic and saw how to compare simulation and theory using a qq-plot.

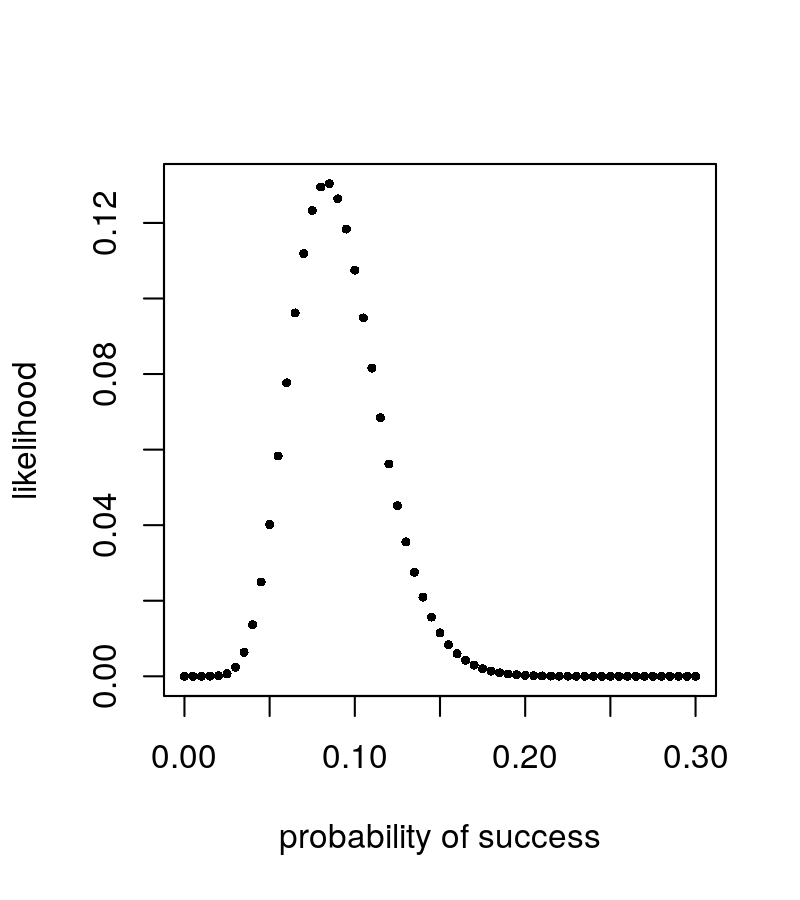
**Estimation** We explained maximum likelihood and Bayesian estimation procedures. These approaches were illustrated on examples involving nucleotide pattern discovery and haplotype estimations.

**Prior and posterior distributions** When assessing data of a type that has been previously studied, such as haplotypes, it can be beneficial to compute the posterior distribution of the data. This enables us to incorporate uncertainty in the decision-making, by way of a simple computation. The choice of the prior has little effect on the result as long as there is sufficient data.

**CpG islands and Markov chains** We saw how dependencies along DNA sequences can be modeled by Markov chain transitions. We used this to build scores based on likelihood ratios that enable us to see whether long DNA sequences come from CpG islands or not. When we made the histogram of scores, we saw in Figure [2.25](https://web.stanford.edu/class/bios221/book/Chap-Models.html#fig:chap2-r-ScoreMixture-1) a noticeable feature: it seemed to be made of two pieces. This **bimodality** was our first encounter with mixtures, they are the subject of Chapter [4](https://web.stanford.edu/class/bios221/book/Chap-Mixtures.html#Chap:Mixtures).

**Maximum likelihood** choose the parameter that makes the observed data the most likely

Example of binomial distribution



1. We want to know: how to calculate the likelihood of a certain parameter

likelihood = **dbinom**(**success\_counts** , prob=success.rate, size= num of trial)

loglikelihood = **function**(theta, n = 300, k = 40) {

115 **+** k **\*** **log**(theta) **+** (n **-** k) **\*** **log**(1 **-** theta)

}

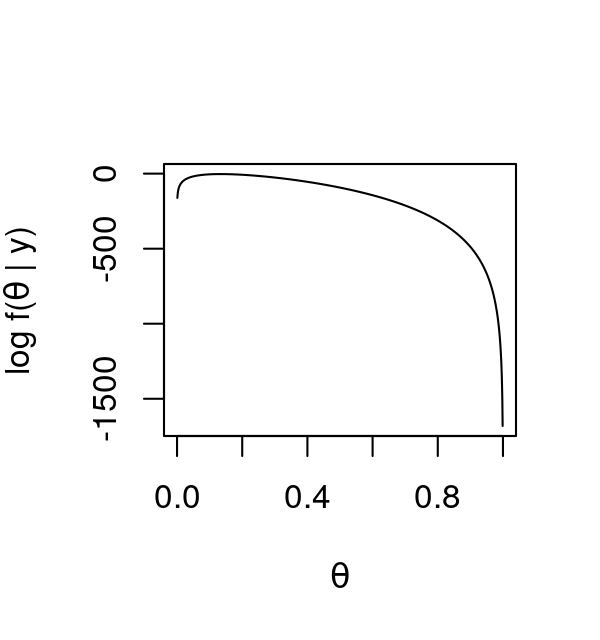
1. Generate the “candidate” parameter

thetas = **seq**(0, 1, by = 0.001)

1. Calculate the likelihood for all candidate parameter and plot

**plot**(thetas, **loglikelihood**(thetas), xlab = **expression**(theta),

ylab = **expression**(**paste**("log f(", theta, " | y)")),type = "l")

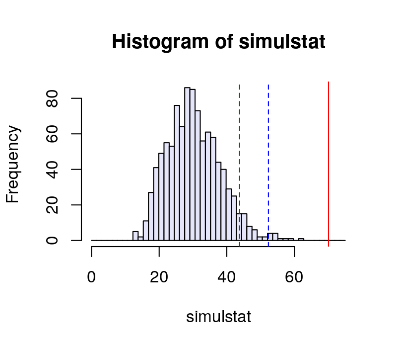


test whether the nucleotides are equally distributed across the four nucleotides for this first gene

stat = **function**(obsvd, exptd = 20 **\*** pvec) {

**sum**((obsvd **-** exptd)**^**2 **/** exptd)

}



The parameters of our distributions are –at least conceptually– definite, knowable, fixed numbers.

we use probability distributions to express our knowledge about the parameters, and use data to *update* this knowledge, for instance by shifting those distributions or making them more narrow; this is provided by the Bayesian paradigm

We have to start with an initial guess for the labels, estimate the parameters and go through several iterations of the algorithm, updating at each step our current best guess of the group labels and the parameters until we see no substantial improvement in our optimizations.

**Chapter4**

**4.2.1 Simple Examples**

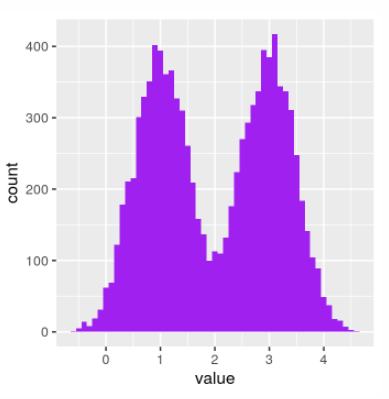
Flip a fair coin:

If it comes up heads

+ Generate a random number from a Normal with mean =1 and variance 0.5.

If it comes up tails

+ Generate a random number from a Normal with mean =3 and variance 0.5.

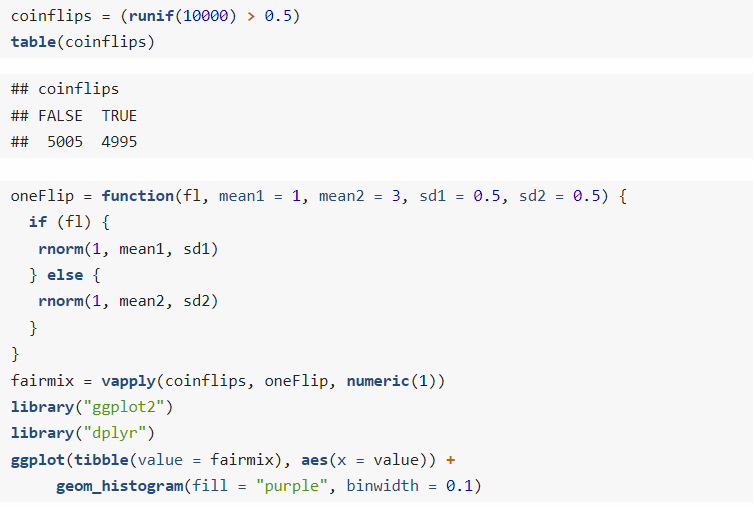


Model:

P(u=0)=0.5, P(u=1)=0.5 λ(u=1)=0.5

u=0, y~NA(1, sd=0.5)

u=1, y~NB(3, sd=0.5)

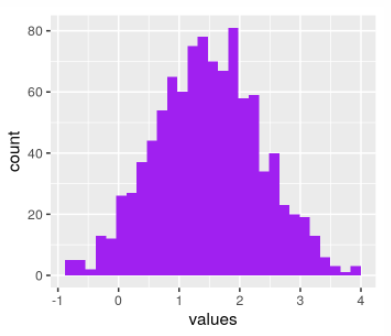


Change parameters：

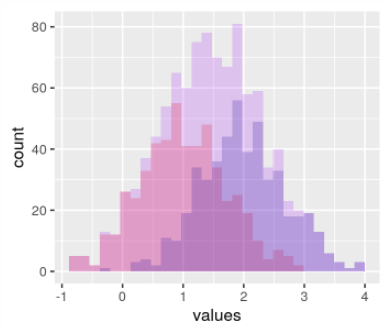
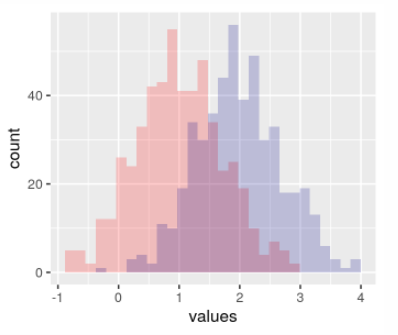
P(u=0)=0.5, P(u=1)=0.5 λ(u=1)=0.5

u=0, y~NA(**1**, sd=0.5)

u=1, y~NB(**2**, sd=0.5)



Colors in red the points that were generated from the heads coin flips and in blue those from the tails



**4.2.2 Discovering the hidden class labels**

**What if : μ1 &μ2 unkonwn, u is labeled**

P(u=0)=0.5, P(u=1)=0.5 🡨λ=0.5

u=0, y~NA(μ1=?, sd=0.5)

u=1, y~NB(μ2=?, sd=0.5)

Find μ1 &μ2: Maximum likelihood(Chapter 2)

**Issue:**

Find maximum likelihood estimates of µ1, µ2, **but u is unknown**

Bivariate distribution: distribution of “couples” (Y , U)

C:\Users\MSI\AppData\Local\Temp\1644469670(1).png

θ stands for the tuple of parameters of the underlying density. In our previous example, θ would be the two means, the two standard deviations, and the mixture fraction λ=0.5; thus θ=(μ1,μ2,σ1,σ2,λ)

*‘We have to start with an initial guess for the labels, estimate the parameters and go through several iterations of the algorithm, updating at each step our current best guess of the group labels and the parameters until we see no substantial improvement in our optimizations.’*

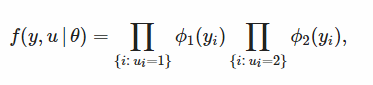
**Expectation-maximization (EM) algorithm**

E `expectaon’ step: guess u (which distribution?)

M `maximization’ step: guess μi (parameters of the component distributions?)

These two iterations (E and M) are repeated until the improvements are small

***Like this:***

**Goal: Maximize**

🡨Likelihood function 2.0 (two levels of parameter)

**Procesure:**

Step “E”: (generate the label u)   
Step “M”: run Maximum Likelihood for completed data, which gives new model µ

Step “E”: compute the probability with which each observation belongs to (?P(u=0), ?P(u=1))

Step “M”: run Maximum Likelihood for completed data, which gives new model µ

Step “E”: compute the probability with which each observation belongs to (?P(u=0), ?P(u=1))

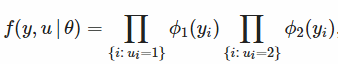
……

The improvement of likelihood is small 🡪 Stop(maximum is found)

**How to find the parameters yield to maximized likelihood/expectation?**

*In chapter 2: Derivative+ Logarithm*

likelihood function (we want to maximize):



C:\Users\MSI\AppData\Local\Temp\1644472459(1).png

C:\Users\MSI\AppData\Local\Temp\1644472540(1).png

Also,

Several R packages provide EM implementations, including [**mclust**](https://cran.r-project.org/web/packages/mclust/), [**EMcluster**](https://cran.r-project.org/web/packages/EMcluster/) and [**EMMIXskew**](https://cran.r-project.org/web/packages/EMMIXskew/).

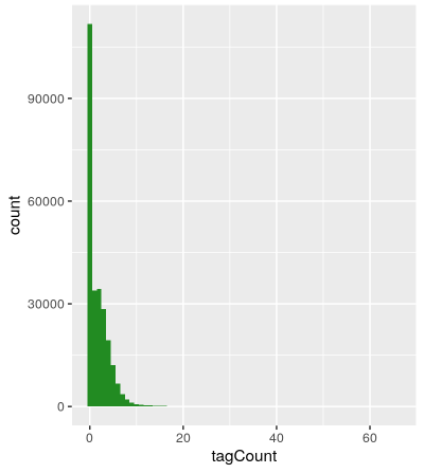
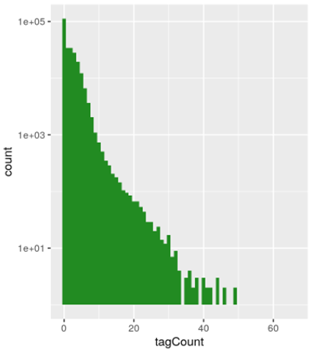
**4.2.3**

***Example1***

**Models for zero-inflated data**

For instance, this may be the number of individuals from each of several species at each of several locations. Such data can often be seen as a mixture of two scenarios: if the species is not present, the count is necessarily zero, but if the species is present, the number of individuals we observe varies, with a random sampling distribution.

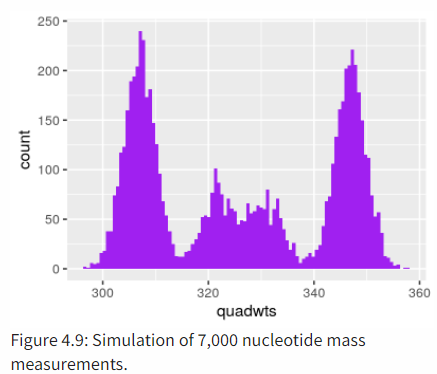
C:\Users\MSI\AppData\Local\Temp\1644830886(1).png

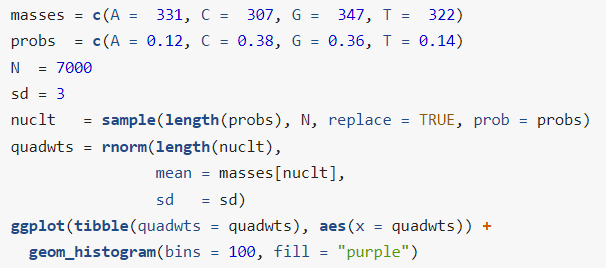
 

**4.2.4**

***Example 2***

Nucleotide info



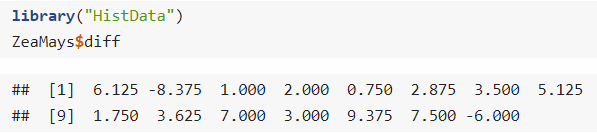


**4.3**

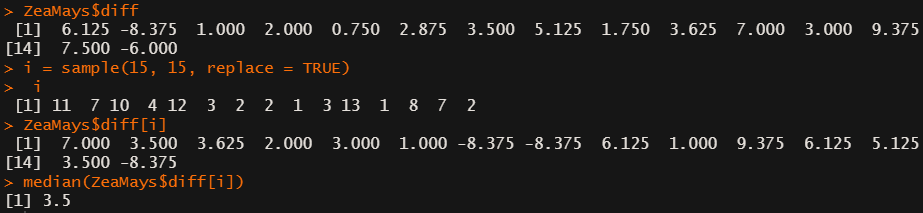
**Bootstrapping**

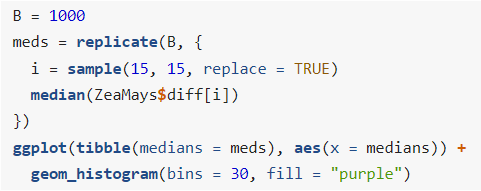
Darwin’s Zea Mays data: in which he compared the heights of 15 pairs of Zea Mays plants

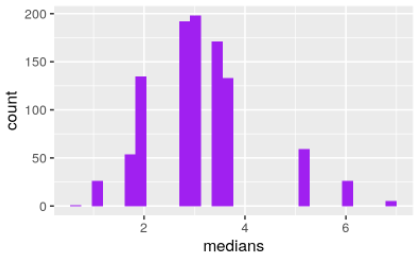
Data:



Goal: Find median







This is called the nonparametric bootstrap resampling approach

In theoretical statics, nonparametric methods are those that have infinitely many

degrees of freedom or parameters. In practice, we do not wait for infinity; when the

number of parameters becomes as large or larger than the amount of data available,

we say the method is nonparametric.

**4.4 Infinite mixtures**

**4.4.1 Infinite mixture of normals**

*If the number of mixture components is as big as (or bigger than) the number of observations, we say we have an****infinite mixture****.*

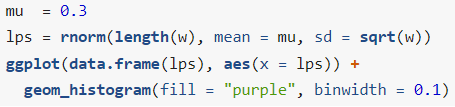
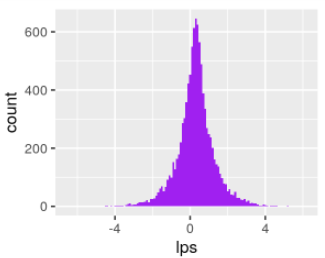
*Laplace distribution*

****

Level 1 Create a sample of Ws from an exponential distribution.

Level 2 The Ws serve as the variances of normal variables with mean μ generated using rnorm.

Ps. rnorm(n, vector of mean, vector of sd)

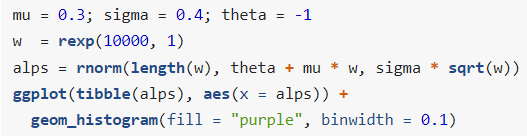
 

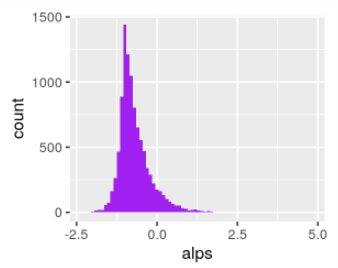
Every observation belongs to its own variance

**Asymmetric Laplace**

Adds another parameter θ that controls the locations or centers of the components

N(*θ+wμ , σw*)

**



X~N(*θ+wμ , σw*)

X∼AL(θ,μ,σ)

Every instance of the data has its own mean and variance

**4.4.2 Infinite mixtures of Poisson variables**

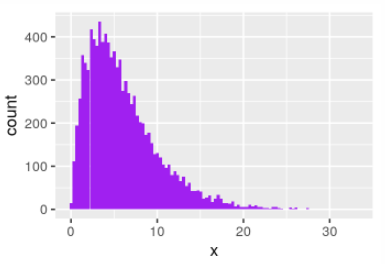
Practical situations: gene expression in microarrays, taxa counts in 16sRNA studies…

In ecology, for instance, we might be interested in variations of fish species in all the lakes in a region. We sample the fish species in each lake to estimate their true abundances, and that could be modeled by a Poisson. But the true abundances will vary from lake to lake. And if we want to see whether, for instance, changes in climate or altitude play a role, we need to disentangle such systematic effects from random lake-to-lake variation.

The different Poisson rate parameters λ can be modeled as coming from a distribution of rates.

**4.4.3 Gamma distribution: two parameters (shape and scale)**

gamma distribution is an extension of the (one-parameter) exponential distribution, but it has two parameters



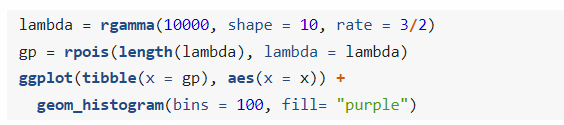
从意义来看：指数分布解决的问题是“要等到一个随机事件发生，需要经历多久时间”，

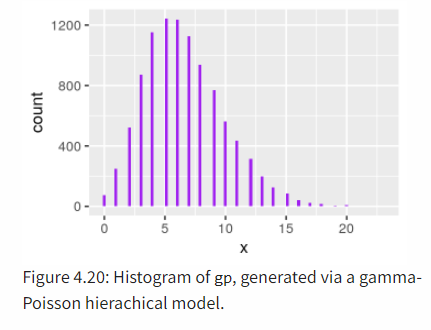
伽玛分布解决的问题是“要等到n个随机事件都发生，需要经历多久时间”

**Gamma–Poisson mixture: a hierarchical model**

AKA. negative binomial distribution

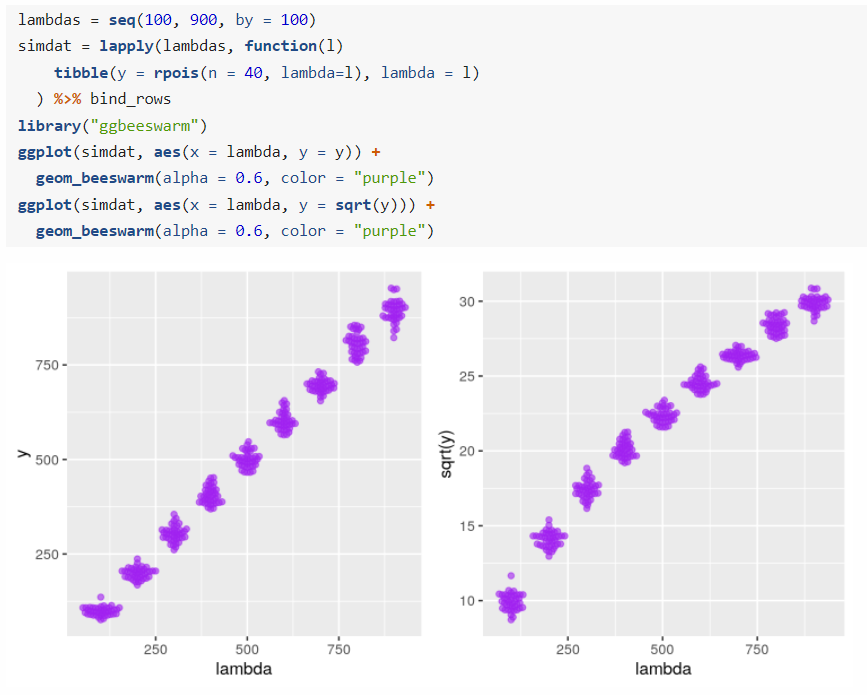
1. Generate a set of parameters: λ1,λ2,...λ1,λ2,... from a gamma distribution.
2. Use these to generate a set of Poisson(λi) random variables, one for each λ1.





**4.4.4 Variance stabilizing transformations**

Issue: how much variability there is between repeated measurements of the same underlying true value?



Left panel：

异方差性Heteroscedasticity: the standard deviations (or, equivalently, the variance) of our data is different in different regions of our data space.

Right panel：

If we apply the square root transformation to the y-variables, then the transformed variables will have approximately the same variance.