Lecture 11: Distributions as Models
STAT GR5206 *Statistical Computing & Introduction to Data Science*

Linxi Liu
Columbia University

November 16, 2018

Course Notes

Last Time

- **Permutation test**. Testing if two distributions are the same.
- In-class example: Non-parametric version of the two-sample t-test.
- **Empirical size**. Simulation based validation of the classic 2-sample t-sample.
- Lab: Sampling distribution, distribution of p-value, empirical size and power.

This Time

- **Distributions as Models:**
- The Method of Moments
- Maximum Likelihood Estimation

Distributions as Models

Cats

The cats dataset includes the heart and body weights of samples of male and female cats. All the cats are adults and over 2 kg in body weight.
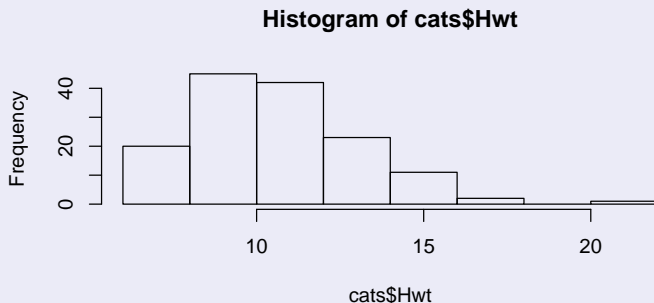
```
> # install.packages("MASS")
> library(MASS)
> head(cats)

  Sex Bwt Hwt
1   F 2.0 7.0
2   F 2.0 7.4
3   F 2.0 9.5
4   F 2.1 7.2
5   F 2.1 7.3
6   F 2.1 7.6
```

The Distribution of the Data

We've studied how to visually inspect the distribution of a continuous random variable.

```
> hist(cats$Hwt)
```

Histogram of cats$Hwt

The Distribution of the Data

R Functions to study the Data's Distribution

quantile(x, probs) calculates the quantiles at probs from x.

```
> quantile(cats$Hwt, c(0.25, 0.5, 0.75))

  25%    50%    75%
8.950 10.100 12.125
```
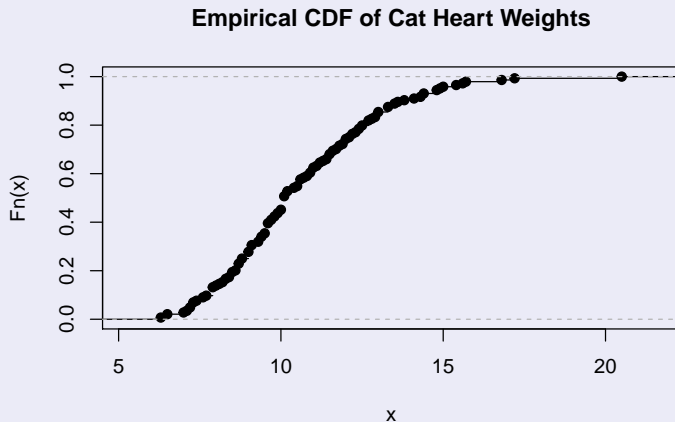
The Distribution of the Data

R Functions to study the Data's Distribution

ecdf(): **e**mprical **c**umulative **d**istribution **f**unction. No assumptions but also no guess about the distribution beyond the observations.

- In math ECDF is written as $\hat{F}$ or $\hat{F}_n$
- Conceptually, quantile() and ecdf() are inverses to each other.

The Distribution of the Data

```
> plot(ecdf(cats$Hwt),
+      main = "Empirical CDF of Cat Heart Weights")
```

Empirical CDF of Cat Heart Weights

The Distribution of the Data

R Functions to study the Data's Distribution

density(x): estimates the density of x by counting how many observations fall in a little window around each point, then smoothing.

- "Bandwidth" = width of window around each point
- AKA calculates a 'kernal density estimate'
- density() returns a collection of x, y values suitable for plotting

The Distribution of the Data

## R Functions to study the Data's Distribution

`density(x)`: estimates the density of x by counting how many observations fall in a little window around each point, then smoothing.

- "Bandwidth" = width of window around each point
- AKA calculates a 'kernal density estimate'
- `density()` returns a collection of x, y values suitable for plotting

Note, `density()` is an *estimate* of the pdf, not the truth.

The Distribution of the Data

```
> hist(cats$Hwt, probability = TRUE, ylim = c(0, 0.17))
> lines(density(cats$Hwt), lty = "dashed")
```
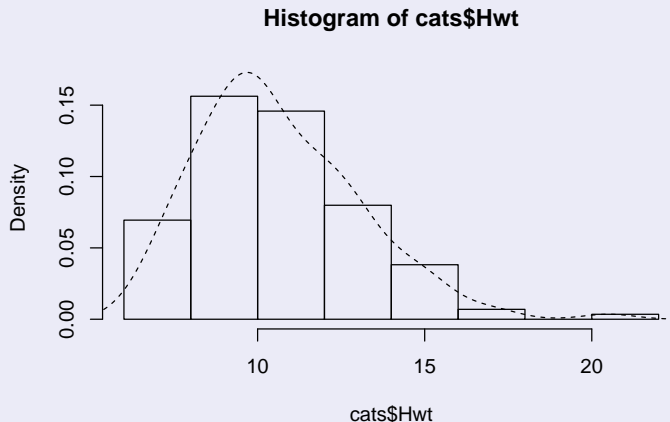

Histogram of cats$Hwt

Why Do We Care About the Distribution of the Data?

- The data itself is too much information and *overly detailed*. Don't need to keep around every single data point.
- Plus, the exact data would never repeat itself if we re-sampled anyways.

Why Do We Care About the Distribution of the Data?

- The data itself is too much information and *overly detailed*. Don't need to keep around every single data point.
- Plus, the exact data would never repeat itself if we re-sampled anyways.
- **Goal**: Store information that *summarizes* what will *generalize* to other situations.
  - Can do this by using a model and only keeping the model's parameters.

How Do We Fit Distributional Models to Data?

Recall that most models are defined by *parameters* (like (μ, σ^2) for the normal). So *fitting* a model to data means finding those parameters such that the model best fits the data.

How Do We Fit Distributional Models to Data?

Recall that most models are defined by *parameters* (like (μ, σ^2) for the normal). So *fitting* a model to data means finding those parameters such that the model best fits the data.

- Match moments (mean, variances, etc.).
- Match other summary statistics.
- Maximize the likelihood.

Recall the Gamma Distribution

- The **gamma** distributions are a family of probability distributions defined by the density functions,

$$f(x) = \frac{x^{a-1}e^{-x/s}}{s^a \Gamma(a)},$$

where the gamma function $\Gamma(a) = \int_0^\infty u^{a-1}e^{-u}du$ is chosen so that the total probability of all non-negative x is 1.

- Parameter a is the **shape**, and s is the **scale**.
- The expected value is as, and the variance as^2.

Method of Moments

- Pick enough moments that they *identify* the parameters. At least one moment per parameter.
- Write equations for the moments in terms of the parameters.
  - E.g. for gamma,
  $$\mu = as, \quad \sigma^2 = as^2.$$
- Solve the moment equations for the parameters (usually done by hand).
  - E.g. for gamma,
  $$a = \frac{\mu^2}{\sigma^2}, \quad s = \frac{\sigma^2}{\mu}.$$

Check Yourself

Tasks

- Write a function `gamma.MMest` that takes as input a data vector and returns estimates of the scale parameters *a* and *s* using the moment equations from the previous slide.

- Plug cat heart weights into your function to get estimates of *a* and *s*.

Check Yourself

Tasks

- Write a function `gamma.MMest` that takes as input a data vector and returns estimates of the scale parameters a and s using the moment equations from the previous slide.
- Plug cat heart weights into your function to get estimates of a and s.

Tasks

```
> gamma.MMest <- function(data) {
+   m <- mean(data)
+   v <- var(data)
+   return(c(a = m^2/v, s = v/m))
+ }
```

Check Yourself

Tasks

- Write a function `gamma.MMest` that takes as input a data vector and returns estimates of the scale parameters a and s.
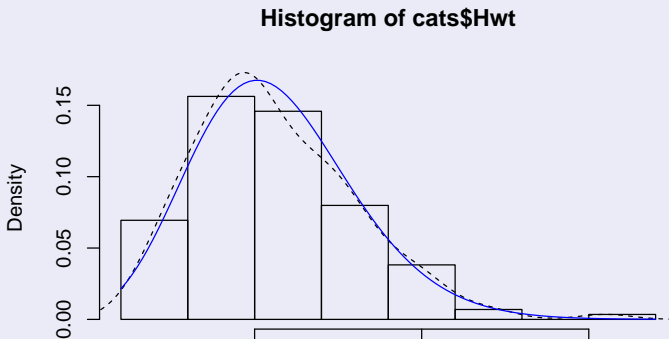- Plug cat heart weights into your function to get estimates of a and s.

Tasks

```
> gamma.MMest(cats$Hwt)

        a           s
19.0653121   0.5575862
```

Method of Moments

```
> hist(cats$Hwt, probability = TRUE, ylim = c(0, 0.17))
> lines(density(cats$Hwt), lty = "dashed")
> cat.MM <- gamma.MMest(cats$Hwt)
> curve(dgamma(x, shape = cat.MM["a"], scale = cat.MM["s"]),
+       add = TRUE, col = "blue")
```

Histogram of cats$Hwt

Method of Moments

- Sometimes we can't solve the moment equations for the parameters by hand. In that case, do it numerically.
- Set up a difference function between the data and the model and then minimize this function.

Method of Moments

- Sometimes we can't solve the moment equations for the parameters by hand. In that case, do it numerically.
- Set up a difference function between the data and the model and then minimize this function.

Cats Example

```
> gamma.mean <- function(a, s) {return(a*s)}
> gamma.var  <- function(a, s) {return(a*s^2)}
> gamma.diff <- function(params, data) {
+    a <- params[1]
+    s <- params[2]
+    return((mean(data) - gamma.mean(a,s))^2
+            + (var(data) - gamma.var(a,s))^2)
+ }
```

Method of Moments

- Sometimes we can't solve the moment equations for the parameters by hand. In that case, do it numerically.
- Set up a difference function between the data and the model and then minimize this function.

Cats Example

```
> nlm(gamma.diff, c(19, 1), data = cats$Hwt)[1:3]

$minimum
[1] 1.899648e-13


$estimate
[1] 19.0653140  0.5575862


$gradient
[1] -5.338805e-09 -2.119918e-07
```

More generally...

Cats Example

- Nothing special about moments. Could match other data summaries too.
  - Examples: the median, quantiles...
- Try to solve for parameters exactly by hand. If you can't set up a discrepancy function and minimize it.

More generally...

Cats Example

- Nothing special about moments. Could match other data summaries too.
  - Examples: the median, quantiles...
- Try to solve for parameters exactly by hand. If you can't set up a discrepancy function and minimize it.
- Just make sure your summaries converge to the population values.
  - How? Simulate then estimate and estimates should converge as the sample grows.

Check Yourself: Checking Your Estimator

Task

- Simulate 100 random variables from a gamma distribution with shape parameter equal to 19 and scale parameter equal to 45. Run the `gamma.MMest` with these values as the input.

- Do the same thing but simulate $10,000$ random variables. Next, $1,000,000$ random variables.

- Does it seem like our estimates are converging to the truth?

Check Yourself: Checking Your Estimator

Solutions

```
> gamma.MMest(rgamma(100, shape = 19, scale = 45))

       a        s
17.59889 46.80859
```

```
> gamma.MMest(rgamma(10000, shape = 19, scale = 45))

       a        s
18.79622 45.39333
```

```
> gamma.MMest(rgamma(1000000, shape = 19, scale = 45))

       a        s
18.96468 45.08403
```

Maximum Likelihood

- Usually we think of parameters, θ, as fixed and consider the probability of different outcomes $f(x, \theta)$ with θ constant and x changing.

Maximum Likelihood

- Usually we think of parameters, θ, as fixed and consider the probability of different outcomes $f(x, \theta)$ with θ constant and x changing.
- **Likelihood** of a parameter value is given by $L(\theta)$: what probability does θ give the data?
  - For continuous variables, use the probability density.
  - Calculate $f(x, \theta)$ letting θ change with data constant.
  - *Not* the probability of θ.

Maximum Likelihood

- Usually we think of parameters, θ, as fixed and consider the probability of different outcomes $f(x, \theta)$ with θ constant and x changing.
- **Likelihood** of a parameter value is given by $L(\theta)$: what probability does θ give the data?
    - For continuous variables, use the probability density.
    - Calculate $f(x, \theta)$ letting θ change with data constant.
    - *Not* the probability of θ.
- **Maximum likelihood** is the guess that the parameter is whatever makes the data most likely.
- Most likely parameter value is the **maximum likelihood estimate** or the **MLE**.

Coding the Likelihood Function

- With independent data points $x_1, x_2, \ldots, x_n$ the likelihood is

$$L(\theta) = \prod_{i=1}^{n} f(x_i, \theta).$$

- Multiplying lots of small numbering is bad, so we usually take the log:

$$\ell(\theta) = \sum_{i=1}^{n} \log f(x_i, \theta).$$

- Note the maximizer is the same for both (though the maximum value will be different).

Check Yourself

Tasks

- Write a function `gamma.ll` which takes as input a parameter vector (with shape and scale) and a data vector and from that returns the log likelihood assuming the data are independent draws from a gamma distribution with scale and shape indicated by the input parameter vec. HINT: Use `dgamma()`.

- Test your function on the cats heart weight data and parameter values scale equals 19 and shape equals 0.5.

Check Yourself

Solution

```
> gamma.ll <- function(params, data) {
+   a <- params[1]
+   s <- params[2]
+   return(sum(dgamma(data, shape = a,
+                     scale = s, log = TRUE)))
+ }
> gamma.ll(c(19, 0.05), cats$Hwt)

[1] -21598.19
```

How do we maximize the likelihood?

How do we maximize it?

- Sometimes, like for the normal distribution, we can do this by hand with calculus.

- Other times we need to use numerical methods... *minimize* the negative log likelihood.

How do we maximize the likelihood?

- Sometimes, like for the normal distribution, we can do this by hand with calculus.

- Other times we need to use numerical methods... *minimize* the negative log likelihood.

```
> nlm(gamma.ll, c(19, 1), data = cats$Hwt)[1:3]

$minimum
[1] -1334280770

$estimate
[1] 409228.0 469356.4

$gradient
[1] -3404.4834  -125.5524
```

How do we maximize the likelihood?

```
> neg.gamma.ll <- function(params, data) {
+   a <- params[1]
+   s <- params[2]
+   return(-sum(dgamma(data, shape = a,
+                      scale = s, log = TRUE)))
+ }
> nlm(neg.gamma.ll, c(19, 1), data = cats$Hwt)$minimum
```

```
[1] 325.5476
```

```
> nlm(neg.gamma.ll, c(19, 1), data = cats$Hwt)$estimate
```

```
[1] 20.299930  0.523674
```

How do we maximize the likelihood?

```
> neg.gamma.ll <- function(params, data) {
+   a <- params[1]
+   s <- params[2]
+   return(-sum(dgamma(data, shape = a,
+                      scale = s, log = TRUE)))
+ }
> nlm(neg.gamma.ll, c(19, 1), data = cats$Hwt)$minimum
```

```
[1] 325.5476
```

```
> nlm(neg.gamma.ll, c(19, 1), data = cats$Hwt)$estimate
```

```
[1] 20.299930  0.523674
```

```
> cat.MM <- gamma.MMest(cats$Hwt)
> neg.gamma.ll(cat.MM, cats$Hwt)
```

Maximum Likelihood

```
> hist(cats$Hwt, probability = TRUE, ylim = c(0, 0.17))
> lines(density(cats$Hwt), lty = "dashed")
> cat.MLE <- nlm(neg.gamma.ll, c(19, 1), data = cats$Hwt)$estimate
> curve(dgamma(x, shape = cat.MM["a"], scale = cat.MM["s"]),
+        add = TRUE, col = "blue")
> curve(dgamma(x, shape = cat.MLE[1], scale = cat.MLE[2]),
+        add = TRUE, col = "red")
```

Maximum Likelihood

Histogram of cats$Hwt
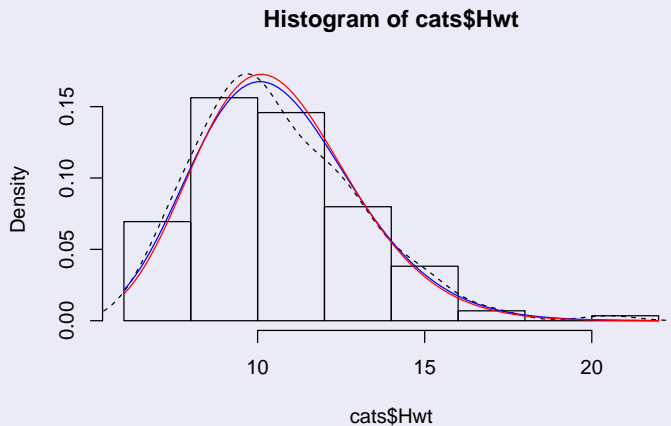
Why the MLE?

- Usually *consistent*: converges to the truth as we get more data.
- Usually *efficient*: converges to the truth as least as fast as anything else.

Checking Fit

- Plot the data with your estimates (like in the last slide).
- Calculate summary statistics not used in fitting and compare with those of the fitted model.
    - Some plotting tools to help with this.
- Use statistical tests.

Checking Fit: Summary Statistics

```
> # Model quantiles
> qgamma(c(0.01, 0.05, 0.95, 0.99), shape = cat.MM["a"],
+        scale = cat.MM["s"])
```

```
[1]  5.795333  6.966974 14.926292 17.097730
```

```
> # Data quantiles:
> quantile(cats$Hwt, c(0.01, 0.05, 0.95, 0.99))
```

```
    1%      5%     95%     99%
 6.500   7.300  14.885  17.028
```

Checking Fit: Summary Statistics

Quantile-Quantile (Q-Q) Plots

- Plots theoretical vs. actual quantiles.
- Ideally, a straight line when the distributions are the same.
- `qqnorm()` and `qqline()` are specialized for checking normality.
- Could also plot quantiles of two samples against each other.
- `qqplot(x,y)` gives a Q-Q plot of one vector against another.

Checking Fit: Summary Statistics

Quantile-quantile Plot

```
> a <- cat.MM["a"]; s <- cat.MM["s"]
> qqplot(cats$Hwt, qgamma((1:99)/100, shape = a, scale = s),
+        ylab = "Theoretical Quantiles")
> abline(0, 1, col = "red")
```

Checking Fit: Summary Statistics
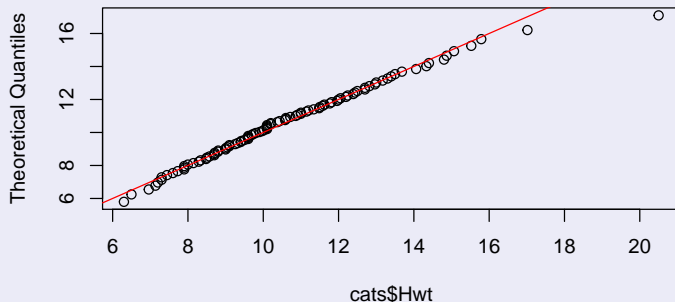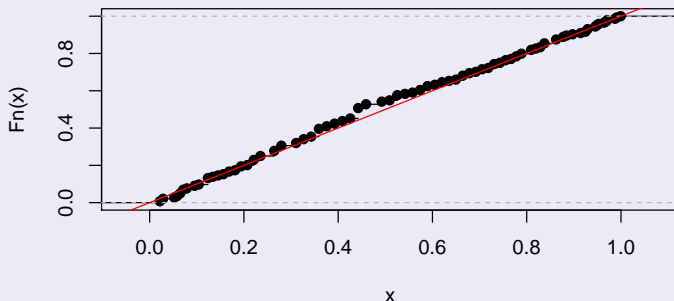
Calibration Plots

- If the distribution is right, 50% of data should be below the median, 90% should be below the 90th percentile, etc.
- **Calibration** probabilities: events with probability p% should happen about p% of the time, not more or less.
- Can look at calibration by calculating the (empirical) CDF and the (theoretical) CDF and plotting.
    - Ideal calibration is a straight line up the diagonal.
    - Systematic deviations should be a warning sign.

Checking Fit: Summary Statistics

Calibration Plots

```
> plot(ecdf(pgamma(cats$Hwt, shape = a, scale = s)),
+        main = "Calibration of gamma distribution for cat hearts")
> abline(0, 1, col = "red")
```


Calibration of gamma distribution for cat hearts

- How much should Q-Q or calibration plot wiggle around the diagonal?

Checking Fit: Kolmogorov-Smirnov Test

- How much should Q-Q or calibration plot wiggle around the diagonal?
- Answer a different question: define the biggest gap between theoretical and empirical CDF

$$D_{KS} = \max_x \left| F(x) - \hat{F}(x) \right|$$

- D_{KS} always has the same distribution *if* the theoretical CDF is fixed and correct.
- Also works for comparing empirical CDF of two samples to see if they come from the same distribution.

Checking Fit: Kolmogorov-Smirnov Test

```
> ks.test(cats$Hwt, pgamma, shape = a, scale = s)

        One-sample Kolmogorov-Smirnov test

data:  cats$Hwt
D = 0.068637, p-value = 0.5062
alternative hypothesis: two-sided
```

Checking Fit: Kolmogorov-Smirnov Test

> **Warning**
> - More complicated and not properly handled by built-in `R` if parameters are estimated by the data.
>   - Fit looks better than it really is.
> - Hack: estimate the model using 90% of the data and check the fit using the K-S test using the other 10% (feels a little bit like *cross*-validation).

Checking Fit: Kolmogorov-Smirnov Test

```
> n        <- length(cats$Hwt)
> train  <- sample(1:n, size = round(.9*n))
> cat.MM <- gamma.MMest(cats$Hwt[train])
> a <- cat.MM["a"]
> s <- cat.MM["s"]
> a
```

```
       a
17.97411
```

```
> s
```

```
        s
0.5900368
```

Checking Fit: Kolmogorov-Smirnov Test

```
> ks.test(cats$Hwt[-train], pgamma, shape = a, scale = s)

        One-sample Kolmogorov-Smirnov test

data:  cats$Hwt[-train]
D = 0.20497, p-value = 0.5988
alternative hypothesis: two-sided
```

Checking Fit: Kolmogorov-Smirnov Test

two sample

Can also test whether two samples come from the same distribution.

```
> ks.test(cats$Hwt[cats$Sex == "F"],
+         cats$Hwt[cats$Sex == "M"])
```

```
        Two-sample Kolmogorov-Smirnov test

data:  cats$Hwt[cats$Sex == "F"] and cats$Hwt[cats$Sex == "M"
D = 0.49419, p-value = 3.847e-07
alternative hypothesis: two-sided
```