Lecture 8: Distributions as Models

STAT GR5206

Statistical Computing & Introduction to Data Science

Cynthia Rush
Columbia University

November 10, 2017

Course Notes

- Homework posted; due November 27. One more homework after this one.
- Lab next week.
- Final Friday, December 15, 1:10pm - 4:00pm. (Location TBD.)

Distributions as Models

Estimation

- Aim: learn about a random process by observing a sample of outcomes.
- $X_1, X_2, \ldots, X_n$ drawn at random. What can we say about the underlying distribution?
- Example: Know distribution is normal, estimate mean and variance.

Two Types of Models

- Parametric: Defined up to a finite dimensional parameter.
- Non-parametric: Number and nature of parameters is flexible.

R Functions to study the Data's Distribution

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

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)
```

The Distribution of the Data

```
hist(cats$Hwt)
quantile(cats$Hwt, c(0.25, 0.5, 0.75))
plot(ecdf(cats$Hwt),
     main = "Empirical CDF of Cat Heart Weights")

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

ggplot(cats) +
 geom_histogram(aes(x = Hwt, y = ..density..)) +
 geom_density(aes(x = Hwt))
```

Why Do We Care About Estimating 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 (parametric estimation).

Estimation

- Given a model and a random sample want a function of the sample that estimates the unknown parameters.
- Estimator, $\theta_n = t(X_1, ..., X_n)$, is only useful for inference if we know how it behaves (under resampling of the data).
- θ_n is a function of a random sample, therefore is a random variable. Behavior will depend on n.

Example

Estimating the population mean of a normally distributed population (for illustrative purposes say $N(10, 9)$).

- Obvious estimation strategy: simply draw a sample and calculate the sample mean.
- Repeat this process with a new sample, get a different estimate.
- The distribution that results from repeated sampling is called the **sampling distribution** of the estimate.

Estimation

```
samp_size  <- 100
samp1      <- rnorm(samp_size, mean = 10, sd = 3)
mmean_est1 <- mean(samp1)

n          <- 500
samp_means <- rep(NA, n)

for (i in 1:n) {
  samp_means[i] <- mean(rnorm(samp_size, mean = 10, sd = 3))
}

ggplot(data.frame(samp = samp_means)) +
  geom_histogram(aes(x = samp, y = ..density..)) +
  geom_density(aes(x = samp))

# Hist: 500 estimates of pop. mean based on sample size 100.
# Estimates are centered around true value; some variation.
```

What's a Good Estimator?

- Obviously want our estimate close to the true value.
- Want θ_n to behave nicely as n increases.
- Want estimate to be more accurate for a large sample than a small sample.
- Formally, a **consistent** estimator will converge to the true parameter value as n increases.

Estimation

```
samp_size  <- c(5, 10, 25, 50, 100)
n          <- 500
samp_means <- cbind(rep(NA, n*length(samp_size)),
                    rep(NA, n*length(samp_size)))
colnames(samp_means) <- c("Values", "SampleSize")

for (j in 1:length(samp_size)) {
  for (i in 1:n) {
    row <- (j-1)*n + i
    samp_means[row, 1] <- mean(rnorm(samp_size[j], 10, 3))
    samp_means[row, 2] <- samp_size[j]
  }
}

ggplot(data.frame(samp_means)) +
  geom_histogram(aes(x = Values, y = ..density..)) +
  geom_density(aes(x = Values)) +
  facet_wrap(~SampleSize)
```

Estimation

Not all estimators will behave as nicely as the sample mean as an estimator for the population mean of a normal!

```
samp_vars <- cbind(rep(NA, n*length(samp_size)),
                   rep(NA, n*length(samp_size)))

colnames(samp_vars) <- c("Values", "SampleSize")

for (j in 1:length(samp_size)) {
  for (i in 1:n) {
    row <- (j-1)*n + i
    samp_vars[row, 1] <- var(rnorm(samp_size[j], 10, 3))
    samp_vars[row, 2] <- samp_size[j]
  }
}

ggplot(data.frame(samp_vars)) +
  geom_histogram(aes(x = Values, y = ..density..)) +
  geom_density(aes(x = Values)) +
  facet_wrap(~SampleSize)
```

In General, How Do We Fit Distributional Models to Data?

Parametric 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.

Some methods...

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

Moments

Let X be a rv according to some probability distribution, then the k^{th} **moment** of a distribution is defined as

$$\mu_k = \mathbb{E}(X^k).$$

E.g. $\mu_1 = \mathbb{E}(X)$ and $\mu_2 = \mathbb{E}(X^2) = \sigma^2 + \mu_1^2$.

The **sample moments** of observations $X_1, X_2, ..., X_n$ independent and identically distributed (iid) from some distribution are defined as,

$$\hat{\mu}_k = \frac{1}{n} \sum_{i=1}^{n} X_i^k.$$

E.g. $\hat{\mu}_1 = \bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i$ is the familiar sample mean and $\hat{\mu}_2 = \hat{\sigma}^2 + \bar{X}^2$ where $\hat{\sigma}^2$ is the sample variance.

Method of Moments

Using the data, our goal is to estimate *parameters* of a model that best fit the data.

- Challenge because there's not usually an obvious way to estimate the parameters using the data.
- Method of moments simply equates the moments of the distribution with the sample moments $\mu_k = \hat{\mu}_k$ and solves for the unknown parameters.
- Note that this implies the distribution must have finite moments.
- Strategy: pick enough **moments** that they identify the parameters. At least one moment per parameter.

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.

Example

- **First**: Write equations for the distribution moments in terms of the parameters.
  - E.g. for gamma,

$$\mu_1 = as, \quad \mu_2 - \mu_1^2 = as^2.$$

- **Second:** Solve the distribution moment equations for the parameters (usually done by hand).
  - E.g. for gamma,

$$a = \frac{\mu_1^2}{\mu_2 - \mu_1^2}, \quad s = \frac{\mu_2 - \mu_1^2}{\mu_1}.$$

- **Third:** Estimate the parameters form the data using the sample moments.
  - E.g. for gamma,

$$\hat{a} = \frac{\hat{\mu}_1^2}{\hat{\mu}_2 - \hat{\mu}_1^2} = \frac{\bar{X}^2}{\hat{\sigma}^2}, \quad \hat{s} = \frac{\hat{\mu}_2 - \hat{\mu}_1^2}{\hat{\mu}_1} = \frac{\hat{\sigma}^2}{\bar{X}}.$$

Check Yourself

Tasks

- Write a function `gam.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, i.e. `gam.MMest(cats$Hwt)` should return estimates $\hat{a} = 19.07$ and $\hat{s} = 0.56$.

Method of Moments

```
cat.MM   <- gam.MMest(cats$Hwt)
gam_args <- list(shape = cat.MM["a"], scale = cat.MM["s"])

ggplot(cats) +
  geom_histogram(aes(x = Hwt, y = ..density..)) +
  geom_density(aes(x = Hwt), linetype = "dashed") +
  stat_function(aes(x = Hwt), fun = dgamma,
                args = gam_args, color = "red")
```

Method of Moments

Before:

- Wrote distributional moment: $\mu_1 = as$, and $\mu_2 - \mu_1^2 = as^2$.
- Solved for parameters:

$$a = \frac{\mu_1^2}{\mu_2 - \mu_1^2}, \quad s = \frac{\mu_2 - \mu_1^2}{\mu_1}.$$

- Estimated parameters from the data:

$$\hat{a} = \frac{\hat{\mu}_1^2}{\hat{\mu}_2 - \hat{\mu}_1^2} = \frac{\bar{X}^2}{\hat{\sigma}^2}, \quad \hat{s} = \frac{\hat{\mu}_2 - \hat{\mu}_1^2}{\hat{\mu}_1} = \frac{\hat{\sigma}^2}{\bar{X}}.$$

- Sometimes can't do the second step by hand. In that case, do it numerically.
- Set up a difference function between the sample moments and the distributional and then minimize this function.

$$\min_{\hat{a},\hat{s}} \left\{ (\hat{\mu}_1 - \hat{a}\hat{s})^2 + (\hat{\sigma}^2 - \hat{a}\hat{s}^2)^2 \right\}$$

Method of Moments

```
gam.mean <- function(a, s) {return(a*s)}
gam.var  <- function(a, s) {return(a*s^2)}

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

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

More generally...

- Nothing special about moments. Could match other data summaries too. Examples: the median, ratios of quantiles (remember your last homework?)...
- Try to solve for parameters exactly by hand. If you can't set up a discrepancy function and minimize it (remember your last homework?).
- 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 `gam.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?

Maximum Likelihood

Let $X_1, X_2, ..., X_n$ be a random vector of observations with joint density function $f(x_1, ...x_n|\theta)$. Then the likelihood of θ as a function of the observed values, $X_i = x_i$, is defined as,

$$\text{lik}(\theta) = f(x_1, ..., x_n|\theta).$$

Maximum Likelihood

- Usually think of θ as fixed and consider probability of changing outcomes $x_1, ...x_n$ given by $f(x_1, ...x_n|\theta)$.
- Likelihood of a parameter lik(θ): what probability does θ give the data?
  - For continuous variables, use the probability density.
  - Calculate $f(x_1, ..., x_n|\theta)$ 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 is the **maximum likelihood estimate** or the **MLE** (value that maximizes the likelihood).

Coding the Likelihood Function

Hint: Often easier to maximize $\log(\text{lik}(\theta))$ or the 'log-likelihood'.

- With independent data points $X_1, X_2, \ldots, X_n$ the likelihood is

$$\text{lik}(\theta) = \prod_{i=1}^{n} f(x_i|\theta).$$

- 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 `gam.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 parameters. HINT: Use `dgamma()`.
- Test your function on the cats heart weight data and parameter values $scale = 19$ and $shape = 0.5$.

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.

How do we maximize the likelihood?

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

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

cat.MM <- gam.MMest(cats$Hwt)

neg.gam.ll(cat.MM, cats$Hwt)
```

Maximum Likelihood

```
cat.MLE  <- nlm(neg.gam.ll, c(19, 1), data = cats$Hwt)$estimate
MM_args  <- list(shape = cat.MM["a"], scale = cat.MM["s"])
MLE_args <- list(shape = cat.MLE[1], scale = cat.MLE[2])

ggplot(cats) +
  geom_histogram(aes(x = Hwt, y = ..density..)) +
  geom_density(aes(x = Hwt), linetype = "dashed") +
  stat_function(aes(x = Hwt), fun = dgamma, args = MM_args, colo
  stat_function(aes(x = Hwt), fun = dgamma, args = MLE_args, col
```

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

Now we have an estimate of a model, how do we know it's a good fit?

Ways to check the 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.

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.

Compares Two Samples, Too

- 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

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

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

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")
```

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")
```

Checking Fit: Kolmogorov-Smirnoff 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-Smirnoff Test

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

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-Smirnoff 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
s

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

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

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

Bootstrap

Sampling Distribution

Question: What is the sampling distribution of our estimator?

- Sampling distribution is the distribution of the estimates around their true value.
- Sometimes we know the sampling distribution (e.g. sample mean for i.i.d. draws of normal rvs), but often computing the sampling distributions of estimates from data can be complex.
- If we could repeat an experiment over and over again, we could actually find a very good approximation to the sampling distribution. Usually not possible.

Bootstrap: Instead of repeatedly obtaining new, independent datasets from the population, we repeatedly obtain datasets from the sample itself, the original dataset.

Bootstrap

Idea of the bootstrap: Use simulation on the data we have to give an estimate of the sampling distribution of the estimator.

General Idea:

- Repeatedly resample the data with replacement and calculate the estimate each time.
- The distribution of these bootstrap estimates approximates the sampling distribution of the estimate.

Bootstrap Algorithm

Given sample data $X_1, X_2, \ldots, X_n$ from some known distribution up to the parameter θ and an estimator $\hat{\theta} = t(X_1, X_2, \ldots, X_n)$ of θ.

- Generate B bootstrap resamples $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_B$. Each bootstrap resample is obtained by sampling n times with replacement from the sample data. (This means datapoints can appear multiple times in any resample $\mathcal{B}_b$.)
- We call the resampled data $X_1^b, X_2^b, \ldots, X_n^b$ for each $b = 1, 2, \ldots, B$.
- Evaluate the estimator on each bootstrap sample: for each $b = 1, 2, \ldots, B$ calculate

$$\hat{\theta}_b = t(X_1^b, X_2^b, \ldots, X_n^b).$$

  That is, we estimate θ pretending $\mathcal{B}_b$ is the data.
- The theory says that the distribution of $\hat{\theta} - \theta$ is well-approximated by the distribution of $\hat{\theta}_b - \theta$.

Bootstrap Example

Let's use the bootstrap to investigate the sampling distribution of the method of moments estimator for the parameters in a gamma distribution.

```
# Recall

cat.MM <- gam.MMest(cats$Hwt)
cat.MM

# A single bootstrap resample from cats$Hwt

n      <- nrow(cats)
resamp <- sample(1:n, n, replace = TRUE)

head(sort(cats$Hwt))
head(sort(cats$Hwt[resamp]))

gam.MMest(cats$Hwt[resamp])
```

Bootstrap Example

```
# 1000 bootstrap resamples from cats$Hwt

B <- 1000
param_ests <- matrix(NA, nrow = B, ncol = 2)
colnames(param_ests) <- c("a", "s")

for (b in 1:B) {
  resamp           <- sample(1:n, n, replace = TRUE)
  param_ests[b, ] <- gam.MMest(cats$Hwt[resamp])
}

head(param_ests)
```

Bootstrap Example

```
# Use histogram to approx sampling distribution

param_ests <- data.frame(param_ests)

ggplot(param_ests) +
  geom_histogram(aes(x = a)) +
  geom_vline(xintercept = mean(a), col = "red")

ggplot(param_ests) +
  geom_histogram(aes(x = s)) +
  geom_vline(xintercept = mean(s), col = "red")

bootstrap.a <- mean(param_ests$a)
bootstrap.s <- mean(param_ests$s)
bootstrap.a
bootstrap.s
```

Bootstrap Example

```
# Use bootstrap replicates to estimate the
# standard error of the estimates

sd(param_ests$a)
sd(param_ests$s)

# Use bootstrap replicates to estimate 95 percent
# confidence intervals for the estimates

CI.a <- quantile(param_ests$a, probs = c(0.025, 0.975))
CI.s <- quantile(param_ests$s, probs = c(0.025, 0.975))
```

Bootstrap Example

```
ggplot(param_ests) +
  geom_histogram(aes(x = a)) +
  geom_vline(xintercept = mean(a), col = "red") +
  geom_vline(xintercept = CI.a[1], col = "red",
             linetype = "dashed") +
  geom_vline(xintercept = CI.a[2], col = "red",
             linetype = "dashed")

ggplot(param_ests) +
  geom_histogram(aes(x = s)) +
  geom_vline(xintercept = mean(s), col = "red") +
  geom_vline(xintercept = CI.s[1], col = "red",
             linetype = "dashed") +
  geom_vline(xintercept = CI.s[2], col = "red",
             linetype = "dashed")
```

Bootstrap Example

Let's convince ourselves this works by going back to the simple estimators for the normal distribution.

```
# Recall
set.seed(1)

samp_size  <- 100
samp1      <- rnorm(samp_size, mean = 10, sd = 3)
mmean_est1 <- mean(samp1)
```

Check Yourself

Tasks

- Calculate 1000 bootstrap resamples from the data `samp1` and for each calculate and store the sample estimate of the mean.
- Calculate the mean and standard deviation of your bootstrap resamples.
- The above is supposed to approximate the sampling distribution, but for the normal distribution, we know the sampling distribution is $\mathcal{N}(10, 9/100)$. How close are the bootstrap estimates of the mean and variance of the sampling distribution to the truth?