ADV MATH METHODS FOR MODELING

Lab Notes on Info-Metrics

Oriol Vallès Codina

April 15, 2020

Time:	W 8-9.50pm
Location:	D909
Instructor:	Oriol Vallès Codina
E-mail:	oriolvallescodina@newschool.edu
Office Hours:	TR 2.30-3.30pm, Room D1126, and W by appointment

1 Derivation of Boltzmann Distribution

The purpose of this derivation is to show, following Boltzmann, that one can derive a well-known physical law (the exponential shape of the probability distribution of energy states in an ideal gas) from mere statistical considerations. One first considers the constraints of the statistical object (the ensemble) in form of constant values for observable macroscopic quantities and then considers all the possible microscopic arrangements that support them. Each set of macroscopic constraints (temperature, pressure, volume, energy, number of particles, etc) defines particular ensembles (microcanonical, canonical, grand canonical, etc).

For instance, the microcanonical ensemble considers all the possible microscopic arrangements that are supported by the same macroscopic properties: a fixed value for energy E, number of particles N, and volume V (i.e. the constraints). Via the ergodic axiom, all such possible microscopic arrangements of the ensemble are considered equiprobable.

Boltzmann's brilliance led him to consider the number of microstates $W(\{N_k\})$ that are supported by N particles and total energy E that are divided into $k = 0, 1, ..., K$ bins of energy ϵ_k and then maximize number W subject to the ensemble constraints

$$E = \sum_k N_k \epsilon_k \tag{1}$$

$$N = \sum_k N_k \tag{2}$$

The kinetic theory of ideal gases envisions an ideal gas as composed of atoms or molecules in random drift within a volume and partaking in elastic collisions between each other where there is no loss of energy so that energy levels ϵ_k can be exchanged. We want to know the distribution of particles $W(\{N_k\})$ across $0, ..., K$ bins of energy levels $\epsilon_0, \epsilon_1, ..., \epsilon_K$, i.e. $W(\{N_0, N_1, ..., N_K\})$. In order to do so, we use the binomial formula to compute *combination*, that is, a selection of items from a collection, such that (unlike permutations) the order of selection does not matter. For a set of n elements, the number of k combinations is:

$$\binom{n}{k} = \frac{n(n-1)...(n-k+1)}{k(k-1)...1} = \frac{n!}{n!(n-k)!}$$

The total number of microstates will be the number of combinations of N_0 particles supported by the whole N particles at energy level ϵ_0, multiplied by the number of N_1 combinations of the remainder particles $N-N_0$ at energy level ϵ_1, multiplied by the number of N_2 combinations supported by the remainder $N-N_0-N_1$ particles at energy level ϵ_2, and so on until we reach maximum energy level ϵ_K:

$$W(\{N_k\}) = \frac{N!}{N_0!(N-N_0)!} \times \frac{(N-N_0)!}{N_1!(N-N_1-N_0)!} \times \frac{(N-N_1-N_0)!}{N_2!(N-N_2-N_1-N_0)!} \times ... \times \frac{(N-...-N_{K-1})!}{N_K!(N-...-N_{K-1})!} \tag{3}$$

so that consecutive terms at the denominator and numerator can vanish, yielding:

$$W(\{N_k\}) = W(\{N_0, N_1, ..., N_K\}) = \frac{N!}{\prod_k N_k!} = \frac{N!}{N_0!N_1!...N_K!} \tag{4}$$

Instead of maximizing $W(\{N_k\})$, we maximize its logarithm $\ln W$ for tractability, on which we can apply Stirling's formula for large numbers,

$$\ln N! \sim N \ln N - N$$

Hence,

$$\ln W = \ln(N!) - \sum_k \ln(N_k!) \sim N \ln N - N - \sum_k N_k \ln N_k + \sum_k N_k = N \ln N - \sum_k N_k \ln N_k \tag{5}$$

We compute now the total differential of $d \ln W$ subject to constraints (1) and (2), multiplied by parameters β and α:

$$dE = \sum_k \epsilon_k dN_k = 0 \tag{6}$$

$$dN = \sum_k dN_k = 0 \tag{7}$$

Knowing that N is a constant and its derivative is zero, the total differential is:

$$d \ln W = d(\underbrace{N \ln N}_{0} - \sum_k N_k \ln N_k) = -\sum_k (1 + \ln N_k) dN_k = -\underbrace{\sum_k dN_k}_{0} - \sum_k \ln N_k dN_k = -\sum_k \ln N_k dN_k \tag{8}$$

which we subject to the constraints introducing two terms that are equal to zero (6) and (7):

$$0 = d \ln W = -\sum_k (\alpha + \beta \epsilon_k + \ln N_k) \, dN_k \tag{9}$$

For each k component of the sum, the term within parentheses is zero, which yields:

$$\ln N_k = -\alpha - \beta \epsilon_k$$

$$N_k = e^{-\alpha - \beta \epsilon_k} = e^{-\alpha} e^{-\beta \epsilon_k}$$

We compute the normalization term $e^{-\alpha}$ or partition function with constraint (2),

$$N = \sum_k N_k = \sum_k e^{-\alpha} e^{-\beta \epsilon_k} = e^{-\alpha} \sum_k e^{-\beta \epsilon_k}$$

$$e^{-\alpha} = \frac{N}{\sum_k e^{-\beta \epsilon_k}}$$

The number of particles N_k at energy level ϵ_k will be:

$$N_k(\epsilon_k) = \frac{N e^{-\beta \epsilon_k}}{\sum_k e^{-\beta \epsilon_k}}$$

Now we can finally obtain the Boltzmann distribution, which gives the probability $p(\epsilon_k)$ that a system will be in a certain state as a function of that state's energy and the temperature of the system:

$$p(\epsilon_k) = \frac{N_k(\epsilon_k)}{N} = \frac{e^{-\beta\epsilon_k}}{\sum_k e^{-\beta\epsilon_k}} \tag{10}$$

where $\beta = \frac{1}{kT}$, k is Boltzmann's constant, and T is the temperature of the system.

2 Probability Theory

2.1 Basics

We have two random variables X and Y with corresponding state spaces N and M.

Joint frequencies depend on the values of all (both) variables:

$$p(x_i, y_j) = p_{ij} \geq 0; \quad \text{with } \sum_i^N \sum_j^M p_{ij} = 1 \tag{11}$$

Marginal frequencies sum frequencies over one variable:

$$p(x_i) = \sum_j^M p(x_i, y_j); \quad p(y_j) = \sum_i^N p(x_i, y_j) \tag{12}$$

Conditional frequencies take one variable as given:

$$p(x_i|y_j) = \frac{p(x_i, y_j)}{p(y_j)} = \frac{p(x_i, y_j)}{\sum_i^N p(x_i, y_j)}; \quad p(y_j|x_i) = \frac{p(x_i, y_j)}{p(x_i)} = \frac{p(x_i, y_j)}{\sum_j^M p(x_i, y_j)} \tag{13}$$

The **chain rule** of probability is:

$$P(X, Y) = P(X|Y)P(Y) = P(Y|X)P(X) \tag{14}$$

2.2 Example: Two-Way Relative Frequency Table

	Boys	**Girls**	**Totals**
Made	7; 21.2%	6; 18.2%	13; 39.4%
Missed	11; 33.3%	9; 27.3%	20; 60.6%
Totals	18; 54.5%	15; 45.5%	33; 100%

The joint probability of a boy missing a shot is $P(boy, missed) = 11/33$. The conditional probability of missing the shot *given being a boy* is the joint probability divided by the marginal probability of being a boy:

$$P(missed|boy) = \frac{P(boy, missed)}{P(boy)} = \frac{11/33}{18/33} = 11/18$$

The joint probability of a girl making a shot is $P(girl, made) = 6/33$. The conditional probability of being a girl given a shot was made is

$$P(girl|made) = \frac{P(girl, made)}{P(made)} = \frac{6/33}{13/33} = 6/13$$

2.3 Bayes' Rule

In the Bayesian (or epistemological) interpretation, probability measures a "degree of belief." Bayes' theorem then links the degree of belief in a proposition before and after accounting for evidence. In the frequentist interpretation, probability (understood as a frequency) measures a "proportion of outcomes", where an experiment is performed many times.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{15}$$

Prove Bayes' Rule through the chain rule. $P(A|B)$ is the posterior, $P(B|A)$ is the likelihood, $P(A)$ is the prior, and $P(B) = \sum P(B|A)P(A)$ works as a normalizing constant and is known as evidence.

2.3.1 Example 1

Using the previous example, the probability of a boy given that the shot was missed is

$$P(boy|missed) = \frac{P(boy, missed)}{P(missed)} = \frac{11/33}{20/33} = 11/20$$

We could have computed this probability from $P(missed|boy)$ with Bayes' Rule:

$$P(boy|missed) = \frac{P(missed|boy)P(boy)}{P(missed)} = \frac{P(missed|boy)P(boy)}{P(missed|boy)P(boy) + P(missed|girl)P(girl)} = \frac{11/18 \times 18/33}{20/33} = 11/20$$

2.3.2 Example 2

The Bayes Rule is especially relevant in medical diagnosis. For instance, we have a drug test with a 99% true-positive rate ("sensibility": i.e. 99% drug users give a true positive) and a 99% true-negative rate ("specificity": i.e. 99% non-users give a true negative). We want to know the probability of being a user given that the test was positive, i.e. $P(user|+)$. We know that 0.5% of the population uses this drug. Hence:

$$P(user|+) = \frac{P(+|user)P(user)}{P(+|user)P(user) + P(+|non-user)P(non-user)}$$

where $P(+|user)$ is the true-positive rate, $P(+|non-user)$ is 1 minus the true-negative rate $P(-|non-user)$, and the denominator equals $P(+)$.

$$P(user|+) = \frac{0.99 \times 0.005}{0.99 \times 0.005 + 0.01 \times 0.995} \sim \frac{1}{3}$$

Only one third of the positives will be positive! From 1000 people, we know that 5 will be users. From 995 non-users, the true-positive rate is 0.99 so there will be around 10 false positives. From 5 users, the true-negative rate is 0.99 so all the positives will be true. In total we will have 15 positives, with 2/3 being false.

The importance of specificity in this example can be seen by calculating that even if sensitivity is raised to 100% and specificity remains at 99% then the probability of the person being a drug user only rises from 33.2% to 33.4%, but if the sensitivity is held at 99% and the specificity is increased to 99.5% then the probability of the person being a drug user rises to about 49.9%.

2.3.3 Example 3: Making inferences about racial disparities in police violence

In PNAS Letters:

A recent PNAS study, Johnson et al., investigates the role of race in fatal police shootings. Unlike previous studies which focused on victim race alone, the paper features original data about the race of officers who

use deadly force and offers a rare accounting of other shooting attributes that contextualize fatal encounters. Johnson et al. discuss possible "discrimination by White officers", but conclude racial diversity in police agencies brings limited benefits–a claim cited by major news outlets and in US Congressional testimony, inflaming an already contentious policy debate.

Despite the value of this much-needed research, its approach is mathematically incapable of supporting its central claims. In this letter, we clarify the gap between what Johnson et als study asserts and what it actually estimates, as well as the implications of that difference for policymaking and future scholarship on race and policing.

3 Information Theory

> The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point. [Shannon, 1948]

Statistical communication theory is generally regarded as having been founded by Shannon in 1948 and Wiener in 1949, who conceived of the communication situation as one in which a signal chosen from a specified class is to be transmitted through a channel, but the output of the channel is not determined by the input. Instead, the channel is described statistically by giving a probability distribution over the set of all possible outputs for each permissible input. For a review of the concepts of information theory, please read chapter 1 of Mackay [MacKay and Mac Kay, 2003].

In reality, knowing the specific location and momentum of a small particle like an atom is in fact epistemologically and technically impossible (Heisenberg's uncertainty principle in quantum mechanics). Hence we should think of atoms as quantum (probability) fields (of position and momentum). In the view of Jaynes (1957), thermodynamic entropy, as explained by statistical mechanics, should be seen as an application of Shannon's information theory: the thermodynamic entropy is interpreted as being proportional to the amount of further Shannon information needed to define the detailed microscopic state of the system, that remains uncommunicated by a description solely in terms of the macroscopic variables of classical thermodynamics, with the constant of proportionality being just the Boltzmann constant. Adding heat to a system increases its thermodynamic entropy because it increases the number of possible microscopic states of the system that are consistent with the measurable values of its macroscopic variables, making any complete state description longer.

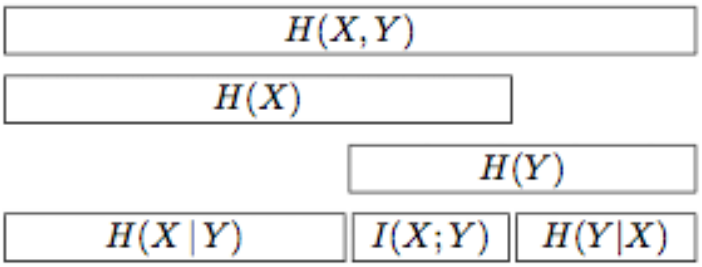


Figure 1: **The relationship between joint information, marginal entropy, conditional entropy and mutual entropy.**

3.1 Self-Information

In information theory, information content, self-information, or surprisal of a random variable or signal is the amount of information gained when it is sampled. Let X be a random variable with discrete probability mass function $p(x_i)$ over $i = 1, ..., N$ states.

The **self-information** or information content of measuring X as outcome x_i is defined as:

$$I(x_i) \equiv -\log(p(x_i)) = \log\left(\frac{1}{p(x_i)}\right) \tag{16}$$

The expected value of the information content of X is the entropy H.

3.2 Entropy as Uncertainty: A Measure of Information

Entropy effectively bounds the performance of the strongest lossless compression possible. Hence it defines the **minimum expected length of the message**. See also Kolmogorov complexity. In practice, compression algorithms deliberately include some judicious redundancy in the form of checksums to protect against errors.

$$H(X) = -\sum_{i}^{N} p_i(X) \log p_i(X) = \langle -\log p(X) \rangle = \langle I(X) \rangle \tag{17}$$

3.3 Units

The units of entropy relate to the base of the logarithm used in the entropy formula, which correspond directly to the length of the vocabulary (ie the number of states N the random variable can attain in each observation).

3.3.1 Example 1: Computer Language in Bits

The bit, or binary digit, is the smallest piece of information that can be processed by a computer. In many systems, such as the American Standard Code for Information Interchange, it can take 8 bits, or 1 byte, to make one character–a letter, numeral or symbol. A bit is either a 1 or 0, a "yes" or "no," or an "on" or "off." The frequency of a signal voltage is measured in cycles per second. One hertz is one complete cycle per second. While higher frequency can mean a faster system, a truer measurement of communication speed is bit rate. Hertz Bits Most data communications systems operate at millions of cycles per second, or megahertz. In high frequencies, such as values in the MHz range, the time the cycle requires is measured in minute fractions of a second.

3.3.2 Example 2: Protein Transcription from DNA (Genetic Information)

> A critical history of technology would show how little any of the inventions of the 18th century are the work of a single individual. Hitherto there is no such book. Darwin has interested us in the history of Nature's Technology, i.e., in the formation of the organs of plants and animals, which organs serve as instruments of production for sustaining life. Does not the history of the productive organs of man, of organs that are the material basis of all social organisation, deserve equal attention? [Marx, 1976]

Important from a complex-systems perspective and historical materialism. Biological evolution is all about information-processing organic systems.

Genes that provide instructions for proteins are expressed in a two-step process. In transcription, the DNA sequence of a gene is "rewritten" in RNA. In eukaryotes, the RNA must go through additional processing steps to become a messenger RNA, or mRNA. In translation, the sequence of nucleotides in the mRNA is "translated" into a sequence of amino acids in a polypeptide (protein chain). Cells decode mRNAs by reading their nucleotides in groups of three, called codons.

- Basic RNA Coding Unit is 3 nucleotides (with 4 possible nucleotides A, C, G, U). Hence we have no bits but "quadrits" ie 4 possible options in each question.

- Size of a codon, the transcription unit from RNA to aminoacids (the basic components of proteins): 3 nucleotides. Why is it so?
- Three is the minimum number of nucleotides per codon needed to encode the existing 20 aminoacids. 20 amino acids are encoded by combinations of 4 nucleotides.
- If a codon were two nucleotides, the set of all combinations could encode only 4x4 = 16 amino acids.
- With three nucleotides, the set of all combinations can encode 4x4x4 = 64 amino acids (i.e. 64 different combinations of four nucleotides taken three at a time), which is greater than the existing 20 aminoacids.
- More Information https://www.khanacademy.org/science/biology/gene-expression-central-dogma/central-dogma-transcription/a/the-genetic-code-discovery-and-properties

3.4 Uniform Distribution has Maximum Uncertainty

$H(p) \leq \log N$ with equality if and only if all $p_i = 1/N$.

3.5 Joint Uncertainty

Let X and Y be random variables with joint probability function $p(x_i, y_j)$ over states $i \in N$ and $j \in M$. We have an experiment with $N \times M$ possibles outcomes of probability $p(x_i, y_j)$. Then the joint uncertainty is:

$$H(X,Y) = -\sum_{i}^{N}\sum_{j}^{M} p(x_i, y_j) \log p(x_i, y_j) \tag{18}$$

3.6 Independent Events Have Additive Uncertainty

$$H(X,Y) \leq H(X) + H(Y) \tag{19}$$

with equality if and only if X and Y are independent.

3.7 Conditional Uncertainty

Let X and Y be two random variables over states N and M. If we are given that $X = x_i$, then the distribution of Y is characterized by the set of conditional probabilities $p(y_i|x_i)$. Conditional uncertainty of Y given that $X = x_i$ is:

$$H(Y|X = x_i) = -\sum_{j=1}^{M} p(y_j|x_i) \log p(y_j|x_i) \tag{20}$$

The *conditional uncertainty of Y given X* is defined as a weighted average of the uncertainties $H(Y|X = x_i)$ (where the weights are $p(x_i)$), yielding:

$$H(Y|X) = -\sum_{i=1}^{N}\sum_{j=1}^{M} \underbrace{p(x_i, y_j)}_{\text{joint}} \log p(y_j|x_i) = -\sum_{i=1}^{N}\sum_{j=1}^{M} p(x_i, y_j) \log \frac{p(x_i, y_j)}{p(x_i)} = \langle -\log p(Y|X) \rangle \tag{21}$$

- **Conditional entropy equals zero** $H(Y|X) = 0$ if and only if the value Y is completely determined by the value of X.
- **Conditional entropy of independent random variables** $H(Y|X) = H(Y)$ if and only if X and Y are independent random variables (conversely, $H(X|Y) = H(X)$.
- **Chain Rule** $H(Y|X) = H(X,Y) - H(X)$

- **Bayes' Rule** $H(Y|X) = H(X|Y) - H(X) + H(Y)$
- $H(Y|X) \leq H(Y)$

3.8 Kullback-Leibler Divergence (Relative Entropy)

The relative entropy or discrimination information between two proper probability distributions p and q for random variables X and Y, i.e. the Kullback-Leibler divergence from q to p is:

$$D_{KL}(p||q) = -\sum_{i}^{N} p_i(x) \log \frac{q(x_i)}{p(x_i)} = \sum_{i}^{N} p(x_i)[\log p(x_i) - \log q(x_i)] = \langle \log p(x_i) - \log q(x_i) \rangle = \underbrace{H_q(p)}_{\text{cross entropy}} - H(p) \tag{22}$$

- $D_{KL}(p||q) \geq 0$ with equality if $p_i = q_i \forall k$.
- **Cross entropy** $H_q(p) = H(p) + D_{KL}(p||q) = \sum_i^N p(x_i) \log\left(\frac{1}{q(x_i)}\right)$ tells us the average length of a message from one distribution using the optimal coding length of another where $-\log q(x_i)$ is the optimal coding length for messages coming from the q distribution while $p(x_i)$ is the cost of sending message x_i to p. distribution.
- **Joint entropy** $H(X, Y)$ tells us the average cost of sending multiple messages simultaneously. Or perhaps more intuitively, the average cost of sending a single message that has multiple parts.

The measure captures the informational distance between two probability distributions, but it is not a true metric (mathematical distance) because it is not symmetric, i.e:

$$D_{KL}(p||q) \neq D(q||p)$$

. In other words, it reflects the gain in information in q resulting from the additional information given by p. In the context of machine learning, $D_{KL}(p||q)$ is often called the information gain achieved if q is used instead of p.

Expressed in the language of Bayesian inference, $D(p||q)$ is a measure of the information gained when one revises one's beliefs from the prior probability distribution q to the posterior probability distribution p. In other words, it is the amount of information lost when q is used to approximate p. p typically represents the "true" distribution of data, observations, or a precisely calculated theoretical distribution, while q typically represents a theory, model, description, or approximation of p. In order to find a distribution q that is closest to p, we can minimize KL divergence.

A common goal in Bayesian experimental design is to maximize the expected KullbackLeibler divergence between the prior $p(A)$ and the posterior $p(A|B)$.

3.9 Mutual Information

Information is a reduction in uncertainty. We define the *information conveyed about X by Y* as:

$$I(X, Y) = H(X) - H(X|Y) = \sum_{i}^{N} \sum_{j}^{M} p(x_i, y_j) \log \frac{p(x_i, y_j)}{p(x_i)q(x_j)} = \langle -\log \frac{p(x_i)}{p(x_i|y_j)} \rangle = H(Y) - H(Y|X) \tag{23}$$

The information conveyed about X by Y may also be interpreted as the difference between the minimum average number of "yes or no" questions required to determine the result of one observation of X *before* Y is observed and the minimum average number of such questions required *after* Y is observed.

Information is symmetric: $I(X, Y) = (Y, X)$. The information conveyed about X by Y is the same that the information conveyed about Y by X.

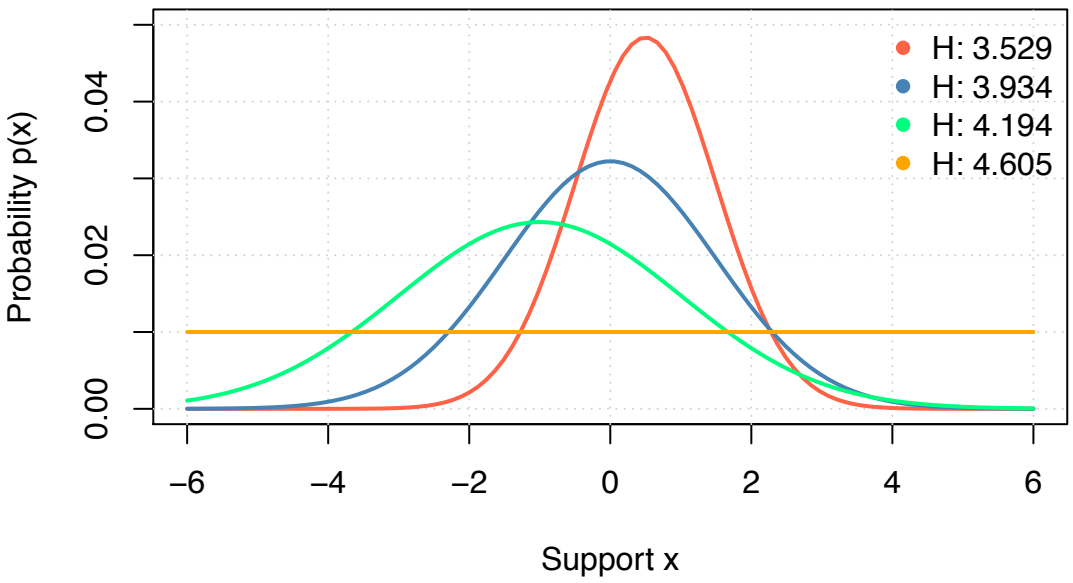
- $H(X,Y) = H(X|Y) + H(Y|X) + I(X,Y)$
- $I(X,Y) \leq H(X)$

Pointwise mutual information (PMI) is a statistical measure of association. It refers to single events, while MI refers to the average of all possible events:

$$\text{pmi}(x;y) \equiv \log \frac{p(x,y)}{p(x)} = \log \frac{p(x|y)}{p(x)}$$

3.10 Example 1

```
# Compute Entropies
entropy<-function(f){return(-sum(f*log(f)))}

# Initialization
colors <- c("tomato", "steelblue", "springgreen", "orange", "black")
x<-seq(-6,6,length=100) # Support
plot(NA,xlim=c(min(x),max(x)),ylim=c(0,0.05),xlab='Support x',ylab='Probability p(x)')
grid()

# Distributions with increasing entropy (due to increasing sd)
p1<-dnorm(x,mean=0.5,sd=1) # Gaussian with mean 0.5, sd 1
p1<-p1/sum(p1)
lines(x,p1,col=colors[1],lwd=2)

p2<-dnorm(x,mean=0,sd=1.5)
p2<-p2/sum(p2)
lines(x,p2,col=colors[2],lwd=2)

p3<-dnorm(x,mean=-1,sd=2)
p3<-p3/sum(p3)
lines(x,p3,col=colors[3],lwd=2)

p4<-rep(1/length(x),length(x)) # Uniform Distribution
lines(x,p4,col=colors[4],lwd=2)

# Add Legend
legend('topright',paste('H:',sapply(list(p1,p2,p3,p4),function(x){round(entropy(x),3)})),pch=16,col=colo
```




```
# Entropy equals the weighted mean (expected value) of the (negative) logarithm
sapply(list(p1,p2,p3,p4),function(x){weighted.mean(log(x),x)})
```

[1] -3.529152 -3.934106 -4.194332 -4.605170

3.11 Example 2

Check Lab 2 Excel Sheet.

3.12 Example: Kullback-Leibler Divergence

- $D_{KL}(Observed||Uniform) = 0.338$
- $D_{KL}(Observed||Binomial) = 0.477$
- $D_{KL}(Binomial||Observed) = 0.330$ (KL divergence is not a distance)

As we can see the information lost by using the binomial approximation is greater than using the uniform approximation. If we have to choose one to represent our observations, we're better off sticking with the uniform approximation. You can compute yourself the KL divergences using the data from here.

3.13 Central Moments of a Distribution

In the linear constraints case of entropy maximization, the constraints take the form of **central moments of the distribution**. If the function represents *physical density*, then the zeroth moment is the total mass, the first moment divided by the total mass is the center of mass, and the second moment is the rotational inertia. If the function is a *probability distribution*, then the zeroth moment is the total probability (i.e. one),

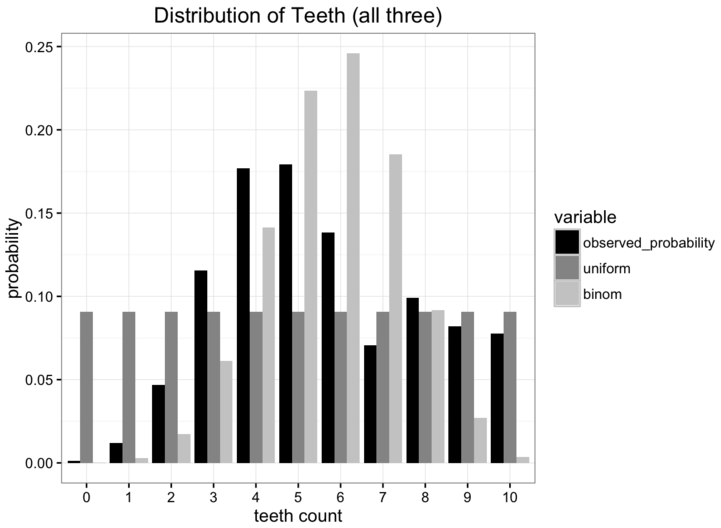


Figure 2: **Approximating uniform and binomial distributions to an observed distribution:** example Here

the first moment is the expected value, the second central moment is the variance, the third standardized moment is the skewness, and the fourth standardized moment is the kurtosis.

The nth central moment of a random variable X is

$$\mu_n(X) \equiv \langle (X - \langle X \rangle)^n \rangle \tag{24}$$

3.13.1 Zeroth Moment: Normalization

$$\sum_i^N p_i = 1 \tag{25}$$

3.13.2 First Moment: Expected Value

$$\mathrm{E}(X) \equiv \langle X \rangle \equiv \sum_i^N p_i x_i \tag{26}$$

3.13.3 Second Moment: Variance

$$\mathrm{var}(X) = \langle (X - \langle X \rangle)^2 \rangle = \sum_i^N p_i (x_i - \langle X \rangle)^2 = \left(\sum_i^N p_i x_i^2 \right) - \langle X \rangle^2 \tag{27}$$

If values are equally likely (i.e $p_i = 1/N$),

$$\mathrm{var}(X) = \frac{1}{N} \sum_i^N (x_i - \langle X \rangle)^2 \tag{28}$$

The variance is also typically designated as σ^2 (it is also the square of the standard deviation). Informally, it measures how far a set of random numbers are spread out from their average value.

The variance can also be thought of as the covariance of a random variable with itself:

$$\mathrm{var}(X) = \mathrm{cov}(X, X) \tag{29}$$

This expression for the variance is useful:

$$\mathrm{var}(X) = \langle (X - \langle X \rangle)^2 \rangle = \langle X^2 \rangle - \langle X \rangle^2 \tag{30}$$

3.13.4 Third Moment: Skewness

In probability theory and statistics, skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean.

$$\tilde{\mu_3} = \langle \left(\frac{X - \mu}{\sigma} \right)^3 \rangle \tag{31}$$

3.13.5 Fourth Moment: Kurtosis

In probability theory and statistics, kurtosis (from Greek: , kyrtos or kurtos, meaning "curved, arching") is a measure of the "tailedness" of the probability distribution of a real-valued random variable.

$$\mathrm{Kurt}(X) = \langle \left(\frac{X - \mu}{\sigma} \right)^4 \rangle = \frac{\langle (X - \mu)^4 \rangle}{\langle (X - \mu)^2 \rangle^2} = \frac{\mu_4}{\sigma^4} \tag{32}$$

4 Types of Distributions

4.1 Uniform

The probability distribution of a continuous uniform distribution between a and b is:

$$f(x) = \frac{1}{b - a} \quad \text{for } a \leq x \leq b \tag{33}$$

4.2 Exponential

The exponential distribution is the probability distribution of the time between events in a Poisson process, i.e., a process in which events occur continuously and independently at a constant average rate λ:

$$f(x) = \lambda e^{-\lambda x} \quad \text{for } x \geq 0 \tag{34}$$

The expected value of the exponential distribution is $1/\lambda$ and variance is $1/\lambda^2$. If you receive phone calls at an average rate of 2 per hour, then you can expect to wait half an hour for every call. Its most characteristic property is **memorylessness**.

The exponential probability distribution of wages, predicted by the statistical equilibrium theory of a labor market developed by Foley in 1996, is supported by empirical data on income distribution in the USA for the majority (about 97%) of population [Drăgulescu and Yakovenko, 2001]. In addition, the upper tail of income distribution (about 3% of population) follows a power law and expands dramatically during financial bubbles, which results in a significant increase of the overall income inequality. A mathematical analysis of the empirical data clearly demonstrates the two-class structure of a society, as pointed out Karl Marx and recently highlighted by the Occupy Movement.

4.3 Power Law (Pareto, Zipf)

A power law is a functional relationship between two quantities, where a relative change in one quantity results in a proportional relative change in the other quantity, independent of the initial size of those quantities: one quantity varies as a power of another. Its characteristic property is **scale invariance**.

$$f(x) = ax^{-k} \quad \text{for } x \geq 0 \tag{35}$$

A power-law has a well-defined mean only if $k > 2$, and it has a finite variance only if $k > 3$; most identified power laws in nature have exponents such that the mean is well-defined but the variance is not, implying they are capable of black swan behavior.

To visualize Zipf's law, we can take a country (e.g., the United States) and order the cities3 by population (e.g., New York as first, Los Angeles as second). Drawing a graph, we place the log of the rank on the y axis (New York has log rank ln1, and Los Angeles has a log rank log 2), and on the x axis, we place the log of the population of the corresponding city, which is called the size of the city. Figure 3 (following Krugman 1996 and Gabaix 1999a) shows the resulting plot for the 135 American metropolitan areas listed in the Statistical Abstract of the United States for 1991.

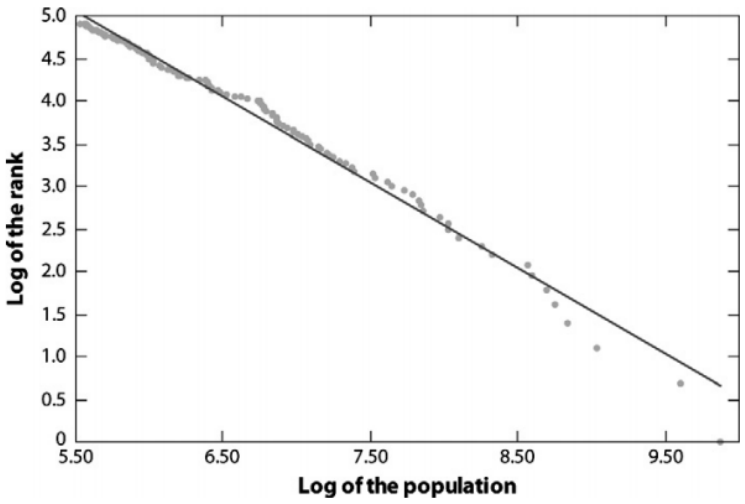


Figure 3: **Log size versus log rank of the 135 American metropolitan areas listed in the Statistical Abstract of the United States for 1991**. Figure taken from Gabaix 1999a [Gabaix, 2009]

The distributions of a wide variety of physical, biological, and man-made phenomena approximately follow a power law over a wide range of magnitudes: these include the sizes of craters on the moon and of solar flares,the foraging pattern of various species, the sizes of activity patterns of neuronal populations, the frequencies of words in most languages, frequencies of family names, the species richness in clades of organisms, the sizes of power outages and earthquakes, criminal charges per convict, volcanic eruptions, human judgements of stimulus intensity and many other quantities. Few empirical distributions fit a power law for all their values, but rather follow a power law in the tail.

In physics, self-organized criticality (SOC) is a property of dynamical systems that have a critical point as an attractor [Bak et al., 1988]. Their macroscopic behavior thus displays the spatial or temporal scale-invariance characteristic of the critical point of a phase transition, but without the need to tune control parameters to a precise value, because the system, effectively, tunes itself as it evolves towards criticality. Its characteristic distribution is in effect a power law.

The Pareto distribution is also a continuous power-law probability distribution that is used in description of social, scientific, geophysical, actuarial, and many other types of observable phenomena. Originally applied to describing the distribution of wealth in a society, fitting the trend that a large portion of wealth is held by a small fraction of the population, the Pareto distribution has colloquially become known and referred to as the Pareto principle, or "80-20 rule", and is sometimes called the "Matthew principle". This rule states that, for example, 80% of the wealth of a society is held by 20% of its population.

4.4 Normal (Gaussian)

A normal distribution is a type of continuous probability distribution for a real-valued random variable. The general form of its probability density function is

$$f(x) = \frac{1}{\sqrt{4\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{4\sigma^2}} \tag{36}$$

The parameter μ is the mean or expectation of the distribution (and also its median and mode); σ is its standard deviation and σ^2 its variance.

Normal distributions are important in statistics and are often used in the natural and social sciences to represent real-valued random variables whose distributions are not known. Their importance is partly due to the central limit theorem, which states that, under some conditions, the average of many samples (observations) of a random variable with finite mean and variance is itself a random variable whose distribution converges to a normal distribution as the number of samples increases.

4.5 Log-normal (Gibrat)

A log-normal distribution results from the logarithm of a random variable following a Gaussian (normal) distribution.

Gibrat's law (sometimes called Gibrat's rule of proportionate growth or the law of proportionate effect) is a rule defined by Robert Gibrat (19041980) in 1931 stating that the proportional rate of growth of a firm is independent of its absolute size. The law of proportionate growth gives rise to a distribution that is log-normal [Kalecki, 1945]. Gibrat's law predicts that firm growth is a purely random effect and therefore should be independent of firm size.

Gibrat's law is also applied to cities size and growth rate, where proportionate growth process may give rise to a distribution of city sizes that is log-normal, as predicted by Gibrat's law. While the city size distribution is often associated with Zipf's law, this holds only in the upper tail, because empirically the tail of a log-normal distribution cannot be distinguished from Zipf's law. A study using administrative boundaries (places) to define cities finds that the entire distribution of cities, not just the largest ones, is log-normal. But this last claim that the lognormal distribution cannot be rejected has been shown to be the result of a statistics with little power: the uniformly most powerful unbiased test comparing the lognormal to the power law shows unambiguously that the largest 1000 cities are distinctly in the power law regime.

5 The Lagrangian Method for Constrained Optimization

The Lagrange multiplier technique lets you find the maximum or minimum of a multivariable function $f(x, y, ...)$ when there is some constraint on the input values you are allowed to use. This technique applies to constraints of the form:

$$g(x, y, ...) = c \tag{37}$$

where g is another multivariable function with the same input space as f and c is some constant. The core idea is to look for points where the contour lines of f and g are tangent to each other (the same problem of finding points where the *gradient* vectors of f and g are parallel to each other). The technique can be boiled down into setting the gradient of a certain function, called the Lagrangian, equal to the zero vector.

In Lagrangian mechanics (a reformulation of classical, Newtonian mechannics), the trajectory of a system of particles is derived by solving the Lagrange equations in one of two forms: either the Lagrange equations of the first kind, which treat constraints explicitly as extra equations, often using Lagrange multipliers; or the Lagrange equations of the second kind, which incorporate the constraints directly by judicious choice of generalized coordinates. In each case, a mathematical function called the Lagrangian is a function of the generalized coordinates, their time derivatives, and time, and contains the information about the dynamics of the system. The Lagrangian is defined here as $L \equiv T - V$ (kinetic energy minus potential energy). If time does not appear explicitly in the Lagrangian, then energy is conserved.

The equivalent in neoclassical economics is the procedure of constrained optimization of **utility** that underpins the marginal theory of value following Walras [Mirowski, 1989, 1991].

5.1 Small Example

Suppose one wants to maximize this function defined on the two-dimensional plane:

$$f(x, y) = 2x + y$$

under the constraint

$$x^2 + y^2 = 1$$

In other words, for which (x, y) on the unit circle is the value $2x + y$ biggest? The Lagrangian takes the form:

$$\mathcal{L}(x, y, \lambda) = \underbrace{2x + y}_{f(x,y)} + \lambda(x^2 + y^2 - 1)$$

We derive the first-order conditions (FOC) by partially differentiating the Lagrangian $\mathcal{L}$ with respect to x, y, λ and set them to zero:

$$\frac{\partial \mathcal{L}}{\partial x} = 2 - 2\lambda x = 0 \rightarrow x = \frac{1}{\lambda}$$

$$\frac{\partial \mathcal{L}}{\partial y} = 1 - 2\lambda y = 0 \rightarrow y = \frac{1}{2\lambda}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = x^2 + y^2 - 1 = 0 \rightarrow x = \pm\sqrt{1 - y^2}$$

By solving this set of three equations and three variables, we find two points $(x, y, \lambda) = (\pm 2/\sqrt{5}, \pm 1/\sqrt{5}, \pm\sqrt{5}/2)$. The positive point is the maximum and the negative point is the minimum of function f.

6 Lagrangian Methods for Constrained Optimization of Entropy

A common complaint against mainstream economics is the use of extraneous assumptions in order to make formal theory tractable. The elegance of the method of entropy maximization is that it *systematizes* the use of assumptions –formalized as constraints– and allows evaluating how much information each of which brings to the model. In this direction, it is a mathematical formalization of Ockham's Razor. Starting from an initial position of maximum ignorance/uncertainty (i.e. the uniform distribution), we subsequently add constraints that bring information to the model in a systematic fashion – which we can compute via the Kullback-Leibler cross entropy.

6.1 Example 0: Expected value (mean) is known

The constrained optimization problem is defined now on **entropy** H as objective function, defined on the space of discrete distribution probabilities $p_i(x)$ for the random variable X labeled over states $i = 1, ..., N$, under **constraints** defined following the distribution moments of p_i: normalization and mean.

In case we know the mean $\langle X \rangle = \mu$ of the random variable, the maximization problem is [Golan, 2018, p.167]:

$$\begin{aligned} \underset{P}{\text{maximize}} \quad & H(P) = -\sum_i p_i \log p_i \\ \text{subject to} \quad & \langle X \rangle \equiv \sum_i p_i x_i = \mu; \\ & \sum_i p_i = 1 \end{aligned}$$

$$\mathcal{L}(p, \lambda_0, \lambda_1) = \underbrace{-\sum_i p_i \log p_i}_{\text{entropy}} + \underbrace{(\lambda_0 - 1)(\sum_i p_i - 1)}_{\text{normalization}} + \underbrace{\lambda_1 \left(\mu - \sum_i p_i x_i \right)}_{\text{expected value}} \tag{38}$$

The first order conditions follow from differentiating the Lagrangian with respect to its parameters p_i, λ_0 and λ_1 and allow us to define the solution for p_i:

$$\frac{\partial \mathcal{L}}{\partial p_i} = -\log p_i - 1 - (\lambda_0 - 1) + \lambda_1 x_i = 0 \tag{39}$$

Solution p_i will have an exponential form due to the derivative of entropy:

$$\log p_i = -\lambda_0 - \lambda_1 x_i \rightarrow p_i = e^{-\lambda_0 - \lambda_1 x_i} \tag{40}$$

The FOC for λ_0 establishes normalization, where Ω is called the **partition function**. In physical statistical mechanics, the partition function describes the partitioning among different microstates and serves as a generator function for all manner of results regarding a process:

$$\frac{\partial \mathcal{L}}{\partial \lambda_0} = \sum_i p_i - 1 = 0 \tag{41}$$

$$\sum_i p_i = \sum_i e^{-\lambda_0 - \lambda_1 x_i} = \sum_i e^{-\lambda_0} e^{-\lambda_1 x_i} = e^{-\lambda_0} \sum_i e^{-\lambda_1 x_i} = 1 \tag{42}$$

$$\Omega(\lambda_1) = e^{-\lambda_0} = \frac{1}{\sum_i e^{-\lambda_1 x_i}} \tag{43}$$

$$p_i = \frac{e^{-\lambda_1 x_i}}{\sum_i e^{-\lambda_1 x_i}} = \frac{e^{-\lambda_1 x_i}}{\Omega(\lambda_1)} \tag{44}$$

The FOC for λ_1 establishes the rate λ_1 in terms of the expected value of X, μ:

$$\frac{\partial \mathcal{L}}{\partial \lambda_1} = \sum_i p_i x_i - \mu = 0 \tag{45}$$

$$\sum_i x_i p_i = \sum_i \frac{x_i e^{-\lambda_1 x_i}}{\sum_i e^{-\lambda_1 x_i}} = \frac{\sum_i x_i e^{-\lambda_1 x_i}}{\Omega(\lambda_1)} = \mu \tag{46}$$

which yields an implicit equation for λ_1 that is solved numerically.

6.2 Example 1: A Coin Toss

What is the probability p_H of one unbiased coin toss falling heads? Our intuition says 50%, instinctively following the maximum-entropy approach. However, we cannot toss a coin infinite times to check what is its frequency following a frequentist approach. Let's try to prove that $p_H = 50\%$ using maxent.

We have two possible events, heads and tails, with probabilities p_H and p_T respectively. Since they are probabilities, we know that

$$\sum_{i=H,T} p_i = p_T + p_H = 1 \tag{47}$$

This is our only constraint: normalization. Entropy H of the distribution p_i is:

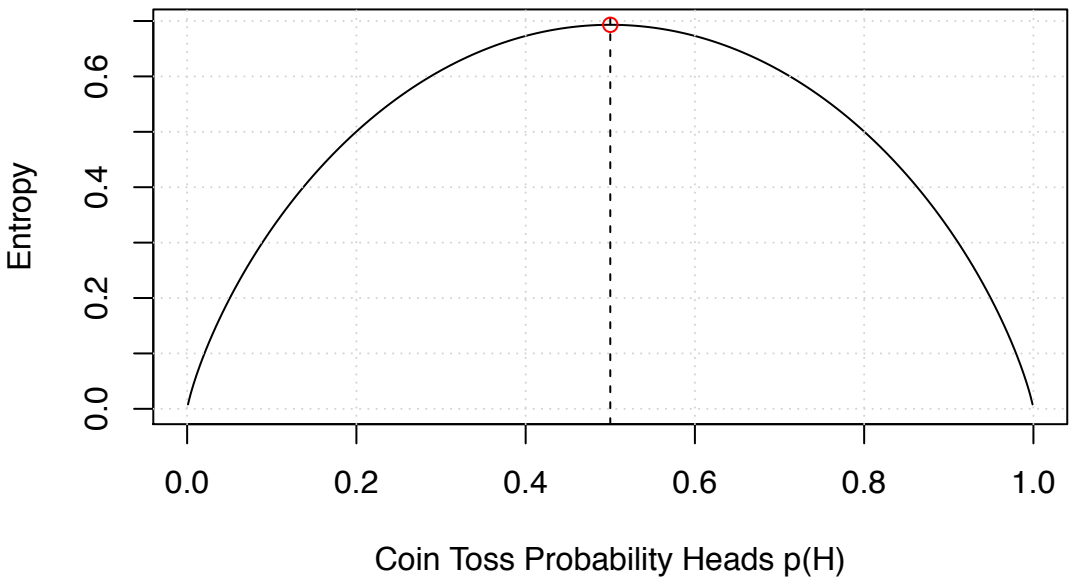
$$H(p) = H(p_H, p_T) = -p_T \log p_T - p_H \log p_H \tag{48}$$

Since we know that $p_T + p_H = 1$, then $p_T = 1 - p_H$, so that entropy is now (where we define p_H as x for simplicity):

$$H(x) = -x \log x - (1 - x) \log x \tag{49}$$

We can plot this function:

```
x<-seq(0,1,by=0.001)
plot(x,-x*log(x)-(1-x)*log(1-x),type='l',xlim=c(0,1),ylim=c(0,log(2)),xlab='Coin Toss Probability Heads
grid()
abline(v=0.5,lty=2)
points(0.5,-0.5*log(0.5)-0.5*log(0.5),col='red')
```



Let's compute the maximum of $H(p_H, p_T)$ using the Lagrangian multiplier method, with only normalization (47) as the single constraint:

$$\mathcal{L}(p_H, p_T, \lambda) = -p_H \log p_H - p_T \log p_T + (1-\lambda)(p_H + p_T - 1) \tag{50}$$

The first-order conditions are the following:

$$\frac{\partial \mathcal{L}}{\partial p_H} = -1 - \log p_H + 1 - \lambda = 0 \rightarrow p_H = e^{-\lambda}$$

$$\frac{\partial \mathcal{L}}{\partial p_T} = -1 - \log p_T + 1 - \lambda = 0 \rightarrow p_T = e^{-\lambda} = p_H$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = 1 - p_H - p_T = 0 \rightarrow 2p_H = 1 \rightarrow p_H = 0.5$$

Hence, entropy $H(p)$ has a maximum at $p_T = p_H = 0.5$ (the uniform distribution), as we expected. At that point, entropy equals $\log 2$.

6.3 Example 2: Generalizing a coin toss to N outcomes

Now instead of two events (heads and tails) we have N potential outcomes, with probabilities p_i for $i = 1, ..., N$. The Lagrangian takes now the form:

$$\mathcal{L}(p_1, ..., p_N, \lambda) = -\sum_i^N p_i \log p_i - (\lambda - 1)\left(\sum_i^N p_i - 1\right) \tag{51}$$

The first-order conditions are the following:

$$\frac{\partial \mathcal{L}}{\partial p_i} = -1 - \log p_i + 1 - \lambda = 0 \rightarrow p_i = e^{-\lambda} \quad \text{for } i = 1, ..., N$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = \sum_i^N p_i - 1 = 0$$

Since all probabilities p_i are equal to $e^{-\lambda}$, we can write

$$\sum_i^N p_i = Np = 1$$

and thus

$$p_i = \frac{1}{N}$$

which is the uniform distribution for discrete outcomes, with entropy is $H = -\sum_i \frac{1}{N} \log 1N = \log N$.

6.4 Numerical Example with $N = 3$ and mean 2.5

Now let's assume that we also know that $N = 3$, outcomes are $x_1 = 1$, $x_2 = 2$, and $x_3 = 3$, and the observed mean is $\langle x \rangle = 2.5$. The Lagrangian takes now the form:

$$\mathcal{L}(p_1, p_2, p_3, \lambda) = -\sum_i^3 p_i \log p_i - (\lambda_0 - 1)\left(\sum_i^3 p_i - 1\right) - \lambda_1 \left(\sum_i^3 p_i x_i - 2.5\right) \tag{52}$$

Now the first order condition for p_i is:

$$\frac{\partial \mathcal{L}}{\partial p_i} = -1 - \log p_i + 1 - \lambda_0 - \lambda_1 x_i = 0 \rightarrow p_i = e^{-\lambda_0 - \lambda_1 x_i} \quad \text{for } i = 1, 2, 3$$

For the Lagrangian multipliers we obtain the constraints:

$$\sum_i^3 p_i = e^{-\lambda_0}(e^{-\lambda_1} + e^{-2\lambda_1} + e^{-3\lambda_1}) = 1 \tag{53}$$

$$\sum_i^3 p_i x_i = e^{-\lambda_0}(e^{-\lambda_1} + 2e^{-2\lambda_1} + 3e^{-3\lambda_1}) = 2.5$$

Now we divide the two last equations (to get rid of λ_0) and re-arranging the terms, we obtain:

$$1.5e^{-\lambda_1} + 0.5e^{-2\lambda_1} - 0.5e^{-3\lambda_1}) = 0$$

If we replace $e^{-\lambda_1}$ by x (so that $\lambda_1 = -\log x$), we obtain a polynomial of order 3:

$$x^3 - x^2 - 3x = x(x^2 - x - 3) = 0$$

which we compute numerically and then retrieve the only possible solution for λ_1:

```
Re(polyroot(c(0,-3,-1,1)))
```

```
[1] 0.000000 -1.302776 2.302776
```

```
-log(Re(polyroot(c(0,-3,-1,1))))
```

```
## Warning in log(Re(polyroot(c(0, -3, -1, 1)))):  NaNs produced
```

[1] Inf NaN -0.8341152
Hence $\lambda_1 = -0.834$. Now we plug $e^{-\lambda_1} + e^{-2\lambda_1} + e^{-3\lambda_1} = 19.82$ into equation (53) to obtain:

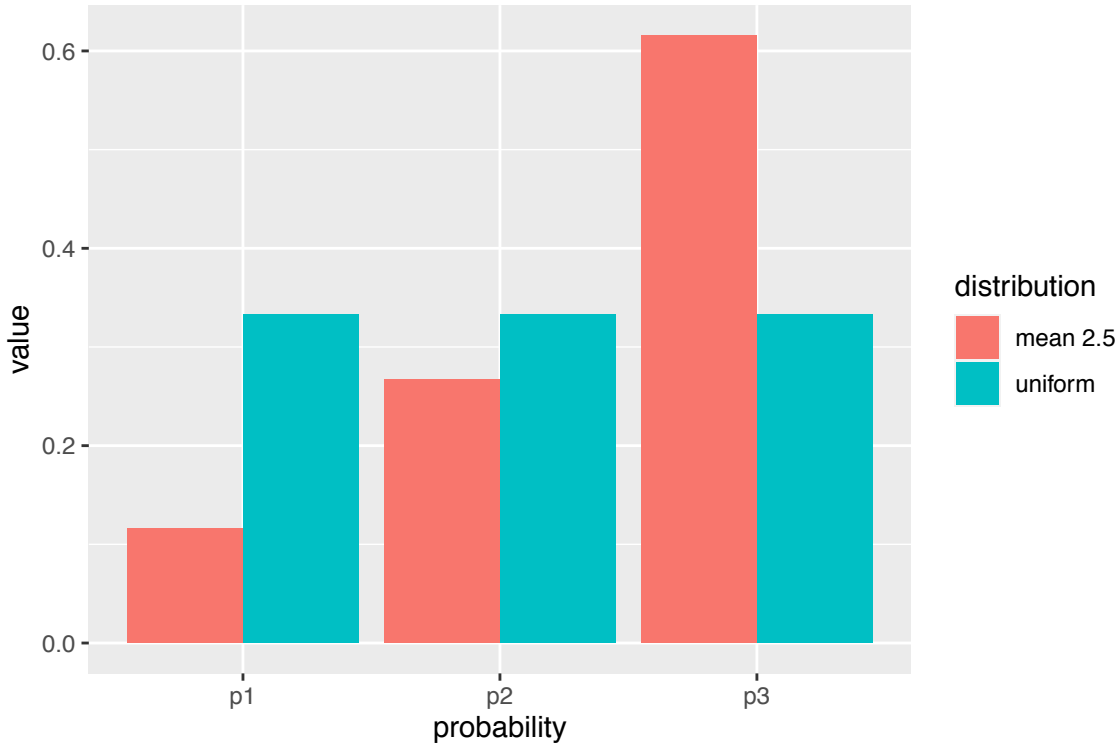
$$e_0^{\lambda} = 19.82 \rightarrow \lambda_0 = \log 19.82 = 2.987$$

Now we can finally compute the probabilities p_1, p_2, and p_3:

```
c(exp(-2.987+0.834),exp(-2.987+2*0.834),exp(-2.987+3*0.834))
```

[1] 0.1161352 0.2674026 0.6156972
and plot both distributions:

```
library(ggplot2)
value<-c(rep(1/3,3),c(exp(-2.987+0.834),exp(-2.987+2*0.834),exp(-2.987+3*0.834)))
distribution<-c(rep('uniform',3),rep('mean 2.5',3))
probability<-rep(paste0('p',1:3),2)
data<-data.frame(probability,distribution,value)
ggplot(data, aes(fill=distribution,x=probability,y=value)) + geom_bar(position="dodge", stat="identity")
```



6.5 Logistic Distribution

Suppose we know a binary $(0, 1)$ variable has mean equal to $3/5$. The generic form of probabilities p_i will be $e^{-\lambda_0 - \lambda_1 x}$. The partition function $e^{-\lambda_0}$ will be:

$$e^{-\lambda_0} = \frac{1}{e^0 + e^{-\lambda_1}} = \frac{1}{1 + e^{-\lambda_1}}$$

Probability $p(x = 1)$ will be:

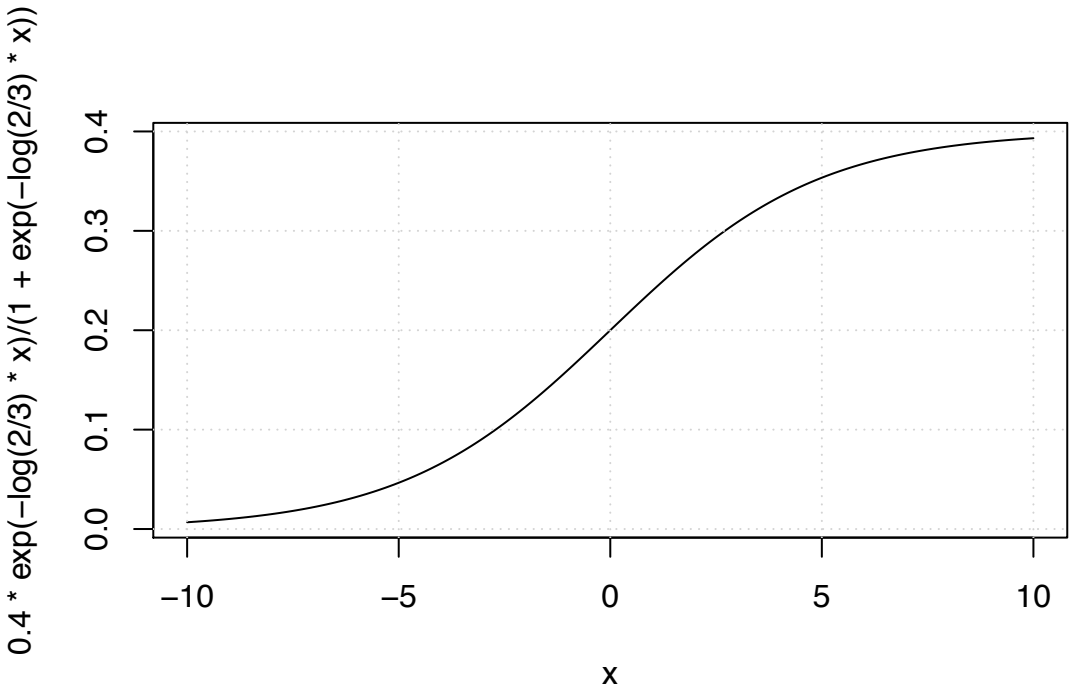
$$p(x = 1) = \frac{e^{-\lambda_1}}{1 + e^{-\lambda_1}} = \frac{1}{1 + e^{\lambda_1}}$$

which we equate to 3/5 to find

$$\lambda_1 = \log 2/3$$

Hence $\lambda_0 = 0.4$

```
x<-seq(-10,10,by=0.01)
plot(x,0.4*exp(-log(2/3)*x)/(1+exp(-log(2/3)*x)),type='l')
grid()
```



6.6 Power Law

Let's compute now the maximum-entropy distribution when we know the average logarithm of the random variable, i.e. $\langle \log x \rangle = \mu$ [Visser, 2013]. The Lagrangian is:

$$\mathcal{L}(p_i, \lambda_0, \lambda_1) = -\sum_i^N p_i \log p_i - (\lambda_0 - 1)\left(\sum_i^N p_i - 1\right) - \lambda_1 \left(\sum^N p_i \log x_i - \mu\right) \tag{54}$$

which yields as first order condition for p_i:

$$-\lambda_1 \log x_i - \lambda_0 - \log p_i = 0 \rightarrow p_i = e^{-\lambda_0} x^{-\lambda_1} \tag{55}$$

The partition function e^{λ_0} happens to be the Riemann zeta function:

$$e^{-\lambda_0} = \frac{1}{\sum_i x^{-\lambda_1}} = \frac{1}{\zeta(\lambda_1)} \tag{56}$$

7 Maxent derivation of a continuous distribution: Laplace

In the notes on information theory, we defined the entropy of variable X over a discrete set of events of probabilities $p(x_i)$:

$$H(X) = -\sum_i p(x_i) \log p(x_i)$$

If the set of events (ie the values that the random variable X can take) is actually continuous, then analytically we write the entropy as an integral, not a sum, over the interval (a, b) in which X is defined:

$$H(X) = -\int_a^b p(x) \log p(x) dx \tag{57}$$

Let's consider that we have a single moment constraint: $\langle |x - \mu| \rangle = b$, which is defined over the whole real line $(-\infty, +\infty)$. Now the Lagrangian takes the form:

$$\mathcal{L} = -\int_{-\infty}^{\infty} p(x) \log p(x) dx - (\lambda_0 - 1) \left(\int_{-\infty}^{\infty} p(x) dx - 1 \right) - \lambda_1 \left(\int_{-\infty}^{\infty} p(x) |x - \mu| dx - b \right)$$

Taking the FOC with respect to $p(x)$, we find the general form for $p(x)$:

$$\frac{\partial \mathcal{L}}{\partial p(x)} = -1 - \log p(x) - \lambda_0 + 1 - \lambda_1 |x - \mu| = 0$$

$$p(x) = \exp\left(-\lambda_0 - \lambda_1 |x - \mu|\right)$$

Now the two constraints look like:

$$\int_{-\infty}^{\infty} \exp\left(-\lambda_0 - \lambda_1 |x - \mu|\right) dx = 1$$

$$\int_{-\infty}^{\infty} |x - \mu| \exp\left(-\lambda_0 - \lambda_1 |x - \mu|\right) dx = b$$

As usual, $e^{-\lambda_0}$ will be the partition function that provides normalization. In order to compute the integral with the absolute function, we separate it into two parts between $-\infty$ and μ and μ and ∞ where we change the sign of x in the former (x is negative for that range). The integral gives the same value as if $\mu = 0$.

$$e^{+\lambda_0} = \int_{-\infty}^{\infty} \exp\left(-\lambda_1 |x - \mu|\right) dx = \int_{-\infty}^{\mu} e^{\lambda_1 (x-\mu)} dx + \int_{\mu}^{\infty} e^{-\lambda_1 (x-\mu)} dx = \int_{-\infty}^{0} e^{\lambda_1 x} dx + \int_{0}^{\infty} e^{-\lambda_1 x} dx$$

$$e^{+\lambda_0} = \frac{e^{\lambda_1 x}}{\lambda_1}|_{-\infty}^{0} - \frac{e^{-\lambda_1 x}}{\lambda_1}|_{0}^{\infty} = \frac{1}{\lambda_1} + \frac{1}{\lambda_1} = \frac{2}{\lambda_1}$$

where we already knew the value of the integral by parts,

$$\int x e^{ax} dx = \frac{e^{ax}}{a^2}(ax - 1)$$

so that the first Lagrange multiplier is:

$$e^{-\lambda_0} = \frac{\lambda_1}{2}$$

The second constraint is

$$b = \int_{-\infty}^{\infty} |x-\mu| e^{(-\lambda_0 - \lambda_1 |x-\mu|)} dx = \frac{\lambda_1}{2} \left(\int_{-\infty}^{0} -x e^{\lambda_1 x} dx + \int_{0}^{\infty} x e^{-\lambda_1 x} dx \right) = -\frac{e^{\lambda_1 x}}{\lambda_1^2}(\lambda_1 x - 1)|_{-\infty}^{0} - \frac{e^{\lambda_1 x}}{-\lambda_1^2}(\lambda_1 x - 1)|_{0}^{\infty}$$

$$b = \frac{\lambda_1}{2}\left(\frac{1}{\lambda_1^2} + \frac{1}{\lambda_1^2})\right) = \frac{1}{\lambda_1}$$

Hence, the Lagrange multipliers are:

$$e^{-\lambda_0} = \frac{1}{2b}$$

$$\lambda_1 = \frac{1}{b}$$

Replacing these values into the general form of $p(x)$, we obtain the Laplace (or double-exponential) distribution:

$$p(x) = \frac{1}{2b}e^{-\frac{|x-\mu|}{b}}$$

The Laplace distribution is the observed distribution for firm profitability, where μ is average profitability, under constraint $\langle|r-\mu|\rangle = b$. What is the significance of b here? Over- and undershooting, maybe? The difference between two independent identically distributed exponential random variables is governed by a Laplace distribution, as is a Brownian motion evaluated at an exponentially distributed random time.

8 Entropy Maximization

8.1 Definition of Variables

Let $j = 1, ..., N'$ be a set of possible states, each of which characterized by a set of random variables $\{X_i\}$ for $i = 1..., N$. In the simplest case, we have only one variable X_i of interest, such as GDP, unemployment, inflation, production costs, or profitability, so that $N = 1$. The set N' of possible states are the values variables $\{X_i\}$ could take: for instance, if $N = 1$, the continuous positive real line for the mentioned variables or 0 and 1 for a binary variable. If $N > 1$, N' comprises the joint values the variables can take. The j labels the probabilities of the distribution, while i labels the variable. In Golan, $k = 1, ..., K$ labels the states or probabilities (here j) and $m = 1, ..., M$ labels the variables(here i). Hence, we have a resulting design matrix X of size $N \times N'$ that includes all observations x_{ij} where variable X_i is in state j, with variables per each row and their states per each column. In Golan, the design matrix is x_{mk} of size $K \times M$.

8.2 Generalized Constraints

In the natural sciences, it is common to use constraints that are central moments of the distribution, for instance the conservation of energy as expected value. In the generalized constraints case, y_i is the expected value of a *functional form* of X_i, i.e. $f_i(X_j)$. Each of these generalized constraints account for a *conservation law*.

$$y_i \equiv \langle f_i(X_i)\rangle = \sum_{j=1}^{N'} p_j f_i(X_j) \tag{58}$$

8.3 The Basic Framework (chapter 4)

In the basic framework, we maximize the entropy of the distribution subject to generalized constraints plus the zeroth normalization constraint:

$$\begin{aligned} \underset{P}{\text{maximize}} \quad & H(P) = -\sum_{j}^{N'} p_j \log p_j \\ \text{subject to} \quad & \langle X_i \rangle \equiv \sum_{j=1}^{N'} p_j f_i(X_j) = \mu; \quad i = 1, ..., N \\ & \sum_{j=1}^{N'} p_j = 1 \\ & p_j \geq 0; \quad j = 1, ..., N' \end{aligned} \tag{59}$$

We adjoin a new parameter λ_i to each one of the constraints or conservation laws $1, ..., N$ plus λ_0 for normalization. These extra parameters, the Lagrange multipliers can be thought of as penalties for constraint violation. The associated first-order conditiobn yield a set of $N + N' + 1$ equations and parameters to be solved. The Lagrangian is:

$$\mathcal{L}(p, \lambda) = -\sum_{i}^{N} p_i \log p_i + (\lambda_0 - 1) \left(\sum_{i}^{N} p_i - 1 \right) + \sum_{i=1}^{N} \lambda_i \left(y_i - \sum_{j=1}^{N'} p_j f_i(X_j) \right) \tag{60}$$

The general form for the solution p_j is obtained from the FOC for p_j:

$$p_j = e^{-\lambda_0 - \sum_{j=1}^{N'} \lambda_i f_i(X_j)} = \exp\left(-\lambda_0 - \sum_{i=1}^{N} \lambda_i f_i(X_j) \right) \tag{61}$$

Normalization parameter λ_0 is associated with the partition function $\Omega(\lambda)$:

$$\lambda_0 = \log \left(\sum_{j=1}^{N'} \exp - \sum_{i=1}^{N} \lambda_i f_i(X_j) \right) = \log \Omega(\lambda) \tag{62}$$

The maximum-entropy solution is finally:

$$p_j = \frac{\exp\left(-\sum_{i=1}^{N} \lambda_i f_i(X_j) \right)}{\sum_{j=1}^{N'} \exp\left(-\sum_{i=1}^{N} \lambda_i f_i(X_j) \right)} = \frac{\exp\left(-\sum_{i=1}^{N} \lambda_i f_i(X_j) \right)}{\Omega(\lambda)} \tag{63}$$

and we can write now the $i = 1, ..., N$ constraints as:

$$y_i = \sum_{j} x_{ij} p_j = \frac{\sum_j x_{ij} \exp\left(-\sum_{i=1}^{N} \lambda_i f_i(X_j) \right)}{\Omega(\lambda)} \quad i = 1, ..., N \tag{64}$$

The entropy then becomes

$$H_{max}(P) = \underbrace{\log \Omega}_{\lambda_0} + \sum_{i=1}^{N} \lambda_i y_i \tag{65}$$

We can also rewrite the constraints as the partial derivative of the logarithm of the partition function (i.e. λ_0), which illustrates the relevance of the Lagrangian multipliers:

$$y_i = -\frac{\partial \log \Omega(\lambda)}{\partial \lambda_i} \tag{66}$$

so that

$$\frac{\partial H_{max}}{\partial y_i} = \lambda_i \quad i = 1, ..., N \tag{67}$$

Now we define matrices B and I of size $N \times N$ so that $B = I^{-1}$ using the chain rule for differentiation:

$$B_{mk} \equiv -\frac{\partial^2 \log \Omega}{\partial \lambda_k \partial \lambda_m} = \frac{\partial y_m}{\partial \lambda_k} = \frac{\partial y_k}{\partial \lambda_m} \quad m, k = 1, ..., N \tag{68}$$

$$I_{mk} \equiv -\frac{\partial^2 H}{\partial \lambda_m \partial \lambda_k} = \frac{\partial \lambda_m}{\partial y_k} = \frac{\partial \lambda_k}{\partial y_m} \quad m, k = 1, ..., N \tag{69}$$

The matrix B is the negative of the covariance of functions $f_i(X_j)$, which shows a universal relationship between the fluctuations of $f_i(X_j)$ and $\partial y_i / \partial \lambda_k$:

$$\langle f_i(X_j) - y_i \rangle \langle f_k(X_j) - y_k \rangle = \langle f_i f_k \rangle - y_i y_k = \frac{\partial^2 \log \Omega}{\partial \lambda_i \partial \lambda_k} = -B_{ik} \quad i, k = 1..., N \tag{70}$$

The relationship is a consequence of the envelope theorem, which also is used in microeconomic theory and relates the Hicksian demand (the solution of the expenditure minimization problem subject to fixed utility) with uncompensated Walrasian demands in the context of the Slutsky matrix (which must be negative semidefinite because we are in an optimization problem). The derivations of Roys Identity and Shepards Lemma, as well as the interpretation of the Lagrange multipliers are all special cases of what is known as the envelope theorem. "The symmetry of $D_p h(p, u)$ is an unexpected property" [Mas-Colell et al., 1995, p.70]. Really? The envelope theorem is just a mathematical result about the differentiability properties of the objective function of a parameterized optimization problem. As we change parameters of the objective, the envelope theorem shows that, in a certain sense, changes in the optimizer of the objective do not contribute to the change in the objective function. The envelope theorem is an important tool for comparative statics of optimization models.

8.4 Lagrange Multipliers and Information

8.4.1 Information View

From the point of view of information and optimization theories, the Lagrange multipliers capture the relative information of each one of the conservation rules (constraints). It is the marginal amount of information a cer- tain constraint contributed to the reduction of the entropy of the inferred distribution. Recalling that the entropy reaches its maximal level when there are no constraints, the Lagrange multipliers capture the amount of entropy that is reduced. The larger the magnitude of that multiplier, the larger its contribution relative to all other information used is shown in equation (67):

$$\frac{\partial H_{max}}{\partial y_i} = \lambda_i \quad i = 1, ..., N$$

But it is only a measure that is relative to the information set used. If, on the other hand, an estimated Lagrange multiplier is practically zero, it means that there is no additional information in the associated constraint; the inferred probability distribution is unaffected by this additional information. That constraint should not be used.

8.4.2 Statistical View

In more statistical terms, we can phrase the above as a statistical hypothesis: a hypothesis about the world around us that is expressed as a question (or a statement) with a yes/no answer. These hypotheses are about the theory or population, and the answer (whether the theory is true or not) is conditional on the observed information. In our context, the hypothesis that a certain piece of information (say constraint i) does not

provide any information about the system analyzed is specified as the hypothesis $\lambda_i = 0$. The answer to this hypothesis is a function of the inferred solution λ_i^*. If, for example, $\lambda_i^* = 0$, it is highly probable that we will fail to reject our hypothesis. If, on the other hand, λ_i^* is very far from zero, we may tend not to accept our hypothesis.

The marginal effects of the Lagrange multipliers λ_i on probabilities p_j can be computed at different quantiles of the inferred distribution:

$$\frac{\partial p_j}{\partial \lambda_i} = -p_i \left(x_{ij} - \sum_{j=1}^{N'} p_j x_{ij} \right) \tag{71}$$

The inferred Lagrange multipliers are the estimated values of the parameters in the probability distribution of interest–the distribution describing the information-generating process.

8.5 The Concentrated Framework

The basic framework presents the *primal model*, i.e. a problem of constrained optimization where optimization is carried with respect to the probabilities p_j. The *dual* problem is the unconstrained, concentrated version of the primal model, where it is concentrated on the minimally necessary set of parameters (the λ_i Lagrange multipliers for $i = 1, ..., N$) required for a full description of the system by ensuring that the conservation rules or constraints are satisfied.

The Lagrangian function (or potential function) is now:

$$\mathcal{L}(p, \lambda) = H(p) + (\lambda_0 - 1) \left(\sum_{i}^{N} p_i - 1 \right) + \sum_{i=1}^{N} \lambda_i \left(y_i - \sum_{j=1}^{N'} p_j f_i(X_j) \right) \tag{72}$$

The *concentrated* model is:

$$\ell(\lambda) = -\sum_{j=1}^{N'} p_j \log p_j + \sum_{i=1}^{N} \lambda_i \left(y_i - \sum_{j=1}^{N'} p_j f_i(X_j) \right) = \underbrace{\log \Omega(\lambda)}_{\lambda_0} + \sum_{i=1}^{N} \lambda_i y_i \tag{73}$$

The dual problem provides a lower bound on the objective value of the primal problem and thus entails *minimization*. Differentiating with respect to the Lagrange multipliers one obtains the conservation rules of the primal problem as first order conditions of the concentrated model:

$$\frac{\partial \ell(\lambda)}{\lambda_i} = y_i - \sum_{j=1}^{N'} p_j f_i(X_j) \tag{74}$$

Normalization factor λ_0, the logarithm of the partition function Ω, a function of the N Lagrange multipliers, is also known as the potential for the problem, whose derivatives are the expected values of the elements of the system.

8.6 The Complete Framework (with Noise; Chapter 9)

Now we must address theory uncertainty and model uncertainty, as well as uncertainty about the observed information. One way to accomplish this is to allow for uncertainty in the constraints themselves. Let $\langle F \rangle$ the actual, underlying theory (possible yet uncertain) represented in terms of expected values and $\langle F' \rangle$ the approximate theory we have based on our input information (on the observed sample) such that

$$\langle F' \rangle = \langle F \rangle + \epsilon$$

where ϵ captures the uncertainty surrounding our model, knowledge, and observed information, i.e. "noise". For $\epsilon = 0$, we retrieve the basic framework without noise. In general terms, for N theories or conservation rules:

$$\langle F_i' \rangle \equiv y_i = \langle F \rangle + \epsilon = \sum_{j=1}^{N'} p_j f_i(X_j) + \epsilon_i \tag{75}$$

In order to define this rigorously, we define error ϵ_i as the expected value of an underlying auxiliary distribution "noise":

$$\epsilon_i \equiv \sum_{k=1}^{K} w_{ik} v_{ik} \tag{76}$$

We consider $\mathcal{V}_i$ as the support of a random variable with mean zero and some unknown distribution so that $\epsilon_i \in \mathcal{V}_i$ in a way that ϵ_i can be viewed as a convex combination of the lower and upper bounds of $\mathcal{V}_i$. For each ϵ_i with $i = 1, ..., N$, we define a K dimensional discrete random variable V_i and a corresponding K dimensional probability distribution w_{ik} such that $w_{ik} \geq 0$ and $\sum_{k=1}^{N'} w_{ik} = 1$. Consequently, we can write:

$$y_i = \sum_{j=1}^{N'} p_j f_i(X_j) + \epsilon_i = \sum_{j=1}^{N'} p_j f_i(X_j) + \sum_{k=1}^{K} w_{ik} v_{ik} \tag{77}$$

and now the complete problem entails inferring not only p but also w, which are conditional on the information we have and on the errors' support as specified by the researcher:

$$\begin{aligned}
\underset{P,W}{\text{maximize}} \quad & H(P,W) = H(P) + H(W) = -\sum_{j}^{N'} p_j \log p_j + -\sum_{k}^{K} w_{ik} \log w_{ik} \\
\text{subject to} \quad & y_i = \sum_{j=1}^{N'} p_j f_i(X_j) + \sum_{k=1}^{K} w_{ik} v_k; \quad i = 1, ..., N \\
& \sum_{j=1}^{N'} p_j = 1; \quad \sum_{k=1}^{K} w_{ij} = 1 \quad i = 1, ..., N \\
& p_j \geq 0; \text{ and } w_{ik} \geq 0; \quad j = 1, ..., N'; \quad k = 1, ..., K
\end{aligned} \tag{78}$$

The Lagrangian function is now:

$$\ell(p, w, \lambda, \mu) = H(P) + H(W) + \sum_{i=1}^{N} \lambda_i \left(y_i - \sum_{j=1}^{N'} p_j f_i(X_j) - \sum_{k=1}^{K} w_{ik} v_k \right) + (\lambda_0 - 1) \left(1 - \sum_{j}^{N'} p_j \right) + \sum_{i}^{N} (\mu_i - 1) \left(1 - \sum_{k}^{K} w_{ik} \right) \tag{79}$$

From the FOC, we obtain $N+1$ partition functions, Ω that captures the information of the underlying theory for p_i, and Ψ_i for $i = 1, ..., N$ for the noise probabilities that capture the uncertainty about the theory:

$$\lambda_0 = \log \left(\sum_{j=1}^{N'} \exp - \sum_{i=1}^{N} \lambda_i f_i(X_j) \right) = \log \Omega(\lambda) \tag{80}$$

$$\mu_i = \log \left(\sum_{k=1}^{K} \exp - \lambda_i v_k \right) \equiv \log \Psi_i(\lambda_i) \quad i = 1, ..., N \tag{81}$$

The general solution for the complete framework is thus:

$$p_j = \frac{\exp\left(- \sum_{i=1}^{N} \lambda_i f_i(X_j) \right)}{\sum_{j=1}^{N'} \exp\left(- \sum_{i=1}^{N} \lambda_i f_i(X_j) \right)} = \frac{\exp\left(- \sum_{i=1}^{N} \lambda_i f_i(X_j) \right)}{\Omega(\lambda)} \quad j = 1, ..., N' \tag{82}$$

$$w_{ik} = \frac{\exp\left(-\lambda_i v_k\right)}{\sum_{k=1}^{K} \exp\left(-\lambda_i v_k\right)} = \frac{\exp\left(-\lambda_i v_k\right)}{\Psi_i(\lambda_i)} \quad i = 1, ..., N; k = 1, ..., K \tag{83}$$

Now the Lagrange multipliers λ_i are not only determined by the expected values y_i, but also the noise $\mathcal{V}_i$:

$$-y_i = \frac{\partial \log \Omega(\lambda)}{\partial \lambda_i} + \frac{\partial \log \Psi_i(\lambda_i)}{\partial \lambda_i} \tag{84}$$

Finally, $\epsilon_i^* = \sum_k w_{ik}^* v_k$ where $\langle \epsilon_i^* \rangle$ is not necessarily zero, but determined from the observed information.

The maximal entropy is also a function of the expected values:

$$H_{max}(P, W) = \underbrace{\log \Omega}_{\lambda_0} + \sum_i \underbrace{\log \Psi_i}_{\mu_i} + \sum_i \lambda_i y_i \tag{85}$$

In the concentrated framework, we have:

$$\ell(\lambda) = \sum_i \lambda_i y_i + \lambda_0 + \sum_i \mu_i \tag{86}$$

The FOC for the λ's reflects the primal-dual relationship, where the optimal conditions of the concentrated model are just the (noisy) conservation rules of the primal problem:

$$\frac{\partial \ell(\lambda)}{\partial \lambda_i} = y_i - \sum_j p_j f_i(X_j) - \sum_k w_{ik} v_k \tag{87}$$

8.7 Linear Regression (Chapter 13)

Consider a sample of independent observations that can be characterized as a set of K dimensional vectors of observations labeled as $i = 1, ..., N$. The design matrix X, the covariates, now has size $N \times K$.

$$y_i = f(x_i; \beta) + \epsilon_i \tag{88}$$

where $f(x_i; \beta)$ is a given function with an unknown K dimensional vector of parameters β and ϵ_i are independent random errors with conditional mean zero and positive conditional variance, i.e $\langle \epsilon_i | x_i \rangle = 0$ and $var(\epsilon_i | x_i) = \sigma^2(x_i)$. In the linear model:

$$f(x_i; \beta) = \sum_k x_{ik} \beta_k$$

The Least Squares solution is just a minimization with respect to the β parameters of the sum of the squares of the errors

$$\sum_{i=1}^{N} \left(y_i - f(x_i; \beta)\right)^2$$

which is achieved by computing the first order conditions:

$$\sum_{i=1}^{N} \frac{\partial f(x_i; \beta)}{\partial \beta} \left(y_i - f(x_i; \beta)\right)$$

While the Least Squares solution does not require any assumption on the shapes of the error terms, the Maximum Likelihood method requires us to specify a likelihood function.

As in chapter 9, Golan allows the model coefficients to be random variables drawn from a frequency distribution over another support set, chosen to include all of the plausible values for the β:

$$\beta_k = \sum_j p_{jk} z_k \tag{89}$$

Once the problem has been generalized to this under-determined form, the constrained maximum entropy method takes over and produces the constrained maximization problem:

$$\begin{aligned}
\underset{P,W}{\text{maximize}} \quad & H(P,W) = H(P) + H(W) \\
\text{subject to} \quad & \sum_j p_{jk} = 1; \quad k = 1, ..., K \\
& \sum_s w_{st} = 1 \quad t = 1, ..., T \\
y_t = \beta_0 + \sum_k \beta_k x_{kt} + \epsilon_t = & \sum_j p_{0j} z_{0j} + \sum_k \sum_j p_{kj} z_{kj} x_{kt} + \sum_s w_{st} v_s
\end{aligned} \tag{90}$$

9 Entropy-constrained quantal response

One important economic example of the constrained maximum entropy method is individual choice of an action, such as buying or selling an asset, or entering or exiting some particular market as a producer in response to a payoff. Suppose an actor has a set of actions $\mathcal{A} = \{a_1, ..., a_K\}$, each of which has a payoff $u(a_k) = u_k$, and acts according to a mixed strategy described by the frequency distribution $\{f_1, ..., f_K\}$. The expected payoff is $\sum_k^K f_k u_k$.

The constrained maximum entropy model of action maximizes the entropy of the frequency distribution subject to a constraint on the minimum level of expected payoff:

$$\begin{aligned}
\underset{f}{\text{maximize}} \quad & -\sum_k^K f_j \log f_j \\
\text{subject to} \quad & \sum_k f_k = 1 \\
& u_{min} \leq max(u_k)
\end{aligned} \tag{91}$$

In the model, the actor is satisficing rather than maximizing.

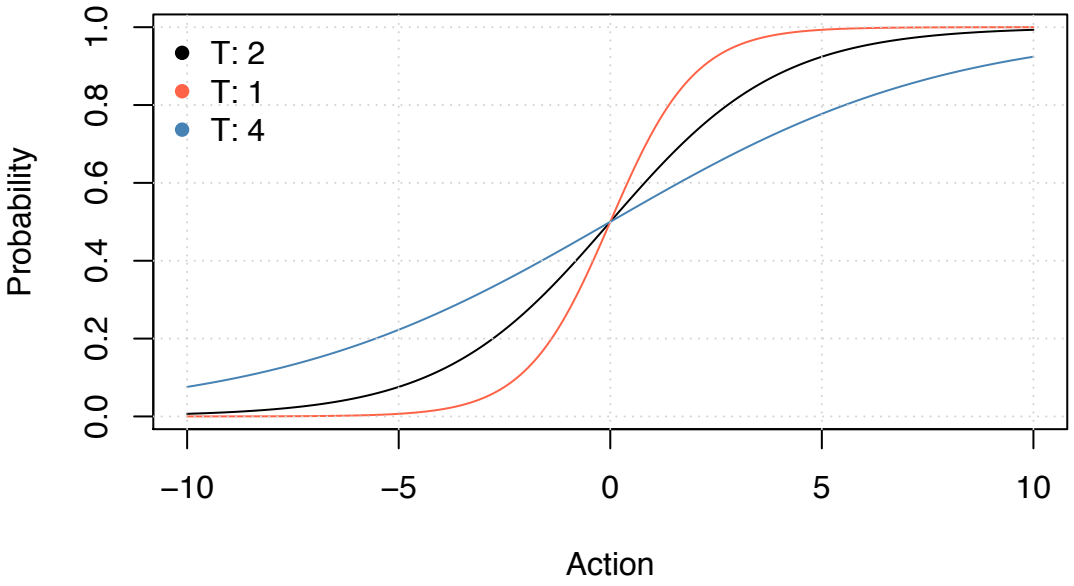
The constrained maximum entropy solution in the action-payoff model is a generalization of the quantal response model that is widely used in econometrics:

$$f_k = \frac{\exp \beta u_k}{\sum_k \exp \beta u_k} \tag{92}$$

where β is the Lagrange multiplier corresponding to the expect payoff constraint. The quantal response model has the intuitive property that the logarithm of the frequency with which we observe any given action is equal to its utility plus a constant offset. The coefficient β is a measure of the responsiveness of the actor to differences in payoff.

For two actions where $u_1 = 0$, $u_2 = 1$, and $\beta = 1/T$?

```
x<-seq(-10,10,by=0.01)
plot(x,1/(1+exp(-0.5*x)),type='l',ylab='Probability',xlab='Action')
lines(x,1/(1+exp(-1*x)),type='l',col='tomato')
lines(x,1/(1+exp(-0.25*x)),type='l',col='steelblue')
grid()
legend('topleft',paste('T:',c(1/0.5,1,1/0.25)),pch=16,col=c('black','tomato','steelblue'),bty='n',bg='t
```



This derivation of logistic quantal response behavior is essentially equivalent to Christopher Sims' theory of "rational inattention" [Sims, 2003].

Logistic quantal response behavior can be regarded as a generalization of rational choice theory, in so far as the decision-maker, as in rational choice theory, has well-defined payoffs over actions, and maximizes expected utility in choosing a mixed strategy, which results in more frequent choices of higher-payoff actions. The new element in entropy-constrained behavior is the behavior temperature, which limits the degree to which the decision-maker can concentrate frequency on the highest-payoff action.

10 Computational Examples of Constrained Optimization

In this section, I introduce some examples of constrained optimization using computational software. An appropriate software to compute constrained optimization is GAMS - you can check examples in the Info-Metrics website. There are a lot of packages in R that allow the user to compute constrained optimization. In our case, we are interested in packages that solve optimization problems with (1) a nonlinear objective function and (2) equality and inequality constraints. The inequality constraints are to define the distribution probabilities, which are the parameters in the objective function, as nonnegative.

The two packages we will be using are *alabama* and *NlcOptim*, with respective functions *auglag* and *solnl*.

```
install.packages('alabama')
```

```
## Error in contrib.url(repos, "source"):  trying to use CRAN without setting a mirror
```

```
install.packages('NlcOptim')
```

```
## Error in contrib.url(repos, "source"):  trying to use CRAN without setting a mirror
```

```
library(alabama)
```

```
## Loading required package:  numDeriv
```

```
library(NlcOptim)

## Loading required package:  MASS
```

Augmented Lagrangian Algorithm

The Augmented Lagrangian method adds additional terms to the unconstrained objective function, designed to emulate a Lagrangian multiplier.

Usage

```
auglag(x0, fn, gr = NULL, lower = NULL, upper = NULL, hin = NULL,
  hinjac = NULL, heq = NULL, heqjac = NULL, localsolver = c("COBYLA"),
  localtol = 1e-06, ineq2local = FALSE, nl.info = FALSE,
  control = list(), ...)
```

Arguments

x0	starting point for searching the optimum.
fn	objective function that is to be minimized.
gr	gradient of the objective function; will be provided provided is `NULL` and the solver requires derivatives.
lower, upper	lower and upper bound constraints.
hin, hinjac	defines the inequalty constraints, `hin(x) >= 0`
heq, heqjac	defines the equality constraints, `heq(x) = 0` .

Figure 4: **Documentation of Function** *auglag*

10.1 3 Events, No Mean

We first define the objective function (the entropy), which will be the same for all the computations, and its gradient. The gradient is a vector of partial derivatives with respect to each of the function variables (the probabilities), which we also obtain in the FOC. The function of the parameter is the vector of probabilities x (p_j in our analytical notation). Because the software computes minimization and not maximization, we eliminate the negative sign of the entropy to search for minima without loss of generality. We also want to avoid the computation of logarithm of zeros (i.e. $0 \log 0 \equiv 0$).

```
entropy<-function(x)
{
  p<-x[x!=0]
  sum(p*log(p))
}
entropy_gr<-function(x) {ifelse(x==0,0,log(x)+1)} # a vector of partial derivatives wrt the p's
```

As in our analytical derivation, we define the values of the 3 events, $x_j = j$ for $j = 1, 2, 3$ in the vector f:

```
f<-c(1,2,3)
```

In order to introduce the constraints for *auglag*, we need a vector for the equality constraints heq (so that $heq_j = 0$ for all j), a vector for the inequality constraints hin (so that $hin_j > 0$ for all j), and their corresponding Jacobian matrices (partial derivatives) of size $N \times N'$, where number of rows N is the number of constraints and number of columns N' is the support space (number of events) of probability distribution p_j:

```
heq <- function(x) { # vector
  h <- rep(NA, 1)
  h[1] <- sum(x) - 1 # zeroth moment constraint: normalization
  # h[2] <- sum(f*x) - 2.5 # first moment constraint: mean
  h
}
heq.jac <- function(x) { # matrix
  j <- matrix(NA, 1, length(x))
  j[1, ] <- c(1, 1, 1) # partial derivatives of the normalization constraint wrt the p's
  # j[2, ] <- c(1,2,3) # values of f, ie partial derivatives of the mean wrt the p's
  j
}
hin <- function(x) { # vector
  h <- x # vector of probabilities must be nonnegative in all entries
  h
}
hin.jac <- function(x) { # matrix
diag(length(x)) # partial derivatives of hin wrt the p's
}
```

Finally we can call the function *auglag*, where the first parameter is the initial vector from which to start the numerical computation:

```
results<-auglag(runif(3,0,.3),entropy,entropy_gr,heq=heq,heq.jac=heq.jac,hin=hin,hin.jac=hin.jac)
```

The results are given in the object *par*:

```
results$par

## [1] 0.3333333 0.3333333 0.3333333
```

10.2 3 Events, Mean 2.5

Now we add a constraint, the mean is 2.5. We only have to change *hin* and *hin.jac*:

```
heq <- function(x) { # vector
  h <- rep(NA, 1)
  h[1] <- sum(x) - 1 # zeroth moment constraint: normalization
  h[2] <- sum(f*x) - 2.5 # first moment constraint: mean
  h
}
heq.jac <- function(x) { # matrix
  j <- matrix(NA, 2, length(x)) # 2 rows now !!
  j[1, ] <- c(1, 1, 1) # partial derivatives of the normalization constraint wrt the p's
  j[2, ] <- c(1, 2, 3) # values of f, ie partial derivatives of the mean wrt the p's
  j
}
results<-auglag(runif(3,0,.3),entropy,entropy_gr,heq=heq,heq.jac=heq.jac,hin=hin,hin.jac=hin.jac)
```

The results are given in the object *par*, which we can compare to the analytical results we obtained earlier:

```
results$par
```

```
## [1] 0.1162041 0.2675917 0.6162041
```

```
c(exp(-2.987+0.834),exp(-2.987+2*0.834),exp(-2.987+3*0.834))
```

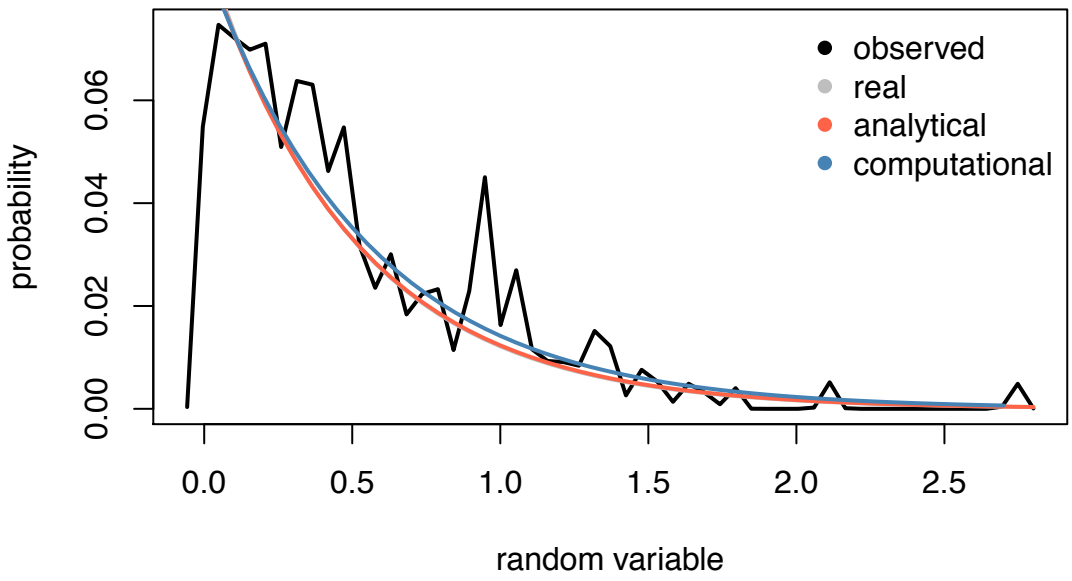
```
## [1] 0.1161352 0.2674026 0.6156972
```

10.3 Exponential Distribution

We generate 200 random values *val* following a exponential distribution with rate 2 (the rate is the Lagrange multiplier and the inverse of the mean, which is 0.5). We define the support space f in bins of size 0.05 between 0 and the maximum random value. In addition, we also define the corresponding equality constraints (we only generalize the functions where now the number of columns is not 3, but the length of the f vector).

```
val<-rexp(200,2)
mu<-mean(val)
f<-seq(0,max(val),by=0.05)

heq <- function(x) {
  h <- rep(NA, 1)
  h[1] <- sum(x) - 1
  h[2] <- sum(f*x) - mu
  h
}
heq.jac <- function(x) {
  j <- matrix(NA, 2, length(x))
  j[1, ] <- rep(1,length(x))
  j[2, ] <- f
  j
}
init<-rnorm(length(f),1/length(f),0.1/length(f))
results_e1<-auglag(init,entropy,entropy_gr,heq=heq,heq.jac=heq.jac,hin=hin,hin.jac=hin.jac)
```

```
tmp<-density(val,bw=0.02,n=length(f)) # distribution of val with length of f as number of bins
x_<-tmp$x
y_<-tmp$y/sum(tmp$y) # we want the discrete sum of actual densities to be 1

p<-exp(-x_/mu)/sum(exp(-x_/mu)) # analytical solution with known mean (which is computed)
p2<-exp(-x_/0.5)/sum(exp(-x_/0.5)) # underlying distribution of the data-generating process

plot(x_,y_,type='l',lwd=2,xlab='random variable',ylab='probability') # plot distribution
lines(x_,p2,col='gray',lwd=2)  # real distribution
lines(x_,p,col='tomato',lwd=2)  # plot analytical solution
lines(f,results_e1$par,col='steelblue',lwd=2) # plot computational solution
legend('topright',c('observed','real','analytical','computational'),col=c('black','gray','tomato','steel
```



The mean is

```
mu
```

[1] 0.5064616

```
sum(f*results_e1$par)
```

[1] 0.5064616

10.4 3 Events with Noise, Known Mean

Now we introduce noise in our former example with $N' = 3$ events. In order to do so, we need to define the support space V for the noise, with the symmetry and 0 requirement (3 values are enough). Now the R parameter x is not only the probability vector p_j with $j = 1, ..., N'$, but $(p_1, p_2, ..., p_{N'}, w_1, w_2, w_3)$, where we add at the end the weights for noise V, which work like probabilities. With this definition, we do not have to change the objective function, which now is $H(p) + H(w)$, or the gradient.

```
f<-c(1,2,3)
v<-c(-5,0,5)
mu<-2.5
heq <- function(x) {
  h <- rep(NA, 1)
  h[1] <- sum(x[1:3]) - 1 # sum of probabilities p is 1
  h[2] <- sum(x[4:6]) - 1 # sum of noise weights w is 1
  h[3] <- sum(f*x[1:3]) + sum(v*x[4:6]) - 2.5 # noisy mean is 2.5
  h
}
```

```
heq.jac <- function(x) {
  j <- matrix(NA, 3, length(x))
  j[1, ] <- c(1, 1, 1, 0, 0, 0)
  j[2, ] <- c(0, 0, 0, 1, 1, 1)
  j[3, ] <- c(1,2,3,-1,0,1) # c(f,v)
  j
}
init<-runif(6,0,.3)
results<-auglag(init,entropy,entropy_gr,heq=heq,heq.jac=heq.jac,hin=hin,hin.jac=hin.jac)
```

The vector of computed probabilities p_j and noise weights w_k is now, which we compare to the analytical solution without noise:

```
results$par

## [1] 0.3005519 0.3510053 0.3484428 0.2948318 0.3199147 0.3852536

c(exp(-2.987+0.834),exp(-2.987+2*0.834),exp(-2.987+3*0.834))

## [1] 0.1161352 0.2674026 0.6156972
```

The real mean and the noise is now:

```
sum(f*results$par[1:3])

## [1] 2.047891

sum(v*results$par[4:6])

## [1] 0.452109
```

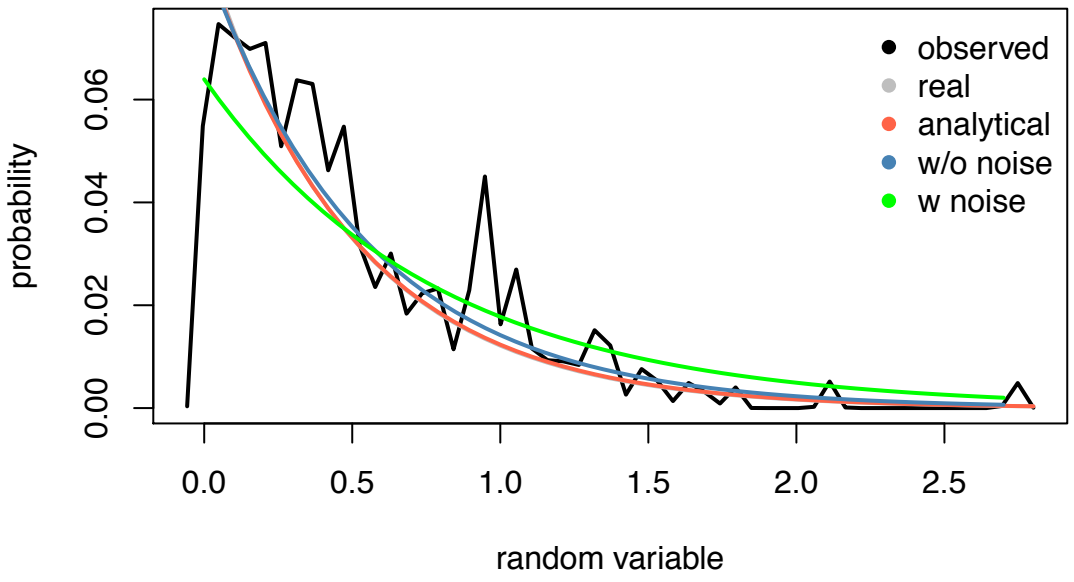
10.5 Exponential Distribution with Noise

Now we calculate the computational solution for the exponential scenario with the same set of random values *val*, but now with noise. The support space for V has now 5 elements, with upper and lower bounds defined by $(-3\sigma, +3\sigma)$.

```
mu<-mean(val)
f<-seq(0,max(val),by=0.05)
v<-c(-3*sd(val),-1.5*sd(val),0,1.5*sd(val),3*sd(val))
v<-c(-sd(val),0,sd(val)) # actually this support space works better, don't @ me

heq <- function(x) {
  h <- rep(NA, 1)
  h[1] <- sum(x[1:length(f)]) - 1
  h[2] <- sum(x[(length(f)+1):(length(f)+length(v))]) - 1
  h[3] <- sum(c(f,v)*x) - mu
  h
}
heq.jac <- function(x) {
  j <- matrix(NA, 3, length(x))
```

```
  j[1, ] <- c(rep(1,length(f)),rep(0,length(v)))
  j[2, ] <- c(rep(0,length(f)),rep(1,length(v)))
  j[3, ] <- c(f,v)
  j
}
init<-c(rnorm(length(f),1/length(f),0.1/length(f)),rep(1/length(v),length(v)))
results_e2<-auglag(init,entropy,entropy_gr,heq=heq,heq.jac=heq.jac,hin=hin,hin.jac=hin.jac)
```

```
plot(x_,y_,type='l',lwd=2,xlab='random variable',ylab='probability') # plot distribution
lines(x_,p2,col='gray',lwd=2)  # real distribution
lines(x_,p,col='tomato',lwd=2)  # analytical solution
lines(f,results_e1$par,col='steelblue',lwd=2) # computational solution w/o noise
lines(f,results_e2$par[1:length(f)],col='green',lwd=2) # computational solution with noise
legend('topright',c('observed','real','analytical','w/o noise','w noise'),col=c('black','gray','tomato'
```



The real mean and the noise is now:

```
sum(f*results_e2$par[1:length(f)])
```

```
## [1] 0.6721069
```

```
sum(v*results_e2$par[(length(f)+1):(length(f)+length(v))])
```

```
## [1] -0.1656452
```

10.6 Linear Regression, with real Gaussian errors

In the linear regression example, we generate a set of 121 coordinates (x, y) (which we label as x_0, y_0) where x goes from -3 to 3 at 0.05 intervals and

$$y = 1 + 0.5x + N(0, 1)$$

Hence in our example, the real intercept is $\beta_0 = 1$ and the real slope is $\beta_1 = 0.5$. We construct three support spaces, each with 3 elements: one for the noise weights V as before, one for the probabilities of the intercept, and one for the probabilities of the slope, i.e. the latter two go into Z, which is problem-specific.

Now the parameter x of the objective function has length $3T+6$, where T is the number of observations: 3 probabilities for the intercept, 3 probabilities for the slope, and 3 noise weights for each of the T observations (in that order). We have now $2T + 2$ constraints: $T + 2$ normalization constraints and one constraint per each of the T observations.

```
x0<-seq(-3,3,by=.05)
y0<-1+0.5*x0+rnorm(length(x0),0,1)
v<-c(-3*sd(y0),0,3*sd(y0))
z<-c(-50,0,50,-30,0,30)

heq <- function(x)
{
  h <- rep(NA, 1)
  h[1] <- sum(x[1:3]) - 1
  h[2] <- sum(x[4:6]) - 1
  for(i in 1:length(x0)){h[i+2] <- sum(x[(3*(i-1)+1+6):(3*i+6)]) - 1}
  for(i in 1:length(x0)){h[i+length(x0)+2]<-sum(x[1:3]*z[1:3])+sum(x[4:6]*z[4:6]*x0[i])+sum(x[(3*(i-1)+
  # h[i+length(x0)+2]<-sum(x[1:3]*z[1:3])+sum(x[4:6]*z[4:6]*x0[i]) +
  # + sum(x[(3*(i-1)+1+6):(3*i+6)]*v) - y0[i] - the complete line in the last 'for' loop
  h
}
heq.jac <- function(x)
{
  j <- matrix(NA, 2*length(x0)+2, length(x))
  j[1, ] <- c(rep(1,3),rep(0,3+3*length(x0)))
  j[2, ] <- c(rep(0,3),rep(1,3),rep(0,3*length(x0)))
  for(i in 1:length(x0))
  {
    j[i+2,] <- 0
    j[i+2,(3*(i-1)+1+6):(3*i+6)]<- 1
  }
  for(i in 1:length(x0))
  {
    j[i+length(x0)+2, ] <- 0
    j[i+length(x0)+2,1:6] <- c(z[1:3],x0[i]*z[4:6])
    j[i+length(x0)+2,(3*(i-1)+1+6):(3*i+6)]<-v
  }
  j
}
init<-rep(1/3,6+3*length(x0))
res_l<-auglag(init,entropy,entropy_gr,heq=heq,heq.jac=heq.jac)
```

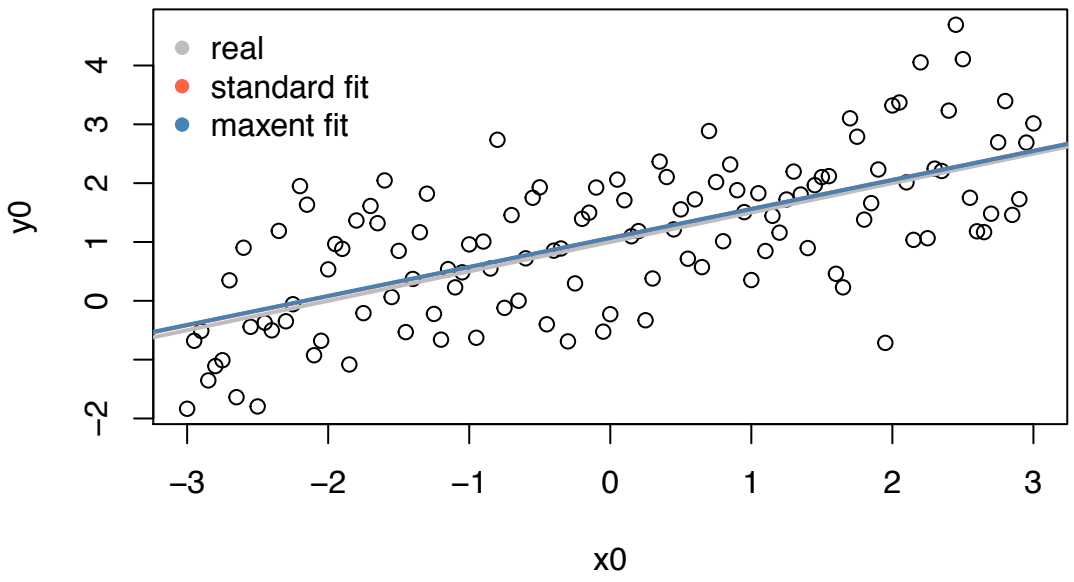
```
beta0<-sum(res_l$par[1:3]*z[1:3])
beta1<-sum(res_l$par[4:6]*z[4:6])
c(beta0=beta0,beta1=beta1)
```

beta0 beta1 1.0689130 0.4926729

```
plot(x0,y0,xlab='x0',ylab='y0')
lfit<-lm(y0 ~ x0)
library(xtable)
print(xtable(summary(lfit)),type='latex')
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	1.0675	0.0884	12.08	0.0000
x0	0.4944	0.0506	9.77	0.0000

```
abline(1,0.5,col='gray',lwd=2) # real line
abline(lfit,col='tomato',lwd=2) # standard fit
abline(beta0,beta1,col='steelblue',lwd=2) # maxent fit
legend('topleft',c('real','standard fit','maxent fit'),col=c('gray','tomato','steelblue'),pch=16,bty='n
```



10.7 Linear Regression, with more elements in the support spaces

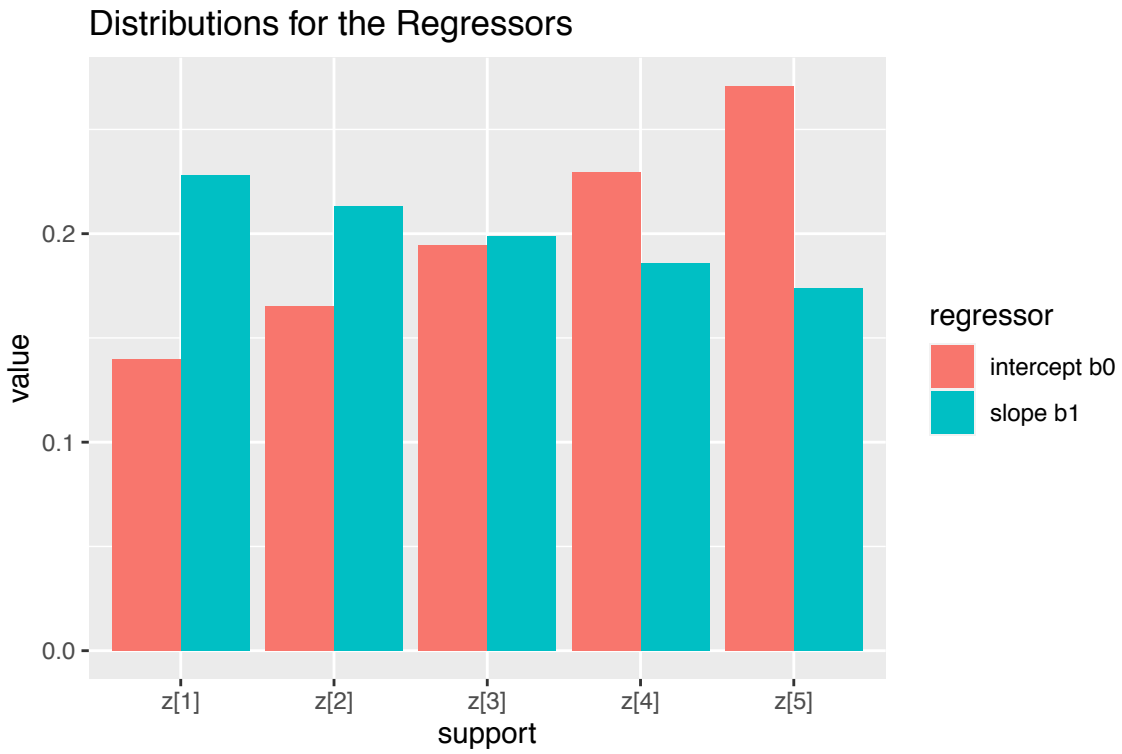
In another linear regression example, we generate a set of 201 coordinates (x, y) (which we label as x_0, y_0) where x goes from -5 to 5 at 0.05 intervals and

$$y = 10 - 2x + N(0, 3)$$

Hence in our example, the real intercept is $\beta_0 = 8$ and the real slope is $\beta_1 = -2$. Once again, we construct three support spaces (for the noise and both regressors), but now each with 5 elements. Our goal here is to obtain more accurate resulting distributions.

```
x0<-seq(-5,5,by=0.05)
y0<-8-2*x0+rnorm(length(x0),0,3)
v<-c(-3*sd(y0),-1.5*sd(y0),0,1.5*sd(y0),3*sd(y0))
z<-c(-50,-25,0,25,50,-30,-15,0,15,30)
b0<-1:(0.5*length(z)) # support space interval for beta0, former example was 1:3
b1<-(0.5*length(z)+1):length(z) # support space interval for beta1, former example was 4:6

heq <- function(x)
{
  h <- rep(NA, 1)
  h[1] <- sum(x[b0]) - 1
  h[2] <- sum(x[b1]) - 1
  for(i in 1:length(x0)) # 2 equality constraints for each obs in the same for loop
  {
    cnt<-(length(v)*(i-1)+1+length(z)):(length(v)*i+length(z))
    h[i+2] <- sum(x[cnt]) - 1
    h[i+length(x0)+2]<-sum(x[b0]*z[b0])+sum(x[b1]*z[b1]*x0[i])+sum(x[cnt]*v) - y0[i]
  }
  h
}
heq.jac <- function(x)
{
  j <- matrix(0, 2*length(x0)+2, length(x)) # "default" value for j is 0, not NA
  j[1,b0] <- 1
  j[2,b1] <- 1
  for(i in 1:length(x0))  # 2 equality constraints for each obs in the same for loop
  {
    cnt<-(length(v)*(i-1)+1+length(z)):(length(v)*i+length(z))
    j[i+2,(length(v)*(i-1)+1+length(z)):(length(v)*i+length(z))]<- 1
    j[i+length(x0)+2,b0] <- z[b0]
    j[i+length(x0)+2,b1] <- x0[i]*z[b1]
    j[i+length(x0)+2,(length(v)*(i-1)+1+length(z)):(length(v)*i+length(z))]<-v
  }
  j
}
init<-rep(1/length(v),length(z)+length(v)*length(x0))
res_l2<-auglag(init,entropy,entropy_gr,heq=heq,heq.jac=heq.jac)
```

```
beta0<-sum(res_l2$par[b0]*z[b0])
beta1<-sum(res_l2$par[b1]*z[b1])

library(ggplot2)
value<-res_l2$par[c(b0,b1)]
regressor<-c(rep('intercept b0',0.5*length(z)),rep('slope b1',0.5*length(z)))
support<-rep(paste0('z[',1:(0.5*length(z)),']'),2)
data<-data.frame(support,regressor,value)
ggplot(data, aes(fill=regressor,x=support,y=value)) + geom_bar(position="dodge", stat="identity") + labs
```



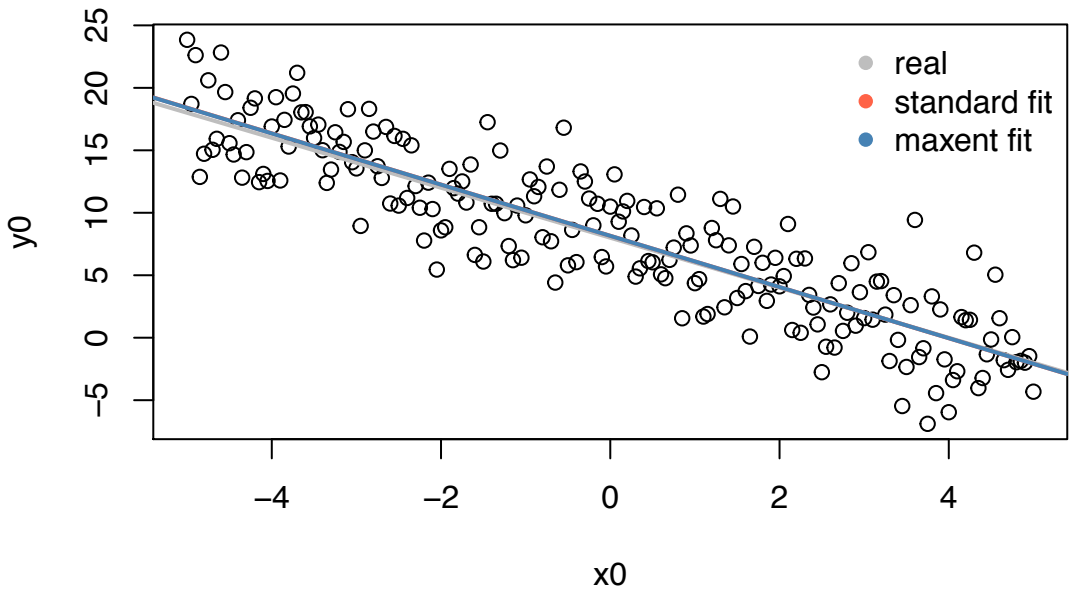

```
c(beta0=beta0,beta1=beta1)
```

beta0 beta1 8.155763 -2.048630

```
plot(x0,y0,xlab='x0',ylab='y0')
lfit2<-lm(y0 ~ x0)
library(xtable)
print(xtable(summary(lfit2)),type='latex')
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	8.1586	0.2156	37.84	0.0000
x0	-2.0503	0.0743	-27.59	0.0000

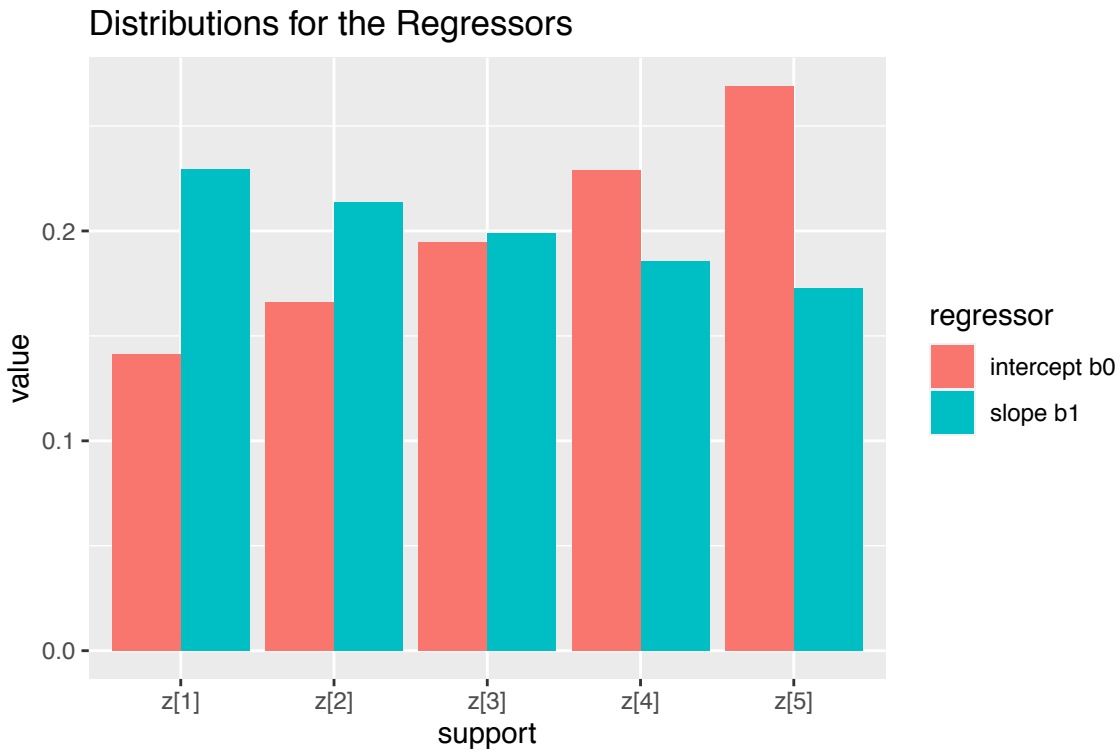
```
abline(8,-2,col='gray',lwd=2) # real line
abline(lfit2,col='tomato',lwd=2) # standard fit
abline(beta0,beta1,col='steelblue',lwd=2) # maxent fit
legend('topright',c('real','standard fit','maxent fit'),col=c('gray','tomato','steelblue'),pch=16,bty='
```



10.8 Linear Regression with Uniformly Distributed Errors

Now we implement the same problem, but the generating random distribution for the errors is no longer a Gaussian, but a uniform distribution:

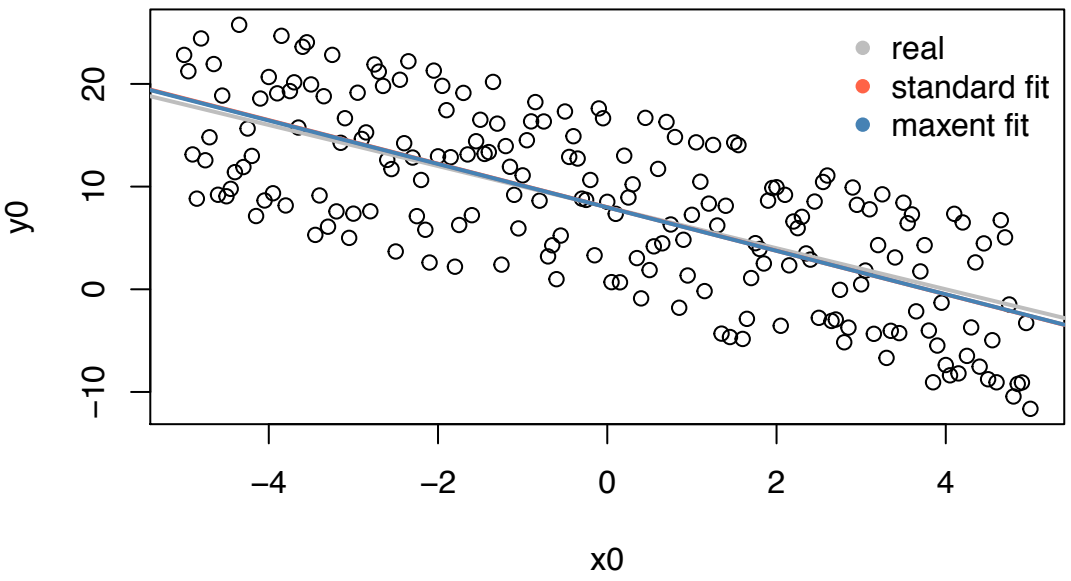
```
x0<-seq(-5,5,by=0.05)
y0<-8-2*x0+runif(length(x0),-10,10)
v<-c(-3*sd(y0),-1.5*sd(y0),0,1.5*sd(y0),3*sd(y0))
z<-c(-50,-25,0,25,50,-30,-15,0,15,30)
res_l3<-auglag(init,entropy,entropy_gr,heq=heq,heq.jac=heq.jac)
```



beta0

beta1 7.975009 -2.115185

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	7.9902	0.4201	19.02	0.0000
x0	-2.1211	0.1448	-14.65	0.0000



10.9 NlcOptim package: solnl function for 3 events, with unknown and known mean 2.5

Solve Optimization Problem With Nonlinear Objective And Constraints

Sequential Quatratic Programming (SQP) method is implemented to find solution for general nonlinear optimization problem (with nonlinear objective and constraint functions). The SQP method can be find in detail in Chapter 18 of Jorge Nocedal and Stephen J. Wright's book. Linear or nonlinear equality and inequality constraints are allowed. It accepts the input parameters as a constrained matrix. The function `solnl` is to solve generalized nonlinear optimization problem:

$$min f(x)$$

$$s.t. ceq(x) = 0$$

$$c(x) \leq 0$$

$$Ax \leq B$$

$$Aeqx \leq Beq$$

$$lb \leq x \leq ub$$

Usage

```
solnl(X = NULL, objfun = NULL, confun = NULL, A = NULL, B = NULL,
  Aeq = NULL, Beq = NULL, lb = NULL, ub = NULL, tolX = 1e-05,
  tolFun = 1e-06, tolCon = 1e-06, maxnFun = 1e+07, maxIter = 4000)
```

Figure 5: **Documentation of Function** *solnl*

```
constraints<-function(x){
  f<-NULL
  f<-rbind(f,sum(x)-1)
  foo<-NULL
  foo<-as.matrix(-x) # inequalities have to be written as <0
  return(list(ceq=f,c=foo))
}

x0<-rnorm(3,1/3,0.1/3)
res_3<-solnl(x0,objfun=entropy,confun=constraints)

val<-c(1,2,3)
mu<-2.5
#constraint function
constraints<-function(x){
  f<-NULL
  f<-rbind(f,sum(x)-1)
  f<-rbind(f,sum(val*x)-mu)
  foo<-NULL
  foo<-as.matrix(-x) # inequalities have to be written as <0
  return(list(ceq=f,c=foo))
}

x0<-rnorm(3,1/3,0.1/3)
res_3_1<-solnl(x0,objfun=entropy,confun=constraints) # analytical values are 0.1161352 0.2674026 0.6156
```

```
res_3$par

##           [,1]
## [1,] 0.3333314
## [2,] 0.3333596
## [3,] 0.3333090

res_3_1$par

##           [,1]
## [1,] 0.1162041
## [2,] 0.2675919
## [3,] 0.6162041
```

11 Priors (Chapter 8)

Prior information is the information that is available in advance of inference and is then used in conjunction with the new observable information for the inference. Prior information incorporates any information that potentially influences the specification of a problem but that arises outside of the determining system. Constructing and quantifying this prior information remains a challenge across all disciplines – especially in the social sciences, where priors are often based not on observed phenomena but rather on underlying unobserved beliefs.

By introducing prior information in the maxent framework, we finally obtain the complete infometrics problem. As in the original template, the setting is a state space with K states, $x_1, ..., x_K$, and a corre-

sponding target distribution f over K (p for Golan). The prior frequency distribution is f_0 or q and may be uniform.

Now the objective is to minimize the Kullback-Leibler divergence between f and f_0 subject to M moment constraints defined by functions g_m, including noise w. We want distributions f and w to bring in minimal information with respect to f_0 and w_0.

$$\begin{aligned} \underset{P}{\text{minimize}} \quad & D_{KL}(f||f_0) + D_{KL}(w||w_0) = \sum_k f_k \log \frac{f_k}{f_{0k}} \\ \text{subject to} \quad & \sum_k f_k g_m(x_k) = y_m \quad m = 1, ..., M; \\ & \sum_j f_k = 1 \end{aligned}$$

11.1 Multi-Parameter Example: Size Distribution - An Industry Simulation

This is the example of Uniformia in Golan's book. First, we solve the problem with standard entropy maximization [Golan, 2018, pp.114-117] and in the second stage we solve the problem with a prior distribution, where we will minimize the Kullback-Leibler divergence with respect to our priors [Golan, 2018, pp.206-207].

Uniformia is a new country we know nothing about except that it has one industry with $K = 10$ different sizes of firms. Each firm uses a single input to produce a single output. We want to infer the size distribution of firms p_k based on all possible information – but we only know the average input $x_1, ..., x_{10}$ and output $y_1, ..., y_{10}$ levels per firm size $k = 1, ..., 10$, as well as their aggregate averages $\bar{x}$ and $\bar{y}$:

$$\langle X \rangle = \bar{X} = \sum_{k=1}^{10} x_k p_k \tag{93}$$

$$\langle Y \rangle = \bar{Y} = \sum_{k=1}^{10} y_k p_k \tag{94}$$

These are two constraints in addition to normalization, $\sum_{k=1}^{10} p_k = 1$. Further, output and input levels are not independent, but related to each other through the production function $y_k = f_k(x_k)$:

$$y_k = \alpha x_k^{\beta_k} + \epsilon_k \tag{95}$$

where $\alpha = 2$, $\beta_k = (.4, .4, .4, .4, .7, .7, .7, 1.5, 1.5)$, ϵ_k is a zero-mean Gaussian error with variance $(1, 1, 1, 1, 1, 2, 2, 3, 3, 3)$. Via the β, the largest two groups have increasing returns to scale, while the rest have decreasing returns to scale.

11.1.1 Without priors

Without priors, the size distribution optimization problem is then:

$$\begin{aligned} \underset{P}{\text{maximize}} \quad & H(p) = -\sum_k p_k \log p_k \\ \text{subject to} \quad & \\ & \langle X \rangle = \bar{X} = \sum_{k=1}^{10} x_k p_k \\ & \langle Y \rangle = \bar{Y} = \sum_{k=1}^{10} y_k p_k \\ & \sum_{k=1}^{10} p_k = 1 \end{aligned}$$

with solution

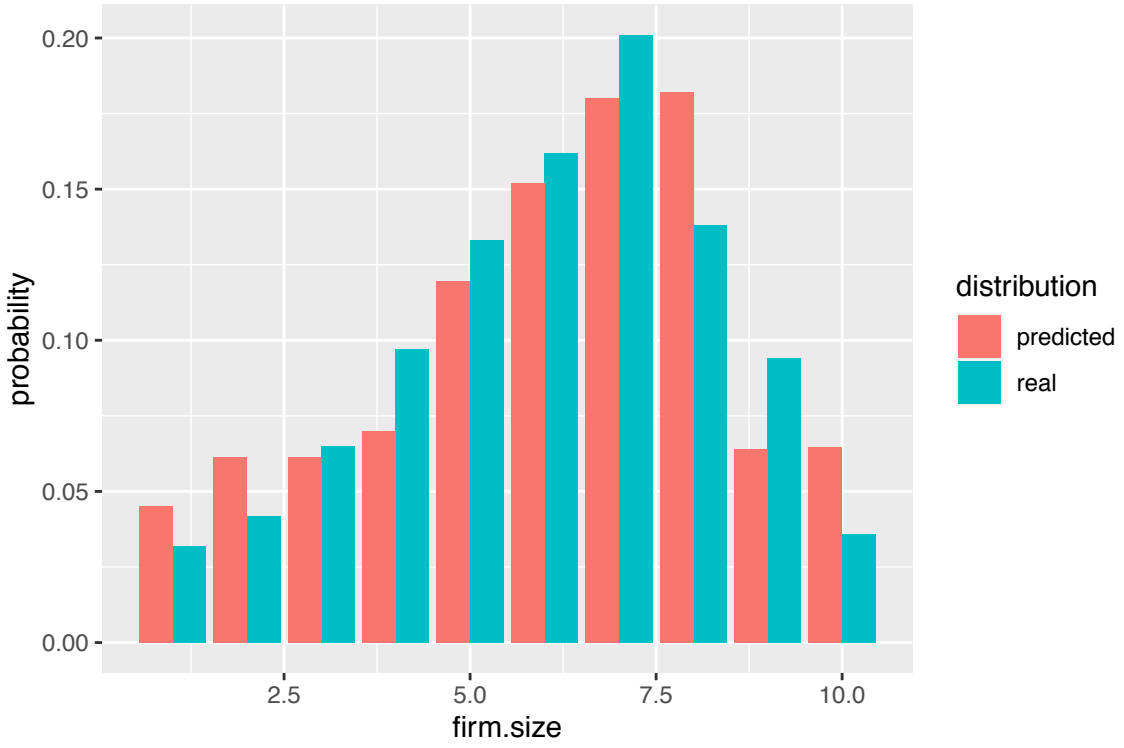
$$p_k^* = \frac{\exp\left(-\lambda_1^* x_k - \lambda_2^* y_k\right)}{\sum^1 0_{k=1} \exp\left(-\lambda_1^* x_k - \lambda_2^* y_k\right)} = \frac{\exp\left(-\lambda_1^* x_k - \lambda_2^* y_k\right)}{\Omega(\lambda_1^*, \lambda_2^*)} \tag{96}$$

Computationally, we set up the problem, where *p_pop* is the real distribution from which we generate the synthetic data, and instead of x and y we write the vectors as *input* and *output*, respectively.

```
K<-10
size<-1:K
sigma<-c(1,1,1,1,sqrt(2),sqrt(2),sqrt(2),sqrt(2),sqrt(3),sqrt(3))
set.seed(10)
error<-rnorm(10,0,.1)*sigma
input<-runif(10,1,100)
input<-input[order(input)]
p_pop = c(.032, .042, .065, .097, .133, .162, .201, .138, .094, .036) # "real" distribution
beta = c(.4, .4, .4, .4, .7, .7, .7, .7, 1.5, 1.5)
a<-2 # alpha
output<-a*input^beta+error
x_bar<-sum(p_pop*input)
y_bar<-sum(p_pop*output)
```

Following the usual procedure, we set up the constraints in *heq*, with the corresponding Jacobian.

```
heq <- function(x) { # vector
  h <- rep(NA, 1)
  h[1] <- sum(x) - 1 # zeroth moment constraint: normalization
  h[2] <- sum(x*input) - x_bar # first moment constraint: input mean
  h[3] <- sum(x*output) - y_bar # first moment constraint: output mean
  h
}
heq.jac <- function(x) { # matrix
  j <- matrix(NA, 3, length(x))
  j[1, ] <- rep(1,K) # partial derivatives of the normalization constraint wrt the p's
  j[2, ] <- input
  j[3, ] <- output
  j
}
results_0<-auglag(rep(1/K,K),entropy,entropy_gr,heq=heq,heq.jac=heq.jac,hin=hin,hin.jac=hin.jac)
```



11.1.2 With priors

If we have a prior distribution q that is not uniform, then instead of maximizing the entropy of p, we minimize the Kullback-Leibler divergence from the prior distribution q to our predicted, "posterior" distribution p. If we have a prior distribution q that is uniform, we are back to maximizing entropy:

$$D_{KL}(p||q) = \sum_k p_k \log \frac{p_k}{q_k} = \sum_k p_k \log \frac{p_k}{1/N} = \sum_k p_k \log(p_k N) = \underbrace{\sum_k \log p_k}_{-H(p)} + \log N \underbrace{\sum_k p_k}_{1}$$

$$D_{KL}(p||q) == \log N - H(p)$$

$\log N$ is the entropy of the uniform distribution, which is a constant that will vanish in the constrained-optimization problem.

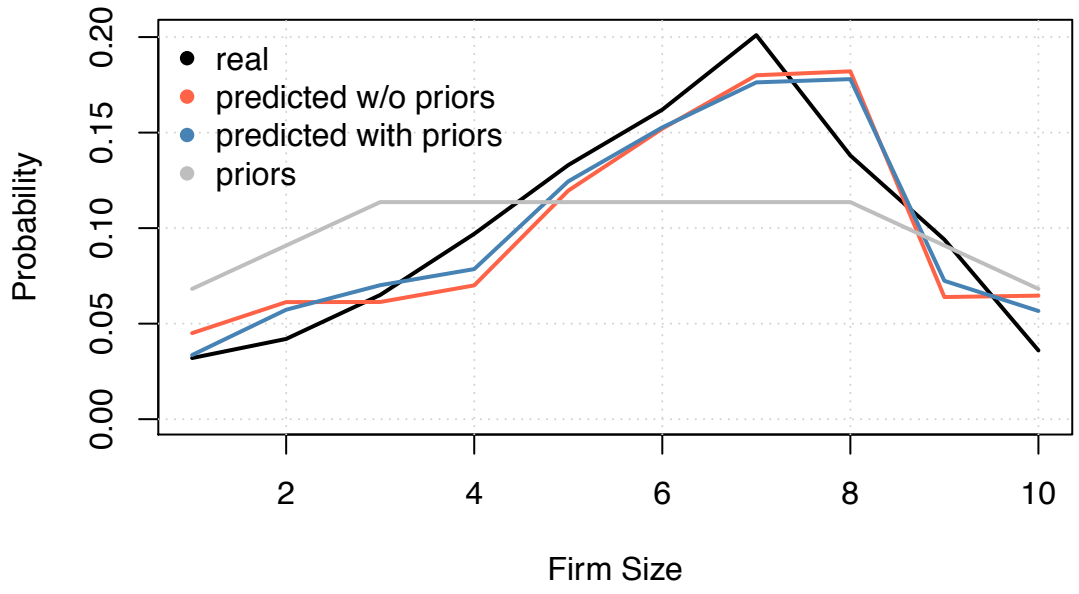
The problem of constrained-optimization for the Uniformia example is now the same as before, but now instead of maximizing entropy we minimize the KL-divergence. Hence, we re-write the objective function and its gradient, while setting up the same priors than Golan [Golan, 2018, pp.206-207]:

```
priors<-c(.0682,0.0909,rep(0.1136,6),0.0909,0.0682)
KL<-function(x){sum(x*log(x/priors))}
KL_gr<-function(x) {log(x/priors)+priors}
results_1<-auglag(rep(1/K,K),KL,KL_gr,heq=heq,heq.jac=heq.jac,hin=hin,hin.jac=hin.jac)
```

We plot the results:

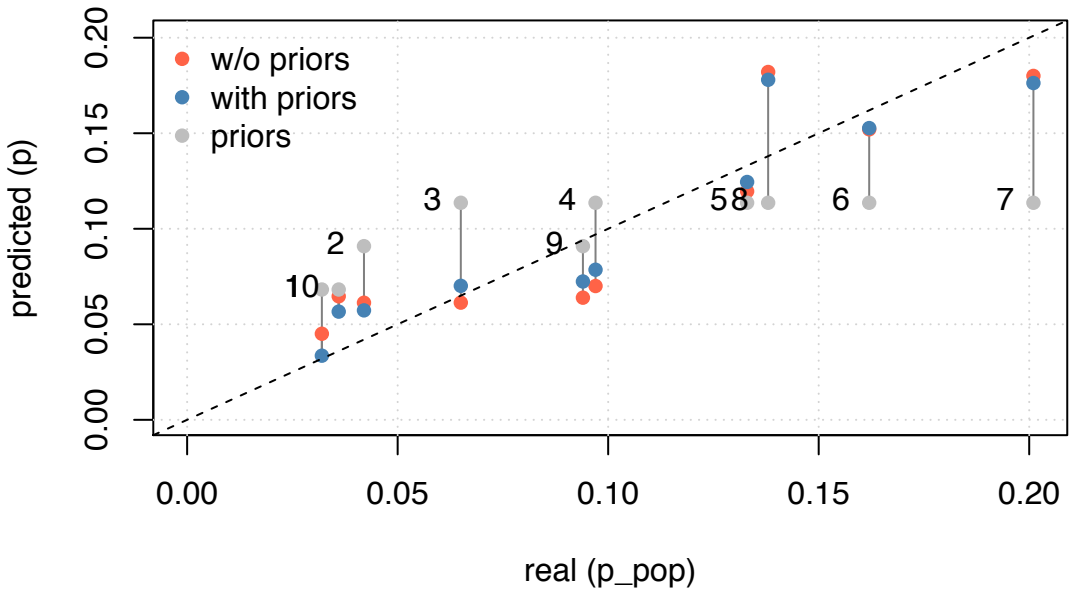
```
plot(NA,ylim=c(0,max(p_pop)),xlim=c(1,10),xlab='Firm Size',ylab='Probability')
grid()
```

```
lines(p_pop,lwd=2)
lines(results_0$par,col='tomato',lwd=2)
lines(results_1$par,col='steelblue',lwd=2)
lines(priors,col='gray',lwd=2)
legend('topleft',c('real','predicted w/o priors','predicted with priors','priors'),col=c('black','tomat
```



In order to facilitate the comparison of our predictions, we can also plot the values of the real distribution *p_pop* vs the predicted distributions with and without priors, and the priors. The numbers next to the priors refer to the firm size:

```
plot(NA,xlim=c(0,max(p_pop)),ylim=c(0,max(p_pop)),xlab='real (p_pop)',ylab='predicted (p)')
grid()
for(k in 1:K){lines(rep(p_pop[k],3),c(results_0$par[k],results_1$par[k],priors[k]),col=gray(0.5))}
points(p_pop,results_0$par,col='tomato',pch=16)
points(p_pop,results_1$par,col='steelblue',pch=16)
points(p_pop,priors,col='gray',pch=16)
abline(a=0,b=1,lty=2)
legend('topleft',c('w/o priors','with priors','priors'),col=c('tomato','steelblue','gray'),pch=16,bty='
text(p_pop,priors,labels=1:K,pos=2)
```



Finally, using our function KL, we can also compute the two KL divergences with respect to the prior distribution for both predicted distributions. As expected, the KL-divergence for the predicted distribution with priors is smaller than without priors.

```
KL(results_0$par)

## [1] 0.0787377

KL(results_1$par)

## [1] 0.07386316
```

12 Markov Processes

In many sciences, **dynamical systems**, such as Markov processes, play a central role in the description of reality. The basic idea is to represent the **state of a system** at a particular time t in terms of measurements, i.e. different variables or dimensions, which is called the **state vector** $x(t)$. The space in which the state vector is defined (usually a finite-dimensional Euclidean space) is called the **state space**. The basic idea of a dynamical system is that certain *laws* govern the evolution of the state vector. Hence, we can define a dynamical system as a state space S, a set of times T, and a rule R that regulates the temporal evolution of the system:

$$R : S \times T \to S \tag{97}$$

The dynamical system is a **Markov process** if and only if the time rule R is probabilistic (i.e. stochastic) and time T and state space S are discrete[1]. In particular, the fundamental Markov property is its **mem-**

[1]For dynamical systems in general, state space S and time T can also be continuous and evolution rule R can be deterministic, for instance in the case of sets of ordinary differential equations.

	$y_j = y_{t+1}$			
$y_i = y_t$	$y_{t+1} = y_1$	$y_{t+1} = y_2$	$y_{t+1} = y_3$	$\sum_j p_{ij}$
$y_t = y_1$	p_{11}	p_{12}	p_{13}	1
$y_t = y_2$	p_{21}	p_{22}	p_{23}	1
$y_t = y_3$	p_{31}	p_{32}	p_{33}	1

Table 1: **Markov Transition Matrix showing conditional probabilities $p_{ij} = p(y_t \to y_{t+1})$, with $K = 3$ states**

orylessness, since the current state of the system S is entirely determined by its previous state – albeit following probability transitions from one state to the other in a stochastic, not deterministic way. Golan addresses Markov processes in pages 250–254 and pages 319–324.

If state space S can take K states, we can characterize the Markov process by a $K \times K$ matrix of transition probabilities $p_{i \to j} = p(j|i) = p_{ij}$ where $i = t$ and $j = t + 1$, two successive instants in discrete time T, that is, $p(y_t \to y_{t+1})$ (see table 1). Let Y be a random variable that can take K mutually exclusive values, i.e. the possible states of the system y_1, y_2, ..., y_K of state space S. We can thus write the Markov process as:

$$P(y_{t+1}|y_t, y_{t-1}, y_{t-2}) = p(y_{t+1}|y_t) \tag{98}$$

where the joint probability of Y is

$$p(y_1, ..., y_t) = p(y_1)p(y_2|y_1)p(y_2|y_3)...p(y_t|y_{t-1})$$

Now in Golan's maximum-entropy framework, consider inferring the transition probabilities of a system from time t to time $t+1$ characterized by a first-order Markov process [Golan, 2018, pp.250-256]. Let $q_{j,t}$ be the frequency of individual states j in time period t so that $\sum_j q_{j,t} = 1$. Using transition probability from i to j p_{ij}, we can thus write

$$q_{j,t+1} = \sum_i p_{ij} q_{j,t}$$

where $\sum_j p_{ij} = 1$ and $q_{j,t}$ and $q_{j,t+1}$ are the observed frequencies of each state at each one of the time periods t and $t + 1$, for $t = 1, ..., T$.

Is it realistic to assume, though, that the system is already in a stationary equilibrium? In fact, it is more realistic to assume that the system is still away from its stationary equilibrium. It is also natural to acknowledge that the transition matrix P, which we want to infer, is just an approximate theory about an evolving process. Consequently, we write

$$q_{j,t+1} = \sum_i p_{ij} q_{j,t} + \epsilon_{j,t} = \sum_i p_{ij} q_{j,t} + \sum_s w_{js,t} v_{s,t} \tag{99}$$

where, as before, noise V_t is a discrete random variable of dimension $S \geq 2$ for each period t, with symmetric-about-zero support space, and $w_{js,t}$ are the weights for that support, i.e $\epsilon_{j,t} = \sum_s w_{js,t} v_{s,t}$. In the particular case of Markov processes, errors $\epsilon_{j,t}$, and thus the support space V_t, are naturally bounded within the interval $[-1, +1]$.

The **info-metrics inferential procedure for Markov processes** is

$$\begin{aligned}
\underset{P}{\text{minimize}} \quad & D_{KL}(p, w||p^0, w^0) = \sum_{ij} p_{ij} \log \frac{p_{ij}}{p^0_{ij}} + \sum_{jst} w_{js,t} \log \frac{w_{js,t}}{w^0_{js,t}} \\
\text{subject to} \quad & q_{j,t+1} = \sum_i p_{ij} q_{i,t} + \epsilon_{j,t} = \sum_i p_{ij} q_{j,t} + \sum_s w_{js,t} v_{s,t}; \quad t = 1, ..., T-1; j = 1, ..., K \\
& \sum_j p_{ij} = 1; \quad i = 1, ..., K \\
& \sum_s w_{js,t} = 1; \quad t = 1, ..., T-1; j = 1, ..., K
\end{aligned} \tag{100}$$

where p^0_{ij} and $w^0_{js,t}$ are the set of prior probabilities for the transition probabilities and the noise weights, respectively. Unless we have more precise information, we take these noise weights to be uniform over a mean zero support, i.e. $w^0_{js,t} = 1/S$ for all t and j.

The general solution for the transition probability matrix is

$$p^*_{ij} = \frac{p^0_{ij} \exp\left(\sum_{t=1}^{T-1} \lambda^*_{j,t} q_{i,t}\right)}{\sum_j p^0_{ij} \exp\left(\sum_{t=1}^{T-1} \lambda^*_{j,t} q_{i,t}\right)} = \frac{p^0_{ij} \exp\left(\sum_{t=1}^{T-1} \lambda^*_{j,t} q_{i,t}\right)}{\Omega_i(\lambda^*)} \tag{101}$$

and for the noise weights

$$w^*_{js,t} = \frac{w^0_{js,t} \exp\left(\lambda^*_{j,t} v_{t,s}\right)}{\sum_s w^0_{js,t} \exp\left(\lambda^*_{j,t} v_{t,s}\right)} = \frac{w^0_{js,t} \exp\left(\lambda^*_{j,t} v_{t,s}\right)}{\Psi_{j,t}(\lambda^*_{j,t})} \tag{102}$$

The inferred probabilities p are the stationary transition probabilities that are as close as possible to the theoretical ones, even though the system may still be slowly evolving and the information may be noisy. We can now evaluate the impact of each state frequency $q_{i,t}$[2] on transition probability p_{ij}:

$$\frac{\partial p^*_{ij}}{\partial q_{i,t}} = p^*_{ij} \left(\lambda^*_{j,t} - \sum_j p^*_{ij} \lambda^*_j \right) \tag{103}$$

In Golan, the possibility of having more additional information in the form of X over space H that is external to the transition is considered, which may be state-specific or inclusive of all states.

```
# Markov Chains
M<-t(matrix(c(.2,.4,.4,.1,.3,.6,.5,.1,.4),nrow=3,ncol=3))
write_matex(M)  # write_matex is an auxiliary function to print matrices in Latex
```

$$\begin{bmatrix} 0.2 & 0.4 & 0.4 \\ 0.1 & 0.3 & 0.6 \\ 0.5 & 0.1 & 0.4 \end{bmatrix}$$

```
# M<-matrix(c(0.9,.15,.25,.075,.8,.25,.025,.05,.5),nrow=3,ncol=3)
eigen(t(M))$values
```

```
[1] 1.00+0.0000000i -0.05+0.2397916i -0.05-0.2397916i
```

```
write_matex(round(eigen(t(M))$vectors,2))
```

$$\begin{bmatrix} -0.52 + 0i & 0.67 + 0i & 0.67 + 0i \\ -0.4 + 0i & -0.42 - 0.4i & -0.42 + 0.4i \\ -0.75 + 0i & -0.25 + 0.4i & -0.25 - 0.4i \end{bmatrix}$$

```
v<-Re(eigen(t(M))$vectors[,1])
v
```

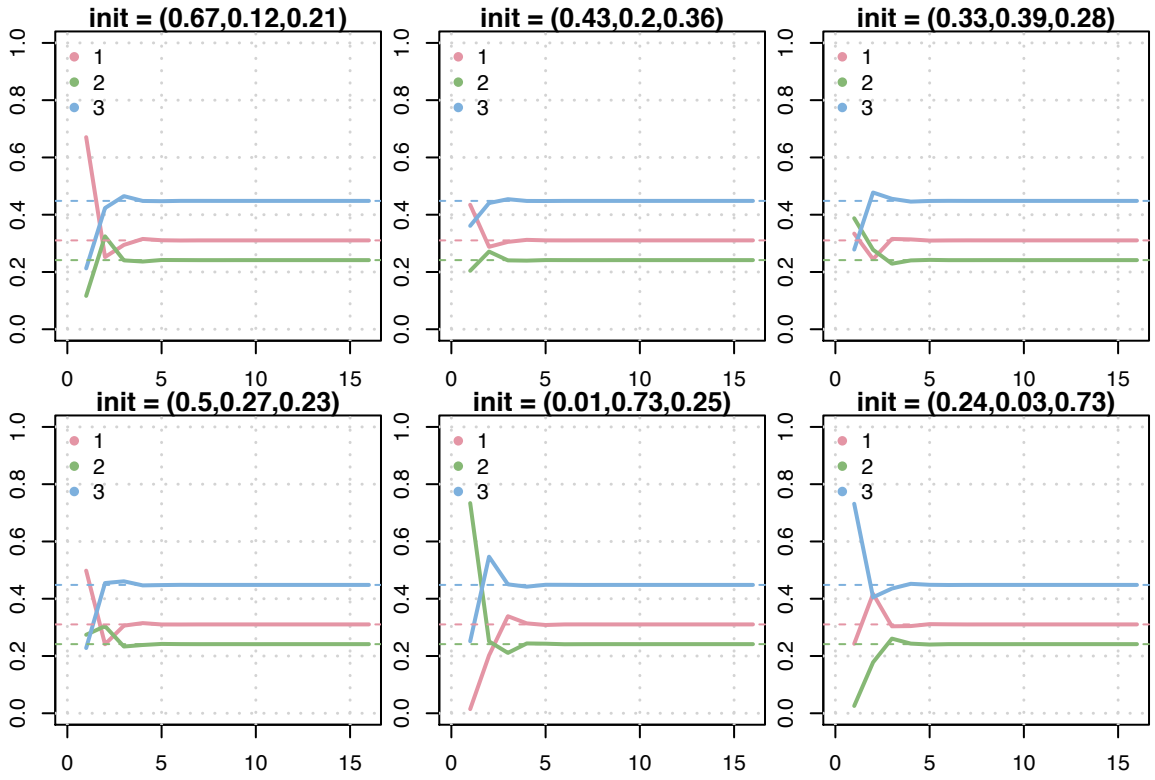
```
[1] -0.5204834 -0.4048205 -0.7518094
```

[2] In Golan, the partial derivative has $q_{j,t}$, but it may be a typo (we write i instead?).

```
ergodic.M<-v/sum(v)
ergodic.M
```

[1] 0.3103448 0.2413793 0.4482759

```
# Function Markov Evolution
markov<-new('markovchain',states=as.character(c(1,2,3)),byrow=T,transitionMatrix=M,name='markov.test')
evo.markov<-function(init,time) {cbind(init,sapply(1:time,function(t){init*markov^t}))}

# Function Plot Markov Evolution
plot.markov<-function(run)
{
  plot(NA,ylim=c(0,1),xlim=c(0,ncol(run)),xlab='iteration',ylab='p(state)',main=paste0('init = (',paste(
  grid(lwd=1.5)
  for(i in 1:nrow(run)) {lines(1:ncol(run),run[i,],lwd=2,col=rainbow_hcl(nrow(run))[i])}
  abline(h=ergodic.M,col=rainbow_hcl(nrow(run)),lty=2)
  legend('topleft',as.character(c(1,2,3)),col=rainbow_hcl(3),pch=16,bty='n')
}

par(mfrow=c(2,3),mar=c(1.85,1.95,1.1,0.35))
for(i in 1:6)
{
  freq.0<-runif(3,0,3)
  plot.markov(evo.markov(freq.0/sum(freq.0),15))
}
```



```
par(mfrow=c(1,1),mar=c(5.1, 4.1, 4.1, 2.1))
```

12.1 Example: Business Cycle

Now let's produce a time series of GDP growth by constructing an economy x_t that grows at rate r and with carrying capacity of the environment K following deterministic logistic growth ,

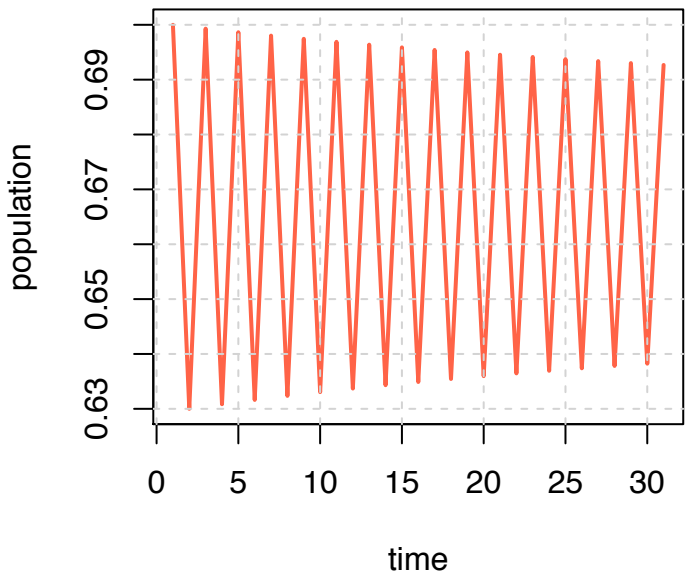
$$x_{t+1} = rx_t \left(1 - \frac{x_t}{K}\right) \tag{104}$$

A simple model on these lines of the Ricardian growth model can be found in Bhaduri and Harris [Bhaduri and Harris, 1987]. Since x can be rescaled by K so that it goes from 0 to 1 instead that from 0 to K, we have the logistic equation

$$x_{t+1} = rx_t \left(1 - x_t\right) \tag{105}$$

This deterministic dynamical system in discrete time, but in continuous space, is well-known in epidemiology in order to understand the spread over time of infectious diseases, for instance the coronavirus. In this case, the carrying capacity K is the population susceptible to infection or final epidemic size.

```
logistic<-function(init,par,timelength)
{
  x<-c()
  x[1]<-init
  for(t in 1:timelength) {x[t+1]<-par*x[t]*(1-x[t])}
  return(x)
}

par(mfrow=c(1,1),mar=c(5.1, 4.1, 4.1, 2.1))
plot(logistic(init=0.7,par=3,timelength=30),type='l',lwd=2,col='tomato',xlab='time',ylab='population')
grid(lty=2)
```

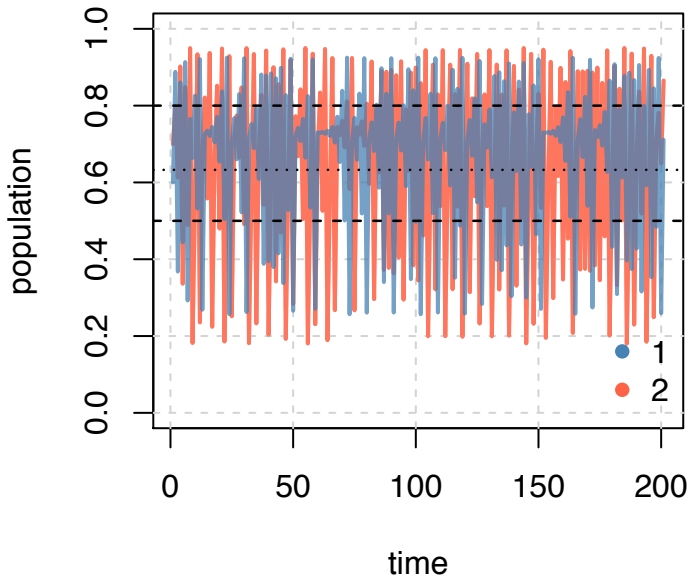


Logistic growth exhibits deterministic chaos within the interval of the parameter r between 3.56 and 4. Deterministic chaos consists of very irregular, aperiodic oscillations over continuous space that cannot be predicted in the long run. Since Markov processes operate on discrete space, we divide the economic regimes of the business cycle in three discrete states, e.g. boom (for $x > b$), bust (for $x < a$), and normal ($a < x < b$) or boom (for $x > \langle x \rangle$) and bust (for $x < \langle x \rangle$):

```
x.1<-logistic(init=0.6,par=3.7,timelength=200)
x.2<-logistic(init=0.7,par=3.8,timelength=200)

a<-0.5
b<-0.8
y<-z<-c()
y[x.2<mean(x.2)]<-'bust'
y[x.2>mean(x.2)]<-'boom'
z[x.1<a]<-'bust' # bust
z[x.1>b]<-'boom' # boom
z[x.1>a & x.1<b]<-'normal' # normal

par(mfrow=c(1,1),mar=c(5.1, 4.1, 4.1, 2.1))
plot(x.2,ylim=c(0,1),type='l',lwd=2,col=alpha('tomato',0.5),xlab='time',ylab='population')
grid(lty=2)
lines(x.2,col=alpha('tomato',0.75),lwd=2)
lines(x.1,col=alpha('steelblue',0.75),lwd=2)
legend('bottomright',c('1','2'),col=c('steelblue','tomato'),pch=16,bty='n')
abline(h=c(mean(x.2),a,b),lty=c(3,2,2),lwd=1.25)
```

Table 2: **Transition Matrix for y (trajectory 2)**

	boom	bust
boom	0.233	0.767
bust	0.814	0.186




```
# Function Compute Transition Probabilities from Observed Evolution
trans.M<-function(y0)
{
  states<-unique(y0)
  t.M<-matrix(0,ncol=length(states),nrow=length(states),dimnames=list(states,states))
  for(i in states)
  {
    trp<-table(y0[which(y0==i)+1])/sum(table(y0[which(y0==i)+1]))
    t.M[i,names(trp)]<-trp
  }
  t.M
}
kable(round(trans.M(y),3),booktabs=T,caption="\\bf \\small Transition Matrix for y (trajectory 2) \\labe
```

```
rowSums(trans.M(y))
```

boom bust 1 1

Table 3: **Transition Matrix for z (trajectory 1)**

	normal	boom	bust
normal	0.643	0.357	0.000
boom	0.241	0.000	0.759
bust	0.477	0.523	0.000

```
kable(round(trans.M(z),3),booktabs=T,caption="\\bf \\small Transition Matrix for z (trajectory 1) \\labe
```

```
rowSums(trans.M(z))
```

normal boom bust 1 1 1

```
# Function Simulate a Markov Trajectory
trial.run<-function(t.M,y0)
{
  y.run<-c()
  y.run[1]<-y0[1]
  for(t in 1:(length(y)-1))
  {
    y.run[t+1]<-sample(colnames(t.M),size=1,prob=t.M[y.run[t],])
  }
  y.run
}

run.y<-trial.run(trans.M(y),y)
table(run.y==y)
```

FALSE TRUE 94 107

```
v<-Re(eigen(t(trans.M(y)))$vectors[,1])
v/sum(v) # Ergodic Distribution
```

[1] 0.515 0.485

```
run.z<-trial.run(trans.M(z),z)
table(run.z==z)
```

FALSE TRUE 115 86

```
v<-Re(eigen(t(trans.M(z)))$vectors[,1])
v/sum(v) # Ergodic Distribution
```

[1] 0.49 0.29 0.22

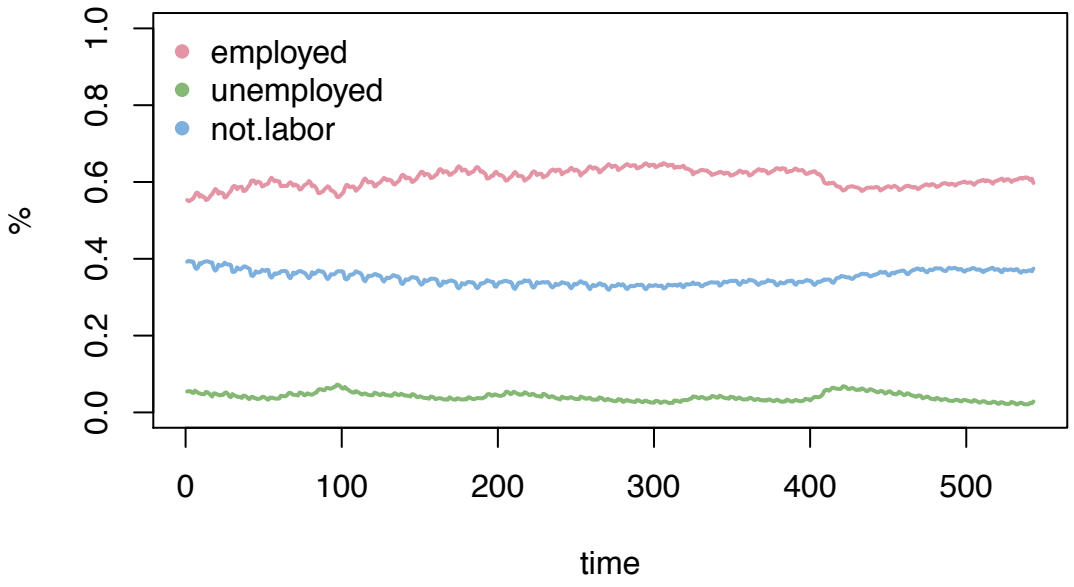
12.2 Example: US Employment

First we download US employment data from the Bureau of Labor Statistics: labor force, employed, unemployed, and not in the labor force.

Table 4: **First 6 Observations of emp data frame**

	employed	unemployed	not.labor
727	0.609	0.026	0.365
728	0.603	0.025	0.373
729	0.605	0.022	0.373
730	0.607	0.022	0.371
731	0.607	0.022	0.371
732	0.604	0.023	0.372

```
labor.force<-read.csv('US_labor_data.csv')
employed<-read.csv('employed_data.csv')
unemployed<-read.csv('unemployed_data.csv')
not.labor<-read.csv('not_labor_force_data.csv')
emp.data<-data.frame(year=labor.force$Year,period=labor.force$Period,labor.force=labor.force$Value,empl
emp.data<-data.frame(emp.data[emp.data$year>1974,],not.labor=not.labor$Value)
emp<-emp.data[,c('employed','unemployed','not.labor')]/emp.data$labor.force
plot(NA,xlim=c(1,nrow(emp)),ylim=c(0,1),xlab='time',ylab='%')
for(i in 1:3){lines(1:nrow(emp),emp[,i],col=rainbow_hcl(3)[i],lwd=2)}
legend('topleft',c('employed','unemployed','not.labor'),pch=16,col=rainbow_hcl(3),bty='n')
```




```
emp<-emp[(nrow(emp)-20):nrow(emp),] # we take only the last 20 time periods
kable(round(head(emp),3),booktabs=T,caption="\\bf \\small First 6 Observations of emp data frame \\label
```

Once again, the **info-metrics inferential procedure for Markov processes** is

$$\begin{aligned}
\underset{P}{\text{minimize}} \quad & D_{KL}(p, w||p^0, w^0) = \sum_{ij} p_{ij} \log \frac{p_{ij}}{p^0_{ij}} + \sum_{jst} w_{js,t} \log \frac{w_{js,t}}{w^0_{js,t}} \\
\text{subject to} \quad & q_{j,t+1} = \sum_i p_{ij} q_{i,t} + \epsilon_{j,t} = \sum_i p_{ij} q_{j,t} + \sum_s w_{js,t} v_{s,t}; \quad t = 1, ..., T-1; j = 1, ..., K \\
& \sum_j p_{ij} = 1; \quad i = 1, ..., K \\
& \sum_s w_{js,t} = 1; \quad t = 1, ..., T-1; j = 1, ..., K
\end{aligned} \tag{106}$$

The transition matrix p_{ij} is $K \times K$ where K is the number of states of the phase space. For our algorithm setup, the objective function that we minimize is the sum of the KL-divergences of K^2 transition probabilities p and $3K(T-1)$ noise weights $w_{js,t}$ where $s = 1, 2, 3$ is the index for the noise support V. Hence our vector x will have length $K^2 + 3K(T-1)$.

We will have $2K(T-1)+K = 2KT-K$ constraints in total: $K(T-1)$ constraints that are obtained from $q_{j,t+1}$ (i.e. K constraints for each time period t), K constraints that normalize the row sum of transition probabilities p_{ij}, and $K(T-1)$ constraints that normalize the noise weights (K constraints for each time period t).

In our example, $K = 3$, $S = 3$, and $T = 21$, so that x will have length $3^2 + 3 * 3 * 20 = 189$: the first K^2 values are for the transition probabilities and the remaining $S * K * (T-1)$ for the noise weights. For each time period t, we have $K = 3$ frequencies $q_{1,t}$, $q_{2,t}$, and $q_{3,t}$. For each of these $K = 3$ frequencies, we have $S = 3$ noise weights, so $K \times S = 9$ noise weights for each time period.

The gradient of the objective function is a vector with the partial derivatives of the KL-divergence with respect to each of the probabilities p and w (which are the same function...).

```
p0<-rep(1/9,9)
v<-c(-1,0,1) # same support space for all T-1 observations
w0<-rep(1/3,3)
init<-c(p0,rep(1/3,3*3*(nrow(emp)-1)))

KL1<-function(x){sum(x[1:9]*log(x[1:9]/p0))+sum(x[10:length(x)]*log(x[10:length(x)]/w0))}
KL_gr1<-function(x)
{
  trp<-c(log(x[1:9]/p0)+p0,log(x[10:length(x)]/w0)+w0)
  trp[is.na(trp)]<-0
  trp
}
```

The *heq* vector includes all the constraints and hence it will have a length of $2KT-K = 2*3*21-3 = 123$. The first $K = 3$ constraints will be for each row of the transition matrix (row normalization), then we assign $K(T-1) = 3 * 20 = 60$ normalization constraints for the noise weights (positions $K+1 = 4$ to $K(T-1)+K = 63$ in the x vector), and finally $K(T-1)$ more constraints for each of the 3 observed frequencies per time period (positions $K(T-1)+K+1 = 64$)

```
heq <- function(x)
{
  h <- rep(NA, 1)

   # 3 constraints for each transition row
  for(i in 1:3){h[i] <- sum(x[(3*i-2):(3*i)]) - 1}
```

```
  pM<-t(matrix(x[1:9],ncol=3,nrow=3)) # auxiliary transition matrix
  for(t in 1:(nrow(emp)-1))
  {
    # selects the 9 weight positions in the x vector corresponding to period t
    cnt<-(9*(t-1)+1+9):(9*t+9)

    # 3 constraints for each time period for the noise weights
    h[3*t+1]<-sum(x[cnt[1:3]]) - 1
    h[3*t+2]<-sum(x[cnt[4:6]]) - 1
    h[3*t+3]<-sum(x[cnt[7:9]]) - 1

    # 1 constraint for each of the 3 observed frequencies per time period
    h[63+1+3*(t-1)]<-sum(pM[,1]*emp[t,])+sum(x[cnt[1:3]]*v)-emp[t+1,1] # q_{1,t+1}
    h[63+2+3*(t-1)]<-sum(pM[,2]*emp[t,])+sum(x[cnt[4:6]]*v)-emp[t+1,2] # q_{2,t+1}
    h[63+3+3*(t-1)]<-sum(pM[,3]*emp[t,])+sum(x[cnt[7:9]]*v)-emp[t+1,3] # q_{3,t+1}
  }
  h
}
heq.jac <- function(x)
{
  j <- matrix(0, 123, length(x)) # 123 rows for the constraints

  # derivative of first 3 constraints wrt the p's is 1
  j[1,1:3]<-1
  j[2,4:6]<-1
  j[3,7:9]<-1

  for(t in 1:(nrow(emp)-1))
  {
    cnt<-(9*(t-1)+1+9):(9*t+9)

    # 3 constraints for each time period for the noise weights
    j[3*t+1, cnt[1:3]]<- 1
    j[3*t+2, cnt[4:6]]<- 1
    j[3*t+3, cnt[7:9]]<- 1

    # 1 constraint for each of the 3 observed frequencies per time period
    # derivative wrt the p columns (j index) equals observed frequencies
    j[63+1+3*(t-1), c(1,4,7)]<-as.numeric(emp[t,]) # q_{1,t+1}
    j[63+2+3*(t-1), c(2,5,8)]<-as.numeric(emp[t,]) # q_{2,t+1}
    j[63+3+3*(t-1), c(3,6,9)]<-as.numeric(emp[t,]) # q_{3,t+1}

    # derivative of q_{j,t+1} constraint wrt noise weights is the noise support v
    j[63+1+3*(t-1), cnt[1:3]]<-v # q_{1,t+1}
    j[63+2+3*(t-1), cnt[4:6]]<-v # q_{2,t+1}
    j[63+3+3*(t-1), cnt[7:9]]<-v # q_{3,t+1}
  }
  j
}
hin <- function(x) { # vector
```

Table 5: **Transition Matrix for US Employment**

	employed	unemployed	not.labor
employed	0.5935	0.0649	0.3416
unemployed	0.3452	0.3168	0.3380
not.labor	0.5073	0.1300	0.3627

```
  h <- x # vector of probabilities must be nonnegative in all entries
  h
}
hin.jac <- function(x) { # matrix
diag(length(x)) # partial derivatives of hin wrt the p's
}
```

```
res.markov<-auglag(init,KL1,KL_gr1,heq=heq,heq.jac=heq.jac,hin=hin,hin.jac=hin.jac)
```

The transition matrix that we obtain is:

```
res.mat<-round(t(matrix(res.markov$par[1:9],ncol=3,nrow=3)),4)
colnames(res.mat)<-rownames(res.mat)<-c('employed','unemployed','not.labor')
kable(res.mat,booktabs=T,caption="\\bf \\small Transition Matrix for US Employment \\label{tab:p.M.3}",e
```

12.3 Example: Input-Output Case

In a 1995 contribution, Leontief and Brody present standard input-output relationships that are complemented by monetary stock-flow data [Leontief and Brody, 1993]. The flow of money is described as a Markov chain. Its ergodic state is equivalent to the economic equilibrium. The definition of the latter requires thus neither labour-theoretic nor marginalist assumptions. The Fisher equation for the velocity of money circulation can be expressed in this input-output context. The average velocity and its dispersion are then determined. The theorems are illustrated on a 5×5 sector Hungarian matrix.

In a 2000 contribution, Leontief's disciple, András Bródy, presents a model of the circulation of money as a Markov chain and a special multiplier is set up in order to describe the transitory propagation of a monetary injection into a given economy [Brody, 2000].

A given unit of money, after staying in state i (i.e. production sector i) for an (probabilistic) interval τ_{ik} turns suddenly into state k with probability a_{ik}. Let's first consider that all intervals τ_{jk} are of unit magnitude. $A = \{a_{jk}\}$ is a stochastic matrix of transition probabilities that can also be considered as the input-output coefficients of a closed Leontief economy.

If the system is self-replacing, then this matrix will have a maximal positive eigenvalue 1 on account of $x = Ax$. Thus, the Markov process generated by A is ergodic, i.e. the process converges to a limiting distribution irrespective of its initial state.

Transition will go either along the consecutive setps of the discrete chain

$$m_{t+1} = Am_t \tag{107}$$

or in continuous form

$$\frac{dm}{dt} = (A - I)m \tag{108}$$

where m_k is the vector of the money stock in each state or production sector at time t. The path of the discrete system will be traced by successively multiplying the initial vector m_0 by matrix A. After a while,

we arrive to the ergodic matrix A^* – Brody argues A^* has rank 1 as a single dyad with the equilibrium price and output vector.

Let's now inject money into the economy, which will leave an irreversible imprint on the system with a characteristic impact and time profile – which is equivalent to constructing a monetary multiplier. We have to consider only the non-equilibrium part of the injection and its propagation. The non-equilibrium part, deviation D, will be:

$$D = A - A^* \tag{109}$$

which propagates by a process in which it will be multiplied, in every step, by the matrix A, i.e. in general D_t will be the tth power of matrix D. We can sum up all the non-equilibrium impulses generated by the injection of new money, which will be equal to the Leontief-inverse of matrix D:

$$\Sigma = I + D + D^2 + ... + D^t = (I - D)^{-1} \tag{110}$$

This series will certainly converge, because the matrix D possesses all lesser eigenvalues of A (expect 1). This series can show the sectoral benefits. All column sums of the powers of D equal zero, thus there will be a clear separation of sectors of production that are winners and losers in the money injection.

In the more general case with differences in turnovers τ_{ik}, let's consider the matrix $B = \{a_{ik}\tau_{ik}\}$ expressing the density of monetary flows from production sector i to k. The column sums of this matrix are the branch-specific average turnover times. The diagonal matrix T expresses the density of the outflow of payments originating in the respective sectors. Column sums of A are of unit magnitude. The summation yields the average of the delay-times. The elements of T are therefore

$$T_i = \sum_k a_{ik}\tau_{ik} \tag{111}$$

The differential equation for the circulation processes may be now set up as

$$\frac{dm}{dt} = (B - T)m \tag{112}$$

which shows that the change in money stocks consists of the payments received Bm minus payments made Tm. In discrete time,

$$m_{t+1} = (I + B - T)m_t \tag{113}$$

An empirical illustration is provided for a small model of the Hungarian economy:

$$\begin{bmatrix} 0.21 & 0.18 & 0.11 & 0.15 & 0.05 \\ 0.2 & 0.24 & 0.51 & 0.17 & 0.37 \\ 0.14 & 0.27 & 0.12 & 0.25 & 0.15 \\ 0.22 & 0.12 & 0.09 & 0 & 0.33 \\ 0.23 & 0.19 & 0.17 & 0.43 & 0.1 \end{bmatrix}$$

and turnover times $\tau = (0.1, 0.05, 0.07, 0.4, 0.8)$ leading to the following monetary multiplier $M = (I - D)^{-1} - I$:

$$\begin{bmatrix} 9.51 & -1.02 & -1.23 & 0.23 & -1.41 \\ -7.82 & 7.19 & -5.36 & -4.9 & -6.12 \\ -2.13 & -2.85 & 9.48 & 0.28 & -2.36 \\ 0.36 & -0.25 & -0.23 & 1.56 & 0.41 \\ 0.08 & -3.07 & -2.66 & 2.83 & 9.48 \end{bmatrix}$$

Injecting money into the 5th sector (government) has only a small spillover effect on the first one (households) and is detrimental to everything else apart from government. What has been demonstrated here is that the classical multiplier can be broken down into a detailed picture that describes the processes both in space (among sectors of production) and through time (if solving the equation for the time path).

References

Bak, P., Tang, C., and Wiesenfeld, K. (1988). Self-organized criticality. *Physical Review A*, 38(1):364.

Bhaduri, A. and Harris, D. J. (1987). The complex dynamics of the simple ricardian system. *The Quarterly Journal of Economics*, 102(4):893–902.

Brody, A. (2000). The monetary multiplier. *Economic Systems Research*, 12(2):215–219.

Drăgulescu, A. and Yakovenko, V. M. (2001). Evidence for the exponential distribution of income in the usa. *European Physical Journal B*, 20(4):585–589.

Gabaix, X. (2009). Power laws in economics and finance. *Annu. Rev. Econ.*, 1(1):255–294.

Golan, A. (2018). *Foundations of info-metrics: Modeling, inference, and imperfect information*. Oxford University Press.

Kalecki, M. (1945). On the gibrat distribution. *Econometrica: Journal of the Econometric Society*, pages 161–170.

Leontief, W. and Brody, A. (1993). Money-flow computations. *Economic Systems Research*, 5(3):225–233.

MacKay, D. J. and Mac Kay, D. J. (2003). *Information theory, inference and learning algorithms*. Cambridge University Press.

Marx, K. (1976). *Capital: Vol. 1, A Critique of Political Economy*. New York: Penguin. Reprint 1990.

Mas-Colell, A., Whinston, M. D., Green, J. R., et al. (1995). *Microeconomic theory*, volume 1. Oxford university press New York.

Mirowski, P. (1989). The rise and fall of the concept of equilibrium in economic analysis. *Recherches Économiques de Louvain/Louvain Economic Review*, 55(04):447–468.

Mirowski, P. (1991). *More heat than light: economics as social physics, physics as nature's economics*. Cambridge University Press.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell Systems Technical Journal*, 27(3):379–423.

Sims, C. A. (2003). Implications of rational inattention. *Journal of monetary Economics*, 50(3):665–690.

Visser, M. (2013). Zipf's law, power laws and maximum entropy. *New Journal of Physics*, 15(4):043021.

ADV MATH METHODS FOR MODELING

Lab Notes on Info-Metrics

Oriol Vallès Codina

May 6, 2020

1 Parameter Estimation for a Double-Exponential Distribution

In the following example, we want to estimate the decay parameter b of a Laplace (or double-exponential) distribution (which we derived in section **??**):

$$p(x) = \frac{1}{2b} e^{-\frac{|x-\mu|}{b}} \tag{1}$$

The Laplace distribution is the observed distribution for firm profitability and appears often in many social and physical processes. Remember that this distribution is generated by a single constraint (in addition to normalization):

$$\langle |x - \mu| \rangle = b \tag{2}$$

We can also write the Laplace distribution in a piece-wise form that emphasizes its double-exponential character:

$$p(x) = \begin{cases} a e^{\lambda(x-\mu)} & \text{if } x < \mu \\ a e^{-\lambda(x-\mu)} & \text{if } x > \mu \end{cases} \tag{3}$$

The first moment of the Laplace distribution, the average μ, also indicates its maximum value around which the distribution is centered, and thus is very easy to find. Without loss of generality and for convenience, we will take $\mu = 0$. We can think that the values we record in our dataset are just $x - \mu$, so that the distribution is centered around zero.

We thus make $\mu = 0$ and take the natural logarithm of (3):

$$\ln p(x) = \begin{cases} \ln a + \lambda x & \text{if } x < 0 \\ \ln a - \lambda x & \text{if } x > 0 \end{cases} \tag{4}$$

We will try to estimate both of these parameters from a sample distribution we generate randomly by applying the maximum entropy problem for linear regressions. Our parameters will be rate $\lambda = 10$ and scale $a = 5$.

We first generate our values following a double-exponential distribution always distinguishing between the positive and the negative branches. Then we check its linearity in the log-linear plot and apply two standard linear fits there.

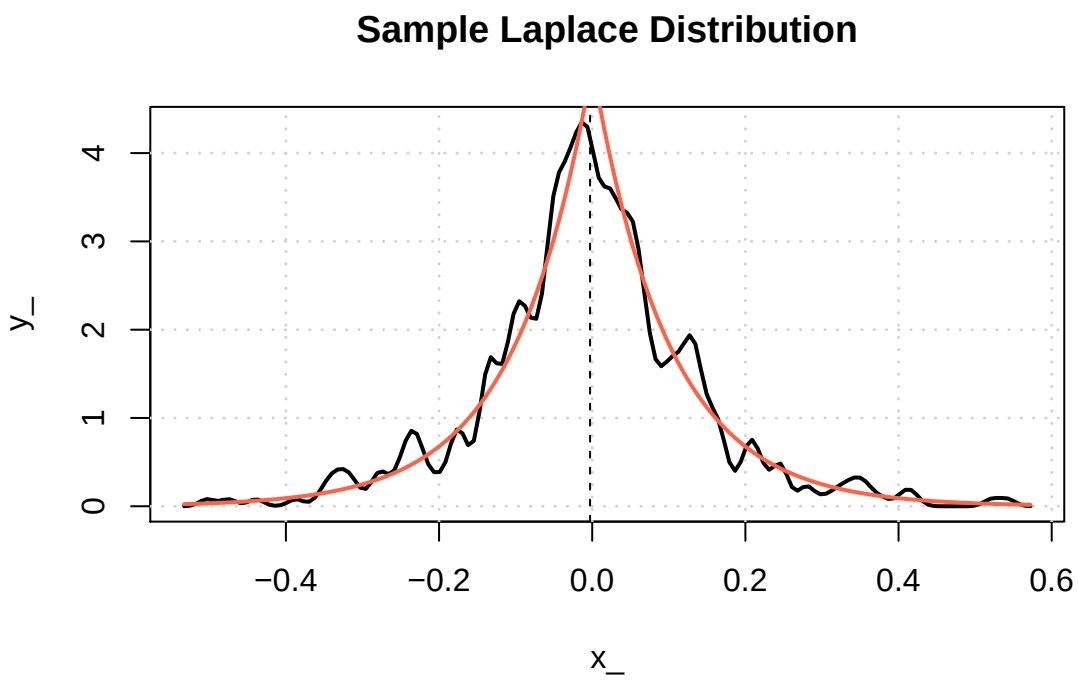
```
library(broom)
library(dplyr)

##
## Attaching package:  'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(xtable)
library(kableExtra)
```

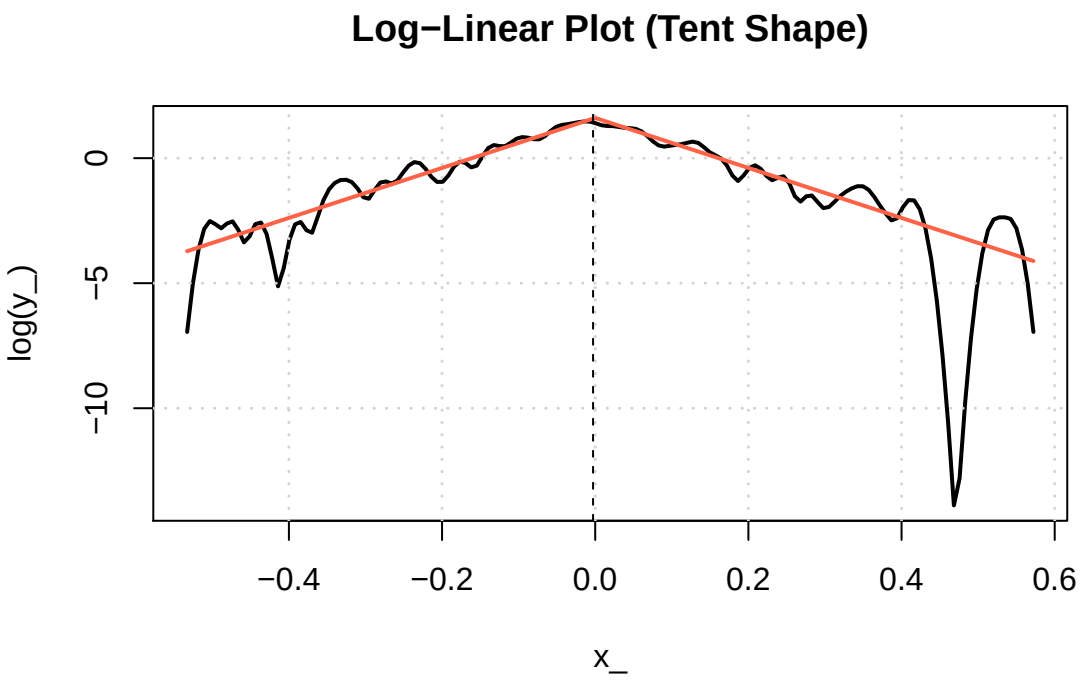
```
##
## Attaching package:  'kableExtra'
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
x<-(rexp(500,rate=10)-rexp(500,rate=10)) # scale parameter should be 10
x_<-density(x,bw=0.01,n=150)$x
y_<-density(x,bw=0.01,n=150)$y
plot(x_,y_,main='Sample Laplace Distribution',col='white')
grid(lwd=1.5)
lines(x_,y_,lwd=2)
abline(v=mean(x),lty=2)
lines(x_,5*exp(-abs(10*x_)),col='tomato',lwd=2) # in fact scale=5 is a better fit...
```




```
plot(x_,log(y_),type='l',lwd=2,main='Log-Linear Plot (Tent Shape)')
grid(lwd=1.5)
```

```
y.li<-log(5)-abs(10*x_)
lines(x_,y.li,col='tomato',lwd=2)
abline(v=mean(x),lty=2)
```




```
fit.positive<-lm(y_[x_>0] ~ x_[x_>0])
fit.negative<-lm(y_[x_<0] ~ x_[x_<0])
print(xtable(fit.positive),type='latex')
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	2.4633	0.1401	17.58	0.0000
x_[x_ > 0]	-5.6374	0.4223	-13.35	0.0000

```
print(xtable(fit.negative),type='latex')
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	2.7719	0.1609	17.22	0.0000
x_[x_ < 0]	6.7324	0.5182	12.99	0.0000

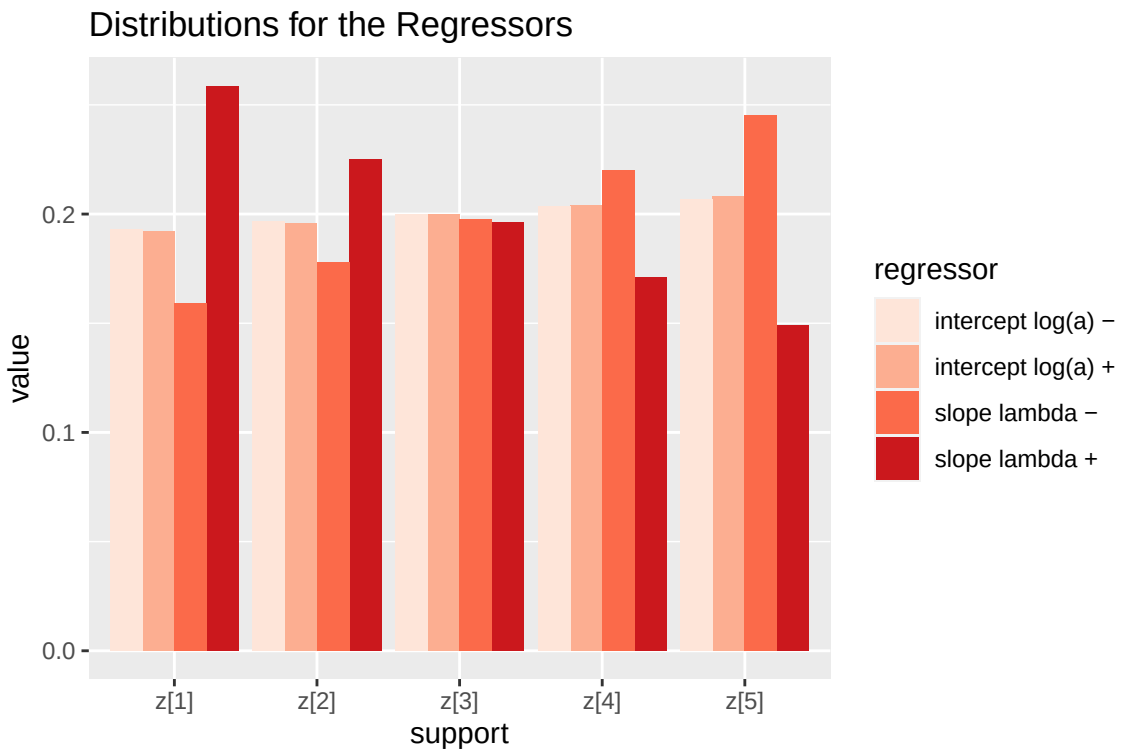
Now we apply the maximum entropy problem for linear regressions in both branches.

```
# POSITIVE BRANCH
library(alabama)
```

```
## Loading required package:  numDeriv
x0<-x_[x_>mean(x)]
y0<-log(y_[x_>mean(x)])
v<-c(-5*sd(y0),-2.5*sd(y0),0,2.5*sd(y0),5*sd(y0))
z<-c(-100,-50,0,50,100,-100,-50,0,50,100)
b0<-1:(0.5*length(z)) # support space interval for beta0, former example was 1:3
b1<-(0.5*length(z)+1):length(z) # support space interval for beta1, former example was 4:6

entropy<-function(x)
{
  p<-x[x!=0]
  sum(p*log(p))
}
entropy_gr<-function(x) {ifelse(x==0,0,log(x)+1)} # a vector of partial derivatives wrt the p's

heq <- function(x)
{
  h <- rep(NA, 1)
  h[1] <- sum(x[b0]) - 1
  h[2] <- sum(x[b1]) - 1
  for(i in 1:length(x0)) # 2 equality constraints for each obs in the same for loop
  {
    cnt<-(length(v)*(i-1)+1+length(z)):(length(v)*i+length(z))
    h[i+2] <- sum(x[cnt]) - 1
    h[i+length(x0)+2]<-sum(x[b0]*z[b0])+sum(x[b1]*z[b1]*x0[i])+sum(x[cnt]*v) - y0[i]
  }
  h
}
heq.jac <- function(x)
{
  j <- matrix(0, 2*length(x0)+2, length(x)) # "default" value for j is 0, not NA
  j[1,b0] <- 1
  j[2,b1] <- 1
  for(i in 1:length(x0))  # 2 equality constraints for each obs in the same for loop
  {
    cnt<-(length(v)*(i-1)+1+length(z)):(length(v)*i+length(z))
    j[i+2,(length(v)*(i-1)+1+length(z)):(length(v)*i+length(z))]<- 1
    j[i+length(x0)+2,b0] <- z[b0]
    j[i+length(x0)+2,b1] <- x0[i]*z[b1]
    j[i+length(x0)+2,(length(v)*(i-1)+1+length(z)):(length(v)*i+length(z))]<-v
  }
  j
}

hin <- function(x) { # vector
  h <- x # vector of probabilities must be nonnegative in all entries
  h
}
hin.jac <- function(x) { # matrix
diag(length(x)) # partial derivatives of hin wrt the p's
}
```

```
init<-rep(1/length(v),length(z)+length(v)*length(x0))
res.positive<-auglag(init,entropy,entropy_gr,heq=heq,heq.jac=heq.jac)

# NEGATIVE BRANCH
x0<-x_[x_<mean(x)]
y0<-log(y_[x_<mean(x)])
res.negative<-auglag(init,entropy,entropy_gr,heq=heq,heq.jac=heq.jac)
```

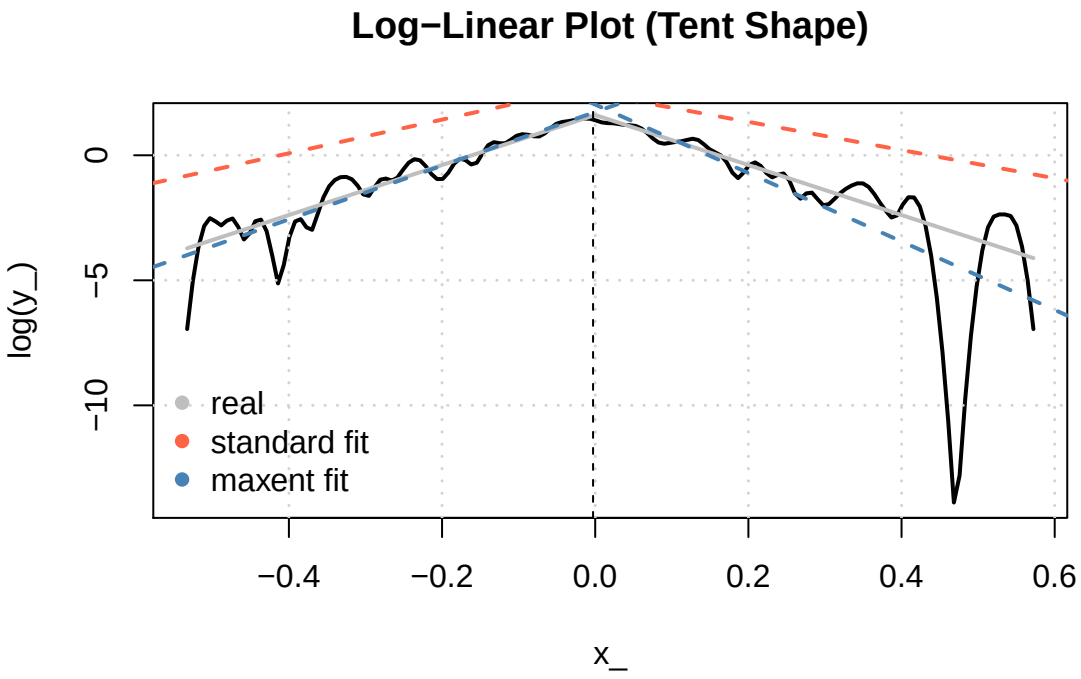
```
beta0.pos<-sum(res.positive$par[b0]*z[b0])
beta1.pos<-sum(res.positive$par[b1]*z[b1])
beta0.neg<-sum(res.negative$par[b0]*z[b0])
beta1.neg<-sum(res.negative$par[b1]*z[b1])

library(ggplot2)
value<-c(res.positive$par[c(b0,b1)],res.negative$par[c(b0,b1)])
regressor<-c(rep('intercept log(a) +',0.5*length(z)),rep('slope lambda +',0.5*length(z)),rep('intercept
support<-rep(paste0('z[',1:(0.5*length(z)),']'),4)

ggplot(data.frame(support,regressor,value), aes(fill=regressor,x=support,y=value)) + geom_bar(position=
```



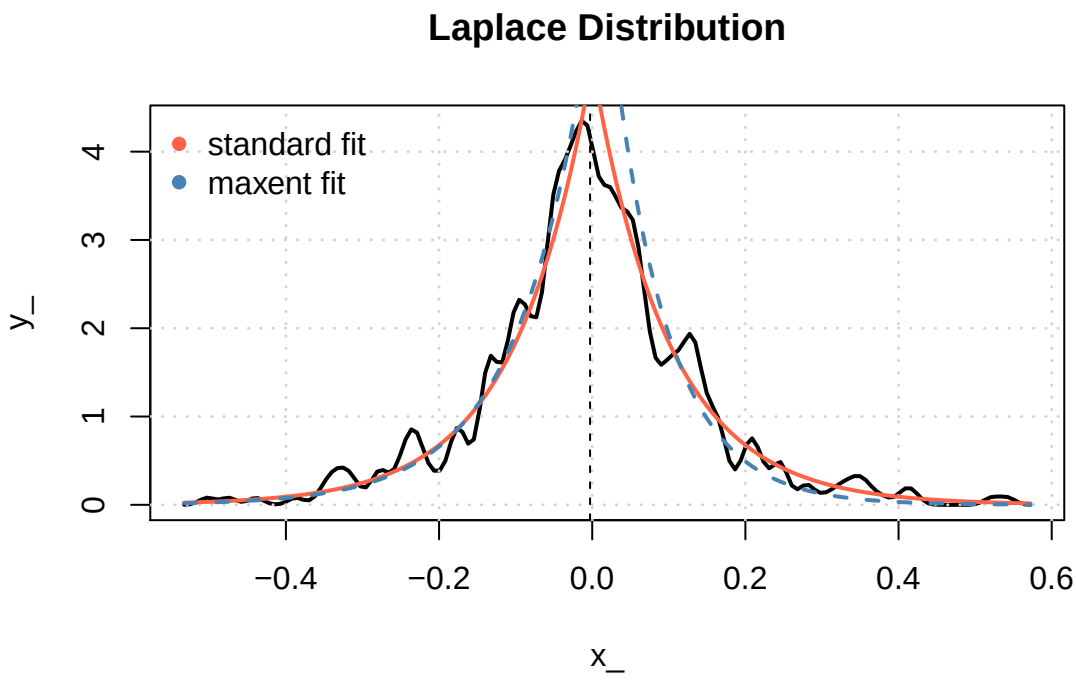

```
res.laplace<-data.frame(parameters=c('a(+)','a(-)','lambda(+)','lambda(-)'),maxent=c(exp(beta0.pos),exp
kable(res.laplace,booktabs=T) %>% row_spec(0,bold=T)
```

parameters	maxent	standard
a(+)	7.600627	11.743040
a(-)	5.642577	15.989444
lambda(+)	-13.687969	-5.637364
lambda(-)	10.735519	6.732429

```
# Visualization
plot(x_,log(y_),type='l',lwd=2,main='Log-Linear Plot (Tent Shape)')
grid(lwd=1.5)
y.li<-log(5)-abs(10*x_)
lines(x_,y.li,col='gray',lwd=2)
abline(fit.positive,col='tomato',lwd=2,lty=2)
abline(fit.negative,col='tomato',lwd=2,lty=2)
abline(beta0.pos,beta1.pos,col='steelblue',lwd=2,lty=2)
abline(beta0.neg,beta1.neg,col='steelblue',lwd=2,lty=2)
abline(v=mean(x),lty=2)
legend('bottomleft',c('real','standard fit','maxent fit'),col=c('gray','tomato','steelblue'),pch=16,bty=
```




```
plot(x_,y_,type='l',lwd=2,main='Laplace Distribution')
grid(lwd=1.5)
abline(v=mean(x),lty=2)
lines(x_,5*exp(-abs(10*x_)),col='tomato',lwd=2) # in fact scale=5 is a better fit...
lines(x_[x_>mean(x)],exp(beta0.pos)*exp(beta1.pos*x_[x_>mean(x)]),col='steelblue',lwd=2,lty=2)
lines(x_[x_<mean(x)],exp(beta0.neg)*exp(beta1.neg*x_[x_<mean(x)]),col='steelblue',lwd=2,lty=2)
legend('topleft',c('standard fit','maxent fit'),col=c('tomato','steelblue'),pch=16,bty='n')
```



References

Bhaduri, A. and Harris, D. J. (1987). The complex dynamics of the simple ricardian system. *The Quarterly Journal of Economics*, 102(4):893–902.

Brody, A. (2000). The monetary multiplier. *Economic Systems Research*, 12(2):215–219.

Leontief, W. and Brody, A. (1993). Money-flow computations. *Economic Systems Research*, 5(3):225–233.

ADV MATH METHODS FOR MODELING

Lab Notes on Info-Metrics

Oriol Vallès Codina

April 22, 2020

1 Construction of a Joint Distribution: Race, Gender, and Income

```
library(kableExtra)
library(xtable)
dat<-as.data.frame(readRDS('IncomeDistr.RDS'))
kable(head(dat))
```

ID	HH_gender	Age	Education	Married	Kids	Labor_force	Race	INCO
31	Male	30	some college or Assoc. degree	Yes	2	Working	white non-Hispanic	12151
34	Male	30	some college or Assoc. degree	Yes	2	Working	white non-Hispanic	11442
51	Male	41	Bachelors degree or higher	Yes	1	Working	white non-Hispanic	11442
52	Male	41	Bachelors degree or higher	Yes	1	Working	white non-Hispanic	11442
53	Male	41	Bachelors degree or higher	Yes	1	Working	white non-Hispanic	11442
54	Male	41	Bachelors degree or higher	Yes	1	Working	white non-Hispanic	11442

```
colnames(dat)
```

```
##  [1] "ID"             "HH_gender"      "Age"            "Education"
##  [5] "Married"        "Kids"           "Labor_force"    "Race"
##  [9] "INCOME"         "WAGE"           "SAVING"         "Liquidity"
## [13] "Expense"        "Household_size" "Age_cat"        "Age_cat_num"
## [17] "Income_cat"     "income_prob"
```

```
# Marginal Probabilities for Discrete Variables
table(dat$HH_gender)/nrow(dat)
```

```
##
##    Female      Male
## 0.2691808 0.7308192
```

```
table(dat$Education)/nrow(dat)
```

```
##
##    no high school diploma/GED    high school diploma or GED
##                     0.1064153                     0.2460988
## some college or Assoc. degree    Bachelors degree or higher
##                     0.2831599                     0.3643260
```

```
table(dat$Race)/nrow(dat)
```

```
##
##     white non-Hispanic black/African American               Hispanic
##              0.69018205             0.15832250             0.10197226
##                   other
##              0.04952319
```

```
# Continuous Variables: Income
bins.income<-density(dat[,'INCOME'],n=21)$x
p.income<-density(dat[,'INCOME'],n=21)$y/sum(density(dat[,'INCOME'],n=21)$y)
-sum(p.income*log(p.income)) # marginal entropy of income
```

```
## [1] 2.280992
```

```
# Joint Probability for Gender and Income
joint<-data.frame()
for(i in 1:length(bins.income)) # you can also write go between bin 2 and bin 19
{
  n.f<-nrow(dat[dat$INCOME>bins.income[i] & dat$INCOME<bins.income[i+1] & dat$HH_gender=='Female',])
  n.m<-nrow(dat[dat$INCOME>bins.income[i] & dat$INCOME<bins.income[i+1] & dat$HH_gender=='Male',])
  joint[i,'Female']<-n.f
  joint[i,'Male']<-n.m
}
joint<-joint/nrow(dat)

kable(joint)
```

Female	Male
0.0000000	0.0000000
0.0539662	0.0493065
0.1115085	0.1547464
0.0532076	0.1303641
0.0174469	0.0846337
0.0063936	0.0565670
0.0109450	0.0769397
0.0047681	0.0403121
0.0018422	0.0245990
0.0032510	0.0407456
0.0006502	0.0126788
0.0011920	0.0163632
0.0005418	0.0079107
0.0005418	0.0092111
0.0002167	0.0053099
0.0001084	0.0040095
0.0014088	0.0088860
0.0009753	0.0065020
0.0002167	0.0017339
0.0000000	0.0000000
0.0000000	0.0000000

```
sum(joint) # should be 1
```

```
## [1] 1
```

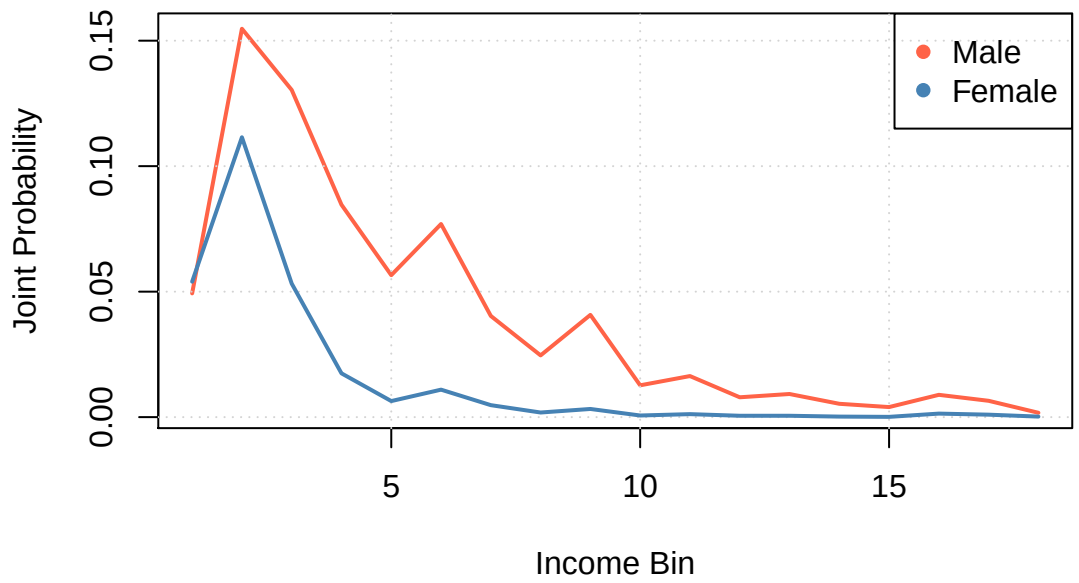
```r
colSums(joint) # returns marginal prob for gender
```

```
##    Female      Male
## 0.2691808 0.7308192
```

```r
rowSums(joint) # returns marginal prob for income
```

```
##           1           2           3           4           5           6
## 0.000000000 0.103272648 0.266254876 0.183571738 0.102080624 0.062960555
##           7           8           9          10          11          12
## 0.087884699 0.045080191 0.026441266 0.043996532 0.013328999 0.017555267
##          13          14          15          16          17          18
## 0.008452536 0.009752926 0.005526658 0.004117902 0.010294755 0.007477243
##          19          20          21
## 0.001950585 0.000000000 0.000000000
```

```r
p<-joint[-c(1,20,21),]
plot(p$Male,col='tomato',lwd=2,type='l',ylab='Joint Probability',xlab='Income Bin')
grid()
lines(p$Female,col='steelblue',lwd=2)
legend('topright',c('Male','Female'),pch=16,col=c('tomato','steelblue'))
```




```r
# 3-Dimensional Joint Probability for Gender, Race and Income
# dimnames=list(c(1:21),unique(dat£Race),c('Female','Male')))
joint.3<-array(NA,dim=c(length(bins.income),length(unique(dat$Race)),2),dimnames=list(c(1:21),unique(da
for(i in 1:length(bins.income))
{
```

```
for(j in unique(dat$Race))
{
n.f<-nrow(dat[dat$INCOME>bins.income[i] & dat$INCOME<bins.income[i+1] & dat$Race==j & dat$HH_gender=='F
n.m<-nrow(dat[dat$INCOME>bins.income[i] & dat$INCOME<bins.income[i+1] & dat$Race==j & dat$HH_gender=='M
    joint.3[i,j,'Female']<-n.f
    joint.3[i,j,'Male']<-n.m
  }
}
sum(joint.3) # total number of observations (ie nrow(dat))

## [1] 9228

joint.3<-joint.3/nrow(dat)
```

Pointwise mutual information (PMI) is a measure of association used in information theory and statistics. In contrast to mutual information (MI) which builds upon PMI, it refers to single events, whereas MI refers to the average of all possible events.

The PMI of a pair of outcomes x and y belonging to discrete random variables X and Y quantifies the discrepancy between the probability of their coincidence given their joint distribution and their individual distributions, *assuming independence*. Mathematically:

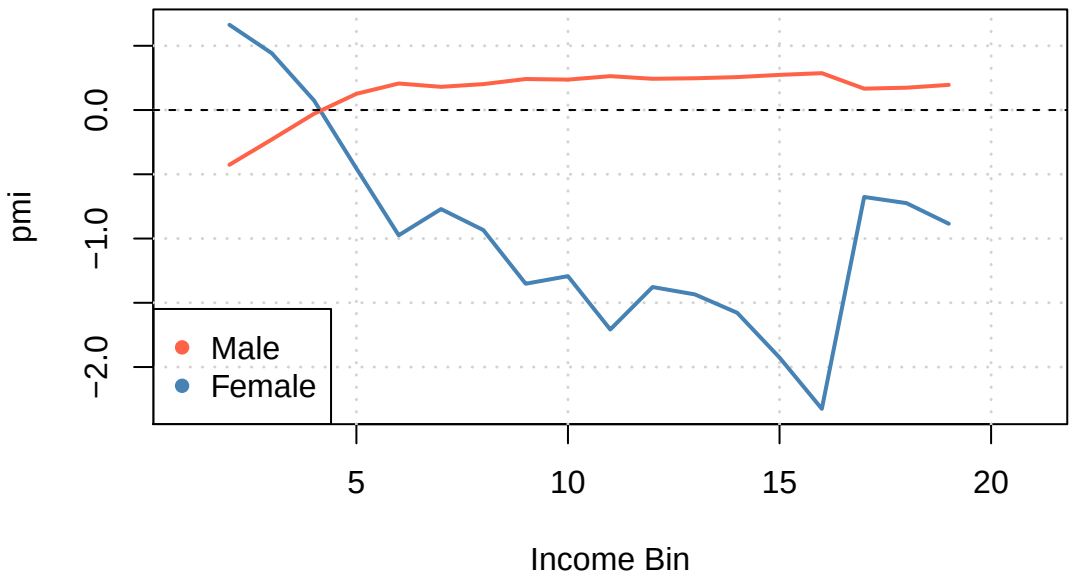
$$\text{pmi}(x,y) = \log \frac{p(x,y)}{p(x)p(y)} = \log p(x,y) - \log p(x) - \log p(y) \tag{1}$$

$$\text{pmi}(x,y) = \log \frac{p(x|y)}{p(x)} = \log \frac{p(y|x)}{p(y)} \tag{2}$$

The mutual information (MI) of the random variables X and Y is the expected value of the PMI (over all possible outcomes):

$$\text{MI}(x,y) = \sum_y \sum_x p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \tag{3}$$

```
# rowSums(joint) returns marginal probability for income
# sum(joint$Female) returns marginal probability for gender female
pmi.f = log(joint$Female/(rowSums(joint)*sum(joint$Female)))
pmi.m = log(joint$Male/(rowSums(joint)*sum(joint$Male)))
plot(NA,xlim=c(1,21),ylim=c(min(c(pmi.f,pmi.m),na.rm=T),max(c(pmi.f,pmi.m),na.rm=T)),xlab='Income Bin',y
grid(lwd=1.5)
lines(pmi.f,col='steelblue',lwd=2)
lines(pmi.m,col='tomato',lwd=2)
abline(h=0,lty=2)
legend('bottomleft',c('Male','Female'),pch=16,col=c('tomato','steelblue'))
```



2 Maximum Entropy for a Distribution with Quantiles as Constraints

This should be useful for when we want to find the (continuous) distribution of income $f(x)$ by extracting the information from the quantiles, as it is often reported in datasets of income inequality. The continuous support for the income variable is $[0, +\infty)$.

If we have the information for $i = 1, ..., N$ quantiles where x_i refers to the income threshold and q_i refers to the quantile percentage, we can write the corresponding N constraints of the maxent problem as:

$$\int_0^\infty [1 - \theta_{x_i}(x)] f(x) dx = q_i \quad i = 1, ..., N \tag{4}$$

where $\theta_{x_i}(x)$ is the Heaviside step function, which is defined in two parts:

$$\theta_{x_i}(x) = \begin{cases} 1 & \text{if } x > x_i \\ 0 & \text{if } x < x_i \end{cases} \tag{5}$$

Hence we can write the quantile constraint as:

$$\int_0^{x_i} f(x) dx = q_i \quad i = 1, ..., N \tag{6}$$

Note that the normalization constraint can also be conceived as the $i = 0$ constraint, where $q_i = 1$ and $x_i = +\infty$ integrating over the whole continuous support.

The Lagrangian will thus be:

$$\mathcal{L} = -\int_0^\infty f(x) \log f(x) dx - (\lambda_0 - 1) \left(\int_0^\infty f(x) dx - 1 \right) - \sum_{i=1}^N \lambda_i \left(\int_0^{x_i} f(x) dx - q_i \right)$$

Taking the FOC with respect to $p(x)$, we find the general form for $p(x)$:

$$\frac{\partial \mathcal{L}}{\partial p(x)} = -1 - \log p(x) - \lambda_0 + 1 - \lambda_1 |x - \mu| = 0$$

$$p(x) = \exp\left(-\lambda_0 - \lambda_1 |x - \mu|\right)$$

Now the two constraints look like:

$$\int_{-\infty}^{\infty} \exp\left(-\lambda_0 - \lambda_1 |x - \mu|\right) dx = 1$$

$$\int_{-\infty}^{\infty} |x - \mu| \exp\left(-\lambda_0 - \lambda_1 |x - \mu|\right) dx = b$$

As usual, $e^{-\lambda_0}$ will be the partition function that provides normalization.

ADV MATH METHODS FOR MODELING

Lab Notes on Info-Metrics

Oriol Vallès Codina

May 6, 2020

1 Income Distribution

The **grouping property** dictates that changing the grouping of the numbers used in an operation without changing the order does not change the result of that operation. This property applies to both addition and multiplication, but not to subtraction and division. The grouping property is also of utmost importance for the entropy function of Boltzmann, Gibbs, and Shannon. The grouping property is instrumental when analyzing systems where the di erent elementary outcomes can be sorted into mutually exclusive groups. Given such a system, and our understanding of that system, we start by sorting the individual outcomes into groups. In that case, identifying a particular outcome means that we first identify the exact group that outcome resides in.

Consider studying the income distribution of a certain population. Our observed information comes from tax returns [Golan, 2018, p.45]. So we know the tax bracket of each individual, the range of incomes (income group) that applies to that tax bracket, and each individual's relative location within that income group. There are potentially many individuals located at the same relative location. We call that "type." Our income distribution is expressed in terms of probability distribution over brackets and types. In terms of our notations, $P(X = x_k|Y = y_j)p_{k|j}$ is the probability of observing an individual with income type k in tax bracket j. The probability of observing an individual in tax bracket j in the type k is w_{kj}. The average over all tax brackets is again $H(X,Y) = H(Y) + H(X|Y)$, where Y is the distribution of the tax bracket groups and X is the distribution of individual types (income levels).

Similarly, we can think of $p_{k|j}$ as the probability of type k's welfare conditional on her family status, her income group, her location (rural, urban, suburban), or in general any subgroup of interest. These are very common problems in the social sciences. A related example from economics is the study of the size distribution of firms, which we already addressed.

The grouping property is also closely related to hierarchical discrete choice (or nested choice) models: think of a decision tree where some choices are made in a sequence but some are not.

1.1 Example: Two-Dice Sum

As a simple example [Golan, 2018, p.200], consider tossing two dice, one black and one white. The two dice are independent of each other. We are interested in the probabilities (and entropy) of the 6×6 two-dice pairs $(1,1), ..., (6,6)$, which is the complete sample space. But our prior information comes from the following groups. Let k enumerate the sum of the two faces, $k = 2, 3, .., 12$; thus there are $K = 11$ events. Even when the dice are perfectly tossed and are bias-free we expect that intermediate values of the sum, say $k = 6, 7, or 8$, are more probable than the extreme values $k = 2$ or 12. The reasoning is that since there are 36 possible outcomes (black-white pairs) that map to 11 sums (events), some events correspond to more than one outcome of the experiment. For example, $k = 3$ can result under two scenarios: when the (black, white) faces of the two dice are (1,2) or when they are (2,1). But the event $k = 2$ can occur only for the single outcome (1,1) while $k = 4$ corresponds to three possible outcomes (1,3; 2,2; 3,1). Our expectation is that the different results of tossing two independent dice (that is, the 36 possible paired outcomes) are

equally probable. That is, their joint probability is the product of their (individual) marginal probabilities. But if so, the probabilities p_k that the sum of the two faces is k cannot be equal for all values of k. Rather, even when no constraining information is available, we expect that $p_k = \frac{n_k}{\sum_k n_k}$ where n_k is the number of events in which the event k can be realized.

```
entropy<-function(x)
{
  p<-x[x!=0]
  sum(p*log(p))
}
entropy_gr<-function(x) {ifelse(x==0,0,log(x)+1)} # a vector of partial derivatives wrt the p's
```

As in our analytical derivation, we define the values of the 3 events, $x_j = j$ for $j = 1, 2, 3$ in the vector f:

```
f<-c(1,2,3)
```

In order to introduce the constraints for *auglag*, we need a vector for the equality constraints *heq* (so that $heq_j = 0$ for all j), a vector for the inequality constraints *hin* (so that $hin_j > 0$ for all j), and their corresponding Jacobian matrices (partial derivatives) of size $N \times N'$, where number of rows N is the number of constraints and number of columns N' is the support space (number of events) of probability distribution p_j:

```
heq <- function(x) { # vector
  h <- rep(NA, 1)
  h[1] <- sum(x) - 1 # zeroth moment constraint: normalization
  # h[2] <- sum(f*x) - 2.5 # first moment constraint: mean
  h
}
heq.jac <- function(x) { # matrix
  j <- matrix(NA, 1, length(x))
  j[1, ] <- c(1, 1, 1) # partial derivatives of the normalization constraint wrt the p's
  # j[2, ] <- c(1,2,3) # values of f, ie partial derivatives of the mean wrt the p's
  j
}
hin <- function(x) { # vector
  h <- x # vector of probabilities must be nonnegative in all entries
  h
}
hin.jac <- function(x) { # matrix
diag(length(x)) # partial derivatives of hin wrt the p's
}
```

Finally we can call the function *auglag*, where the first parameter is the initial vector from which to start the numerical computation:

```
results<-auglag(runif(3,0,.3),entropy,entropy_gr,heq=heq,heq.jac=heq.jac,hin=hin,hin.jac=hin.jac)

## Error in auglag(runif(3, 0, 0.3), entropy, entropy_gr, heq = heq, heq.jac = heq.jac, :  could
not find function "auglag"
```

References

Golan, A. (2018). *Foundations of info-metrics: Modeling, inference, and imperfect information.* Oxford University Press.

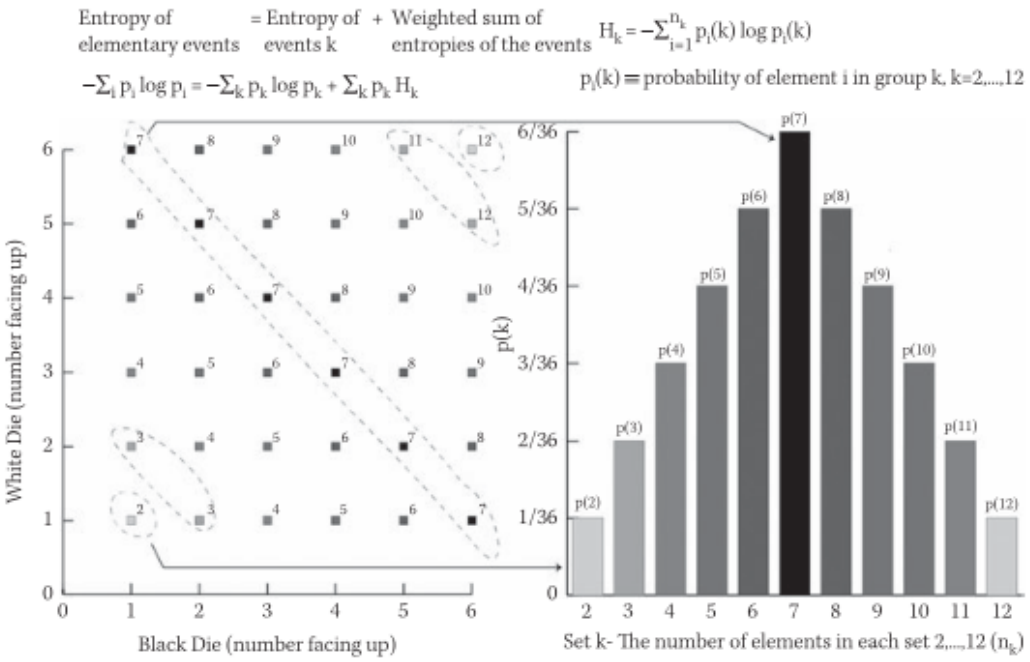


FIGURE 8.3. The two-dice example, featuring a graphical representation of the relationship between elementary outcomes (represented by dots) and events. Each event contains at least one elementary outcome. Elementary outcomes with the same number on their top-right are members of the same set. For example, the two outcomes (1, 2) and (2, 1) are members of the event 'k = 3' presented as a number on the top right of each outcome on the left panel. The left panel captures the elementary outcomes—events relationship (the complete 6×6 sample space). Each outcome is a two dimensional point capturing the number on the top face of each die. Each event is the sum of the numbers on the top faces of the two dice. The right panel shows the probability distribution of the events (the partial sample space of the 11 groups). The total number of outcomes in each set is n_k, known as the degeneracy of event k. From (8.8) and (8.10) we can calculate the different entropies in this case. The first term of the left hand side of (8.8), the entropy of the elementary elements, $-\sum_i p_i \log p_i$, reaches a maximum at 3.584, where all the p_k are 1/36. These probabilities imply that the event probabilities for the p_k's are 0.028, 0.056, 0.083, 0.111, 0.139, 0.167, 0.139, 0.111, 0.083, 0.056, and 0.028, respectively for k =2, ... ,12 (see right panel). So the first term on the right hand side, which is the entropy of the events k, $\sum_k p_k \log p_k$ is 2.270. The second term on the right hand side of (8.8), $\sum_k p_k H_k$, is 1.314 $(3.584 = 2.27 + 1.314)$. Stated differently, for uniform probabilities within each group k, the entropy of the group is $\log(n_k)$ for k =2, ... ,12, and that of the elementary elements is $\log(36)$. Putting it all together within (8.8) we have $\log(36) = -\sum_k p_k \log p_k + \sum_k \log(n_k)$.