Fundamentals of Statistical Modeling (VT21)

Andrea Discacciati
Karolinska Institutet
Stockholm, Sweden

Lab 1

```
. version 14
```

Exercise 0 (local macros)

In this course, we'll make extensive use of Stata's `local macros`. If they're new to you, get familiar with the help of the code below. Copy and paste it in a new do-file and run it one chunk at a time.

The Stata command `display` (abbreviated with `di`) prints strings in the Results window and can also be used as a hand calculator (`help display`).

```
* Chunk 1
local a = "Fundamentals of statistical modeling."
di "`a'"

* Chunk 2
local a = "statistical modeling"
di "Fundamentals of `a'."

* Chunk 3
local a = "Fundamentals of"
local b = "statistical"
local c = "modeling."
di "`a' `b' `c'"

* Chunk 4
local a = "Fundamentals"
local b = "`a' of"
local c = "`b' statistical"
local d = "`c' modeling."
di "`d'"

* Chunk 5
di exp(2) * exp(-2)

* Chunk 6
local a = "exp(2)"
local b = "exp(-2)"
di `a' * `b'

* Chunk 7
local a = "2"
local b = "exp(`a')"
local c = "exp(-`a')"
```

```
di `b' * `c'

* Chunk 8
sysuse auto, clear
describe

local yvar = "weight"
local xvars = "length mpg"
local xvars2 = "price"
regress `yvar' `xvars' `xvars2'
```

Exercise 1

Some of the distributions we'll be working with in this course are:

- Normal: `https://en.wikipedia.org/wiki/Normal_distribution`
- Skew-normal: `https://en.wikipedia.org/wiki/Skew_normal_distribution`
- Gamma: `https://en.wikipedia.org/wiki/Gamma_distribution`
- Beta: `https://en.wikipedia.org/wiki/Beta_distribution`
- Exponential: `https://en.wikipedia.org/wiki/Exponential_distribution`
- Chi-squared: `https://en.wikipedia.org/wiki/Chi-squared_distribution`
- Weibull: `https://en.wikipedia.org/wiki/Weibull_distribution`
- Bernoulli: `https://en.wikipedia.org/wiki/Bernoulli_distribution`
- Negative binomial: `https://en.wikipedia.org/wiki/Negative_binomial_distribution`

Get familiar with the distributions above.

- Are they useful to model continuous or discrete variables?
- What's their support?
- By how many parameters are they parametrized?
- What does their shape look like?
- Which Stata functions implement their probability mass / density functions? See `help density_functions`.

Exercise 2

Load the data

```
. version 14
. use https://raw.githubusercontent.com/anddis/fsm/master/data/lab1.dta, clear
```

We assume that $f(y_n)$ is normal. Estimate the parameters μ and σ. Use the `normalden()` function.

```
. local f = "normalden(y_n, {mu}, {sigma})"
. mlexp (ln(`f´))
initial:       log likelihood =     -<inf>  (could not be evaluated)
feasible:      log likelihood = -43652.817
rescale:       log likelihood = -1800.6375
rescale eq:    log likelihood = -1328.2447
Iteration 0:   log likelihood = -1328.2447  (not concave)
Iteration 1:   log likelihood = -1120.8299
Iteration 2:   log likelihood = -1064.4927
Iteration 3:   log likelihood = -1060.4292
Iteration 4:   log likelihood = -1059.3324
Iteration 5:   log likelihood = -1059.3319
Iteration 6:   log likelihood = -1059.3319
Maximum likelihood estimation
Log likelihood = -1059.3319                     Number of obs     =        300

------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         /mu |   178.4911    .4772459   374.00   0.000     177.5557    179.4265
      /sigma |   8.266142    .3374638    24.49   0.000     7.604725    8.927559
------------------------------------------------------------------------------
```

Estimate again the parameters μ and σ, but this time constrain the parameter σ to be positive by replacing $\sigma = \exp(\theta)$. You'll get now an estimate of θ. Recover the MLE of σ. (Note that the MLE $\hat{\sigma}$ hasn't changed – see slide 33 – but this "trick" improves computational stability).

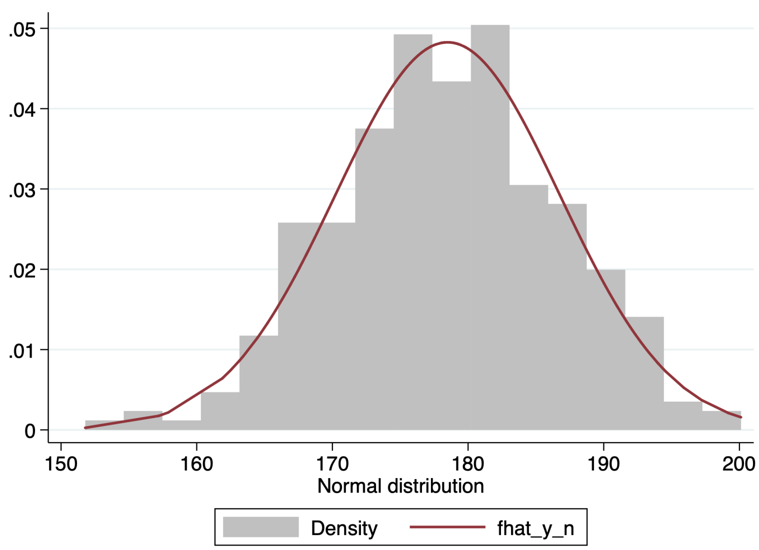
```
. local sigma = "exp({theta})"
. local f = "normalden(y_n, {mu}, `sigma´)"
. mlexp(ln(`f´))
initial:       log likelihood =     -<inf>  (could not be evaluated)
feasible:      log likelihood = -32398.765
rescale:       log likelihood = -1981.1218
rescale eq:    log likelihood = -1440.3171
Iteration 0:   log likelihood = -1440.3171  (not concave)
Iteration 1:   log likelihood = -1112.6119
Iteration 2:   log likelihood = -1085.7986
Iteration 3:   log likelihood = -1059.4172
Iteration 4:   log likelihood =  -1059.332
Iteration 5:   log likelihood = -1059.3319
Maximum likelihood estimation
Log likelihood = -1059.3319                     Number of obs     =        300
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
/mu	178.4911	.4772459	374.00	0.000	177.5557	179.4265
/theta	2.112168	.0408248	51.74	0.000	2.032153	2.192183

```
.
. di exp(_b[/theta])
8.2661406
```

Plot the estimated density $\hat{f}(y_n)$ over the sample histogram.

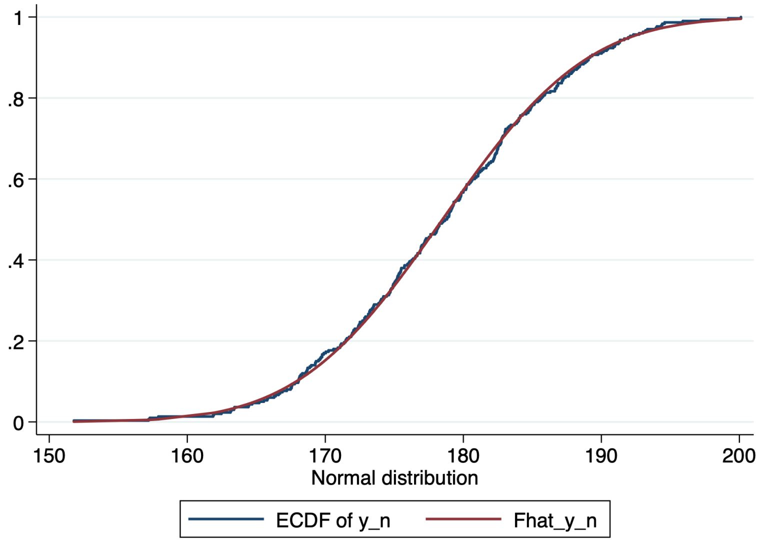
```
. gen fhat_y_n = normalden(y_n, _b[/mu], exp(_b[/theta]))
. tw (hist y_n) (line fhat_y_n y_n, sort), name(y_n, replace)
. graph export y_n.png, replace
(file y_n.png written in PNG format)
```



Plot the model-based estimated CDF $\hat{F}(y_n)$ (see slide 38) over the empirical CDF (see slide 20). The function `normal()` returns the CDF of a standard normal distribution.

```
. gen Fhat_y_n = normal((y_n - _b[/mu])/exp(_b[/theta]))
. cumul y_n, gen(sampleF_y_n)
.
. // This is what -cumul- does under the hood
. // sort y_n
. // gen sampleF_y_n = sum(_cons)
```

```
. // su sampleF_y_n
. // replace sampleF_y_n = sampleF_y_n / r(max)
.
. tw (line sampleF_y_n Fhat_y_n y_n, connect(J l) sort), name(Fy_n, replace)
. graph export Fy_n.png, replace
(file Fy_n.png written in PNG format)
```



Exercise 3

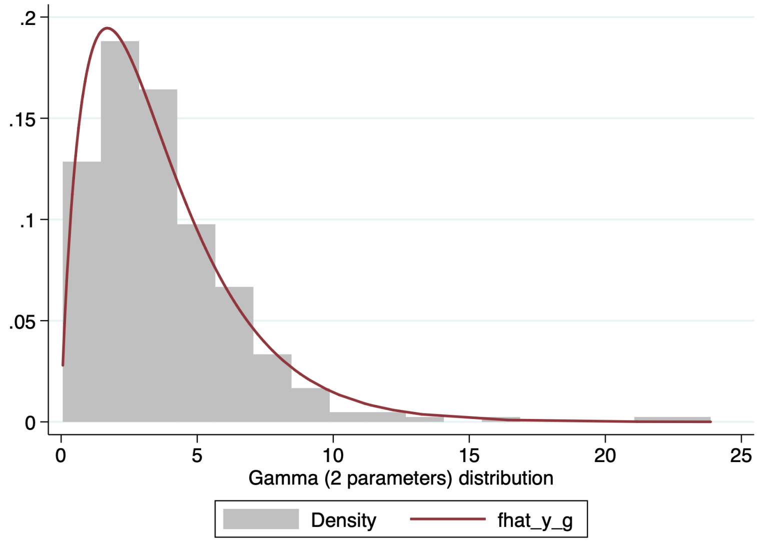
Assume that $f(y_g)$ is (2-parameter) gamma. Estimate the parameters α and β using the `gammaden()` function. Fix the location parameter g (the third argument of the `gammaden()` function), equal to 0.

```
. local f = "gammaden({alpha}, {beta}, 0, y_g)"
. mlexp(ln(`f´))
initial:       log likelihood =     -<inf>  (could not be evaluated)
feasible:      log likelihood = -2465.2095
rescale:       log likelihood = -670.37304
rescale eq:    log likelihood = -670.37304
Iteration 0:   log likelihood = -670.37304
Iteration 1:   log likelihood = -668.66319
Iteration 2:   log likelihood = -668.35073
Iteration 3:   log likelihood = -668.34648
Iteration 4:   log likelihood = -668.34648
Maximum likelihood estimation
Log likelihood = -668.34648                     Number of obs     =        300
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
/alpha	1.830878	.138033	13.26	0.000	1.560338	2.101418
/beta	2.042909	.1769745	11.54	0.000	1.696046	2.389773

Plot the estimated density $\hat{f}(y_g)$ over the sample histogram.

```
. gen fhat_y_g = gammaden(_b[/alpha], _b[/beta], 0, y_g)
. tw (hist y_g) (line fhat_y_g y_g, sort), name(y_g, replace)
. graph export y_g.png, replace
(file y_g.png written in PNG format)
```



Exercise 4

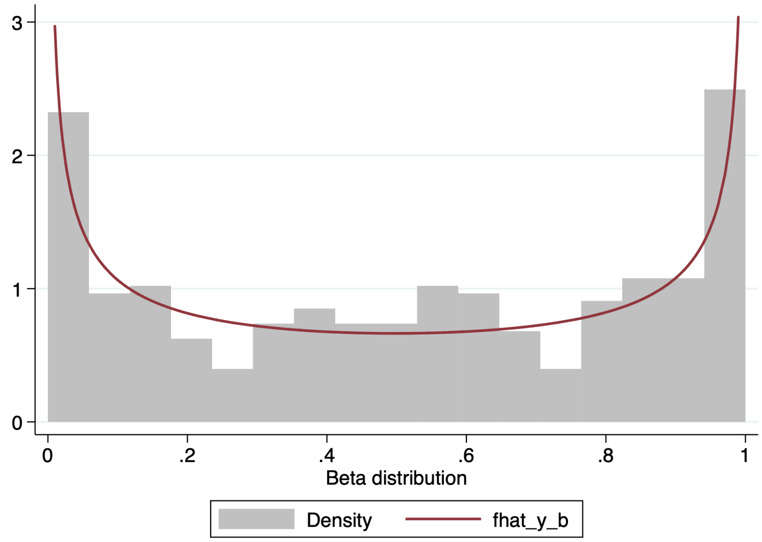
We assume that $f(y_b)$ is beta. Estimate the parameters α and β using the `betaden()` function.

```
. local f = "betaden({alpha}, {beta}, y_b)"
. mlexp(ln(`f´))
initial:       log likelihood =     -<inf>  (could not be evaluated)
feasible:      log likelihood =  57.883521
rescale:       log likelihood =  57.883521
rescale eq:    log likelihood =  57.883521
Iteration 0:   log likelihood =  57.883521
Iteration 1:   log likelihood =  58.345242
Iteration 2:   log likelihood =  58.346286
Iteration 3:   log likelihood =  58.346286
Maximum likelihood estimation
Log likelihood =  58.346286                     Number of obs     =        300
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf. Interval]	
/alpha	.5346582	.0394503	13.55	0.000	.457337	.6119794
/beta	.5286525	.0388869	13.59	0.000	.4524355	.6048694

Plot the estimated density $\hat{f}(y_b)$ over the sample histogram. (OBS: We need to be careful when plotting the estimated density function close to the boundaries of the distribution's support).

```
. gen fhat_y_b = betaden(_b[/alpha], _b[/beta], y_b)
. tw (hist y_b) (line fhat_y_b y_b if inrange(y_b, .01, .99), sort), name(y_b, replace)
. graph export y_b.png, replace
(file y_b.png written in PNG format)
```



Exercise 5

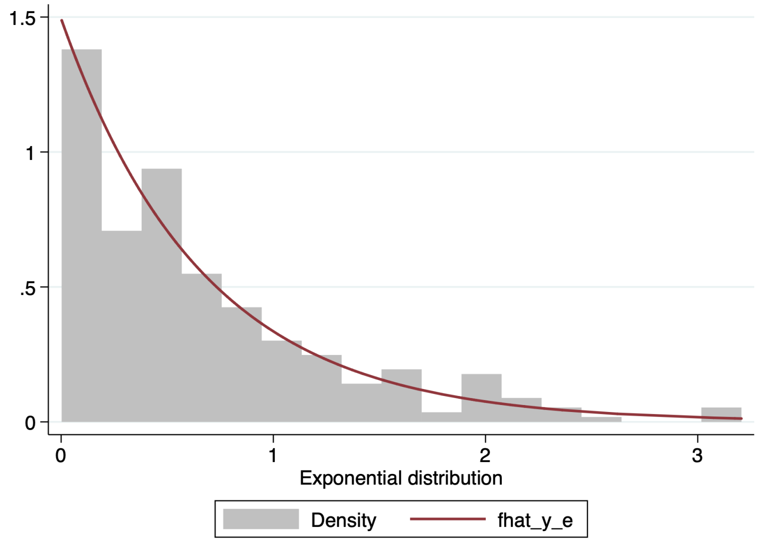
We assume that $f(y_e)$ is exponential. Estimate the parameter λ. You'll need to code the density of an exponential distribution yourself: $f(y;\lambda) = \lambda \exp(\lambda y)$

```
. local f = "{lambda}*exp(-y_e * {lambda})"
. mlexp(ln(`f´))
initial:       log likelihood =     -<inf>  (could not be evaluated)
feasible:      log likelihood =  -308.3512
rescale:       log likelihood = -193.68405
Iteration 0:   log likelihood = -193.68405
Iteration 1:   log likelihood = -181.67541
Iteration 2:   log likelihood = -179.57917
Iteration 3:   log likelihood = -179.57914
Iteration 4:   log likelihood = -179.57914
Maximum likelihood estimation
Log likelihood = -179.57914                     Number of obs     =        300
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
/lambda	1.493919	.0862515	17.32	0.000	1.324869	1.662969

Plot the estimated density $\hat{f}(y_e)$ over the sample histogram.

```
. gen fhat_y_e = _b[/lambda] * exp(-y_e * _b[/lambda])
. tw (hist y_e) (line fhat_y_e y_e, sort), name(y_e, replace)
. graph export y_e.png, replace
(file y_e.png written in PNG format)
```



Exercise 6

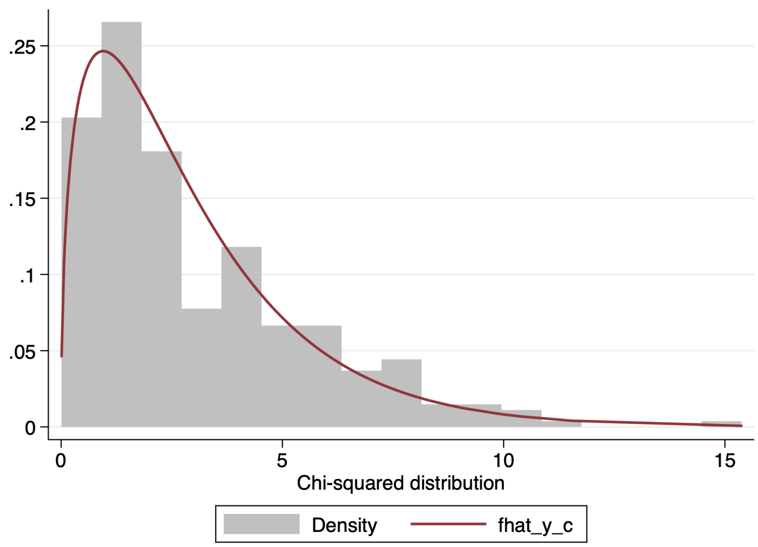
We assume that $f(y_c)$ is chi-squared. Estimate the parameter k using the `chi2den()` function.

```
. local f = "chi2den({k}, y_c)"
. mlexp(ln(`f´))
initial:       log likelihood =     -<inf>  (could not be evaluated)
feasible:      log likelihood = -1053.9877
rescale:       log likelihood = -660.78037
Iteration 0:   log likelihood = -660.78037
Iteration 1:   log likelihood = -628.29724
Iteration 2:   log likelihood = -626.40531
Iteration 3:   log likelihood = -626.40048
Iteration 4:   log likelihood = -626.40048
Maximum likelihood estimation
Log likelihood = -626.40048                     Number of obs     =        300
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf. Interval]	
/k	2.950201	.1181046	24.98	0.000	2.718721	3.181682

Plot the estimated density $\hat{f}(y_c)$ over the sample histogram.

```
. gen fhat_y_c = chi2den(_b[/k], y_c)
. tw (hist y_c) (line fhat_y_c y_c, sort), name(y_c, replace)
. graph export y_c.png, replace
(file y_c.png written in PNG format)
```


.25
.2
.15
.1
.05
0
0
5
10
15
Chi-squared distribution
Density
fhat_y_c