#+TITLE: # Sensitivity analysis using lava simulation

#+SUBTITLE: French R Meeting

#+AUTHOR: Thomas Alexander Gerds
Section of Biostatistics, Department of Public Health, University of Copenhagen

#+DATE: Anglet, 29 June 2017

Outline

LAVA

The `lava` package by Klaus K. Holst was developed to analyse (linear) latent variable models.[1]

In this tutorial we will **not** discuss the gorgeous functionality of `lava` for estimating parameters of a structural equation system.

We will study the `lava` language for a different purpose:

```
 ___(_)_ __ ___  _   _| | __ _| |_(_) ___  _ __
/ __| | '_ ` _ \| | | | |/ _` | __| |/ _ \| '_ \
\__ \ | | | | | | |_| | | (_| | |_| | (_) | | | |
|___/_|_| |_| |_|\__,_|_|\__,_|\__|_|\___/|_| |_|
```

[1]Klaus K. Holst and Esben Budtz-Jørgensen (2013). Linear Latent Variable Models: The lava-package Computational Statistics 28 (4), pp. 1385-1452.

Applications

The basic idea is to simulate data that are *alike* some observed real data (that we have) such that regression results obtained in the simulated data resemble the real data results.

Besides the intrinsic beauty and elegance of the `lava` language and its functionality, the following methods can be useful for

- simulation of a complex (biological) system
- sample size and power calculation
- sensitivity analysis
- analysis of small sample properties of a new statistical method in a realistic setting

Install the most recent version of the lava package

```
# devtools::install_github('kkholst/lava')
library(lava)
packageVersion("lava")
# need one of the following to display models
packageVersion("visNetwork")
packageVersion("igraph")
packageVersion("Rgraphviz")
# set some lava options
lava.options(layout="fdp")
lava.options(plot.engine="Rgraphviz")
```

```
[1] ‘1.5.1’
[1] ‘2.0.0’
[1] ‘1.0.1’
[1] ‘2.14.0’
```

Remark: `Rgraphviz` requires that the program `graphviz` is installed on your computer

Lava language

Create an empty lava object

```r
m <- lvm()
print(m)
```

```
Latent Variable Model

Empty
NULL
```

A normal distributed variable

The following statement has two effects:

1. a variable named age is added to the model
2. the distribution of the variable is set to be normal with mean 50 and standard deviation 10

```
distribution(m,~age) <- normal.lvm(mean=50,sd=10)
print(m)
```

```
Latent Variable Model

Exogenous variables:
  age        gaussian(identity)
```

A log-normal distributed variable

We add a log-normal variable named `bili` with mean 0.58 and standard deviation 1.03

```
distribution(m,~bili) <- lognormal.lvm(mean=0.58,sd
    =1.03)
print(m)
```

```
Latent Variable Model

Exogenous variables:
  age          gaussian(identity)
  bili         log-normal
```

A binary variable

We add a binomial variable named `sex` with success probability 0.12:

```
distribution(m,~sex) <- binomial.lvm(p=0.12)
print(m)
```

```
Latent Variable Model

Exogenous variables:
  age         gaussian(identity)
  bili        log-normal
  sex         binomial(logit)
```

Simulate data from object

At any time during the building of the object we can check what happens when we simulate from the object.

```
set.seed(13)
print(sim(m,10),digits=2)
```

```
   age bili sex
1   48 2.74   0
2   64 6.34   0
3   51 2.28   0
4   49 1.23   0
5   57 5.58   1
6   53 0.58   0
7   68 2.87   0
8   54 0.44   0
9   40 0.26   0
10  56 1.14   0
```

A categorical variable

We add a categorical variable named `stage`:

Category	Probability
1/2	27%
3	38%
4	35%.

```
m <- categorical(m,~stage, K=3,p=c(0.38,0.35),
                 labels=c("1/2","3","4"))
print(m)
```

```
Latent Variable Model

Exogenous variables:
  age             gaussian(identity)
  bili            log-normal
  sex             binomial(logit)
  stage           categorical
```

Time to event variable

We add a time to event variable named `t.death` with a Cox-Weibull distribution[2]

```
distribution(m,~t.death) <- coxWeibull.lvm(
    scale=0.00000033,
    shape=1.45)
print(m)
```

```
Latent Variable Model

Exogenous variables:
  age             gaussian(identity)
  bili            log-normal
  sex             binomial(logit)
  stage           categorical
  t.death         weibull(1.45,0.00000033)
```

[2]Bender et al. (2005) Statistics in Medicine. Vol. 24, p:1713–1723

Censoring time

We add a censoring time named `t.cens` with another Cox-Weibull distribution:

```
distribution(m,~t.cens) <- coxWeibull.lvm(
    scale=0.0000000000091,
    shape=3.14)
print(m)
```

```
Latent Variable Model

Exogenous variables:
  age            gaussian(identity)
  bili           log-normal
  sex            binomial(logit)
  stage          categorical
  t.death        weibull(1.45,0.00000033)
  t.cens         weibull(3.14,0.0000000000091)
```

Simulate data from object

At any time during the building of the object we can check what happens when we simulate from the object.

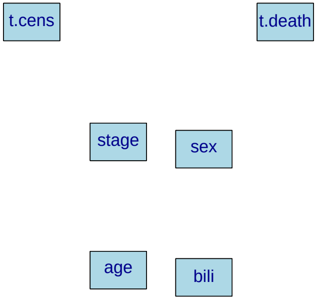
```r
set.seed(13)
print(sim(m,10),digits=2)
```

```
   age bili sex stage t.death t.cens
1   44 2.08   0     3   26599   4085
2   47 0.40   0     3   27514   2795
3   48 0.22   1     4    5382   2430
4   41 0.60   0     3   10133   2846
5   42 0.84   0     4   21129   4364
6   51 1.77   1     3   51346   2595
7   66 4.28   0     4   25055   4870
8   56 1.20   0     3   62335   3025
9   66 1.04   0   1/2   12149   3392
10  45 1.35   0     4   55198   2557
```

Plot I

There is a nice graphical display which shows the variables in the model.

```
plot(m)
```



Further variables

Treatment:

```
distribution(m,~trt) <- binomial.lvm(p=0.5)
```

Standardised blood clotting time:

```
distribution(m,~protime) <- lognormal.lvm(mean=2.37,sd
    =0.09)
```

Liver transplantation is a competing risk:

```
distribution(m,~t.trans) <- coxWeibull.lvm(scale
    =0.0000021,shape=1.9)
```

Transformed variables

Design (aka dummy) variables

We generate two binary design variables which indicate the stages "3" and "4" (stage "1/2" serves as reference group):

```
transform(m,stage3~stage) <- function(x){
    1*(x[["stage"]]==3)
}
transform(m,stage4~stage) <- function(x){
    1*(x[["stage"]]==4)
}
```

Design (aka dummy) variables

We generate two binary design variables which indicate the stages "3" and "4" (stage "1/2" serves as reference group):

```
transform(m,stage3~stage) <- function(x){
    1*(x[["stage"]]==3)
}
transform(m,stage4~stage) <- function(x){
    1*(x[["stage"]]==4)
}
```

Check simulation result

```
set.seed(18)
d=sim(m,5)
d[,grep("stage",names(d))]
```

```
  stage stage3 stage4
1   1/2      0      0
2   1/2      0      0
3   1/2      0      0
4     3      1      0
5     4      0      1
```

A factor variable

We tend to forget if `1` means `male` or `female` ...

```
transform(m,Sex~sex) <- function(x){
    factor(x[["sex"]],
     levels=c(0,1),labels=c("f","m"))
}
```

A factor variable

We tend to forget if `1` means `male` or `female` ...

```
transform(m,Sex~sex) <- function(x){
    factor(x[["sex"]],
     levels=c(0,1),labels=c("f","m"))
}
```

```
d=sim(m,5)
d[,grep("ex",names(d))]
```

```
  sex Sex
1   1   m
2   0   f
3   0   f
4   0   f
5   0   f
```

A categorized variable

We categorize age as ageCat with 4 categories:

```
transform(m,ageCat~age) <- function(x){
    cut(x[["age"]],
    c(-Inf,40,50,60,Inf),
    labels=c("<40","40-50","50-60",">60"))}
```

A categorized variable

We categorize age as ageCat with 4 categories:

```
transform(m,ageCat~age) <- function(x){
    cut(x[["age"]],
    c(-Inf,40,50,60,Inf),
    labels=c("<40","40-50","50-60",">60"))}
```

Check simulation result

```
set.seed(19)
d=sim(m,5)
d[,grep("^age",names(d))]
```

```
       age ageCat
1 47.39415  40-50
2 44.27494  40-50
3 64.06558    >60
4 55.04271  50-60
5 42.99719  40-50
```

Event time

We calculate the time to what comes first: transplant, death or end of study (censored) and derive the corresponding status variable

```
m <- eventTime(m,
           time~min(t.cens=0,t.trans=1,t.death=2),
           eventName="status")
```

Event time

We calculate the `time` to what comes first: transplant, death or end of study (censored) and derive the corresponding `status` variable

```
m <- eventTime(m,
           time~min(t.cens=0,t.trans=1,t.death=2),
           eventName="status")
```

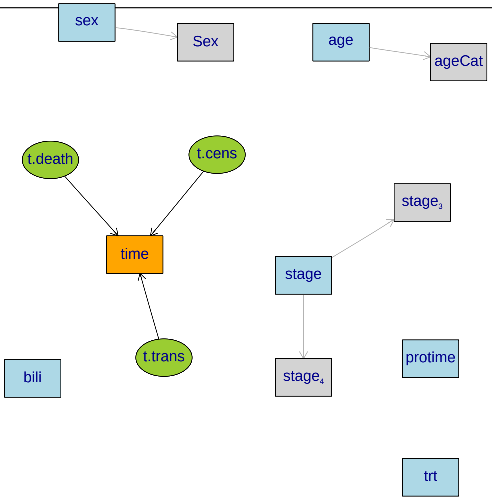
Check simulation result

```
set.seed(13)
d=sim(m,5)
d[,grep("^time|status|t\\.",names(d))]
```

```
   t.death   t.cens  t.trans     time status
1 51345.68 3128.879 611.8234 611.8234      1
2 25054.57 3178.120 332.2064 332.2064      1
3 62334.90 1496.004 313.1335 313.1335      1
4 12148.65 2003.784 723.2410 723.2410      1
5 55198.44 2813.327 691.1188 691.1188      1
```

Plot II

```
plot(m)
```



Further derived variables

Grouped blood clotting time:

```
transform(m,protimegrp~protime) <- function(x){
    cut(x[["protime"]], c(-Inf,10,11,Inf), labels=c(
    "<=10","10-11",">11"))
}
transform(m,protimegrp1~protimegrp) <- function(x){
    1*(x[["protimegrp"]]=="10-11")
}
transform(m,protimegrp2~protimegrp) <- function(x){
    1*(x[["protimegrp"]]==">11")
}
```

The log-transformed values of the variable `bili`

```
transform(m,logbili~bili) <- function(x){
    log(x[["bili"]])
}
```

Regression

Excursion: my Publish package

```r
# devtools::install_github('tagteam/Publish')
packageVersion("Publish")
library(Publish)
```

This package collects results of linear, logistic, poisson and Cox regression analyses in *publishable* table format.

(log)-Linear regression

The following line adds a negative effect of `sex` on `bili`

```
regression(m,bili~sex) <- -0.22
```

(log)-Linear regression

The following line adds a negative effect of `sex` on `bili`

```
regression(m,bili~sex) <- -0.22
```

Check if this had the expected effect:

```
set.seed(18)
d <- sim(m,1000)
publish(lm(log(bili)~Sex,data=d))
```

```
   Variable Units Coefficient         CI.95  p-value
(Intercept)              0.57   [0.51;0.64]  <0.0001
        Sex     f        0.00   [0.00;0.00]   1.0000
                m       -0.13  [-0.33;0.06]   0.1901
```

(log)-Linear regression

The following line adds a small positive effect of age on `bili`:

```
regression(m,bili~age) <- 0.002
```

(log)-Linear regression

The following line adds a small positive effect of age on `bili`:

```
regression(m,bili~age) <- 0.002
```

Check if this had the expected effect:

```
set.seed(16)
d <- sim(m,1000)
publish(lm(log(bili) ~ age,data=d))
```

```
   Variable Units Coefficient        CI.95  p-value
(Intercept)              0.33 [-0.08;0.75]   0.1187
        age              0.01 [-0.00;0.01]   0.1985
```

Logistic regression

We add effects as log odds ratios for `sex` and `age` on the probability of the treatment `trt==1`:

```
or <- c(0.98,1.003)
regression(m,trt~sex+age) <- log(or)
```

Logistic regression

We add effects as log odds ratios for `sex` and `age` on the probability of the treatment `trt==1`:

```
or <- c(0.98,1.003)
regression(m,trt~sex+age) <- log(or)
```

Check if this had the expected effect:

```
set.seed(18)
d <- sim(m,1000)
publish(glm(trt~Sex+age,data=d,family="binomial"))
```

```
Variable Units OddsRatio       CI.95  p-value
     Sex     f      1.00 [1.00;1.00]   1.0000
             m      1.09 [0.76;1.54]   0.6453
     age            1.01 [0.99;1.02]   0.3020
```

Cox regression

We add effects as log hazard ratios for sex, age, protime and bili on the Cox-Weibull distribution of the variable t.death.

```
hr2 <- c(0.93,1.03,2.5,1.51,1.89,1.47,2.27)
regression(m, t.death~sex+age+logbili+protimegrp1+
    protimegrp2 +stage3+stage4) <- log(hr2)
```

Note that t.death is a latent variable which is not observed for all patients in real life.

Cox regression

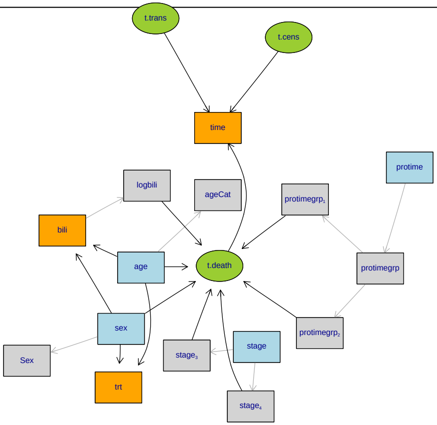
Check if this had the expected effect:

```
d <- sim(m,1000)
publish(coxph(Surv(time,status==2)~Sex+age+logbili+
    protimegrp+stage,data=d))
```

```
  Variable Units HazardRatio        CI.95   p-value
       Sex     f        1.00 [1.00;1.00]   1.00000
               m        1.35 [0.80;2.27]   0.25551
       age              1.04 [1.03;1.06]   < 0.001
   logbili              2.54 [2.14;3.01]   < 0.001
protimegrp  <=10        1.00 [1.00;1.00]   1.00000
           10-11        1.64 [1.02;2.64]   0.04228
             >11        2.46 [1.54;3.93]   < 0.001
     stage   1/2        1.00 [1.00;1.00]   1.00000
               3        1.66 [1.13;2.43]   0.00993
               4        1.87 [1.25;2.78]   0.00215
```

Plot III

```
require(visNetwork)
## lava.options(plot.engine="visNetwork")
plot(m)
```



Further regression effects

Add effects on the hazard rate of transplant:

```
hr1 <- c(0.31,0.91,2.28,0.37,0.33,2.37,5.5)
regression(m, t.trans~sex+age+logbili+protimegrp1
+protimegrp2+stage3+stage4) <- log(hr1)
```

```
d <- sim(m,1000)
publish(coxph(Surv(time,status==1)~Sex+age+logbili+
    protimegrp+stage,data=d),org=1L,units=list("age"="
    year"))
```

Variable	Units	HazardRatio	CI.95	p-value
Sex	f	1.00	[1.00;1.00]	1.00000
	m	0.34	[0.15;0.77]	0.00965
age	year	0.90	[0.88;0.92]	$<$ 0.001
logbili		2.14	[1.72;2.65]	$<$ 0.001
protimegrp	<=10	1.00	[1.00;1.00]	1.00000
	10-11	0.34	[0.22;0.53]	$<$ 0.001
	>11	0.37	[0.24;0.56]	$<$ 0.001
stage	1/2	1.00	[1.00;1.00]	1.00000
	3	2.94	[1.82;4.75]	$<$ 0.001
	4	6.29	[3.90;10.15]	$<$ 0.001

Simulating data alike `pbc` data

PBC data

For the purpose of illustration we consider the Mayo Clinic trial data in primary biliary cirrhosis (PBC) of the liver conducted between 1974 and 1984[3]

```r
library(survival)
data(pbc)
# ?pbc
pbc <- na.omit(pbc[,c("time","status","age","sex","stage"
    ,"bili","protime","trt")])
head(pbc)
```

```
  time status      age sex stage bili protime trt
1  400      2 58.76523   f     4 14.5    12.2   1
2 4500      0 56.44627   f     3  1.1    10.6   1
3 1012      2 70.07255   m     4  1.4    12.0   1
4 1925      2 54.74059   f     4  1.8    10.3   1
5 1504      1 38.10541   f     3  3.4    10.9   2
6 2503      2 66.25873   f     3  0.8    11.0   2
```

[3] Therneau and Grambsch (2000). Modeling Survival Dat

Data preparation

```r
pbc$stage <- factor(pbc$stage)
levels(pbc$stage) <- list("1/2"=c(1,2),"3"=3,"4"=4)
pbc$logbili <- log(pbc$bili)
pbc$logprotime <- log(pbc$protime)
pbc$protimegrp <- cut(pbc$protime,c(-Inf,10,11,Inf),labels=c("
    <=10","10-11",">11"))
pbc$trt <- factor(pbc$trt)
print(head(pbc),digits=1)
```

```
  time status age sex stage bili protime trt logbili logprotime protimegrp
1  400      2  59   f     4 14.5      12   1     2.7          3        >11
2 4500      0  56   f     3  1.1      11   1     0.1          2      10-11
3 1012      2  70   m     4  1.4      12   1     0.3          2        >11
4 1925      2  55   f     4  1.8      10   1     0.6          2      10-11
5 1504      1  38   f     3  3.4      11   2     1.2          2      10-11
6 2503      2  66   f     3  0.8      11   2    -0.2          2      10-11
```

Table 1

```
Publish::org(Publish::utable(trt∼age+sex+bili+protimegrp+stage,data=
    pbc))
```

Variable	Level	trt = 1 (n=158)	trt = 2 (n=154)	Total (n=312)	p-value
age	mean (sd)	51.4 (11.0)	48.6 (10.0)	50.0 (10.6)	0.01707
sex	m	21 (13.3)	15 (9.7)	36 (11.5)	
	f	137 (86.7)	139 (90.3)	276 (88.5)	0.42123
bili	mean (sd)	2.9 (3.6)	3.6 (5.3)	3.3 (4.5)	0.12992
protimegrp	<=10	40 (25.3)	49 (31.8)	89 (28.5)	
	10-11	82 (51.9)	57 (37.0)	139 (44.6)	
	>11	36 (22.8)	48 (31.2)	84 (26.9)	0.02915
stage	1/2	47 (29.7)	36 (23.4)	83 (26.6)	
	3	56 (35.4)	64 (41.6)	120 (38.5)	
	4	55 (34.8)	54 (35.1)	109 (34.9)	0.37731

Cox regression for mortality hazard rate

```
publish(coxph(Surv(time,status==2)~trt+sex+age+
    logbili+protimegrp+stage,data=pbc))
```

```
  Variable Units HazardRatio       CI.95   p-value
       trt                1.11 [0.77;1.61]   0.57628
       sex     m          1.00 [1.00;1.00]   1.00000
               f          0.93 [0.57;1.53]   0.78084
       age                1.03 [1.01;1.05]   < 0.001
   logbili                2.61 [2.13;3.19]   < 0.001
protimegrp  <=10          1.00 [1.00;1.00]   1.00000
           10-11          1.43 [0.79;2.56]   0.23469
             >11          1.71 [0.94;3.11]   0.08061
     stage   1/2          1.00 [1.00;1.00]   1.00000
               3          1.49 [0.84;2.64]   0.17395
               4          2.29 [1.30;4.06]   0.00441
```

Compare results in real and simulated data

```
set.seed(82)
d <- sim(m,312)
A <- publish(coxph(Surv(time,status==2)~trt+sex+age+logbili+
    protimegrp+stage,data=pbc),print=0L)
B <- publish(coxph(Surv(time,status==2)~trt+Sex+age+logbili+
    protimegrp+stage,data=d),print=0L)
publish(cbind(A$regressionTable,B$regressionTable[,-c(1:2)]))
```

```
 Variable Units HazardRatio       CI.95   p-value HazardRatio       CI.95
      trt              1.11 [0.77;1.61]   0.57628        0.74 [0.52;1.05]
      sex     m        1.00 [1.00;1.00]   1.00000        1.00 [1.00;1.00]
              f        0.93 [0.57;1.53]   0.78084        0.55 [0.28;1.10]
      age              1.03 [1.01;1.05]   < 0.001        1.03 [1.01;1.05]
  logbili              2.61 [2.13;3.19]   < 0.001        2.41 [1.96;2.96]
protimegrp <=10        1.00 [1.00;1.00]   1.00000        1.00 [1.00;1.00]
          10-11        1.43 [0.79;2.56]   0.23469        1.82 [1.09;3.02]
            >11        1.71 [0.94;3.11]   0.08061        1.89 [1.15;3.09]
    stage   1/2        1.00 [1.00;1.00]   1.00000        1.00 [1.00;1.00]
              3        1.49 [0.84;2.64]   0.17395        1.31 [0.85;2.02]
              4        2.29 [1.30;4.06]   0.00441        2.18 [1.42;3.36]
```

Setup a simulation study

Set up a simulation study for $n = 312$ based on the lvm object m.

```
run <- function(...,n=312) {
    d <- simulate(m,n=n)
    f <- coxph(Surv(time,status==2)~trt+sex+age+
    logbili+protimegrp+stage,data=d)
    structure(c(exp(coef(f)["trt"]),
        exp(coef(f)["logbili"])),
          names=c("trt","logbili"))
}
```

We study the estimates of the mortality hazard ratios for treatment trt and log-transformed bilirubin logbili

Running the simulation study

Calling the function `run` once returns the `coxph` estimates of the hazard ratios for `trt` and `logbili`.

```
run()
```

```
     trt  logbili
1.141812 2.253563
```

Running the simulation study

Calling the function `run` once returns the `coxph` estimates of the hazard ratios for `trt` and `logbili`.

```
run()
```

```
     trt  logbili
1.141812 2.253563
```

The following code runs the simulation 100 times and creates an R-object with the simulation results for which `lava` provides nice summary and plot functions.

```
set.seed(17)
simres <- sim(run,100,mc.cores=1)
# mc.cores=parallel::detectCores()
```

The estimated values

Results corresponding to the 100 simulated data sets:

```r
print(simres)
```

```
    trt    logbili
1   1.3034 2.9569
2   0.7955 2.4934
3   0.9765 2.9780
4   1.1604 2.2383
5   0.8884 2.4086
---
96  1.418  2.768
97  1.058  2.486
98  0.933  2.674
99  1.270  2.894
100 1.016  2.502
```

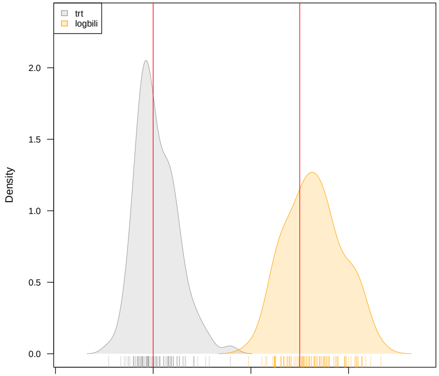
Summary of simulation results

```r
summary(simres,
    estimate=c("trt","logbili"),
    true=c(1,2.5))
```

```
100 replications                                    Time: 1.291s

             trt logbili
Mean    1.034356 2.64306
SD      0.211906 0.28886

Min     0.544908 1.97696
2.5%    0.711276 2.15632
50%     0.993638 2.63439
97.5%   1.499242 3.15971
Max     1.789160 3.33055

Missing 0.000000 0.00000

True    1.000000 2.50000
Bias    0.034356 0.14306
RMSE    0.214673 0.32234
```

Plot of simulation results (correctly specified model)

```r
density(simres)
abline(v=c(1,2.5),col=2)
```



Sensitivity analysis

Sensitivity analysis

We perform a sensitivity analysis to check the robustness of the estimates when the treatment decision depends on the bilirubin value via an unobserved confounder `U`. For this we

- add a latent variable named `U` with standard normal distribution
- add a regression effect with value `0.9` of `U` on `bili`
- add a regression effect with odds ratio `0.5` of `U` on `trt`

Adding the latent variable U

```
distribution(m,~U) <- normal.lvm(mean=0,sd=1)
latent(m) <- ~U
regression(m, bili~U) <- 0.9
regression(m, trt~U) <- log(0.5)
```

Running sensitivity analysis 1

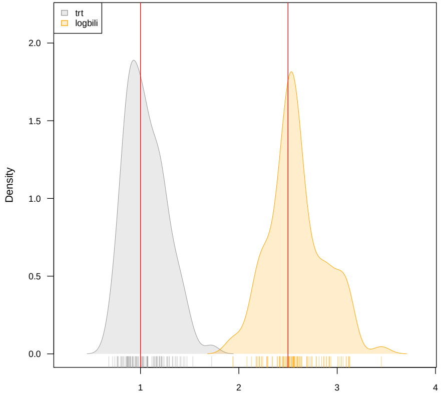
Now the `lvm` object `m` has changed and we can see the effect of `U` by re-running the simulation code:

```
set.seed(17)
simres1 <- sim(run,100,mc.cores=1)
summary(simres1,estimate=c("trt","logbili"),true=c(1,2.5))
```

```
100 replications                                        Time: 1.42s

             trt  logbili
Mean    1.053635 2.586743
SD      0.204659 0.284647

Min     0.677933 1.939730
2.5%    0.749897 2.104110
50%     1.021316 2.555836
97.5%   1.462270 3.120859
Max     1.723871 3.449172

Missing 0.000000 0.000000

True    1.000000 2.500000
Bias    0.053635 0.086743
RMSE    0.211570 0.297571
```

Results sensitivity analysis 1

```
density(simres1)
abline(v=c(1,2.5),col=2)
```



Sensitivity analysis 2

To provoke a more serious deviation from the assumptions of the Cox model we

- add a regression effect of `U` on `t.trans` with hazard ratio value 1.4
- add a regression effect of `U` on `t.death` with hazard ratio value 0.7

```
regression(m, t.trans~U) <- log(1.4)
regression(m, t.death~U) <- log(0.7)
```

Running sensitivity analysis 2

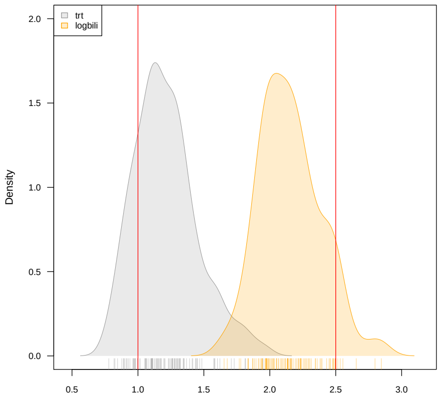
Again the lvm object m has changed and we can see the effect of letting U affect the event times by re-running the simulation code:

```
set.seed(17)
simres2 <- sim(run,100,mc.cores=1)
summary(simres2,estimate=c("trt","logbili"),true=c(1,2.5)
    )
```

```
100 replications                                    Time: 1.315s

            trt  logbili
Mean    1.21174  2.15489
SD      0.23731  0.23192

Min     0.77911  1.65335
2.5%    0.83491  1.77381
50%     1.17405  2.13609
97.5%   1.77336  2.60748
Max     1.94970  2.84601

Missing 0.00000  0.00000

True    1.00000  2.50000
Bias    0.21174 -0.34511
RMSE    0.31804  0.41580
```

Results of sensitivity analysis 2

```
density(simres2)
abline(v=c(1,2.5),col=2)
```



Summary and conclusion

- `lava` provides a very powerful and flexible simulation engine
- many other nice features not shown today
- the help pages can be improved
- regression effects on transformed variables do not work

```
 _____ _                 _          _  ___
|_   _| |__   __ _ _ __ | | _____  | |/ / | __ _ _   _ ___
  | | | '_ \ / _` | '_ \| |/ / __| | ' /| |/ _` | | | / __|
  | | | | | | (_| | | | |   <\__ \ | . \| | (_| | |_| \__ \
  |_| |_| |_|\__,_|_| |_|_|\_\___/ |_|\_\_|\__,_|\__,_|___/
```