

CT561: Systems Modelling and Simulation

Week 11: Sensitivity Analysis and Statistical Screening in R

<https://github.com/JimDuggan/CT561>

Dr. Jim Duggan,
Information Technology,
School of Engineering & Informatics

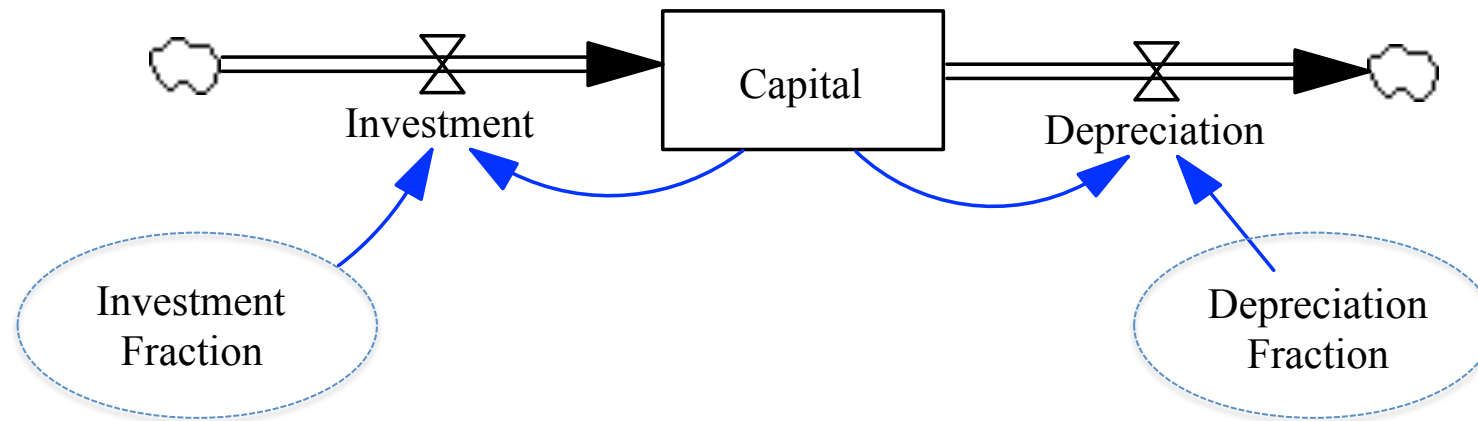


Sensitivity Analysis Process

- **Precondition:** Model equations finalised.
- Identify Uncertain Parameters, their distribution, and range of values
- Decide on number of runs (NRUNS)
- Let $n = 1$
- While $n \leq \text{NRUNS}$
 - Generate Parameter Values (e.g. uniform distribution)
 - Run Simulation and store results
 - Let $n = n + 1$
- Plot results and analyse

Capital Model:

Experiment by varying the parameters



```
Capital = INTEG( Investment - Depreciation , 100)
Depreciation = Capital * Depreciation Fraction
Depreciation Fraction = 0.02
Investment = Capital * Investment Fraction
Investment Fraction = 0.07 - step ( 0.06, 10)
```

R Code – Notice the parameter ranges (Min and Max)

```
library(deSolve)
library(ggplot2)
library(scales)
```

```
START<-2015; FINISH<-2035; STEP<-0.25;
```

```
simtime <- seq(START, FINISH, by=STEP)
```

```
INV_FRACT_MIN <-0.03; INV_FRACT_MAX <-0.12
```

```
DEP_FRACT_MIN <-0.01; DEP_FRACT_MAX <-0.04
```

```
INIT_CAP_MIN <-9000; INIT_CAP_MAX <-10100
```

```
NRUNS<-100
```

Need an outer function

```
sensRun<-function(run){  
  model <- function(time, stocks, auxs)  
    {  
      stocks <- c(sCapital=runif(1,INIT_CAP_MIN,INIT_CAP_MAX))  
  
      # create the vector of auxiliaries  
      auxs <- c(aInvestmentFraction=  
                runif(1, INV_FRACT_MIN,  
                      INV_FRACT_MAX),  
                aDepreciationFraction=  
                runif(1,DEP_FRACT_MIN,  
                      DEP_FRACT_MAX))  
  
      o<-data.frame(ode(y=stocks, simtime, func = model,  
                      parms=auxs, method="euler"))  
  
      o$run<-run  
      return (o)  
    }  
}
```

The model file

```
model <- function(time, stocks, auxs)
{
  with(as.list(c(stocks, auxs)),{

    fInvestment <- sCapital * aInvestmentFraction
    fDepreciation <- sCapital * aDepreciationFraction

    dC_dt <- fInvestment - fDepreciation

    return (list(c(dC_dt),
                  Investment=fInvestment,
                  Depreciation=fDepreciation,
                  InvFraction=aInvestmentFraction,
                  DepFraction=aDepreciationFraction))
  })
}
```

The controlling loop

```
o1<-sensRun(1)
for(n in 2:NRUNS)
{
  o1<-rbind(o1,sensRun(n))
}
```

```
> dim(o1)
```

```
[1] 8100    7
```

```
> head(o1)
```

	time	sCapital	Investment	Depreciation	InvFraction	DepFraction	run
1	2015.00	9206.398	947.8448	250.2711	0.102955	0.02718447	1
2	2015.25	9380.791	965.7995	255.0119	0.102955	0.02718447	1
3	2015.50	9558.488	984.0943	259.8425	0.102955	0.02718447	1
4	2015.75	9739.551	1002.7356	264.7646	0.102955	0.02718447	1
5	2016.00	9924.044	1021.7301	269.7799	0.102955	0.02718447	1
6	2016.25	10112.031	1041.0843	274.8902	0.102955	0.02718447	1

Data snapshot

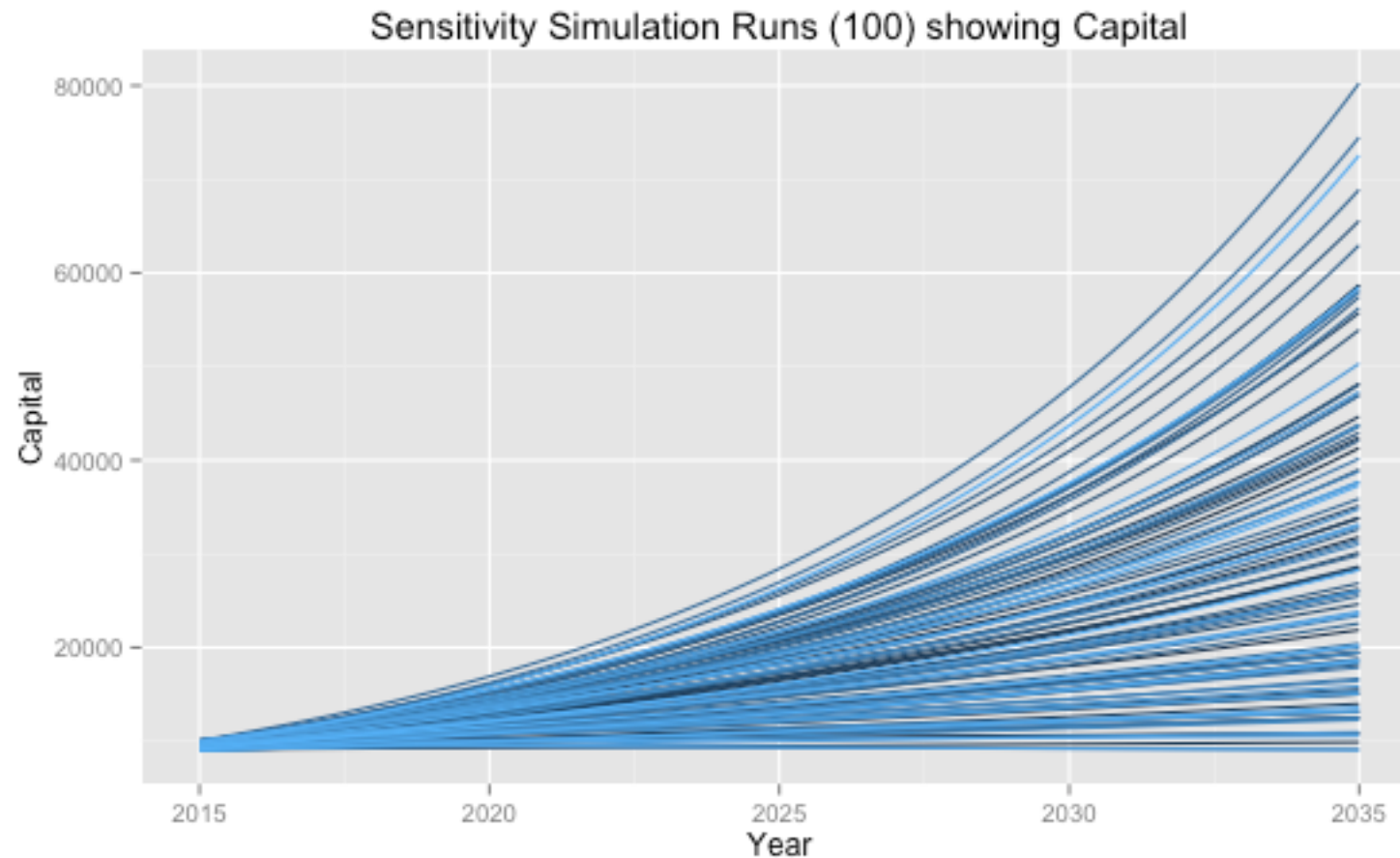
```
> o1[3000:3005,]
```

	time	sCapital	Investment	Depreciation	InvFraction	DepFraction	run
3000	2015.50	10352.49	875.0018	112.8731	0.08452086	0.01090299	38
3001	2015.75	10543.03	891.1057	114.9505	0.08452086	0.01090299	38
3002	2016.00	10737.07	907.5060	117.0661	0.08452086	0.01090299	38
3003	2016.25	10934.68	924.2082	119.2206	0.08452086	0.01090299	38
3004	2016.50	11135.92	941.2178	121.4148	0.08452086	0.01090299	38
3005	2016.75	11340.87	958.5404	123.6494	0.08452086	0.01090299	38

```
> tail(o1)
```

	time	sCapital	Investment	Depreciation	InvFraction	DepFraction	run
8095	2033.75	26370.07	1970.675	473.7822	0.07473148	0.01796666	100
8096	2034.00	26744.30	1998.641	480.5057	0.07473148	0.01796666	100
8097	2034.25	27123.83	2027.004	487.3247	0.07473148	0.01796666	100
8098	2034.50	27508.75	2055.770	494.2404	0.07473148	0.01796666	100
8099	2034.75	27899.13	2084.944	501.2543	0.07473148	0.01796666	100
8100	2035.00	28295.06	2114.531	508.3677	0.07473148	0.01796666	100

Visualise Output (Stock)



```
p1<-ggplot(o1,aes(x=time,y=sCapital,color=run,group=run)) + geom_line() +  
  ylab("Capital") +  
  xlab("Year") + guides(color=FALSE) +  
  ggtitle("Sensitivity Simulation Runs (100) showing Capital")
```

Statistical Screening

- A pragmatic method to search for the most important of uncertain parameters.
- The goal is to learn which of the many uncertain inputs stands out as most influential

$$r = \frac{\sum (X - \bar{X})(Y - \bar{Y})}{\sqrt{[\sum (X - \bar{X})^2][\sum (Y - \bar{Y})^2]}}$$

Measures of Association

Sample Pearson Correlation Coefficient

- The quantitative measure of the strength in the linear relationship between two variables
- The values of r can range from a perfect positive correlation of 1, to a perfect negative correlation of -1
- Care should be taken when interpreting results, just because two variables are correlated does not guarantee a cause and effect situation
- Assumption that data follows a normal distribution

$$r = \frac{\sum(X - \bar{X})(Y - \bar{Y})}{\sqrt{[\sum(X - \bar{X})^2][\sum(Y - \bar{Y})^2]}}$$

Example (where r should be = 1)

$$r = \frac{\sum(X - \bar{X})(Y - \bar{Y})}{\sqrt{[\sum(X - \bar{X})^2][\sum(Y - \bar{Y})^2]}}$$

```
> x
[1] 1 2 3
> y
[1] 3 6 9
> sum((x-mean(x))*(y-mean(y)))
[1] 6
```

```
> t1<-sum((x-mean(x))^2)
> t1
[1] 2
> t2<-sum((y-mean(y))^2)
> t2
[1] 18
```

```
> sum((x-mean(x))*(y-mean(y)))/sqrt(t1*t2)
[1] 1
```

Process

- Identify the variable of interest
- Run a suite of n sensitivity runs, with parameters taking on a range of value
 - Organise the full data set by time (i.e. n observations for each variable at a given time)
 - Use the `cor()` function to calculate the correlation coefficient
 - Repeat for each time step
- Evaluate the overall result

From earlier...

```
> dim(o1)
```

```
[1] 8100    7
```

```
> head(o1)
```

	time	sCapital	Investment	Depreciation	InvFraction	DepFraction	run
1	2015.00	9206.398	947.8448	250.2711	0.102955	0.02718447	1
2	2015.25	9380.791	965.7995	255.0119	0.102955	0.02718447	1
3	2015.50	9558.488	984.0943	259.8425	0.102955	0.02718447	1
4	2015.75	9739.551	1002.7356	264.7646	0.102955	0.02718447	1
5	2016.00	9924.044	1021.7301	269.7799	0.102955	0.02718447	1
6	2016.25	10112.031	1041.0843	274.8902	0.102955	0.02718447	1

```
> tail(o1)
```

	time	sCapital	Investment	Depreciation	InvFraction	DepFraction	run
8095	2033.75	26370.07	1970.675	473.7822	0.07473148	0.01796666	100
8096	2034.00	26744.30	1998.641	480.5057	0.07473148	0.01796666	100
8097	2034.25	27123.83	2027.004	487.3247	0.07473148	0.01796666	100
8098	2034.50	27508.75	2055.770	494.2404	0.07473148	0.01796666	100
8099	2034.75	27899.13	2084.944	501.2543	0.07473148	0.01796666	100
8100	2035.00	28295.06	2114.531	508.3677	0.07473148	0.01796666	100

The split() function

Description

`split` divides the data in the vector `x` into the groups defined by `f`. The replacement forms replace values corresponding to such a division. `unsplit` reverses the effect of `split`.

```
> runs<-split(o1,o1$time)
```

```
> head(runs[[1]])
```

	time	sCapital	Investment	Depreciation	InvFraction	DepFraction	run
1	2015	9206.398	947.8448	250.2711	0.10295501	0.02718447	1
82	2015	9535.836	334.9849	117.4564	0.03512906	0.01231737	2
163	2015	9699.243	694.6379	114.4674	0.07161774	0.01180168	3
244	2015	9647.232	1070.9941	190.2613	0.11101568	0.01972185	4
325	2015	9962.860	907.6735	381.1013	0.09110572	0.03825220	5
406	2015	10014.443	384.6826	347.9108	0.03841278	0.03474091	6

```
> head(runs[[2]])
```

	time	sCapital	Investment	Depreciation	InvFraction	DepFraction	run
2	2015.25	9380.791	965.7995	255.0119	0.10295501	0.02718447	1
83	2015.25	9590.218	336.8953	118.1262	0.03512906	0.01231737	2
164	2015.25	9844.286	705.0255	116.1791	0.07161774	0.01180168	3
245	2015.25	9867.416	1095.4379	194.6037	0.11101568	0.01972185	4
326	2015.25	10094.503	919.6670	386.1370	0.09110572	0.03825220	5
407	2015.25	10023.636	385.0358	348.2302	0.03841278	0.03474091	6

supply()... to process the list.

```
corInv<-sapply(runs,function(l){cor(l$sCapital, l$InvFraction )})  
corDep<-sapply(runs,function(l){cor(l$sCapital, l$DepFraction )})
```

```
> corDep[1:7]
```

2015	2015.25	2015.5	2015.75	2016	2016.25	2016.5
0.142374549	0.071538932	0.005488386	-0.051454845	-0.098256220	-0.135845238	-0.165839818

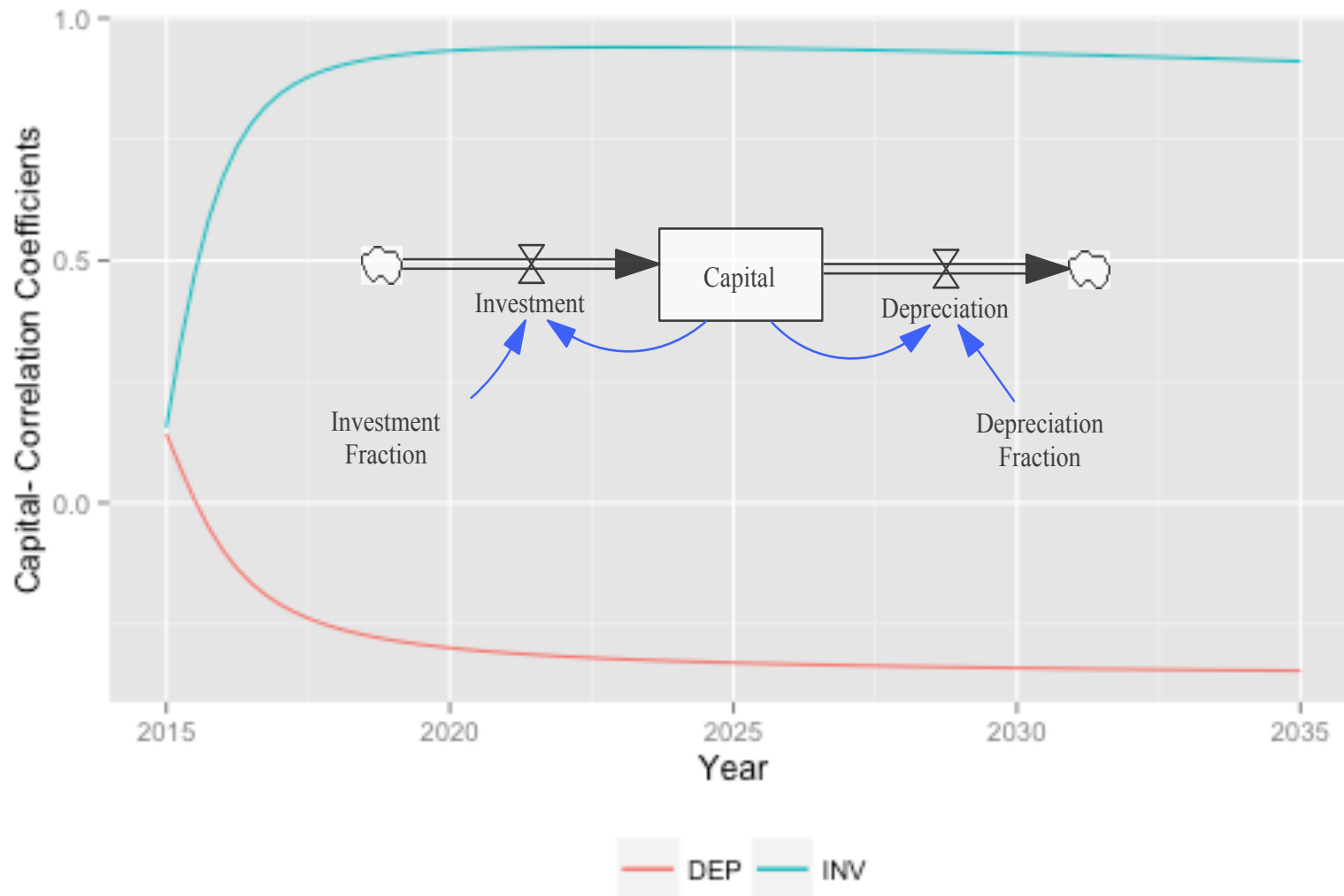
```
> corInv[1:7]
```

2015	2015.25	2015.5	2015.75	2016	2016.25	2016.5
0.1553017	0.3271642	0.4723357	0.5864322	0.6723759	0.7359221	0.7827813

```
p2<-ggplot()+  
  geom_line(aes(simtime,corInv,color="INV"))+  
  geom_line(aes(simtime,corDep,color="DEP"))+  
  scale_y_continuous(labels = comma)+  
  ylab("Capital- Correlation Coefficients")+  
  xlab("Year") +  
  labs(color="")+  
  theme(legend.position="bottom")
```

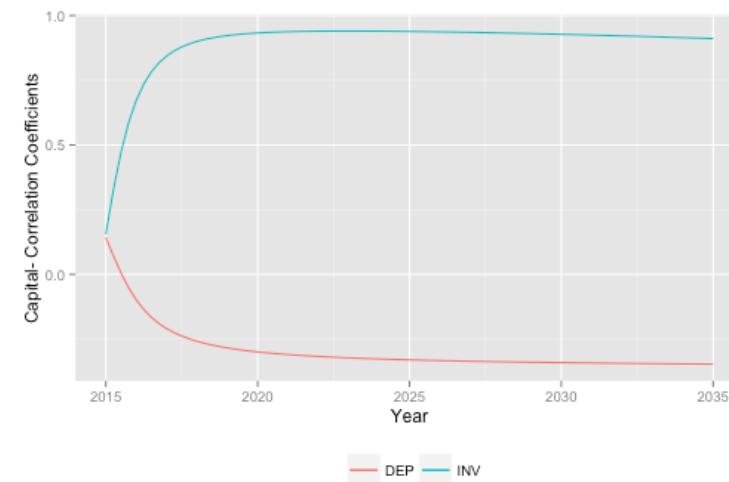
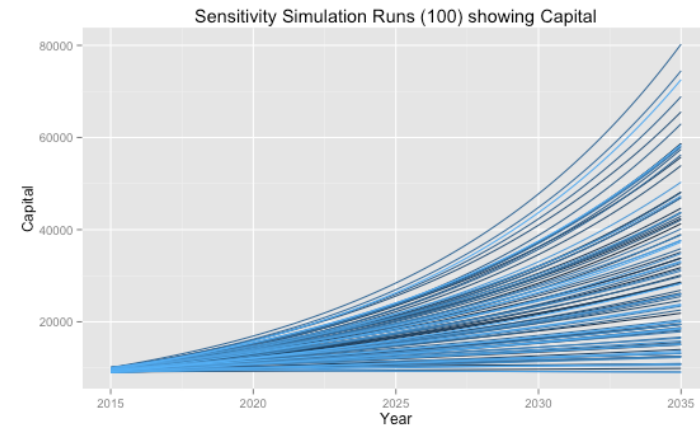

Analysis of Output:

Investment Fraction more influential



Summary

- Sensitivity Analysis supports exploring parameter uncertainty
- R supports generating large data sets of runs
- **split()** and **cor()** functions can be used to calculate correlation coefficients
- Analysis shows most influential parameters



Reference

Statistical screening of system dynamics models

Andrew Ford^{a*} and Hilary Flynn^b

Abstract

This paper describes a pragmatic method of searching for the key inputs to a system dynamics model. This analysis is known as screening. The goal is to learn which of the many uncertain inputs stand out as most influential. The method is implemented with readily available software and relies on the simple correlation coefficient to indicate the relative importance of model inputs at different times in the simulation. The screening is demonstrated with two examples with step-by-step instructions. The paper recommends that screening analysis be used in an iterative process of screening and model expansion to arrive at tolerance intervals on model results. The appendices compare screening analysis with analytical methods to identify the key inputs to system dynamics models. Copyright © 2005 John Wiley & Sons, Ltd.

Syst. Dyn. Rev. **21**, 273–303, (2005)