The simulation of pseudo-data

Simulating data is a tricky business. If you ever try to sell your simulations as real data, your career will certainly be ruined and your self-esteem will drop to zero. Who wants to end like that?
Nevertheless, simulating data can be useful in many ways. You can learn to think in data structures, in *data generating processes*, which helps you in designing experiments and will make it easier to think ahead in terms of how you will analyse your data. Also a power analysis such as the one of a previous exercise often consists of simulations.
Many statistical inference methods also use a simulation component to arrive at conclusions. The jackknife and bootstrap are examples of methods where pseudo-data are essential.

Today, we will not use or write the most beautiful code ever written, but it will help you to get started in data simulations. A helper file with R commands is available on blackboard. It is called "datasims2006.txt".

♣ Resampling Methods

Methods such as the jackknife and bootstrap resample from the original data collected, to generate sets of pseudo-data on which an analysis is carried out each time.
I give the simplest possible example here of resampling.
We generate 40 numbers from a standard normal distribution

```
x <- rnorm(40)
```

Then we resample these data, either with or without replacement.

```
sample(x)
```

```
sample(x,replace=TRUE)
```

Resampling using the function sample() can go wrong sometimes. You better use functions from the "boot" library for more involved sampling schemes.

♣ How to get real simulated data...

(Tegeltjeswijsheden)
- It is impossible to simulate data without a model for the data-generating process.
- All models can be simulated.

You have to determine which are the independent variables and the dependent (or response) variables that together make up the data generating process.
You have to determine which independent variables will be controlled by the experimenter, and which ones will be random themselves.
You have to determine which probability distributions you will use to generate randomness in the data.
You will have to decide which variables will actually be "measured" or "observed".
You have to model the measurement error.

We will construct a model from a paper, bit by bit. Kark and Sol (2005) is on the establishment success of introduced bird species, and you can find the *pdf* on blackboard.
First, go through this document and run all the different versions of the model one by one.
Then start from **HERE** for a second round, and modify the code such that you obtain the model of Table 2 in the paper.

The response variable, is a binomially distributed random variable. In models for such variables, a polynomial with independent variables is linked to the response via a non-linear transformation as explained in the exercise on liability traits.
The transformation we use today is the most commonly used transformation called the "logit" transformation.

```
logit<-function(p)log(p/(1-p))
invlogit<-function(pol)exp(pol)/(1+exp(pol))
```

First we use the data in Table one of Kark and Sol (2005) to build a very simplified model.
Let's look at those data of Table 1 first:

```
data<-cbind(success=c(29,26,31,10),failure=c(42-29,0,63-31,24-10))
data
```

We can calculate the probability of establishment per region as follows:

```
data[,1]/(data[,1]+data[,2])
```

First, we will build a model assuming that the probability of establishment is identical in all regions.
We use a quite involved route to do so. It will make it easier to turn the resulting model into a more elaborate one.

The overall probability of establishment is

```
sum(data[,1])/sum((data[,1]+data[,2]))
```

We can fit a *generalized linear model* to the data.

"~1" means that we only fit an intercept parameter to the data, nothing else (no dummy variables, no covariates, nothing). The default setting in this glm for the transformation is the logit, that's why we don't have to write it anywhere.

```
glm1<-glm(data~1,family=binomial)
summary(glm1)

Call:
glm(formula = data ~ 1, family = binomial)

Deviance Residuals:
     1          2          3          4
 0.9629     4.9912    -2.0491    -2.0063

Coefficients:
             Estimate   Std. Error  z value    Pr(>|z|)
(Intercept)  0.4868     0.1654      2.943      0.00325 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 34.063  on 3  degrees of freedom
Residual deviance: 34.063  on 3  degrees of freedom
AIC: 48.336

Number of Fisher Scoring iterations: 4
```

In the summary table, we see that only an intercept parameter is estimated. We can transform that estimate to the average probability of establishment, by using the invlogit() function we already defined.

```
invlogit(glm1$coef)
```

We can also simulate a new dataset on the basis of this model.

```
simintro1<-rbinom(155,1,prob=invlogit(glm1$coef))
sum(simintro1)/155
```

In this second step, we build a model with separate probabilities of establishment per region. We need an extra variable ("regions") which lists the names of the regions. It has to be a "factor" variable, to make it clear to R that the names stand for groups.

```
regions<-factor(c("California","MedBasin","Australia","Cape"))
```

We fit a second glm:

```
glm2<-glm(data~regions,family=binomial)
summary(glm2)
```

This model has 4 parameters, so that you can calculate 4 different predicted probabilities of establishment for the groups. For the purpose of simulating data, it is easier to remove the intercept from the model:

```
glm2<-glm(data~regions-1,family=binomial)
summary(glm2)
glm2$coef
```

Here we simulate data according to this second model:

```
simintro2<-rep(0,155)
simregions<-
factor(c(rep("California",42),rep("MedBasin",26),rep("Australia",63),rep("Cape",24)))

table(simregions)

for(i in 1:155){
out<-simintro2;
out[i]<-
ifelse(simregions[i]=="Australia",rbinom(1,1,prob=invlogit(glm2$coef[1])),out[i]);
out[i]<-
ifelse(simregions[i]=="California",rbinom(1,1,prob=invlogit(glm2$coef[2])),out[i]);
out[i]<-ifelse(simregions[i]=="Cape",rbinom(1,1,prob=invlogit(glm2$coef[3])),out[i]);
out[i]<-
ifelse(simregions[i]=="AMedBasin",rbinom(1,1,prob=invlogit(glm2$coef[4])),out[i]);
simintro2<-out
}
```

The result:

```
table(simintro2,simregions)
```

We can simulate from models with additional explanatory variables. Here's a way of simulating the effect of introduction mode. We assume that introductions are assigned randomly to models of introduction, and that 1/3 of introductions are deliberate.

```
rmode1<-rbinom(length(simintro),1,1/3)
table(rmode1,simregions)

simmode<-ifelse(rmode1==1,"deliberate",NA)
```

You need to be careful when you code additional models of introduction, that your way of programming doesn't create observations for which two modes of introduction occur.

You can also add so-called "random effects", which will not be observed directly, but of which we know which observations share a specific random effect. In the statistical analysis of random effects, you are mainly interested in the variance between groups, and not really in the actual effect value of each group. Nevertheless, when simulating, you need to simulate a random effect per level of the grouping variable.
We simulate a random effect of species, assuming that there are twenty species.

```
species<-gl(20,1) # list of species "names"
simspecies<-sample(species, 155, replace=TRUE) # species variable

specieseff<-rnorm(20) # species random effects
```

Plot a histogram of the species effects. We now distribute all the species effects across the observations, using a loop.

```
simspecieseffect<-rep(0,155)

for(i in 1:155){
simspecieseffect[i]<-specieseff[simspecies[i]]
}
```

Here we simulate a model with region-specific probabilities, and a random effect of species:

```
simintro3<-rep(0,155)

for(i in 1:155){
out<-simintro3;
out[i]<-
ifelse(simregions[i]=="Australia",rbinom(1,1,prob=invlogit(glm2$coef[1]+simspecieseff
ect[i])),out[i]);
out[i]<-
ifelse(simregions[i]=="California",rbinom(1,1,prob=invlogit(glm2$coef[2]+simspeciesef
fect[i])),out[i]);
out[i]<-
ifelse(simregions[i]=="Cape",rbinom(1,1,prob=invlogit(glm2$coef[3]+simspecieseffect[i
])),out[i]);
out[i]<-
ifelse(simregions[i]=="AMedBasin",rbinom(1,1,prob=invlogit(glm2$coef[4]+simspecies
effect[i])),out[i]);
simintro3<-out
}
```

The authors analysed their data with this kind of mixed model approach:

```
library(lme4)
mm<-lmer(simintro3~simregions+(1|simspecies)-1,family=binomial)
summary(mm)
```

Now go for round two, and simulate the model of the data in the paper!
Note that Kark and Sol did not list their intercept parameter in Table 2, we will assume it is equal to our glm1$coef estimate. There is one simplification allowed: You have to code only a single random effect of species, we will not code the family random effect.
You can simplify your simulation code a lot by using dummy variables to denote group membership. Then the last loop in which the random probabilities are drawn becomes very simple.

♠ Some types of biological data are random variables, but they are very complicated to simulate. Fortunately there is always some software to help you. For example:

Seqgen – helps you to simulate DNA sequence data.

Tom Van Dooren 09/2007
t.j.m.van.dooren@biology.leidenuniv.nl