Simulation and parallelization in R

Author: Nicholas G Reich, Andrea S Foulkes, Gregory J Matthews

Biostatistics in Practice: High-Performance Computing with R
7 February 2014

This material is part of the **statsTeachR** *project*

Module learning goals

At the end of this module you should be able to...

- simulate data from a parametric distribution.
- Design and implement a resampling simulation experiment to test a hypothesis.
- Run simulations in parallel, when appropriate.

What is simulation?

Definitions

- Broadly: "The technique of imitating the behaviour of some situation or process (whether economic, military, mechanical, etc.) by means of a suitably analogous situation or apparatus, esp. for the purpose of study or personnel training." (from the *OED*)
- In science: Creating a model that imitates a physical or biological process.
- In statistics: The generation of data from a model using rules of probability.

Simple examples of simulations

- Drawing pseudo-random numbers from a probability distribution (e.g. proposal distributions, ...).
- Generating data from a specified model (e.g. building a template dataset to test a method, calculating statistical power).
- Resampling existing data (e.g. permutation, bootstrap).

What simulations have you run?

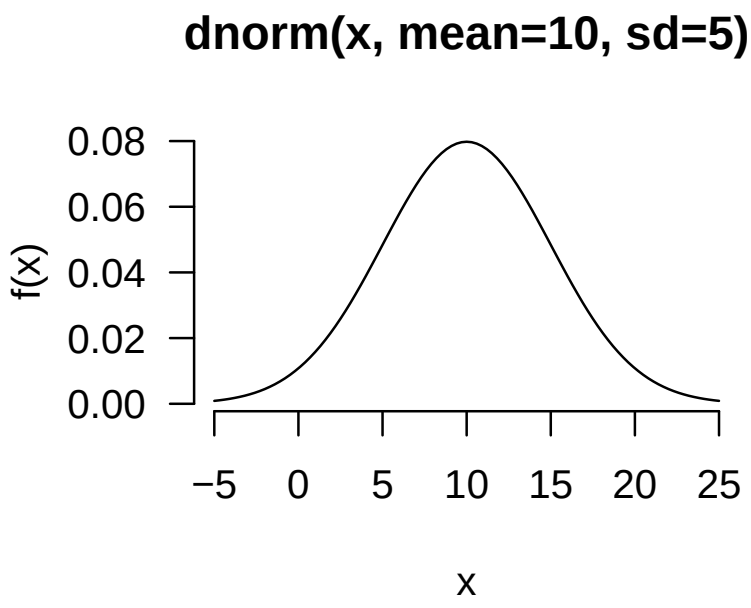
Random number generation in R

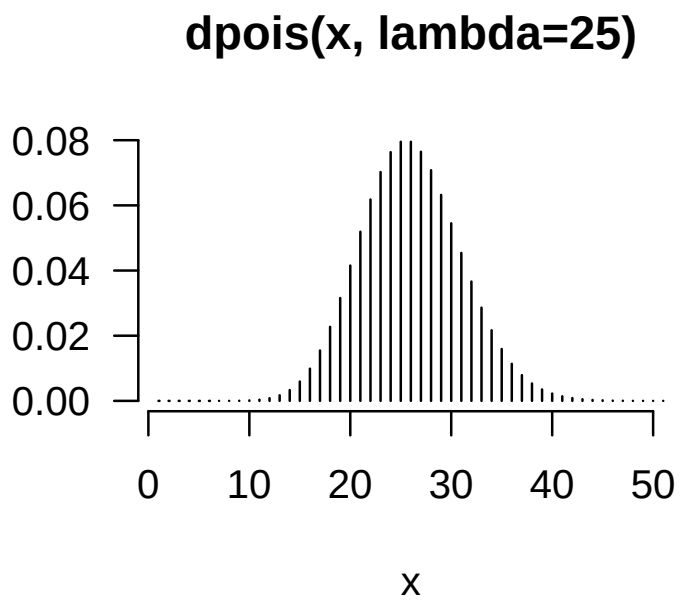
`rnorm()`, `rpois()`, etc...

Built-in functions for simulating from parametric distributions.

```
y <- rnorm(100, mean = 10, sd = 5)
(p <- rpois(5, lambda = 25))

## [1] 23 22 22 28 27
```





Resampling data in R

`sample()`

Base R function for sampling data (with or without replacement).

```
p
```

```
## [1] 23 22 22 28 27
```

```
sample(p, replace = FALSE)
```

```
## [1] 22 28 23 27 22
```

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

```
## [1] 27 27 22 23 27
```

Generating data from a model

A Simple Linear Regression model

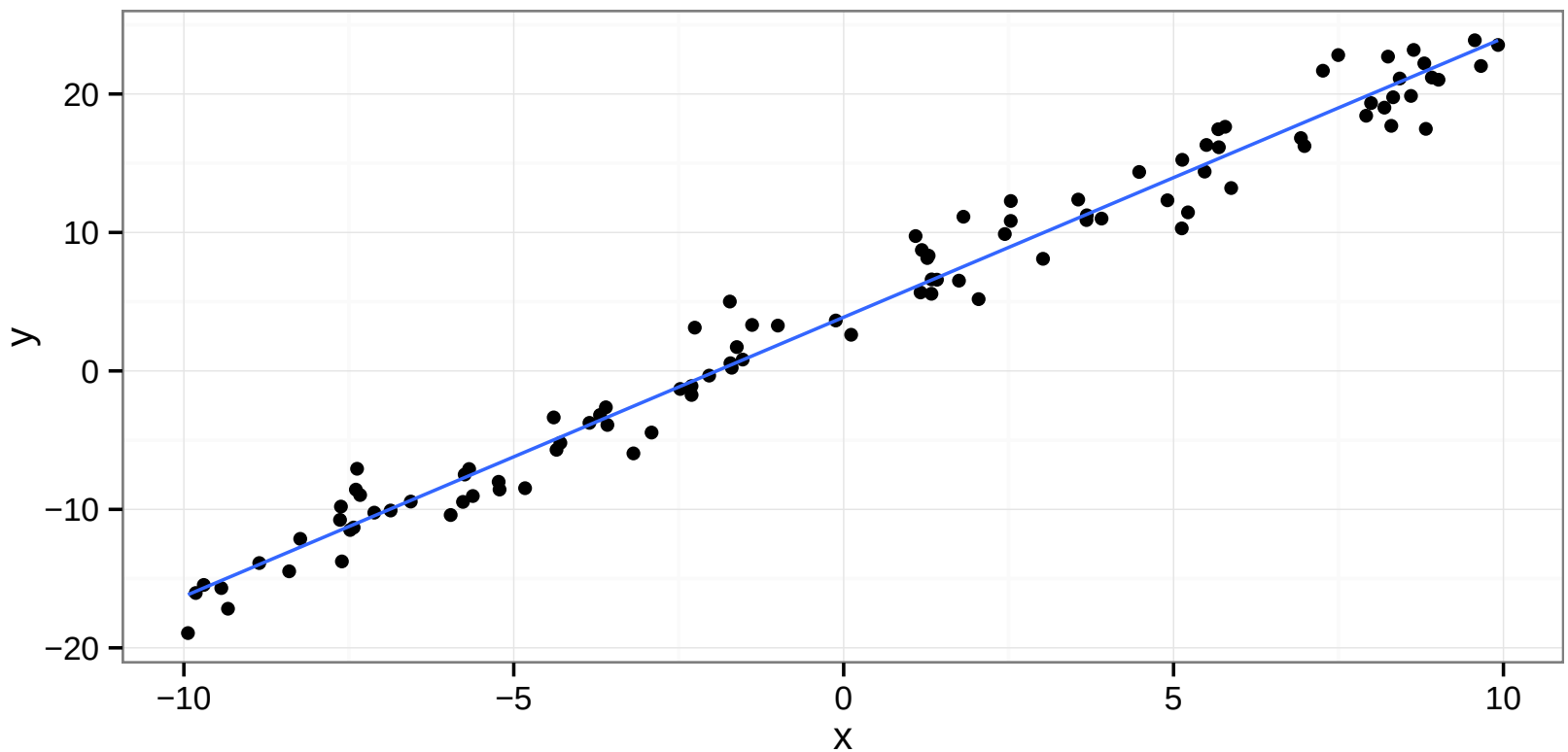
$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

What is needed to simulate data (i.e. Y_i) from this model?

- The X_i: fixed quantities.
- Error distribution: e.g. $\epsilon_i \overset{iid}{\sim} N(0, \sigma^2)$.
- Values for parameters: β_0, β_1, σ^2.

Generating data from $Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$

```
require(ggplot2)
n <- 100; b0=4; b1=2; sigma=2      ## define parameters
x <- runif(n, -10, 10)             ## fix the X's
eps <- rnorm(n, sd=sigma)          ## simulate the e_i's
y <- b0 + b1*x + eps               ## compute the y_i's
qplot(x, y, geom=c("point", "smooth"), method="lm", se=FALSE)
```



Example data: heights of mothers and daughters

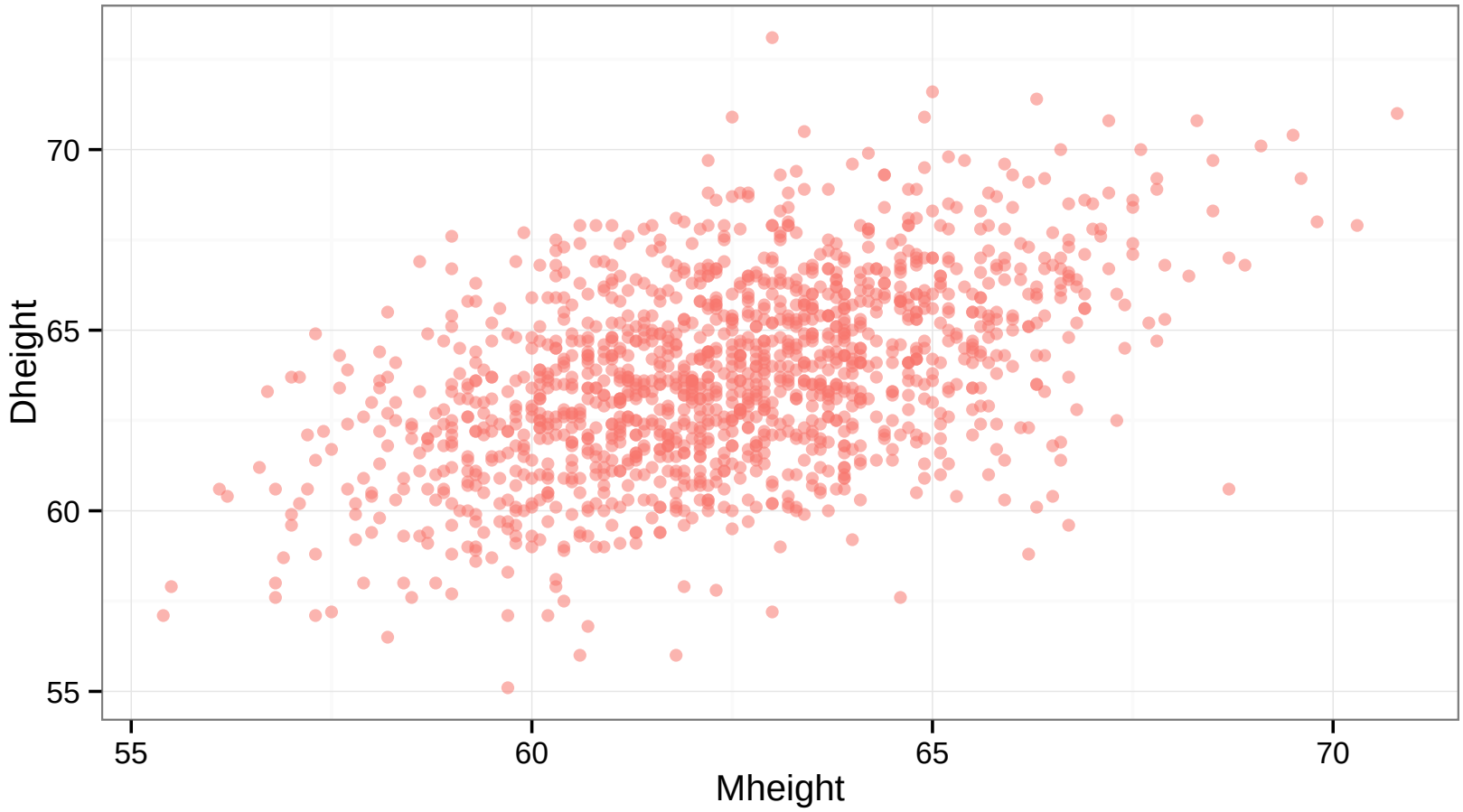
Heights of $n = 1375$ mothers in the UK under the age of 65 and one of their adult daughters over the age of 18 (collected and organized during the period 1893–1898 by the famous statistician Karl Pearson)

```
require(alr3)
data(heights)
head(heights)

##   Mheight Dheight
## 1    59.7    55.1
## 2    58.2    56.5
## 3    60.6    56.0
## 4    60.7    56.8
## 5    61.8    56.0
## 6    55.5    57.9
```

Example data: heights of mothers and daughters

```
qplot(Mheight, Dheight, data=heights, col="red", alpha=.5) +
        theme(legend.position="none")
```



One way to draw inference about height association

Using normality assumptions and simple linear regression

$$Dheight_i = \beta_0 + \beta_1 \cdot Mheight_i + \epsilon_i$$

```
mod1 <- lm(Dheight ~ Mheight, data = heights)
summary(mod1)$coefficients
```

```
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept)  29.9174    1.62247   18.44 5.212e-68
## Mheight       0.5417    0.02596   20.87 3.217e-84
```

Another way to draw inference about height association

Using a simulation-based permutation test

- This can evaluate evidence for/against a null hypothesis.
- We are interested in $H_0 : \beta_1 = 0$, i.e. there is no relationship between heights of mother and daughter.
- The trick: we can easily simulate multiple sets of data that we know have no association!
- All we need is `sample()`.

```
resampDheight <- sample(heights$Dheight, replace = FALSE)
```

Single permutation results

We can then fit this model

$$Dheight_i^* = \beta_0 + \beta_1 \cdot Mheight_i + \epsilon_i$$

where $Dheight_i^*$ are the permuted daughter heights.
This essentially "generates" data from the null model:

$$Dheight_i^* = \beta_0 + 0 \cdot Mheight_i + \epsilon_i$$

```
mod2 <- lm(resampDheight ~ Mheight, data = heights)
summary(mod2)$coefficients
```

```
##             Estimate Std. Error t value   Pr(>|t|)
## (Intercept) 65.35658    1.86157 35.1083 3.063e-193
## Mheight     -0.02571    0.02979 -0.8631  3.883e-01
```

Permutation tests require repeated samples!

A permutation test algorithm

- Run original analysis (i.e. fit our linear model), store $\hat{\beta}_1$.
- For i in $1, 2, \ldots, N$:
  - Permute the Ys.
  - Re-run original analysis, store $\hat{\beta}_1^{(i)}$.
- Calculate fraction of the $\hat{\beta}_1^{(i)}$ as or more "extreme" than $\hat{\beta}_1$, from our "null distribution" of $\hat{\beta}_1$s.

Hands-on exercise

- We have provided code for you to adapt and run a permutation test.
- `BiPSandbox/module2/permTest2.Rmd`.

Shifting gears from statistics to computation...

Parallel computation in R

What is parallel computing?

- Wikipedia says: "Parallel computing is a form of computation in which many calculations are carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently ("in parallel")."
- Not every computational problem is parallelizable!
- Today, we will focus on problems that are "embarrassingly parallel" in nature.
- We will make a distinction between "local" and "distributed" parallel computing, but these are our loose terminology.

Local vs. distributed

Local parallelization

- Within a `.R` script, call multiple cores (on your laptop, on a cluster node, etc...).
- Two simple implementations:
  - `foreach()` requiring `doMC` (or `doNWS` or `doParallel`) and `foreach` packages
  - `mclapply()` requiring `parallel` package

Distributed parallelization

- Within a `.sh` script, call multiple `.R` files/jobs/scripts to run on separate nodes simultaneously.
- Each individual job could have local parallelization as well.

How many cores do you have?

- Simple way to check:

```
require(parallel)
detectCores()

## [1] 4
```

- Caveat 1: # of cores may be less than the # of simultaneous computational threads you could run.
- Caveat 2: You may not want to eat up all your local CPU with a computational job.
- Caveat 3: Some knowledgable people say noy to run things in parallel in the R GUI app. Instead, run using R CMD BATCH command line.

Hands-on exercise

- We have provided code for you to adapt and run a permutation test in parallel:
  `BiPSandbox/module2/permTest2_doMC.R`
  `BiPSandbox/module2/permTest2_doParallel.R`
  `BiPSandbox/module2/permTest2_mclapply.R`
- Let's try to learn about different parallel computing speeds through an old-school distributed computing experiment: see `BiPSandbox/module2/localParallelData.csv`

Recapping module learning goals

We have...

- simulated data from a parametric distribution.
- used resampling theory and simulation techniques to conduct a permutation test.
- run a simulation in parallel, on our laptops.
- learned something about parallel computing along the way.