Data Science for Actuaries (ACT6100)

Arthur Charpentier

Supervisé # 3 (simulations)

automne 2Q20

https://github.com/freakonometrics/ACT6100/

Random Numbers?

TABLE OF RANDOM DIGITS 1

00000	10097	32533	76520	13586	34673	54876	80959	09117	39292	74945
00001	37542	04805	64894	74296	24805	24037	20636	10402	00822	91665
00002	08422	68953	19645	09303	23209	02560	15953	34764	35080	33606
00003	99019	02529	09376	70715	38311	31165	88676	74397	04436	27659
00004	12807	99970	80157	36147	64032	36653	98951	16877	12171	76833
00005	66065	74717	34072	76850	36697	36170	65813	39885	11199	29170
00006	31060	10805	45571	82406	35303	42614	86799	07439	23403	09732
00007	85269	77602	02051	65692	68665	74818	73053	85247	18623	88579
00008	63573	32135	05325	47048	90553	57548	28468	28709	83491	25624
00009	73796	45753	03529	64778	35808	34282	60935	20344	35273	88435
00010	98520	17767	14905	68607	22109	40558	60970	93433	50500	73998
00011	11805	05431	39808	27732	50725	68248	29405	24201	52775	67851
00012	83452	99634	06288	98083	13746	70078	18475	40610	68711	77817
00013	88685	40200	86507	58401	36766	67951	90364	76493	29609	11062
00014	99594	67348	87517	64969	91826	08928	93785	61368	23478	34113
00015	65481	17674	17468	50950	58047	76974	73039	57186	40218	16544
00016	80124	35635	17727	08015	45318	22374	21115	78253	14385	53763
00017	74350	99817	77402	77214	43236	00210	45521	64237	96286	02655
00018	69916	26803	66252	29148	36936	87203	76621	13990	94400	56418
00019	09893	20505	14225	68514	46427	56788	96297	78822	54382	14598
00020	91499	14523	68479	27686	46162	83554	94750	89923	37089	20048
00021	80336	94598	26940	36858	70297	34135	53140	33340	42050	82341
00022	44104	81949	85157	47954	32979	26575	57600	40881	22222	06413
00023	12550	73742	11100	02040	12860	74697	96644	89439	28707	25815
00024	63606	49329	16505	34484	40219	52563	43651	77082	07207	31790
00025	61196	90446	26457	47774	51924	33729	65394	59593	42582	60527

Source A Million Random Digits with 100,000 Normal Deviates, RAND, 1955.

Random Numbers?

Here random means a sequence of numbers do not exhibit any discernible pattern, i.e. successively generated numbers can not be predicted.

A random sequence is a vague notion... in which each term is unpredictable to the uninitiated and whose digits pass a certain number of tests traditional with statisticians... Derrick Lehmer, quoted in Knuth (1997)

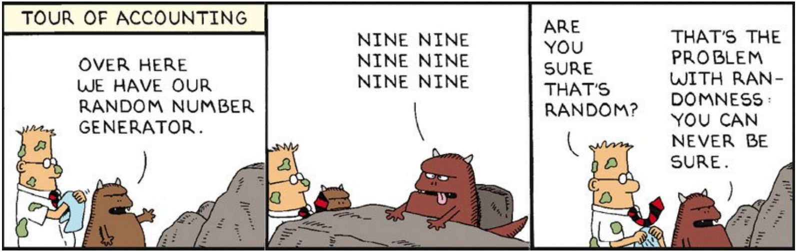
The goal of Pseudo-Random Numbers Generators is to produce a sequence of numbers in $[0,1]$ that imitates ideal properties of random number.

```
> runif(50)
 [1] 0.27 0.37 0.57 0.91 0.20 0.90 0.94 0.66 0.63 0.06
[11] 0.21 0.18 0.69 0.38 0.77 0.50 0.72 0.99 0.38 0.78
[21] 0.93 0.21 0.65 0.13 0.27 0.39 0.01 0.38 0.87 0.34
[31] 0.48 0.60 0.49 0.19 0.83 0.67 0.79 0.11 0.72 0.41
[41] 0.82 0.65 0.78 0.55 0.53 0.79 0.02 0.48 0.73 0.69
```

Random Numbers?

```
> y <- runif(1)
> y
[1] 0.6578972
> y
[1] 0.6578972
```

```
> x %<a-% runif(1)
> x
[1] 0.3973465
> x
[1] 0.2266485
```



Source Dibert, 2001.

Randomness

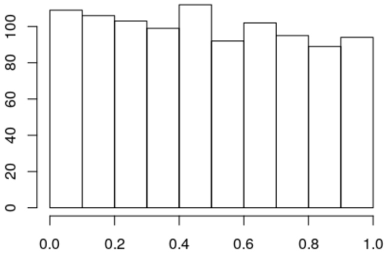
Heuristically,

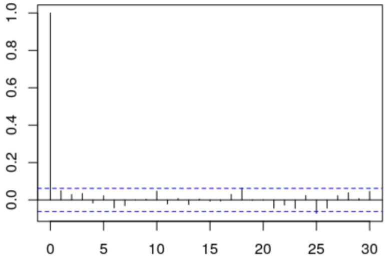
1. calls should provide a uniform sample:

$$\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{u_i\in(a,b)} = b-a \text{ with } b>a,$$

2. calls should be independent: for $b>a$ and $d>c$.

$$\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{u_i\in(a,b),u_{i+k}\in(c,d)} = (b-a)(d-c)\ \forall k\in\mathbb{N},$$





How to create randomness?

Linear Congruential Method

Given $a, b, m \in \mathbb{N}$ and $x_0 \in \{0, 1, \cdots, m\}$, define

$$x_{i+1} = (ax_i + b) \text{ modulo } m,$$

and set $u_i = x_i / m$.

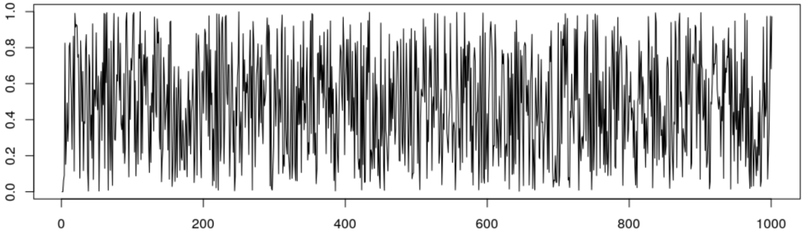
```
> a = 13; b = 43; m = 100; x = 77; u = rep(NA, 40)
> for (i in 1:40) {x = (a * x + b) %% m
+      u[i] = x / m }
> u
 [1] 0.44 0.15 0.38 0.37 0.24 0.55 0.58 0.97 0.04 0.95
[11] 0.78 0.57 0.84 0.35 0.98 0.17 0.64 0.75 0.18 0.77
[21] 0.44 0.15 0.38 0.37 0.24 0.55 0.58 0.97 0.04 0.95
```

Problem: not all values in $\{0, \cdots, m-1\}$ are obtained, and there is a cycle here.

Solution: (very) large values for m and choose properly a and b.

How to create randomness?

E.g. $m = 2^{32} - 1$, $a = 16807$ $(= 7^5)$ and $b = 0$ (used in Matlab).



See L'Ecuyer (2017) for an historical perspective,

Note See McCullough & Heiser (2008) or Mélard (2014) about MS Excel and randomness

Random Numbers?

```
> runif(1)
[1] 0.6696012
> runif(1)
[1] 0.6721922
> runif(1)
[1] 0.2026715
> runif(1)
[1] 0.7540288
```

```
> set.seed(123)
> runif(1)
[1] 0.2875775
> runif(1)
[1] 0.7883051
> runif(1)
[1] 0.4089769
```

```
> set.seed(123)
> runif(1)
[1] 0.2875775
> runif(1)
[1] 0.7883051
> runif(1)
[1] 0.4089769
```

For parallel computations, it can be tricky,

```
> library(parallel)
> set.seed(123)
> mclapply(X = 1:10,
  FUN = function(x) runif
    (1), mc.cores = 2, mc
    .set.seed = TRUE)
[[1]]
[1] 0.5607899

[[2]]
[1] 0.4849712
```

```
> set.seed(123)
> mclapply(X = 1:10,
  FUN = function(x) runif
    (1), mc.cores = 2, mc
    .set.seed = TRUE)
 [[1]]
[1] 0.216951

[[2]]
[1] 0.1005274
```

How to replicate randomness?

There are (at least) two generators of random numbers in SAS see sas.com

- Fishman and Moore (1982) used for function RANUNI
- Mersenne-Twister used for the RAND function, based on Matsumoto and Nishimura (1997)

For instance, with SAS, to generate a random sequence, use

```
DATA FRANUNI (KEEP = x) ;
seed = 123 ;
DO REP = 1 TO 10;
  CALL RANUNI(seed, x);
  OUTPUT;
END;
RUN;
PROC PRINT DATA = FRANUNI ;
RUN ;
```

Obs	REP	X
1	1	0.75040
2	2	0.32091
3	3	0.17839
4	4	0.90603
5	5	0.35712
6	6	0.22111
7	7	0.78644
8	8	0.39808
9	9	0.12467
10	10	0.18769

How to replicate randomness?

```
> library(randtoolbox)
> setSeed(123)
> (U = congruRand(n=10, dim = 1, mod = 2^31-1, mult =
    397204094))
 [1] 0.7503961 0.3209120 0.1783896 0.9060334 0.3571171
 [6] 0.2211140 0.7864383 0.3980819 0.1246652 0.1876858
```

To replicate Tufféry (2013), use

```
> url="http://freakonometrics.free.fr/german_credit.
    csv"
> credit = read.csv(url, header = TRUE, sep = ",")
> index = sort(which(rank(U)<=644))
> table(credit$class[index])
  0   1
451 193
```

The training sample will be based on the 644 observations, and the remaining 356 will be used as validation sample

```
> train.db = credit[index,]
> valid.db = credit[-index,]
```

Monte Carlo

From the law of large numbers, if $U_1, U_2, \cdots$ is a sequence of i.i.d random variables, uniformly distributed on $[0,1]$, and some mapping $h : [0,1] \to \mathbb{R}$,

$$\frac{1}{n}\sum_{i=1}^{n} h(U_i) \xrightarrow{\text{a.s.}} \mu = \int_{[0,1]} h(u)\,\mathrm{d}u = \mathbb{E}[h(U)], \text{ as } n \to \infty$$

and from the central limit theorem

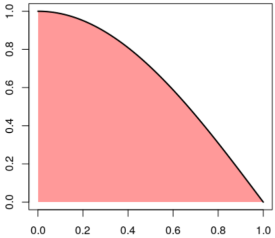
$$\sqrt{n}\left(\left(\frac{1}{n}\sum_{i=1}^{n} h(U_i)\right) - \mu\right) \xrightarrow{\mathcal{L}} \mathcal{N}\left(0, \sigma^2\right)$$

where $\sigma^2 = \text{Var}[h(U)]$, and $U \sim \mathcal{U}_{[0,1]}$.

Monte Carlo

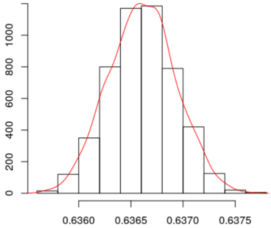
Consider $h(u) = \cos(\pi u/2)$,

```
> h=function(u) cos(u*pi/2)
> integrate(h,0,1)
0.6366198 with absolute error<7.1e
    -15
> mean(h(runif(1e6)))
[1] 0.6363378
```



We can actually repeat that a thousand time

```
> M=rep(NA,1000)
> for(i in 1:1000) M[i]=mean(h(runif
    (1e6)))
> mean(M)
[1] 0.6366087
> sd(M)
[1] 0.000317656
```



Which sum? (on importance sampling)

If $X \sim f$, $\mathbb{E}[h(X)] = \int h(x)f(x)dx$

If $\{x_1, \cdots, x_n\}$, $\widehat{s}_n = \frac{1}{n}\sum_{i=1}^{n} h(x_i)$, from the law of large numbers

$$\widehat{s}_n = \frac{1}{n}\sum_{i=1}^{n} h(x_i) \to \mathbb{E}[\widehat{s}_n] = \int h(x)f(x)dx, \quad \text{with } x_i \sim f$$

and the precision is given by the variance

$$\text{Var}[\widehat{s}_n] = \frac{1}{n}\text{Var}[h(X)]$$

$\to$ proxy of the numerical error approximation

Which sum? (on importance sampling)

Importance sampling is based on the property

$$h(x)f(x) = \frac{h(x)f(x)}{g(x)}g(x) = \widetilde{h}(x)g(x)$$

and consider

$$\widetilde{s}_n = \frac{1}{n}\sum_{i=1}^{n}\widetilde{h}(y_i) \to \int h(x)f(x)dx = \int \widetilde{h}(y)g(y)dy, \quad \text{with } y_i \sim g$$

which could have a lower variance

see https://ewfrees.github.io/Loss-Data-Analytics/ to go further

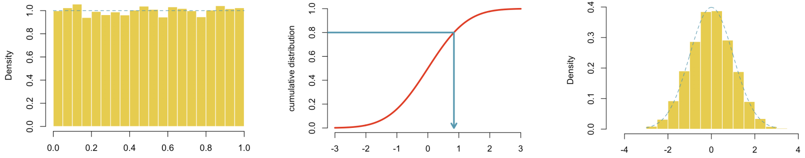
A probabilistic result

If F is a cdf, and if $U \sim \mathcal{U}([0,1])$, $X = F^{-1}(U)$ has cdf F
(see inverse method sampling)

Proof: Let $x \in \mathbb{R}$, $\mathbb{P}[X \leq x]$ is equal to

$$\mathbb{P}[F^{-1}(U) \leq x] = \mathbb{P}[F\big(F^{-1}(U)\big) \leq F\big(x\big)] = \mathbb{P}[U \leq F\big(x\big)] = F(x)$$

where $F^{-1}(u) = \inf\,\{x \mid F(x) \geq u\}$ for $u \in (0,1)$.



A probabilistic result

```
> U = runif(100)
 [1] 0.27 0.37 0.57 0.91 0.20 0.90 0.94 0.66 0.63 0.06
[11] 0.21 0.18 0.69 0.38 0.77 0.50 0.72 0.99 0.38 0.78
[21] 0.93 0.21 0.65 0.13 0.27 0.39 0.01 0.38 0.87 0.34
[31] 0.48 0.60 0.49 0.19 0.83 0.67 0.79 0.11 0.72 0.41
[41] 0.82 0.65 0.78 0.55 0.53 0.79 0.02 0.48 0.73 0.69
[51] 0.48 0.86 0.44 0.24 0.07 0.10 0.32 0.52 0.66 0.41
[61] 0.91 0.29 0.46 0.33 0.65 0.26 0.48 0.77 0.08 0.88
[71] 0.34 0.84 0.35 0.33 0.48 0.89 0.86 0.39 0.78 0.96
```

```
> Q(U)
 [1] -0.63 -0.33  0.18  1.33 -0.84  1.27  1.60  0.41
 [9]  0.33 -1.54 -0.82 -0.93  0.49 -0.29  0.74 -0.01
[17]  0.58  2.40 -0.31  0.76  1.51 -0.80  0.39 -1.15
[25] -0.62 -0.29 -2.21 -0.30  1.12 -0.41 -0.04  0.25
[33] -0.02 -0.89  0.94  0.44  0.82 -1.24  0.59 -0.22
[41]  0.92  0.38  0.78  0.13  0.07  0.80 -1.99 -0.06
[49]  0.62  0.50 -0.06  1.09 -0.16 -0.69 -1.47 -1.28
[57] -0.48  0.05  0.42 -0.24  1.36 -0.54 -0.10 -0.43
[65]  0.39 -0.65 -0.05  0.73 -1.38  1.15 -0.41  0.99
```

Notations & Results

Given a sample $\{x_1, \cdots, x_n\}$ i.i.d. from F,

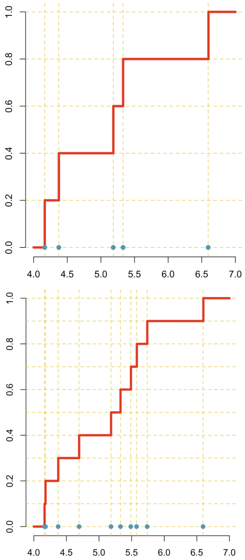
$$F(x) = \mathbb{P}[X \leq x],$$

the empirical cumulative distribution function is

$$\widehat{F}_n(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(x_i \leq x),\ \ x \in \mathbb{R}$$

Glivenko-Cantelli: $\widehat{F}_n \rightarrow F$ as $n \rightarrow \infty$,
or more precisely, almost surely

$$\|\widehat{F}_n - F\|_\infty = \sup_{x\in\mathbb{R}} |\widehat{F}_n(x) - F(x)| \longrightarrow 0$$



A probabilistic result

The inverse method with $\widehat{F}_n$ simply means resampling within $\{x_1, \cdots, x_n\}$ with equal probabilities $1/n$ (or *with replacement*)

```
> x
[1] 4.164 4.374 5.184 5.330 6.595
> Qemp(U)
 [1] 6.60 6.60 6.60 5.33 4.37 5.33 5.33 4.16 6.60 5.33
[11] 4.37 4.37 4.37 6.60 5.33 5.18 5.33 5.18 6.60 5.18
[21] 5.18 4.37 6.60 4.37 4.16 6.60 4.16 6.60 5.33 4.16
[31] 4.16 6.60 4.37 4.37 5.33 5.18 5.18 5.18 5.33 5.33
[41] 4.37 5.18 5.33 5.18 4.37 5.18 5.18 5.18 5.33 5.18
[51] 5.33 4.37 4.37 4.16 5.18 5.18 5.18 5.18 4.16 5.18
[61] 4.37 4.16 4.16 4.16 6.60 4.37 4.37 5.33 5.18 4.16
[71] 5.33 4.16 6.60 5.18 4.16 4.16 5.18 4.16 5.18 4.16
```

called bootstrapping

Bootstrap

Real World:

- distribution F
- data $\{x_1, \cdots, x_n\}$, i.i.d., F
- empirical distribution $\widehat{F}_n$
- parameter $\theta = t(F)$
- estimate $\widehat{\theta}_n = t(\widehat{F}_n)$
- error $\widehat{\theta}_n - \theta$
- standardized error $\dfrac{\widehat{\theta}_n - \theta}{s(\widehat{F}_n)}$

Bootstrap World ($\star$):

- distribution $\widehat{F}_n$
- data $\{x_1^\star, \cdots, x_n^\star\}$, i.i.d., $\widehat{F}_n$
- empirical distribution $\widehat{F}_n^\star$
- parameter $\widehat{\theta}_n = t(\widehat{F}_n)$
- estimate $\widehat{\theta}_n^\star = t(\widehat{F}_n^\star)$
- error $\widehat{\theta}_n^\star - \widehat{\theta}_n$
- standardized error $\dfrac{\widehat{\theta}_n^\star - \widehat{\theta}_n}{s(\widehat{F}_n^\star)}$

The sampling distribution of $\widehat{\theta}_n$ depends on (unknown) F
Use $\widehat{F}_n$ as a proxy for F: we cannot resample from F, but we can from $\widehat{F}_n$

Bootstrap

Example: mean, $\theta = t(F) = \int x dF(x)$

$$\widehat{\theta}_n = t(\widehat{F}_n) = \int x d\widehat{F}_n(x) = \frac{1}{n}\sum_{i=1}^n x_i$$

Example: variance, $\theta = t(F) = \int x^2 dF(x) - \left(\int x dF(x)\right)^2$

$$\widehat{\theta}_n = t(\widehat{F}_n) =$$

$$\int x^2 d\widehat{F}_n(x) - \left(\int x d\widehat{F}_n(x)\right) = \frac{1}{n}\sum_{i=1}^n x_i^2 - \left(\frac{1}{n}\sum_{i=1}^n x_i\right)^2$$

Bootstrap & Bias Estimation

Consider some statistic $\widehat{\theta}(\boldsymbol{y})$ (define on a sample $\boldsymbol{y}$).

$$\hat{\theta}_B = \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}_{(b)} \text{ where } \hat{\theta}_{(b)} = \hat{\theta}(\boldsymbol{y}^{(b)})$$

Recall that $\text{Bias}[\hat{\boldsymbol{\theta}}] = \mathbb{E}[\hat{\boldsymbol{\theta}}] - \theta$

the bootstrap estimate of the bias of the estimator $\hat{\boldsymbol{\theta}}$ is obtained by replacing $\mathbb{E}[\hat{\boldsymbol{\theta}}]$ with $\hat{\theta}_B$ and θ with $\hat{\theta}$:

$$\text{Bias}_{\text{bs}}[\hat{\boldsymbol{\theta}}] = \hat{\theta}_B - \hat{\theta}$$

Then, since $\theta = E[\hat{\boldsymbol{\theta}}] - \text{Bias}[\hat{\boldsymbol{\theta}}]$, the bootstrap bias corrected estimate is

$$\hat{\theta}_{\text{bs}} = \hat{\theta} - \text{Bias}_{\text{bs}}[\hat{\boldsymbol{\theta}}] = \hat{\theta} - (\hat{\theta}_B - \hat{\theta}) = 2\hat{\theta} - \hat{\theta}_B$$

Bootstrap & Bias Estimation

Example: $\displaystyle\frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})^2$ is a biased estimate of $\sigma^2=\text{Var}[X]$

```
> n = 20
> data = rnorm(n, 0, 1)
> set.seed(123)
> variance = sum((data - mean(data))^2)/n
> boot_vars = rapply(as.list(1:999), function(x) {
+   data_b = sample(data, n, replace=T)
+   sum((data_b - mean(data_b))^2)/n
+ })
> mean(boot_vars) - variance
[1] -0.05576993
```

where the true value is $-n^{-1}\sigma^2$ (here -0.05).

Bootstrap & Bias Estimation

Consider X with mean $\mu = \mathbb{E}(X)$. Let $\theta = \exp[\mu]$, then $\widehat{\theta} = \exp[\overline{x}]$ is a biased estimator of θ, see Horowitz (1998)

Idea 1 : Asymptotic Approximation, i.e. if $\sqrt{n}[\widehat{\tau}_n - \tau] \xrightarrow{\mathcal{L}} \mathcal{N}(0, \sigma^2)$, then, if $g'(\tau)$ exists and is non-null,

$$\sqrt{n}[g(\widehat{\tau}_n) - g(\tau)] \xrightarrow{\mathcal{L}} \mathcal{N}(0, \sigma^2[g'(\tau)]^2)$$

so $\widehat{\theta}_1 = \exp[\overline{x}]$ is asymptotically unbiased.

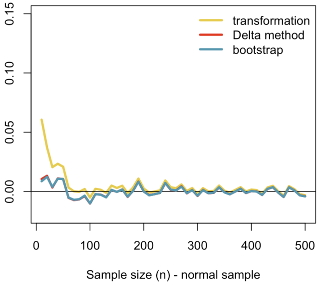
Idea 2 : Delta Method correction,

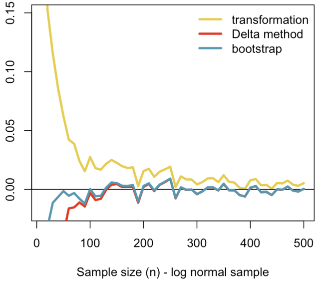
based on $\widehat{\theta}_2 = \exp\left[\overline{x} - \frac{s^2}{2n}\right]$ where $s^2 = \dfrac{1}{n}\displaystyle\sum_{i=1}^{n}[x_i - \overline{x}]^2$.

Idea 3 : Use Bootstrap, $\widehat{\theta}_3 = \dfrac{1}{B}\displaystyle\sum_{b=1}^{B}\exp[\overline{x}^{(b)}]$

Bootstrap & Bias Estimation

```
simu=function(n = 10){
    get_i = function(i){
    x = rnlorm(n,sd=1)-exp(1/2);
    S = matrix(sample(x, size=n*500, replace=TRUE),ncol=500)
    ThetaBoot = exp(colMeans(S))
    Bias = mean(ThetaBoot)-exp(mean(x))
    theta=exp(mean(x))/exp(.5*var(x)/n)
    c(exp(mean(x)),exp(mean(x))-Bias, theta)
    }
    res = lapply(1:500, get_i)
    res = do.call(rbind, res)
    bias = colMeans(res-1)
    return(bias)
 }
```

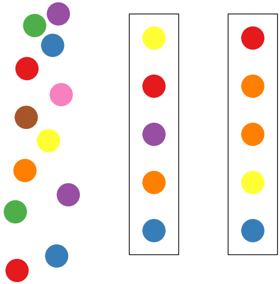




Régression Linéaire & Bootstrap (1)

Dataset $\{\boldsymbol{z}_i = (y_i, \boldsymbol{x}_i)\}$, $i = 1, \cdots, n$.
Use paired sampling by (repeatedly) resampling $\{\boldsymbol{z}_1^\star, \cdots, \boldsymbol{z}_n^\star\}$.

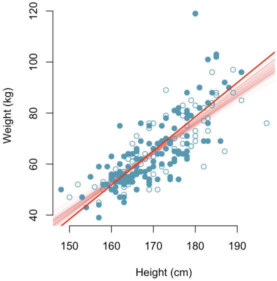
Idea :
$\{(y_i, \boldsymbol{x}_i)\}$ is obtained from (unknown) $\mathbb{P}$
Based on n observations, we observe $\mathbb{P}_n$
We generate other samples by resampling from $\mathbb{P}_n$

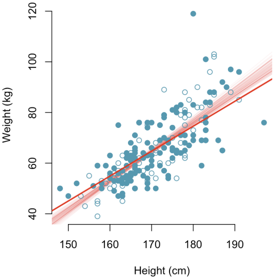


1. sample $\{i_1^{(b)}, \cdots, i_n^{(b)}\}$ randomly with replacement in $\{1, 2, \cdots, n\}$
2. consider dataset $(\boldsymbol{x}_i^{(b)}, y_i^{(b)}) = (\boldsymbol{x}_{i^{(b)}}, y_{i^{(b)}})$'s, and fit a model
3. let $\widehat{\boldsymbol{\beta}}^{(b)}$ denote the estimated values, or $\widehat{y}_{n+1}^{(b)}$ some prediction

Régression Linéaire & Bootstrap (1)

```
> BETA = matrix(NA,1000,2)
> for(s in 1:1000){
  idx = sample(1:nrow(Davis),nrow(
    Davis),replace=TRUE)
  reg_sim = lm(weight~height, data=
    Davis[idx,])
  BETA[s,] = reg_sim$coefficients
}
> hist(BETA[,1])
> hist(BETA[,2])
```

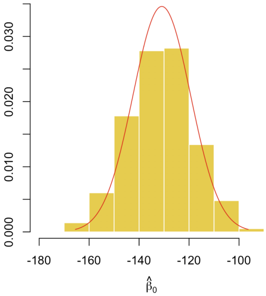


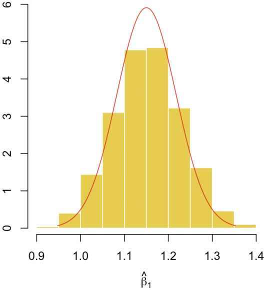


Régression Linéaire & Bootstrap (1)

Alternative code

```
> library(boot)
> coef = function(formula, data,
    indices) {
  d = data[indices,]
  fit = lm(formula, data=d)
  return(coef(fit))
}
> results = boot(data=Davis, statistic
    =coef, R=1000, formula=weight~
    height)
> plot(results, index=1)
> plot(results, index=2)
```





Régression Linéaire & Bootstrap (2)

As an alternative model-based resampling

1. sample $\widehat{\varepsilon}_1^{(b)},\cdots,\widehat{\varepsilon}_n^{(b)}$ resample from $\{\widehat{\varepsilon}_1,\widehat{\varepsilon}_2,\cdots,\widehat{\varepsilon}_n\}$
2. set $y_i^{(b)}=\widehat{\beta}_0+\widehat{\beta}_1x_i+\widehat{\varepsilon}_i^{(b)}=\widehat{y}_i+\widehat{\varepsilon}_i^{(b)}$
3. consider dataset $(\boldsymbol{x},y^{(b)})=(\boldsymbol{x}_i,y_i^{(b)})$'s and fit a model
4. let $\widehat{\boldsymbol{\beta}}^{(b)}$ denote estimated values

Note in a simple regression

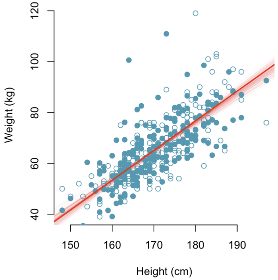
$$\widehat{\beta}_1^{(b)}=\frac{\sum[x_i-\overline{x}]\cdot y_i^{(b)}}{\sum[x_i-\overline{x}]^2}=\widehat{\beta}_1+\frac{\sum[x_i-\overline{x}]\cdot\widehat{\varepsilon}_i^{(b)}}{\sum[x_i-\overline{x}]^2}$$

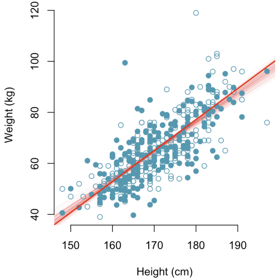
hence $\mathbb{E}[\widehat{\beta}_1^{(b)}]=\widehat{\beta}_1$, while

$$\text{Var}[\widehat{\beta}_1^{(b)}]=\frac{\sum[x_i-\overline{x}]^2\cdot\text{Var}[\widehat{\varepsilon}_i^{(b)}]}{\left(\sum[x_i-\overline{x}]^2\right)^2}\sim\frac{\sigma^2}{\sum[x_i-\overline{x}]^2}$$

Régression Linéaire & Bootstrap (2)

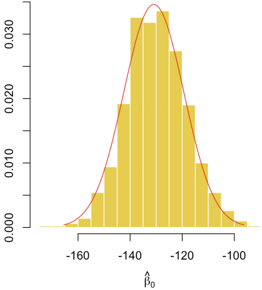
```
> BETA = matrix(NA,1000,2)
> reg = lm(weight~height, data=Davis)
> epsilon = residuals(reg)
> for(s in 1:1000){
  eps = sample(epsilon,nrow(Davis),
    replace=TRUE)
  Davis_s = data.frame(height =
    Davis$height, weight =predict(reg)
    +eps)
  reg_sim = lm(weight~height, data=
    Davis_s)
  BETA[s,] = reg_sim$coefficients
}
> hist(BETA[,1])
> hist(BETA[,2])
```

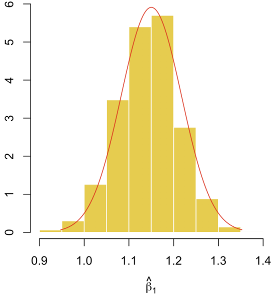




Régression Linéaire & Bootstrap (2)

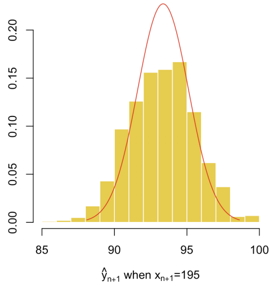
```
> BETA = matrix(NA,1000,2)
> reg = lm(weight~height, data=Davis)
> epsilon = residuals(reg)
> for(s in 1:1000){
  eps = sample(epsilon,nrow(Davis),
    replace=TRUE)
  Davis_s = data.frame(height =
    Davis$height, weight =predict(reg)
    +eps)
  reg_sim = lm(weight~height, data=
    Davis_s)
  BETA[s,] = reg_sim$coefficients
}
> hist(BETA[,1])
> hist(BETA[,2])
```

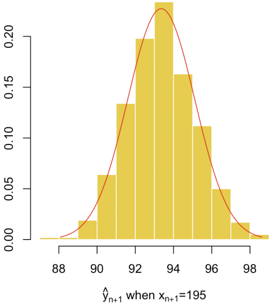




Régression Linéaire & Bootstrap (1/2)

```
> PRED = matrix(NA,1000,2)
> nwDavis = data.frame(height = 195)
> for(s in 1:1000){
+    idx = sample(1:n,n,replace=TRUE)
+    reg_sim = lm(weight~height, data=
   Davis[idx,])
+    PRED[s,1] = predict(reg_sim,
   newdata=nwDavis)
+    eps = sample(epsilon,nrow(Davis),
   replace=TRUE)
+    Davis_s = data.frame(height =
   Davis$height, weight =predict(reg)
   +eps)
+    reg_sim = lm(weight~height, data=
   Davis_s)
+    PRED[s,2] = predict(reg_sim,
   newdata=nwDavis)
+ }
```

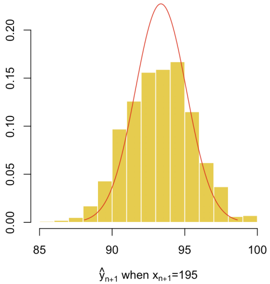


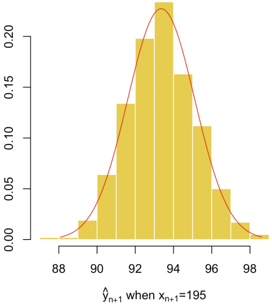


Régression Linéaire & Bootstrap (1/2)

```
> apply(PRED,2,function(x) quantile(x,.025))
[1] 89.04203 89.63030
> apply(PRED,2,function(x) quantile(x,.975))
[1] 97.60345 97.02423
> predict(lm(weight~height, data=Davis), newdata=nwDavis,interval="confidence",se.fit = TRUE)
$fit
       fit      lwr      upr
1 93.35749 89.89571 96.81927

$se.fit
[1] 1.755451
```





Bootstrap heuristics

Here $\widehat{\boldsymbol{\beta}} = \boldsymbol{\beta} + (\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{\varepsilon} = T_\beta(\boldsymbol{\varepsilon})$ or $\widehat{y}_{n+1} = \boldsymbol{x}'_{n+1}\widehat{\boldsymbol{\beta}} = T_y(\boldsymbol{\varepsilon})$

Use simulations, we draw n values $\{\epsilon_1, \cdots, \epsilon_n\}$ and

- $\mathbb{E}\left[\dfrac{1}{n}\displaystyle\sum_{i=1}^n T(\epsilon_i)\right] = \mathbb{E}[T(\varepsilon)]$ (unbiased)
- $\dfrac{1}{n}\displaystyle\sum_{i=1}^n T(\epsilon_i) \xrightarrow{\mathcal{L}} \mathbb{E}[T(\varepsilon)]$ as $n \to \infty$ (consistent)

Bootstrap & Tests

Consider the test of $H_0 : \beta_j = 0$,

1. compute $t_n = \dfrac{(\widehat{\beta}_j - \beta_j)^2}{\widehat{\sigma}_j^2}$
2. generate B boostrap samples, under the null assumption H_0
3. for each boostrap sample, compute $t_n^{(b)} = \dfrac{(\widehat{\beta}_j^{(b)} - \widehat{\beta}_j)^2}{\widehat{\sigma}_j^{2(b)}}$
4. reject H_0 if $\displaystyle\frac{1}{B}\sum_{i=1}^{B} \mathbf{1}(t_n > t_n^{(b)}) < \alpha$.

Bootstrap & Tests

What does "generate B boostrap samples, under the null assumption H_0" mean ?

Example : $y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$ with $H_0 : \beta_1 = 0$.

2.1. Estimate the model under H_0, i.e. $y_i = \beta_0 + \eta_i$, and save $\{\widehat{\eta}_1, \cdots, \widehat{\eta}_n\}$

2.2. Define $\widetilde{\boldsymbol{\eta}} = \{\widetilde{\eta}_1, \cdots, \widetilde{\eta}_n\}$ with $\widetilde{\eta} = \sqrt{\dfrac{n}{n-1}}\widehat{\eta}$

2.3. Draw (with replacement) residuals $\widetilde{\boldsymbol{\eta}}^{(b)} = \{\widetilde{\eta}_1^{(b)}, \cdots, \widetilde{\eta}_n^{(b)}\}$

2.4. Set $y_i^{(b)} = \widehat{\beta}_0 + \widetilde{\eta}_i^{(b)}$

2.5. Estimate the regression model $y_i^{(b)} = \beta_0^{(b)} + \beta_1^{(b)} x_i + \varepsilon_i^{(b)}$

3. for each boostrap sample, compute $t_n^{(b)} = \dfrac{(\widehat{\beta}_j^{(b)} - \widehat{\beta}_j)^2}{\widehat{\sigma}_j^{2(b)}}$

4. reject H_0 if $\dfrac{1}{B}\displaystyle\sum_{i=1}^{B} \mathbf{1}(t_n > t_n^{(b)}) < \alpha$.