

IntRoduction générale

Techniques avancées en programmation statistique R

Patrick Fournier

Automne 2020

Université du Québec à Montréal

Un peu d'histoire

~→ Fondé en 1925 par la American Telephone & Telegraph Company (AT&T).

Bell labs

- ~> Fondé en 1925 par la American Telephone & Telegraph Company (AT&T).
- ~> Mecque du développement technologique au 20 siècle.

Bell labs

- ~> Fondé en 1925 par la American Telephone & Telegraph Company (AT&T).
- ~> Mecque du développement technologique au 20 siècle.
- ~> 9 prix nobels et 4 prix Turing [2].

Bell labs

- ~> Fondé en 1925 par la American Telephone & Telegraph Company (AT&T).
- ~> Mecque du développement technologique au 20 siècle.
- ~> 9 prix nobels et 4 prix Turing [2].
- ~> Y ont été développés (entre autres) [8]

- ~> Fondé en 1925 par la American Telephone & Telegraph Company (AT&T).
- ~> Mecque du développement technologique au 20 siècle.
- ~> 9 prix nobels et 4 prix Turing [2].
- ~> Y ont été développés (entre autres) [8]
 - ~> ordinateur binaire
(1939)

Bell labs

- ~> Fondé en 1925 par la American Telephone & Telegraph Company (AT&T).
- ~> Mecque du développement technologique au 20 siècle.
- ~> 9 prix nobels et 4 prix Turing [2].
- ~> Y ont été développés (entre autres) [8]
 - ~> ordinateur binaire (1939)
 - ~> transistor (1947)

- ~> Fondé en 1925 par la American Telephone & Telegraph Company (AT&T).
- ~> Mecque du développement technologique au 20 siècle.
- ~> 9 prix nobels et 4 prix Turing [2].
- ~> Y ont été développés (entre autres) [8]
 - ~> ordinateur binaire (1939)
 - ~> transistor (1947)
 - ~> téléphonie cellulaire (1947)

- ~> Fondé en 1925 par la American Telephone & Telegraph Company (AT&T).
- ~> Mecque du développement technologique au 20 siècle.
- ~> 9 prix nobels et 4 prix Turing [2].
- ~> Y ont été développés (entre autres) [8]
 - ~> ordinateur binaire (1939) (1954)
 - ~> transistor (1947)
 - ~> téléphonie cellulaire (1947)
 - ~> cellule photovoltaïque

Bell labs

- ~> Fondé en 1925 par la American Telephone & Telegraph Company (AT&T).
- ~> Mecque du développement technologique au 20 siècle.
- ~> 9 prix nobels et 4 prix Turing [2].
- ~> Y ont été développés (entre autres) [8]
 - ~> ordinateur binaire (1939) (1954)
 - ~> transistor (1947)
 - ~> téléphonie cellulaire (1947)
 - ~> cellule photovoltaïque
 - ~> laser (1957)

- ~> Fondé en 1925 par la American Telephone & Telegraph Company (AT&T).
- ~> Mecque du développement technologique au 20 siècle.
- ~> 9 prix nobels et 4 prix Turing [2].
- ~> Y ont été développés (entre autres) [8]
 - ~> ordinateur binaire (1939)
 - ~> transistor (1947)
 - ~> téléphonie cellulaire (1947)
 - ~> cellule photovoltaïque
 - ~> laser (1957)
 - ~> satellite de communication (1962)

- ~> Fondé en 1925 par la American Telephone & Telegraph Company (AT&T).
- ~> Mecque du développement technologique au 20 siècle.
- ~> 9 prix nobels et 4 prix Turing [2].
- ~> Y ont été développés (entre autres) [8]
 - ~> ordinateur binaire (1939)
 - ~> transistor (1947)
 - ~> téléphonie cellulaire (1947)
 - ~> cellule photovoltaïque (1954)
 - ~> laser (1957)
 - ~> satellite de communication (1962)
 - ~> Unix, C (1969 - 1972)

Bell labs

- ~> Fondé en 1925 par la American Telephone & Telegraph Company (AT&T).
- ~> Mecque du développement technologique au 20 siècle.
- ~> 9 prix nobels et 4 prix Turing [2].
- ~> Y ont été développés (entre autres) [8]
 - ~> ordinateur binaire (1939)
 - ~> transistor (1947)
 - ~> téléphonie cellulaire (1947)
 - ~> cellule photovoltaïque
 - (1954)
 - ~> laser (1957)
 - ~> satellite de communication (1962)
 - ~> Unix, C (1969 - 1972)
 - ~> C++ (Années 1980)

~→ Inventé par John Chambers au sein des Bell labs dans la seconde moitié des années '70.

- ~> Inventé par John Chambers au sein des Bell labs dans la seconde moitié des années '70.
- ~> Développé spécifiquement à des fins de programmation statistique.

- ~> Inventé par John Chambers au sein des Bell labs dans la seconde moitié des années '70.
- ~> Développé spécifiquement à des fins de programmation statistique.
 - ~> Auparavant, les procédures étaient implémentées en Fortran.

- ~> Inventé par John Chambers au sein des Bell labs dans la seconde moitié des années '70.
- ~> Développé spécifiquement à des fins de programmation statistique.
 - ~> Auparavant, les procédures étaient implémentées en Fortran.
- ~> Objectif [4] : “to turn ideas into software, quickly and faithfully.”

```
1      CALL FIT B(TAG C)
2  C INITIAL ESTIMATES OF PARAMETERS
3  100  IF(.NOT.FIT AQ(TAG C)) GO TO 150
4  C TEST FOR ASSESSMENT ON THIS STEP
5      CALL FIT A(TAG C)
6  C REPORT ON CURRENT MODEL
7  150  IF(.NOT.FIT CQ(TAG C)) GO TO 200
8  C TEST FOR COMPLETION OF ITERATION
9      CALL FIT C(TAG C)
10 C FINAL REPORT
11     RETURN
12 200  CALL FIT S(TAG C)
13 C TAKE NEXT STEP IN ITERATION
14     GO TO 100
```

Listing 1 : Sous-programme d'optimisation du système FIT, John Chambers, 1969 [3].

Qu'est-ce qu'un langage de programmation ?



Figure 1 : Platon et Aristote, probablement en train de débattre du meilleur langage de programmation entre R et SAS, Raphaël, *L'École d'Athènes*, 1512.

Langage machine

↪ Couture à l'aiguille.

Langage machine

- ↪ Couture à l'aiguille.
- ↪ Directement exécuté par la machine \Rightarrow *aucune* abstraction.

Langage machine

- ↪ Couture à l'aiguille.
- ↪ Directement exécuté par la machine ⇒ *aucune* abstraction.
- ↪ *Très* difficile à comprendre pour un humain.

Langage machine

- ↪ Couture à l'aiguille.
- ↪ Directement exécuté par la machine ⇒ *aucune* abstraction.
- ↪ *Très* difficile à comprendre pour un humain.

Exemple : Calcul d'un nombre de Fibonacci sur x86 [6] :

```
8B542408 83FA0077 06B80000 0000C383
FA027706 B8010000 00C353BB 01000000
B9010000 008D0419 83FA0376 078BD989
C14AEBF1 5BC3
```

Langage assembleur

~→ Couture à l'aiguille, mais avec un dé à coudre.

Langage assembleur

- ~> Couture à l'aiguille, mais avec un dé à coudre.
- ~> Combinaisons de bits représentés par des *symboles* ⇒ très légère abstraction.

Langage assembleur

- ↪ Couture à l'aiguille, mais avec un dé à coudre.
- ↪ Combinaisons de bits représentés par des *symboles* ⇒ très légère abstraction.
- ↪ Très près du langage machine, mais “compréhensible” par un “humain”.

Langage assembleur

- ↪ Couture à l'aiguille, mais avec un dé à coudre.
- ↪ Combinaisons de bits représentés par des *symboles* ⇒ très légère abstraction.
- ↪ Très près du langage machine, mais “compréhensible” par un “humain”.
- ↪ Exemples : voir `assembleur.ipynb`

Langage de bas niveau

↪ Machine à coudre.

Langage de bas niveau

↪ Machine à coudre.

↪ Relativement éloigné du langage machine \Rightarrow bon niveau d'abstraction.

Langage de bas niveau

- ↪ Machine à coudre.
- ↪ Relativement éloigné du langage machine \Rightarrow bon niveau d'abstraction.
- ↪ Nécessite une véritable phase de compilation.

Langage de bas niveau

- ↪ Machine à coudre.
- ↪ Relativement éloigné du langage machine \Rightarrow bon niveau d'abstraction.
 - ↪ Nécessite une véritable phase de compilation.
- ↪ Relativement près du langage naturel tout en reposant sur des opérations de bas niveau (ex. arithmétique des pointeurs).

Langage de bas niveau

- ↪ Machine à coudre.
- ↪ Relativement éloigné du langage machine \Rightarrow bon niveau d'abstraction.
 - ↪ Nécessite une véritable phase de compilation.
- ↪ Relativement près du langage naturel tout en reposant sur des opérations de bas niveau (ex. arithmétique des pointeurs).
- ↪ Exemples : voir `c.ipynb`

Langage de haut niveau

↪ Machine à coudre électrique.

Langage de haut niveau

- ~> Machine à coudre électrique.
- ~> Éloigné du langage machine, voir n'a presque aucun lien avec celui-ci.

Langage de haut niveau

- ~> Machine à coudre électrique.
- ~> Éloigné du langage machine, voir n'a presque aucun lien avec celui-ci.
- ~> Conçu pour être indépendant de l'architecture sur lequel il est implémenté et faciliter la vie du programmeur.

Langage de haut niveau

Exemple : Addition de polynômes en Haskell [1] :

```
1  type Poly = [(Int,Int)]
2
3  addPoly :: Poly -> Poly -> Poly
4  addPoly [] ys = ys
5  addPoly xs [] = xs
6  addPoly ((a,b):xs) ((c,d):ys)
7      | a == c = ((a,b+d):(addPoly xs ys))
8      | a < c = ((a,b):(addPoly xs ((c,d):ys)))
9      | a > c = ((c,d):(addPoly ((a,b):xs) ys))
```

Langage : compilation vs. interprétation

~→ Ordinateur \neq magie ; ultimement, machine qui n'accomplit qu'une unique tâche : *exécution de code machine*.

Langage : compilation vs. interprétation

- ~> Ordinateur \neq magie; ultimement, machine qui n'accomplit qu'une unique tâche : *exécution de code machine*.
- ~> Tout langage autre ne peut être exécuté par un ordinateur
⇒ *compilation* ou *interprétation*.

Langage : compilation vs. interprétation

- ~> Ordinateur \neq magie ; ultimement, machine qui n'accomplit qu'une unique tâche : *exécution de code machine*.
- ~> Tout langage autre ne peut être exécuté par un ordinateur
 \Rightarrow *compilation* ou *interprétation*.

Compilation

“Traduction” en langage machine (éventuellement, optimisations).

Langage : compilation vs. interprétation

- ~> Ordinateur \neq magie; ultimement, machine qui n'accomplit qu'une unique tâche : *exécution de code machine*.
- ~> Tout langage autre ne peut être exécuté par un ordinateur
 \Rightarrow *compilation* ou *interprétation*.

Compilation

“Traduction” en langage machine (éventuellement, optimisations).

Interprétation

Exécution du code par un programme, l'*interpréteur*.

Language : spécification vs. implémentation

Qu'est-ce qu'un langage de programmation ?

Language : spécification vs. implémentation

Qu'est-ce qu'un langage de programmation ?

~→ Premiers ordinateurs modernes : années '40.

Language : spécification vs. implémentation

Qu'est-ce qu'un langage de programmation ?

~> Premiers ordinateurs modernes : années '40.

~> Contraintes de performances \Rightarrow programmes écrits en langage *assembleur* ou *machine*.

Language : spécification vs. implémentation

Qu'est-ce qu'un langage de programmation ?

- ~> Premiers ordinateurs modernes : années '40.
- ~> Contraintes de performances \Rightarrow programmes écrits en langage *assembleur* ou *machine*.
- ~> Spécification et implémentation se confondent.

Language : spécification vs. implémentation

Qu'est-ce qu'un langage de programmation ?

- ~> Premiers ordinateurs modernes : années '40.
 - ~> Contraintes de performances \Rightarrow programmes écrits en langage *assembleur* ou *machine*.
 - ~> Spécification et implémentation se confondent.
- ~> En 1948, Konrad Zuse publie une *spécification* d'un des premiers langage de plus haut niveau, le Plankalkül [10].

Language : spécification vs. implémentation

Qu'est-ce qu'un langage de programmation ?

- ~> Premiers ordinateurs modernes : années '40.
 - ~> Contraintes de performances \Rightarrow programmes écrits en langage *assembleur* ou *machine*.
 - ~> Spécification et implémentation se confondent.
- ~> En 1948, Konrad Zuse publie une *spécification* d'un des premiers langage de plus haut niveau, le Plankalkül [10].
 - ~> Malgré cela, le Plankalkül ne sera pas *implémenté* avant 1975, soit *27 ans plus tard* !

R



Qu'est-ce que R?

R est...

un langage de haut niveau

Implémente des *concepts mathématiques abstraits*; le programmeur ne se soucie pas de détails d'implémentation.

Qu'est-ce que R?

R est...

un langage de haut niveau

Implémente des *concepts mathématiques abstraits*; le programmeur ne se soucie pas de détails d'implémentation.

un langage interprété

Code exécuté par l'*interpréteur R*.

Qu'est-ce que R?

R est...

un langage de haut niveau

Implémente des *concepts mathématiques abstraits*; le programmeur ne se soucie pas de détails d'implémentation.

un langage interprété

Code exécuté par l'*interpréteur R*.

une implémentation de S [5]

Tout comme le “vieux S” et le “nouveau S”, R implémente les spécifications du langage S.

Origines et influences

~→ R est fortement influencé par une famille de langages extrêmement importante dans l'histoire de l'informatique : *Lisp*.

Origines et influences

- ~> R est fortement influencé par une famille de langages extrêmement importante dans l'histoire de l'informatique : *Lisp*.
- ~> Dialecte important de Lisp : *Common Lisp*.

Origines et influences

- ~> R est fortement influencé par une famille de langages extrêmement importante dans l'histoire de l'informatique : *Lisp*.
- ~> Dialecte important de Lisp : *Common Lisp*.
 - ~> Programmation orientée objet (CLOS).

Origines et influences

- ~> R est fortement influencé par une famille de langages extrêmement importante dans l'histoire de l'informatique : *Lisp*.
- ~> Dialecte important de Lisp : *Common Lisp*.
 - ~> Programmation orientée objet (CLOS).
 - ~> Réflectivité.

Origines et influences

- ~> R est fortement influencé par une famille de langages extrêmement importante dans l'histoire de l'informatique : *Lisp*.
- ~> Dialecte important de Lisp : *Common Lisp*.
 - ~> Programmation orientée objet (CLOS).
 - ~> Réflectivité.
 - ~> Programmation fonctionnelle (& array programming).

Origines et influences

- ~> R est fortement influencé par une famille de langages extrêmement importante dans l'histoire de l'informatique : *Lisp*.
- ~> Dialecte important de Lisp : *Common Lisp*.
 - ~> Programmation orientée objet (CLOS).
 - ~> Réflectivité.
 - ~> Programmation fonctionnelle (& array programming).
- ~> Influence plus récente : *XLispStat*; basé sur *XLisp* qui étend *Scheme*, un dialecte minimaliste de *Lisp*.

Origines et influences

- ~> R est fortement influencé par une famille de langages extrêmement importante dans l'histoire de l'informatique : *Lisp*.
- ~> Dialecte important de Lisp : *Common Lisp*.
 - ~> Programmation orientée objet (CLOS).
 - ~> Réflectivité.
 - ~> Programmation fonctionnelle (& array programming).
- ~> Influence plus récente : *XLispStat*; basé sur *XLisp* qui étend *Scheme*, un dialecte minimaliste de *Lisp*.
 - ~> Procédures statistiques de haut niveau (régressions, modèles linéaires généralisés, ...).

Origines et influences

- ~> R est fortement influencé par une famille de langages extrêmement importante dans l'histoire de l'informatique : *Lisp*.
- ~> Dialecte important de Lisp : *Common Lisp*.
 - ~> Programmation orientée objet (CLOS).
 - ~> Réflectivité.
 - ~> Programmation fonctionnelle (& array programming).
- ~> Influence plus récente : *XLispStat*; basé sur *XLisp* qui étend *Scheme*, un dialecte minimaliste de *Lisp*.
 - ~> Procédures statistiques de haut niveau (régressions, modèles linéaires généralisés, ...).
 - ~> Visualisation de données (statique et même dynamique, voir exemple).

Une influence importante : Lisp



Figure 2 : John McCarthy, concepteur de Lisp et pionnier de l'intelligence artificielle, \approx 1967.



Figure 3 : Lisp machine, ordinateur conçu pour l'exécution directe de code Lisp, années '70.

Une influence importante : XLispStat

```
1 (def h (histogram abrasion-loss))
2 (sequence-slider-dialog
3   (order hardness) :action
4   #'(lambda (i)
5     (send h :unselect-all-points)
6     (send h :point-selected i t)))
```

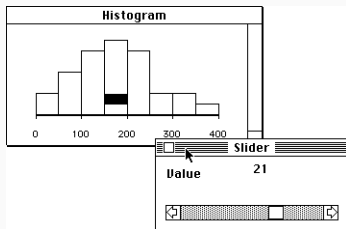


Figure 4 : Programme XLispStat et sa sortie graphique dynamique, Luke Tierney, 1989 [9].

R est multi-paradigmes :

Programmation impérative

Programme \simeq modifications successives de son propre état.

R est multi-paradigmes :

Programmation impérative

Programme \simeq modifications successives de son propre état.

Programmation procédurale

Possibilité de faire appel à des procédures (fonctions).

Paradigmes endossés par R

R est multi-paradigmes :

Programmation impérative

Programme \simeq modifications successives de son propre état.

Programmation procédurale

Possibilité de faire appel à des procédures (fonctions).

Programmation fonctionnelle (Purrr)

Programme \simeq application de fonctions.

Paradigmes endossés par R

R est multi-paradigmes :

Programmation impérative

Programme \simeq modifications successives de son propre état.

Programmation procédurale

Possibilité de faire appel à des procédures (fonctions).

Programmation fonctionnelle (Purrr)

Programme \simeq application de fonctions.

Programmation réflexive

Un programme peut examiner et modifier sa structure.

R est multi-paradigmes :

Programmation orientée objet (OOP)

Le concept d'objet joue un rôle central.

R est multi-paradigmes :

Programmation orientée objet (OOP)

Le concept d'objet joue un rôle central.

Array programming

Opérations sur des ensembles de valeurs.

Paradigmes endossés par R

R est multi-paradigmes :

Programmation orientée objet (OOP)

Le concept d'objet joue un rôle central.

Array programming

Opérations sur des ensembles de valeurs.

Programmation lettrée (Knitr & Sweave)

Explications du programmes données en conjonction avec le code source.

Exemple : impératif vs. déclaratif

Prolog (déclaratif)

```
1  car(X) :- toyota(X).
2  car(X) :- honda(X).
3
4  toyota(prius) :- true.
5  toyota(patrick) :- false.
6  honda(patrick) :- false.
7
8  humain(X) :- not(car(X)).
9
10 ?- car(prius).
11 true
12
13 ?- car(patrick).
14 false.
15
16 ?- humain(patrick).
17 true.
```

Exemple : impératif vs. déclaratif

R (impératif) Voir `r.ipynb`

Exemple : impératif vs. fonctionnel

Pascal (impératif)

```
1  Program autoCor1;
2  var
3      uniforms, uniforms1: array [1..10000] of Real;
4      kk: Integer;
5      avg, prod, sumSq, res: Real;
6  begin
7      avg := 0; prod := 0; sumSq := 0;
8      for kk := 1 to 10000 do uniforms[kk] := random;
9      for kk := 1 to 10000 do avg := avg + uniforms[kk] / 10000;
10     for kk := 1 to 10000 do uniforms[kk] := uniforms[kk] - avg;
11     for kk := 2 to 10000 do uniforms1[kk - 1] := uniforms[kk];
12     uniforms1[10000] := uniforms[1];
13
14     for kk := 1 to 10000 do
15         prod := prod + uniforms[kk] * uniforms1[kk];
16     for kk := 1 to 10000 do
17         sumSq := sumSq + uniforms[kk] * uniforms[kk];
18
19     res := prod / sumSq
20 end.
```


Exemple : impératif vs. fonctionnel

R (fonctionnel) Voir `r.ipynb`

Exemple : scalar vs. array

Pascal (scalar)

```
1  Program autoCor1;
2  var
3      uniforms, uniforms1: array [1..10000] of Real;
4      kk: Integer;
5      avg, prod, sumSq, res: Real;
6  begin
7      avg := 0; prod := 0; sumSq := 0;
8      for kk := 1 to 10000 do uniforms[kk] := random;
9      for kk := 1 to 10000 do avg := avg + uniforms[kk] / 10000;
10     for kk := 1 to 10000 do uniforms[kk] := uniforms[kk] - avg;
11     for kk := 2 to 10000 do uniforms1[kk - 1] := uniforms[kk];
12     uniforms1[10000] := uniforms[1];
13
14     for kk := 1 to 10000 do
15         prod := prod + uniforms[kk] * uniforms1[kk];
16     for kk := 1 to 10000 do
17         sumSq := sumSq + uniforms[kk] * uniforms[kk];
18
19     res := prod / sumSq
20 end.
```

Exemple : scalar vs. array

R (array) Voir `r.ipynb`

Mathématiques appliquées?

Définition

Utilisation des mathématiques pour résoudre des problèmes provenant d'autre domaines.

Définition

Utilisation des mathématiques pour résoudre des problèmes provenant d'autres domaines.

Discipline faisant face à un *ensemble distinct de problèmes*.

Définition

Utilisation des mathématiques pour résoudre des problèmes provenant d'autre domaines.

Discipline faisant face à un *ensemble distinct de problèmes*.

~> Contraintes d'implémentation.

Définition

Utilisation des mathématiques pour résoudre des problèmes provenant d'autres domaines.

Discipline faisant face à un *ensemble distinct de problèmes*.

↪ Contraintes d'implémentation.

↪ Précision

Définition

Utilisation des mathématiques pour résoudre des problèmes provenant d'autres domaines.

Discipline faisant face à un *ensemble distinct de problèmes*.

~> Contraintes d'implémentation.

~> Précision

~> Efficacité

Définition

Utilisation des mathématiques pour résoudre des problèmes provenant d'autres domaines.

Discipline faisant face à un *ensemble distinct de problèmes*.

↪ Contraintes d'implémentation.

↪ Précision

↪ Efficacité

↪ Temporelle

Définition

Utilisation des mathématiques pour résoudre des problèmes provenant d'autres domaines.

Discipline faisant face à un *ensemble distinct de problèmes*.

↪ Contraintes d'implémentation.

↪ Précision

↪ Efficacité

↪ Temporelle

↪ Spatiale

Exemple : régression linéaire

Contexte

Problème

Étant donné y_n : réponse aléatoire, $X_{n \times p}$: prédicteurs,
 $n > p$ trouver β tel que

$$y = X\beta$$

Exemple : régression linéaire

Contexte

Problème

Étant donné y_n : réponse aléatoire, $X_{n \times p}$: prédicteurs, $n > p$ trouver β tel que

$$y = X\beta$$

Habituellement, erreurs de mesure \Rightarrow système indéterminé.

Exemple : régression linéaire

Contexte

Problème

Étant donné y_n : réponse aléatoire, $X_{n \times p}$: prédicteurs, $n > p$ trouver β tel que

$$y = X\beta$$

Habituellement, erreurs de mesure \Rightarrow système indéterminé.

Outils

Exemple : régression linéaire

Contexte

Problème

Étant donné y_n : réponse aléatoire, $X_{n \times p}$: prédicteurs,
 $n > p$ trouver β tel que

$$y = X\beta$$

Habituellement, erreurs de mesure \Rightarrow système indéterminé.

Outils

\rightsquigarrow Méthode des moindres carrés.

Exemple : régression linéaire

Contexte

Problème

Étant donné y_n : réponse aléatoire, $X_{n \times p}$: prédicteurs, $n > p$ trouver β tel que

$$y = X\beta$$

Habituellement, erreurs de mesure \Rightarrow système indéterminé.

Outils

- \rightsquigarrow Méthode des moindres carrés.
- \rightsquigarrow Maximum de vraisemblance.

Exemple : régression linéaire

Contexte

Problème

Étant donné y_n : réponse aléatoire, $X_{n \times p}$: prédicteurs, $n > p$ trouver β tel que

$$y = X\beta$$

Habituellement, erreurs de mesure \Rightarrow système indéterminé.

Outils

- \rightsquigarrow Méthode des moindres carrés.
- \rightsquigarrow Maximum de vraisemblance.
- \rightsquigarrow ???

Méthode des moindres carrés

~> Développée par nul autre que Carl Friedrich Gauss (et Adrien-Marie Legendre de manière indépendante) en 1795, publiée en 1805.

Méthode des moindres carrés

- ↪ Développée par nul autre que Carl Friedrich Gauss (et Adrien-Marie Legendre de manière indépendante) en 1795, publiée en 1805.
- ↪ Estimateur :

$$\hat{\beta} = \arg \max_{\beta} \|y - X\beta\|^2 = (X^T X)^{-1} X^T y$$

Exemple : régression linéaire

Maximum de vraisemblance

↪ Histoire plus alambiquée [7].

Maximum de vraisemblance

- ↪ Histoire plus alambiquée [7].
- ↪ Idée : on suppose que y suit une certaine distribution puis on maximise la vraisemblance comme fonction de β .

Exemple : régression linéaire

Maximum de vraisemblance

- ↪ Histoire plus alambiquée [7].
- ↪ Idée : on suppose que y suit une certaine distribution puis on maximise la vraisemblance comme fonction de β .
- ↪ Il est bien connu que

$$y \sim \mathcal{N} \Rightarrow \hat{\beta} = (X^T X)^{-1} X^T y$$

Exemple : régression linéaire

Maximum de vraisemblance

- ↪ Histoire plus alambiquée [7].
- ↪ Idée : on suppose que y suit une certaine distribution puis on maximise la vraisemblance comme fonction de β .
- ↪ Il est bien connu que

$$y \sim \mathcal{N} \Rightarrow \hat{\beta} = (X^T X)^{-1} X^T y$$

Donc, ces deux méthodes nécessitent une inversion de matrice.

Inversion de matrice

Sur un ordinateur standard, l'inversion de matrice est

Exemple : régression linéaire

Inversion de matrice

Sur un ordinateur standard, l'inversion de matrice est

Imprécise

Perte de précision due à l'encodage des nombres à virgule flottante.

Exemple : régression linéaire

Inversion de matrice

Sur un ordinateur standard, l'inversion de matrice est

Imprécise

Perte de précision due à l'encodage des nombres à virgule flottante.

Inefficace

Exemple : régression linéaire

Inversion de matrice

Sur un ordinateur standard, l'inversion de matrice est

Imprécise

Perte de précision due à l'encodage des nombres à virgule flottante.

Inefficace

↪ Méthodes gaussiennes : $\mathcal{O}(n^3)$

Exemple : régression linéaire

Inversion de matrice

Sur un ordinateur standard, l'inversion de matrice est

Imprécise

Perte de précision due à l'encodage des nombres à virgule flottante.

Inefficace

↪ Méthodes gaussiennes : $\mathcal{O}(n^3)$

↪ Meilleures méthodes : $\mathcal{O}(> n^{2.3})$

Exemple : régression linéaire

Décomposition QR

Toute matrice X peut se décomposer en une matrice Q orthogonale et R triangulaire supérieure de sorte que

$$X = QR.$$

Exemple : régression linéaire

Décomposition QR

Toute matrice X peut se décomposer en une matrice Q orthogonale et R triangulaire supérieure de sorte que

$$X = QR.$$

Précis

Perte de précision moindre que l'inversion de matrice.

Exemple : régression linéaire

Décomposition QR

Toute matrice X peut se décomposer en une matrice Q orthogonale et R triangulaire supérieure de sorte que

$$X = QR.$$

Précis

Perte de précision moindre que l'inversion de matrice.

Efficace

Multiple optimisation possible (entre autre, pas besoin de calculer explicitement Q).

Exemple : régression linéaire

Estimation et décomposition QR

En posant $X = QR$, on obtient.

Exemple : régression linéaire

Estimation et décomposition QR

En posant $X = QR$, on obtient.

$$X^T X \beta = X^T y$$

Exemple : régression linéaire

Estimation et décomposition QR

En posant $X = QR$, on obtient.

$$X^T X \beta = X^T y$$

$$\Leftrightarrow R^T R \beta = R^T Q^T y$$

Exemple : régression linéaire

Estimation et décomposition QR

En posant $X = QR$, on obtient.

$$X^T X \beta = X^T y$$

$$\Leftrightarrow R^T R \beta = R^T Q^T y$$

$$\Leftrightarrow R \beta = Q^T y.$$

Exemple : régression linéaire

Estimation et décomposition QR

En posant $X = QR$, on obtient.

$$X^T X \beta = X^T y$$

$$\Leftrightarrow R^T R \beta = R^T Q^T y$$

$$\Leftrightarrow R \beta = Q^T y.$$

~> On a transformé un problème surdéterminé en problème déterminé!

Exemple : régression linéaire

Estimation et décomposition QR

En posant $X = QR$, on obtient.

$$X^T X \beta = X^T y$$

$$\Leftrightarrow R^T R \beta = R^T Q^T y$$

$$\Leftrightarrow R \beta = Q^T y.$$

- ↪ On a transformé un problème surdéterminé en problème déterminé!
- ↪ Le système d'équations est échelonné.

Exemple : régression linéaire

Estimation et décomposition QR

En posant $X = QR$, on obtient.

$$X^T X \beta = X^T y$$

$$\Leftrightarrow R^T R \beta = R^T Q^T y$$

$$\Leftrightarrow R \beta = Q^T y.$$

- ↪ On a transformé un problème surdéterminé en problème déterminé!
- ↪ Le système d'équations est échelonné.
- ↪ Seules les p premières lignes de R sont non nulles.

Exemple : régression linéaire

Estimation et décomposition QR

En posant $X = QR$, on obtient.

$$X^T X \beta = X^T y$$

$$\Leftrightarrow R^T R \beta = R^T Q^T y$$

$$\Leftrightarrow R \beta = Q^T y.$$

- ↪ On a transformé un problème surdéterminé en problème déterminé!
- ↪ Le système d'équations est échelonné.
- ↪ Seules les p premières lignes de R sont non nulles.
- ↪ Donc, on peut résoudre ce système d'équations directement par backsolving ($\mathcal{O}(m^2)$).

Exemple : régression linéaire

Variance de $\hat{\beta}$

\rightsquigarrow On sait que $\hat{V}[\hat{\beta}] = s^2(X^T X)^{-1}$.

Exemple : régression linéaire

Variance de $\hat{\beta}$

↪ On sait que $\hat{V}[\hat{\beta}] = s^2(X^T X)^{-1}$.

↪ Toutefois, on peut faire mieux que d'inverser $X^T X$!

Exemple : régression linéaire

Variance de $\hat{\beta}$

↪ On sait que $\hat{V}[\hat{\beta}] = s^2(X^T X)^{-1}$.

↪ Toutefois, on peut faire mieux que d'inverser $X^T X$!

↪ Si $X = QR$, on a que

$$\hat{V}[\hat{\beta}] = s^2(R^T R)^{-1}.$$

Exemple : régression linéaire

Variance de $\hat{\beta}$

↪ On sait que $\hat{V}[\hat{\beta}] = s^2(X^T X)^{-1}$.

↪ Toutefois, on peut faire mieux que d'inverser $X^T X$!

↪ Si $X = QR$, on a que

$$\hat{V}[\hat{\beta}] = s^2(R^T R)^{-1}.$$

↪ On remarque que R^T est une matrice triangulaire inférieure de sorte que $R^T R$ est la décomposition de Cholesky de $X^T X$!

Exemple : régression linéaire

Variance de $\hat{\beta}$

↪ On sait que $\hat{V}[\hat{\beta}] = s^2(X^T X)^{-1}$.

↪ Toutefois, on peut faire mieux que d'inverser $X^T X$!

↪ Si $X = QR$, on a que

$$\hat{V}[\hat{\beta}] = s^2(R^T R)^{-1}.$$

↪ On remarque que R^T est une matrice triangulaire inférieure de sorte que $R^T R$ est la décomposition de Cholesky de $X^T X$!

↪ Il existe des algorithmes efficaces (analogue au backsolving) pour l'inversion d'une matrice dont on possède la décomposition de Cholesky.

Exemple : nombres de Fibonacci

\rightsquigarrow Soit

$$M = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$$

Exemple : nombres de Fibonacci

↪ Soit

$$M = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$$

↪ On peut montrer que le n^{e} nombre de Fibonacci, $n = 0, 1, \dots$, est l'entrée supérieure gauche de M^n .

Exemple : nombres de Fibonacci

↪ Soit

$$M = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$$

- ↪ On peut montrer que le n^{e} nombre de Fibonacci, $n = 0, 1, \dots$, est l'entrée supérieure gauche de M^n .
- ↪ Il n'est pas nécessaire de calculer explicitement les puissances de M .

Exemple : nombres de Fibonacci

↪ Les valeurs propres de M sont φ et $-\varphi^{-1}$ où $\varphi = \frac{1+\sqrt{5}}{2}$ est le nombre d'or.

Exemple : nombres de Fibonacci

↪ Les valeurs propres de M sont φ et $-\varphi^{-1}$ où $\varphi = \frac{1+\sqrt{5}}{2}$ est le nombre d'or.

↪ De plus, des vecteurs propres de M sont

$$u_1 = \begin{pmatrix} \varphi \\ 1 \end{pmatrix}, \quad u_2 = \begin{pmatrix} -\varphi^{-1} \\ 1 \end{pmatrix}.$$

Exemple : nombres de Fibonacci

↪ Les valeurs propres de M sont φ et $-\varphi^{-1}$ où $\varphi = \frac{1+\sqrt{5}}{2}$ est le nombre d'or.

↪ De plus, des vecteurs propres de M sont

$$u_1 = \begin{pmatrix} \varphi \\ 1 \end{pmatrix}, \quad u_2 = \begin{pmatrix} -\varphi^{-1} \\ 1 \end{pmatrix}.$$

↪ On obtient la décomposition spectrale suivante :

$$M = UDU^{-1} = \frac{1}{\sqrt{5}} \begin{pmatrix} \varphi & -\varphi^{-1} \\ 1 & 1 \end{pmatrix} \begin{pmatrix} \varphi & 0 \\ 0 & -\varphi^{-1} \end{pmatrix} \begin{pmatrix} 1 & \varphi^{-1} \\ -1 & \varphi \end{pmatrix}.$$

Exemple : nombres de Fibonacci

↪ La puissance n de M peut alors se calculer efficacement :

$$M^n = UD^nU^{-1}$$

Exemple : nombres de Fibonacci

↪ La puissance n de M peut alors se calculer efficacement :

$$M^n = UD^nU^{-1}$$

↪ En fait, comme on n'a besoin que de l'entrée supérieure gauche de M , on a que le n^{e} nombre de Fibonacci est

$$\frac{1}{\sqrt{5}} \begin{pmatrix} \varphi^{n+1} & (-\varphi)^{-(n+1)} \end{pmatrix} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \frac{\varphi^{n+1} - (-\varphi)^{-(n+1)}}{\sqrt{5}}$$

Exemple : nombres de Fibonacci

↪ La puissance n de M peut alors se calculer efficacement :

$$M^n = UD^nU^{-1}$$

↪ En fait, comme on n'a besoin que de l'entrée supérieure gauche de M , on a que le n^{e} nombre de Fibonacci est

$$\frac{1}{\sqrt{5}} \begin{pmatrix} \varphi^{n+1} & (-\varphi)^{-(n+1)} \end{pmatrix} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \frac{\varphi^{n+1} - (-\varphi)^{-(n+1)}}{\sqrt{5}}$$

↪ On peut donc calculer les nombres de Fibonacci directement : pas besoin d'algorithmes dynamiques, si sophistiqués soient-ils !

Références

- [1] *Add polynomials*. English. Mai 2009. URL : https://wiki.haskell.org/Add_polynomials.
- [2] *Bell Labs*. en. Page Version ID : 908476575. Juil. 2019. URL : https://en.wikipedia.org/w/index.php?title=Bell_Labs&oldid=908476575.
- [3] John M CHAMBERS. “A computer system for fitting models to data”. In : *Journal of the Royal Statistical Society : Series C (Applied Statistics)* 18.3 (1969), p. 249-263.
- [4] John M CHAMBERS. *Programming with data : A guide to the S language*. Springer Science & Business Media, 1998.

- [5] Kurt HORNIK. *R FAQ – What are the differences between R and S?* English. Oct. 2018. URL : https://cran.r-project.org/doc/FAQ/R-FAQ.html#What-are-the-differences-between-R-and-S_003f.
- [6] *Low-level programming language - Wikipedia*. URL : https://en.wikipedia.org/wiki/Low-level_programming_language#Machine_code.
- [7] Stephen M. STIGLER. “The Epic Story of Maximum Likelihood”. EN. In : *Statistical Science* 22.4 (nov. 2007), p. 598-620. ISSN : 0883-4237, 2168-8745. DOI : 10.1214/07-STS249. URL : <https://projecteuclid.org/euclid.ss/1207580174>.

- [8] *The Top Bell Labs Innovations - Part I : The Game-Changers*. en-US. URL :
<http://blog.tmcnet.com/next-generation-communications/,%20http://blog.tmcnet.com/next-generation-communications/2011/08/the-top-bell-labs-innovations---part-i-the-game-changers.html>.

- [9] Luke TIERNEY. *XLISP-STAT A Statistical Environment Based on the XLISP Language (Version 2.0)*. English. Technical Report 528. Minnesota, United States of America : University of Minnesota, School of Statistics, juil. 1989. URL : <https://homepage.divms.uiowa.edu/~luke/xls/tutorial/techreport/techreport.html>.
- [10] Konrad ZUSE. "Über den Allgemeinen Plankalkül als Mittel zur Formulierung schematisch-kombinativer Aufgaben". de. In : *Archiv der Mathematik* 1.6 (nov. 1948), p. 441-449. ISSN : 1420-8938. DOI : 10.1007/BF02038459. URL : <https://doi.org/10.1007/BF02038459>.