BIOSTAT M280/BIOMATH 280/STAT M230: Statistical Computing

Tue/Thu 1:00pm-2:50pm, CHS 51-279

Instructor: Dr. Hua Zhou, huazhou@ucla.edu

1 Lecture 1: Jan 5

Today

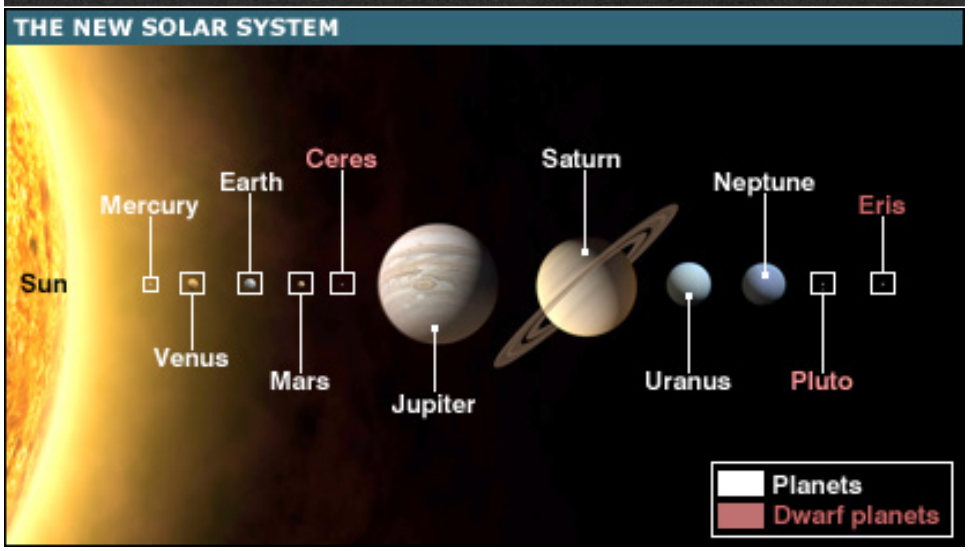
- Introduction and course logistics
- Computer storage and arithmetic
- Homework (not graded): fill out a short survey at https://www.surveymonkey.com/r/G8BCVVM
- Homework (not graded): Read the Introduction and Foundations of *Advanced R* by Hadley Wickham http://adv-r.had.co.nz. Install R. Try examples while you read the book. Do the quizzes.

What is statistics?

- People collect data in order to answer certain questions. (Bio)statisticians's job is to help make sense of data.
- Statistics, the science of *data analysis*, is the applied mathematics in the 21st century.
- Read papers *The future of data analysis* by John Tukey (http://hua-zhou.github.io/teaching/biostatm280-2016winter/readings/Tukey61FutureDataAnalysis.pdf) and *50 years of data sicence* by David Donoho (http://hua-zhou.github.

io/teaching/biostatm280-2016winter/readings/Donoho15FiftyYearsDataScience.pdf).

How Gauss became famous?

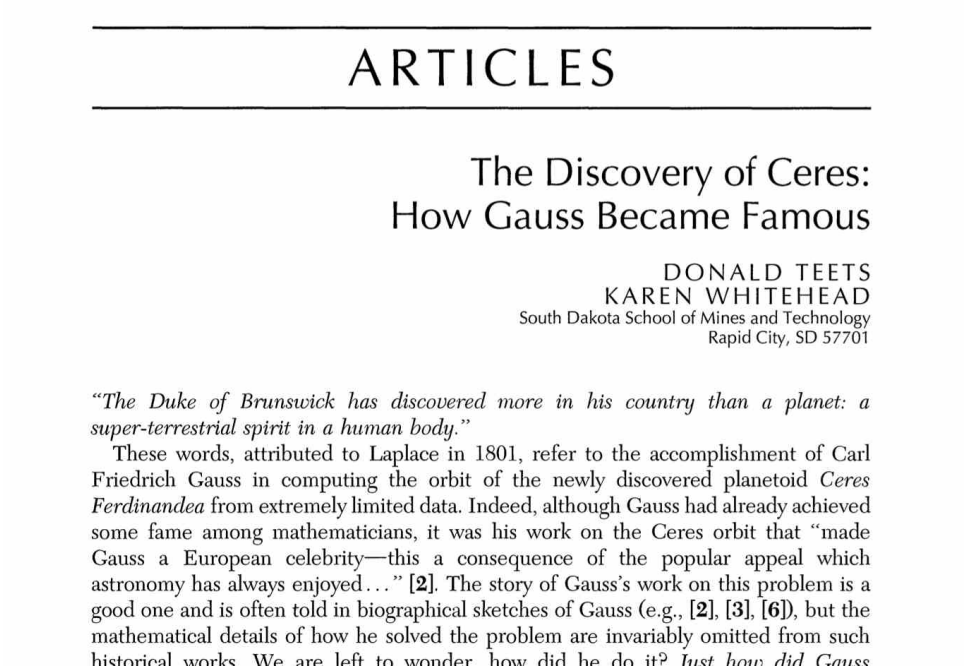


- 1801, *Dr. Carl Friedrich Gauss*, 24; proved Fundamental Theorem of Algebra; wrote the book *Disquisitiones Arithmetic*, which is still being studied today.
- 1801, Jan 1-Feb 11 (41 days), astronomer Piazzi observed Ceres (a dwarf planet), which was then lost behind sun.
- 1801, Aug–Sep, futile search by top astronomers; Laplace claimed it unsolvable.

- 1801, Oct–Nov, Gauss did calculations by *method of least squares*.
- 1801, Dec 31, astronomer von Zach re-located Ceres according to Gauss' calculation.
- 1802, *Summarische Übersicht der Bestimmung der Bahnen der beiden neuen Hauptplaneten angewandten Methoden*, considered the origin of linear algebra.
- 1807, Professor of Astronomy and (the first) Director of Göttingen Observatory in remainder of his life.
- 1809, *Theoria motus corporum coelestium in sectionibus conicis solum ambientium* (Theory of motion of the celestial bodies moving in conic sections around the Sun); birth of the Gaussian (normal) distribution, as an attempt to rationalize the method of least squares.
- 1810, Laplace consolidated the importance of Gaussian distribution by proving the central limit theorem.
- 1829, Gauss-Markov Theorem. Under Gaussian error assumption (actually only uncorrelated and homoscedastic needed), least square solution is the best linear unbiased estimate (BLUE), i.e., it has the smallest variance and thus MSE among all linear unbiased estimators. Note other estimators such as the James-Stein estimator may have smaller MSE, but they are *nonlinear*.

For more details of the story

- `http://www.keplersdiscovery.com/Asteroid.html`
- Teets and Whitehead (1999)



ARTICLES

The Discovery of Ceres: How Gauss Became Famous

DONALD TEETS
KAREN WHITEHEAD
South Dakota School of Mines and Technology
Rapid City, SD 57701

"The Duke of Brunswick has discovered more in his country than a planet: a super-terrestrial spirit in a human body."

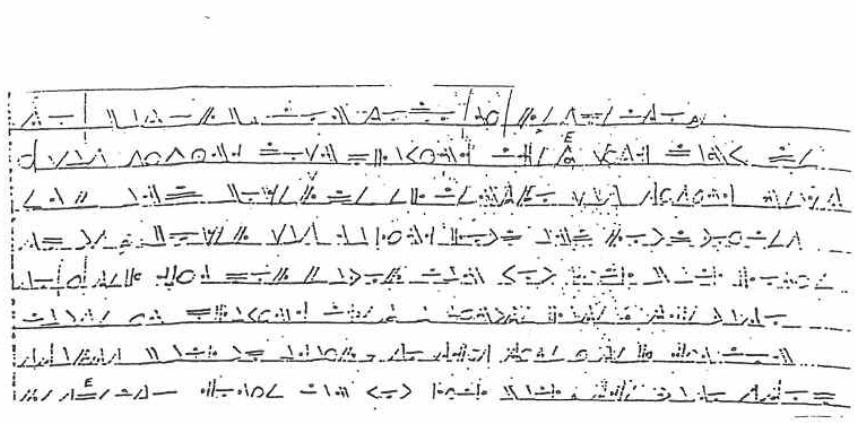
These words, attributed to Laplace in 1801, refer to the accomplishment of Carl Friedrich Gauss in computing the orbit of the newly discovered planetoid *Ceres Ferdinandea* from extremely limited data. Indeed, although Gauss had already achieved some fame among mathematicians, it was his work on the Ceres orbit that "made Gauss a European celebrity—this a consequence of the popular appeal which astronomy has always enjoyed..." **[2]**. The story of Gauss's work on this problem is a good one and is often told in biographical sketches of Gauss (e.g., **[2]**, **[3]**, **[6]**), but the mathematical details of how he solved the problem are invariably omitted from such historical works. We are left to wonder how did he do it? *Just how did Gauss*

- Stigler (1986) gives a more comprehensive account of the origin of the method of least squares.

Gauss' story

- Motivated by a real problem.
- Heuristic solution: method of least squares.
- Solution readily verifiable: Ceres was re-discovered!
- *Algorithmic development*: linear algebra, Gaussian elimination, FFT (fast Fourier transform).
- Theoretical justification: Gaussian distribution, Gauss-Markov theorem.

A sampler by Marc Coram



ENTER HAMLET HAM TO BE OR NOT TO BE THAT IS THE QUESTION WHETHER TIS
NOBLER IN THE MIND TO SUFFER THE SLINGS AND ARROWS OF OUTRAGEOUS
FORTUNE OR TO TAKE ARMS AGAINST A SEA OF TROUBLES AND BY OPPOSING END

100 ER ENOHDLAE OHDLO UOZEOUNORU O UOZEO HD OITO HEOQSET IUROFHE HENO ITORUZAEN
200 ES ELOHRNDE OHRNO UOVEOULOSU O UOVEO HR OITO HEOQAET IUSOPHE HELO ITOSUVDEL
300 ES ELOHANDE OHANO UOVEOULOSU O UOVEO HA OITO HEOQRET IUSOFHE HELO ITOSUVDEL
400 ES ELOHINME OHINO UOVEOULOSU O UOVEO HI OATO HEOQRET AUSOWHE HELO ATOSUVKEL
500 ES ELOHINME OHINO UODEOULOSU O UODEO HI OATO HEOQRET AUSOWHE HELO ATOSUDKEL
600 ES ELOHINME OHINO UODEOULOSU O UODEO HI OATO HEOQRET AUSOWHE HELO ATOSUDKEL
900 ES ELOHANME OHANO UODEOULOSU O UODEO HA OITO HEOQRET IUSOWHE HELO ITOSUDKEL
1000 IS ILOHANMI OHANO RODIORLOSR O RODIO HA OETO HIOQUIT ERSOWHI HILO ETOSRDMIL
1100 ISTILOHANMITOHANOT ODIO LOS TOT ODIOTHATOEROTHIOQUIRTE SOWHITHILOTEROS DMIL
1200 ISTILOHANMITOHANOT ODIO LOS TOT ODIOTHATOEROTHIOQUIRTE SOWHITHILOTEROS DMIL
1300 ISTILOHARMITOHAROT ODIO LOS TOT ODIOTHATOENOTHIOQUINTE SOWHITHILOTENOS DMIL
1400 ISTILOHAMRITOHAMOT OFIO LOS TOT OFIOTHATOENOTHIOQUINTE SOWHITHILOTENOS FRIL
1600 ESTEL HAMRET HAM TO CE OL SOT TO CE THAT IN THE QUENTIOS WHETHEL TIN SOCREL
1700 ESTEL HAMRET HAM TO BE OL SOT TO BE THAT IN THE QUENTIOS WHETHEL TIN SOBREL
1800 ESTER HAMLET HAM TO BE OR SOT TO BE THAT IN THE QUENTIOS WHETHER TIN SOBLER
1900 ENTER HAMLET HAM TO BE OR NOT TO BE THAT IS THE QUESTION WHETHER TIS NOBLER
2000 ENTER HAMLET HAM TO BE OR NOT TO BE THAT IS THE QUESTION WHETHER TIS NOBLER

to bat-rb. con todo mi respeto. i was sitting down playing chess with danny de emf and boxer de el centro was sitting next to us. boxer was making loud and loud voices so i tell him por favor can you kick back homie cause im playing chess a minute later the vato starts back up again so this time i tell him con respecto homie can you kick back. the vato stop for a minute and he starts up again so i tell him check this out shut the f**k up cause im tired of your voice and if you got a problem with it we can go to celda and handle it. i really felt disrespected thats why i told him. anyways after i tell him that the next thing I know that vato slashes me and leaves. dy the time i figure im hit i try to get away but the c.o. is walking in my direction and he gets me right dy a celda. so i go to the hole. when im in the hole my home boys hit doxer so now "b" is also in the hole. while im in the hole im getting schoold wrong and

- A consulting project by Marc Coram (then a graduate student in statistics at Stanford); customer is a professor in political science, who wants to understand a cryptic message circulated in a state prison.
- Marc modeled letter sequence by a Markov chain (26×26 transition matrix) and estimated transition probabilities from *War and Peace*.

- Now each mapping σ yields a likelihood $f(\sigma)$ of the symbol sequence.
- Find the σ that maximizes f. Sample space is at least $26! = 4.0329 \times 10^{26}$. Combinatorial optimization – hard!
- *Metropolis algorithm*: At each iteration, generate a new σ' by random transposition of two letters; accept σ' with probability $\min\left\{\frac{f(\sigma')}{f(\sigma)}, 1\right\}$.

Marc Coram's story

- Motivated by a real problem.
- Solution readily verifiable: we can read it!
- *Algorithm development*: Metropolis sampler is one of top 10 algorithms in the 20th century.
- Read Diaconis (2009) for more details.

What is this course about?

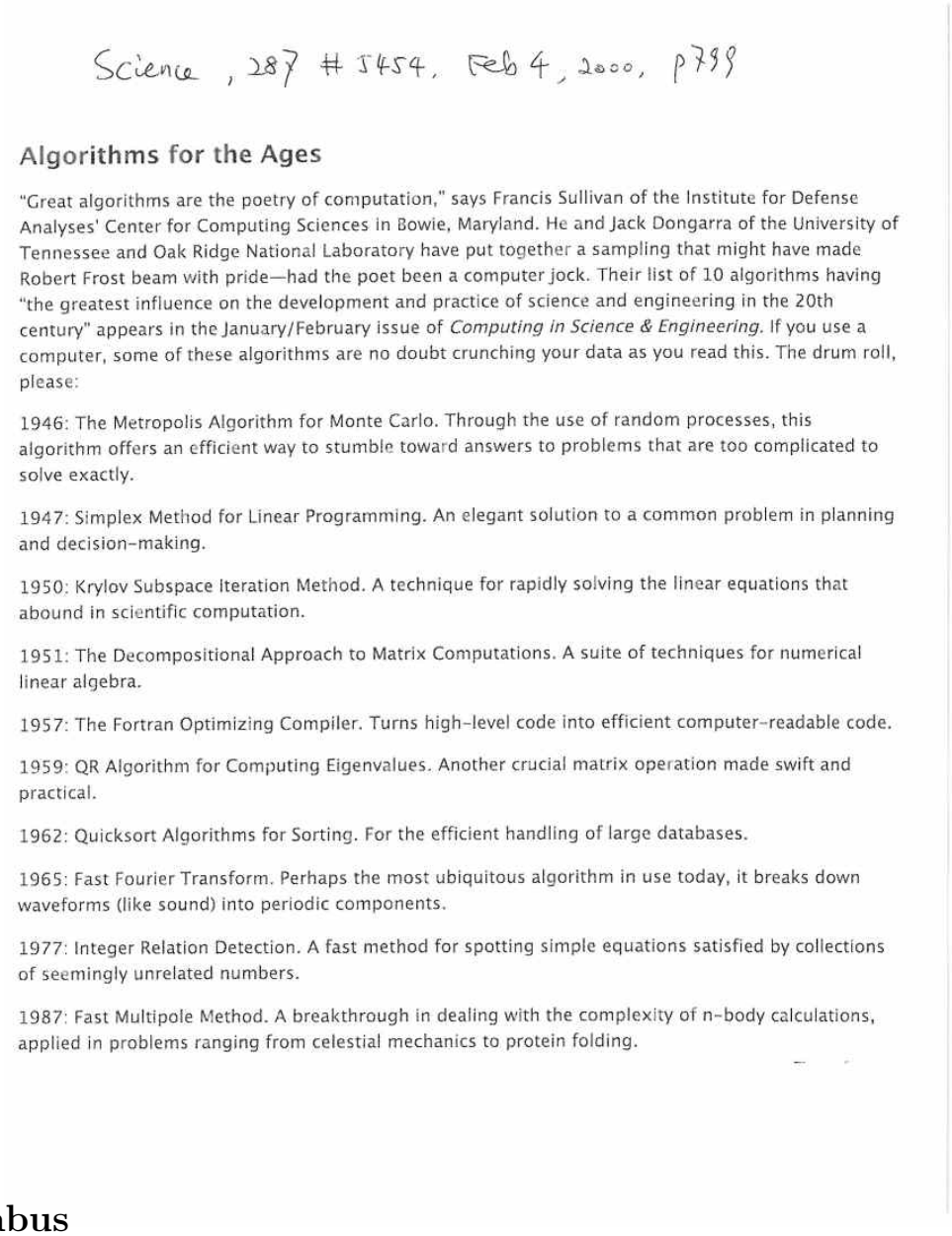
- Not a course on "packages and languages for data analysis". It does not answer questions such as "How to fit a linear mixed model in SAS, SPSS or R?"
- Not a programming course, although programming is *extremely* important and we do homework in R.
- This course is about "numerical methods in statistics". Our focus is on *algorithms*.

> The form of a mathematical expression and the way the expression should be evaluated in actual practice may be quite different.
>
> James Gentle

For a common numerical task in statistics, say the least squares solution $\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y}$, we need to know which methods/algorithms are out there and what are their advantages and disadvantages. You will *fail* this course if you use

```
solve(t(X) %*% X) %*% t(X) %*% y
```

Using `lm(y ~ X)` is correct (and efficient) but is not the purpose of this course. We want to understand what is going on when calling the `lm()` function.



Science, 287 #5454, Feb 4, 2000, p799

Algorithms for the Ages

"Great algorithms are the poetry of computation," says Francis Sullivan of the Institute for Defense Analyses' Center for Computing Sciences in Bowie, Maryland. He and Jack Dongarra of the University of Tennessee and Oak Ridge National Laboratory have put together a sampling that might have made Robert Frost beam with pride—had the poet been a computer jock. Their list of 10 algorithms having "the greatest influence on the development and practice of science and engineering in the 20th century" appears in the January/February issue of *Computing in Science & Engineering*. If you use a computer, some of these algorithms are no doubt crunching your data as you read this. The drum roll, please:

1946: The Metropolis Algorithm for Monte Carlo. Through the use of random processes, this algorithm offers an efficient way to stumble toward answers to problems that are too complicated to solve exactly.

1947: Simplex Method for Linear Programming. An elegant solution to a common problem in planning and decision-making.

1950: Krylov Subspace Iteration Method. A technique for rapidly solving the linear equations that abound in scientific computation.

1951: The Decompositional Approach to Matrix Computations. A suite of techniques for numerical linear algebra.

1957: The Fortran Optimizing Compiler. Turns high-level code into efficient computer-readable code.

1959: QR Algorithm for Computing Eigenvalues. Another crucial matrix operation made swift and practical.

1962: Quicksort Algorithms for Sorting. For the efficient handling of large databases.

1965: Fast Fourier Transform. Perhaps the most ubiquitous algorithm in use today, it breaks down waveforms (like sound) into periodic components.

1977: Integer Relation Detection. A fast method for spotting simple equations satisfied by collections of seemingly unrelated numbers.

1987: Fast Multipole Method. A breakthrough in dealing with the complexity of n-body calculations, applied in problems ranging from celestial mechanics to protein folding.

Syllabus

Check course website frequently for updates and announcements.
`http://hua-zhou.github.io/teaching/biostatm280-2016winter`
Lecture notes will be updated and posted after each lecture.

Computer storage and arithmetic

Elementary units of computer storage:

- *bit* = "binary" + "digit" (coined by statistician John Tukey).
- *byte* = 8 bits.
- kB = kilobyte = 10^3 bytes.
- MB = megabytes = 10^6 bytes.
- GB = gigabytes = 10^9 bytes.
- TB = terabytes = 10^{12} bytes.
- PB = petabytes = 10^{15} bytes.

Storage of characters

ASCII control characters

00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

ASCII printable characters

32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

Extended ASCII characters

128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

- Plain text files are stored in the form of characters: `.r`, `.c`, `.cpp`, `.tex`, `.html`, ...
- ASCII (American Code for Information Interchange): 7 bits, only $2^7 = 128$ characters, "Hua" corresponds to "48 75 61 (Hex) = 72 117 97 (Dec) = 1001000 1110101 1100001".

- Extended ASCII: 8 bits, $2^8 = 256$ characters.
- Unicode: UTF-8, UTF-16 and UTF-32 support many more characters including foreign characters; last 7 digits conform to ASCII. UTF-8 is the current dominant character encoding on internet.

2 Lecture 2, Jan 7

Announcements

- Location change: class meeting is changed to CHS 51-279, effective Jan 7.
- Enrollment cap is raised to 38 (open).
- Use RStudio for R programming.

Last time

- Introduction. Gauss (least squares to find Ceres)–optimization, Marc Coram (decipher a note circulating in jail)–sampling. Two major modes of statistical computing.
- Course content, logistics.
- Computer representation of characters (ASCII, unicode) and integers (fixed point number system).

Today

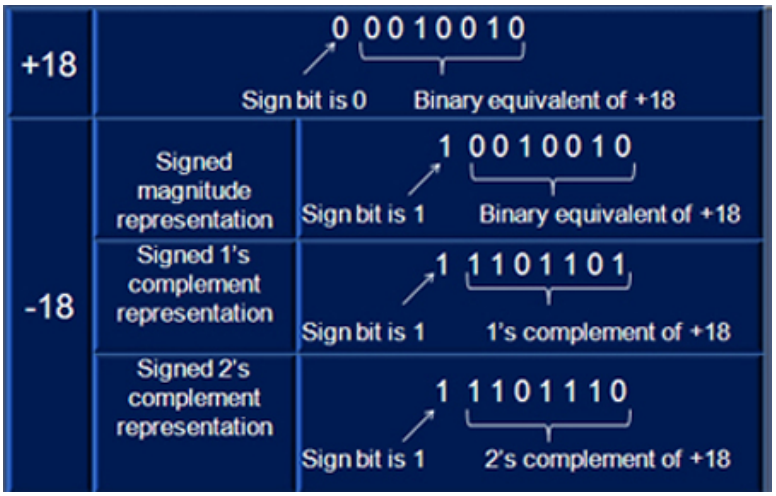
- Computer representation and arithmetic of integers: fixed-point numbers.
- Computer representation and arithmetic of real numbers: floating-point numbers.

Fixed-point number system

Fixed-point number system $\mathbb{I}$ is a computer model for integers $\mathbb{Z}$. One storage unit may be $M = 8/16/32/64$ bit.

- The number of bits and method of representing negative numbers vary from system to system. The `integer` type in R has $M = 32$ bits. MATLAB has (u)int8, (u)int16, (u)int32, (u)int64. JULIA has even more choices such as `Int128`, `UInt128`, `BigInt`, ...
- First bit indicates sign: 0 for nonnegative numbers, 1 for negative numbers.

- "two's complement representation" for negative numbers. (i) Sign bit is set to 1, (ii) remaining bits are set to opposite values, (iii) 1 is added to the result.

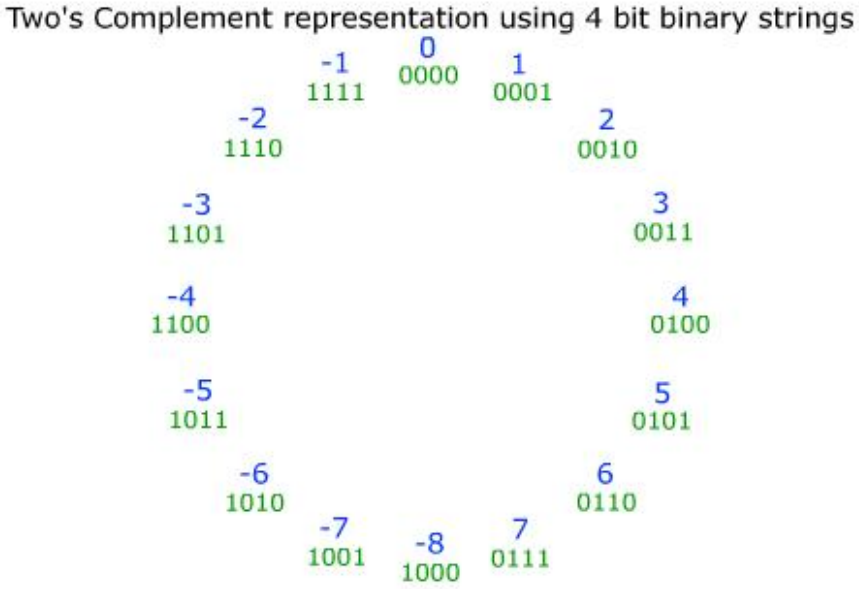


- Range of representable integers by M-bit storage is $[-2^{M-1}, 2^{M-1}-1]$ (don't need to represent 0 anymore so *could* have capacity for 2^{M-1} negative numbers).

- For $M = 8$, $[-128, 127]$.
  For $M = 16$, $[-65536, 65535]$.
  For $M = 32$, $[-2147483648, 2147483647]$.

- The smallest representable integer in R is $-2^{31} + 1 = -2147483647$.

- For unsigned integers such as in MATLAB, the range is $[0, 2^M - 1]$.

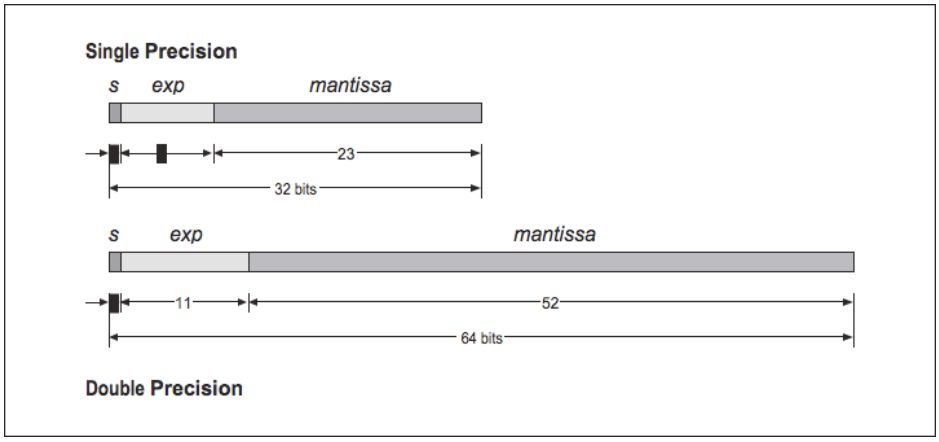


- An integer $-i$ in the interval $[-2^{M-1}, -1]$ would be represented by the same bit pattern by which the nonnegative integer $2^M - i$ is represented, treating the sign bit as a regular numeric bit. For example, (-18) = 11101110, $(2^8) - (18)$=(238)=11101110. Two's complement is subtracting the nonnegative integer i from $1 \ldots 1 + 1 = 10 \ldots 0 = (2^M)$.

- Addition and subtraction are much simpler in two's complement representation. Why? It respects modular arithmetic nicely (look at the diagram). E.g., (3)+(4) = 0011 + 0100 = 0111 = (7), (3)+(-5) = 0011 + 1011 = 1110 = (-2), (3) + (7) = 0011 + 0111 = 1010 = (-6) (overflow), (-3) + (-7) = 1101 + 1001 = 0110 = (6) (underflow).

- Keep track of overflow and underflow. If the result of a summation is R, which must be in the set $[-2^{M-1}, 2^{M-1}+1]$, there are only three possibilities for the true sum: R, $R+2^M$ (overflow), or $R-2^M$ (underflow).
  - When adding two nonnegative integers: 0XX...X + 0YY...Y, where X and Y are arbitrary binary digits, we only need to keep track of overflow in this case. If the resulting binary number has a leading bit of 1, we know overflow occurs since the sum cannot be negative. We should treat that sign bit as a regular numeric bit. In other words, we crossed the upper boundary 100 and should add 2^M to the result. If the resulting binary number has a leading bit of 0, no overflow occurs.
  - When adding two negative integers: 1XX...X + 1YY...Y, where X and Y are arbitrary binary digits, we only need to keep track of underflow in this case. If the resulting binary number has a leading bit of 0, we know underflow occurs since the sum cannot be positive. We crossed the lower boundary 000 and should subtract 2^M from the result. If the resulting binary number has a leading bit of 1, no underflow occurs.
  - When adding a negative integer and a nonnegative integer: 1XX...X + 0YY...Y, the result is always between the two summands. No overflow or underflow could happen.
- R reports `NA` for integer overflow and underflow. Other languages, e.g., Julia simply outputs the result according modular arithmetic.

Floating-point number system



Floating-point number system $\mathbb{F}$ is a computer model for real numbers $\mathbb{R}$.

- A real number is represented by $\pm d_0.d_1d_2\cdots d_p \times b^e$ (scientific notation).
- Parameters for a floating-point number system: *base* (or *radix*), range of *fraction* (or *mantissa*, *significand*), range of *exponent*.
- Non-uniqueness of representation. *normalized/denormalized* significant digits. E.g., $+18 = +1.0010 \times 2^4$(normalied) $= +0.10010 \times 2^5$(denormalized).
- *Bias* (or *excess*): actual exponent is obtained by subtracting bias from the value of exponent evaluated regardless of sign digit.
- IEEE 754.
  - *Single precision* (32 bit): base 2, $p = 23$ (23 significant bits), $e_{\max} = 127$, $e_{\min} = -126$ (8 exponent bits), bias=127. $e_{\min}-1$ and $e_{\max}+1$ are reserved for special numbers. This implies a maximum magnitude of $\log_{10}(2^{127}) \approx 38$ and precision to $\log_{10}(2^{23}) \approx 7$ decimal point. $\pm 10^{\pm 38}$.
  - *Double precision* (64 bit): base 2, $p = 52$ (52 significant bits), $e_{\max} = 1023$, $e_{\min} = -1022$ (11 exponent bits), bias=1023. This implies a maximum magnitude of $\log_{10}(2^{1023}) \approx 308$ and precision to $\log_{10}(2^{52}) \approx 16$ decimal point. $\pm 10^{\pm 308}$.

- "$(+18) = (2^4 + 2^1) = +1.0010 \times 2^4$ in single precision

  $$(0)(10000011)(00100000000000000000).$$

  First is sign bit. Next 8 bits are exponent 131 in ordinary base 2 with a bias of 127. Remaining 23 bits represent the fraction beyond the leading bit, known to be 1. In summary it represents $(+18)$ as $+1.0010 \times 2^4$ in the binary format. (-18) is represented by the same bits except changing the sign bit to 1.

- Special floating-point numbers:
  - Exponent $e_{\max} + 1$ plus a mantissa of 0 means $\pm\infty$.
  - Exponent $e_{\max} + 1$ plus a nonzero mantissa means NaN. NaN could be produced from `0 / 0`, `0 * Inf`, ... In general $NaN \neq NaN$ bitwise.
  - Exponent $e_{\min} - 1$ with a mantissa of all 0s represents the real number 0.
  - Exponent $e_{\min} - 1$ with a nonzero mantissa are for numbers less than $b^{e_{\min}}$. Numbers are de-normalized in the range $(0, b^{e_{\min}})$ – "graceful underflow".

- $\mathbb{F}$ is not a subset of $\mathbb{R}$, although $\mathbb{I} \subset \mathbb{Z}$.

- For the history of establishment of IEEE-754 standard, see an interview with William Kahan.
  `http://www.cs.berkeley.edu/~wkahan/ieee754status/754story.html`

To summarize

- Single precision: range $\pm 10^{\pm 38}$ with precision up to 7 decimal digits.

- Double precision: range $\pm 10^{\pm 308}$ with precision up to 16 decimal digits.

- The floating-point numbers do not occur uniformly over the real number line.

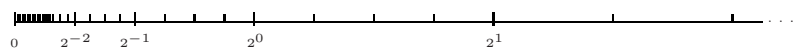


Fig. 2.4. The Floating-Point Number Line, Nonnegative Half

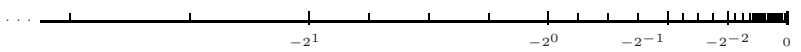


Fig. 2.5. The Floating-Point Number Line, Nonpositive Half

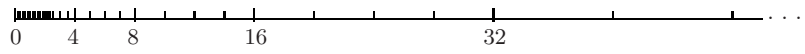


Fig. 2.6. The Floating-Point Number Line, Nonnegative Half; Another View

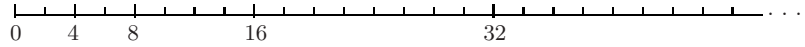


Fig. 2.7. The Fixed-Point Number Line, Nonnegative Half

- *Machine epsilons* are the spacings of numbers around 1. $\epsilon_{\min} = b^{-p}$ and $\epsilon_{\max} = b^{1-p}$.

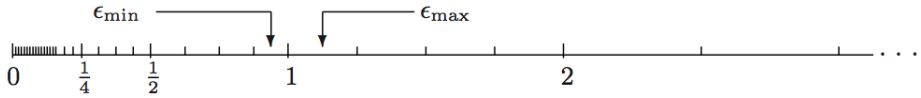


Fig. 2.8. Relative Spacings at 1: "Machine Epsilons"

- The variable `.Machine` in R contains numerical characteristics of the machine.
- How to test `inf` and `nan`? In R, `is.nan()`, `is.finite()`, `is.infinite()`. In MATLAB, `isinf()`, `isnan()`. In JULIA, `isinf()`, `isnan()`.

Integers and floating-point numbers in Julia

- R only uses double (64-bit) and 32-bit integer. It can be a downside when dealing with big data.

- Julia offers a rich collection of primitive data types. Read (the excellent) documentation. `http://docs.julialang.org/en/release-0.4/manual/integers-and-floating-`

  Notable things include: `Int128`, `UInt128`, half precision numbers `Float16`, arbitrary precision numbers `BigInt` and `BigFloat`, rational numbers, ...

Consequences of computer storage/arithmetic

- Be memory conscious when dealing with big data. E.g., human genome has about 3×10^9 bases, each of which belongs to $\{A, C, T, G\}$. How much storage if we store 10^6 SNPs (single nucleotide polymorphisms) of 1000 individuals (1000 Genome Project) as characters (1GB), single (4GB), double (8GB), int64(8GB), int32 (4GB), int16 (2GB), int8 (1GB), PLINK binary format 2bit/SNP (250MB)?

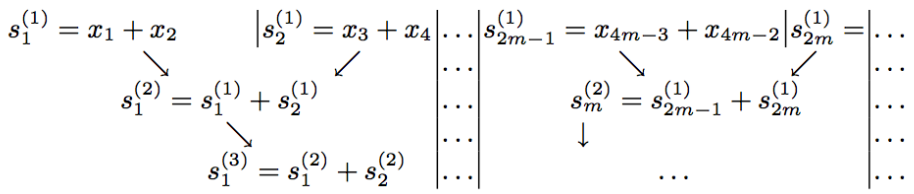
- Know the limit. *Overflow* and *Underflow*. For double precision, $\pm 10^{\pm 308}$. In most situations, underflow is preferred over overflow. Overflow often causes crashes. Underflow yields zeros. E.g., in logistic regression, $p_i = \frac{\exp(\boldsymbol{x}_i^T \boldsymbol{\beta})}{1+\exp(\boldsymbol{x}_i^T \boldsymbol{\beta})} = \frac{1}{1+\exp(-\boldsymbol{x}_i^T \boldsymbol{\beta})}$. The former expression can easily lead to $\infty/\infty = NaN$, while the latter expression leads to *graceful underflow*.

- Be aware of non-uniform distribution of floating-point numbers, in contrast to fixed-point numbers. There are the same number of floating-point numbers in $[b^i, b^{i+1}]$ and $[b^{i+1}, b^{i+2}]$ for $e_{\min} \le i \le e_{\max} - 2$. It is more dense when closer to zero.

- "Catastrophic cancellation 1". Addition or subtraction of two numbers of widely different magnitudes: $a + b$ or $a - b$ where $a \gg b$ or $a \ll b$. We loose the precision in the number of smaller magnitude. Consider $a = x.xxx... \times 2^0$ and $b = y.yyy... \times 2^{-53}$. What happens when computer calculates $a + b$? We get $a + b = a$!

- Another example: What happens when compute $\sum_{x=1}^{\infty} x$ in order? Will the partial sum reach `Inf`? "A divergent series converges."

- Always try to add numbers of similar magnitude. Rule 1: add small numbers together before adding larger ones. Rule 2: add numbers of like magnitude together (paring). When all numbers are of same sign and similar magnitude, add in pairs so each stage the summands are of similar magnitude.



- "Catastrophic cancellation 2". Subtraction of two nearly equal numbers eliminates significant digits. $a - b$ where $a \approx b$. Consider $a = x.xxxxxxxxxx1ssss$, $b = x.xxxxxxxxxx0tttt$. The result is $1.vvvvu...u$ where u are unassigned digits.

- E.g., evaluating e^{-20} by Taylor series

$$e^{-x} = 1 - x + x^2/2! - x^3/3! + \cdots$$

  gets `6.138e-09`, while the true value is about `2.061e-09`. Many cancellations accumulate. Anyway, it's *not* the way to evaluate e^x!

- Sometimes catastrophic cancellation can be avoided. Roots of the quadratic function $ax^2 + bx + c$ are

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

  When one root is close to 0, cancellation can happen. We may evaluate one of the root (away from 0) by the formula and then compute the other by relationship $x_1 x_2 = c/a$.

- Reading: *What every computer scientist should know about floating-point arithmetic* by David Goldberg.
  `http://hua-zhou.github.io/teaching/biostatm280-2016winter/readings/Goldberg91FloatingPoint.pdf`

References

Diaconis, P. (2009). The Markov chain Monte Carlo revolution. *Bull. Amer. Math. Soc. (N.S.)*, 46(2):179–205.

Golub, G. H. and Van Loan, C. F. (2013). *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, fourth edition.

Huber, P. J. (1994). Huge data sets. In *COMPSTAT 1994 (Vienna)*, pages 3–13. Physica, Heidelberg.

Huber, P. J. (1996). Massive data sets workshop: The morning after. In *Massive Data Sets: Proceedings of a Workshop*, pages 169–184. National Academy Press, Washington.

Knuth, D. E. (2005). *The Art of Computer Programming. Vol. 1. Fasc. 1.* Addison-Wesley, Upper Saddle River, NJ. MMIX, a RISC computer for the new millennium.

Lange, K. (2010). *Numerical Analysis for Statisticians*. Statistics and Computing. Springer, New York, second edition.

Stigler, S. M. (1986). *The History of Statistics*. The Belknap Press of Harvard University Press, Cambridge, MA. The measurement of uncertainty before 1900.

Teets, D. and Whitehead, K. (1999). The discovery of Ceres: how Gauss became famous. *Math. Mag.*, 72(2):83–93.