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

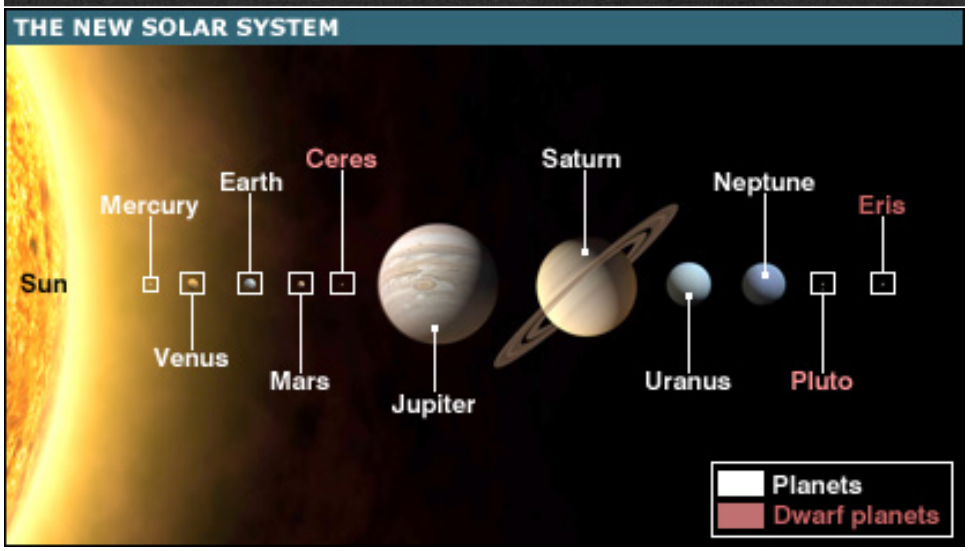
- Introduction and course logistics
- Computer storage and arithmetic
- Homework (not graded): fill out a short survey at https://www.surveymonkey.com/r/G8BCVVM
- Homework (not graded): Read the Introduction and Foundations of *Advanced R* by Hadley Wickham http://adv-r.had.co.nz. Install R. Try examples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

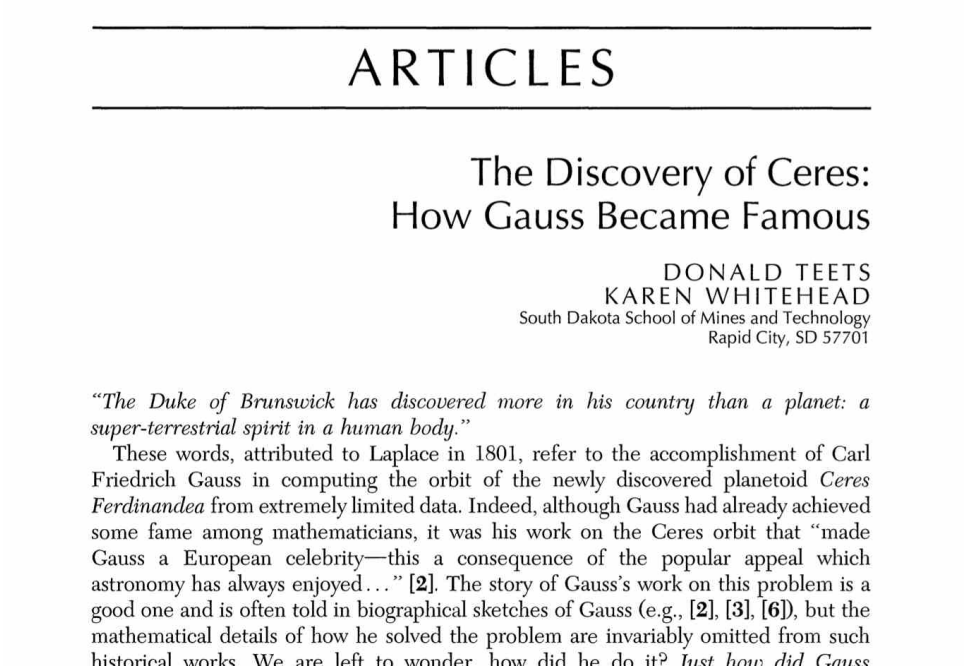


- 1801, *Dr. Carl Friedrich Gauss*, 24; proved Fundamental Theorem of Algebra; wrote the book *Disquisitiones Arithmetic*, which is still being studied today.
- 1801, Jan 1-Feb 11 (41 days), astronomer Piazzi observed Ceres (a dwarf planet), which was then lost behind sun.
- 1801, Aug–Sep, futile search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

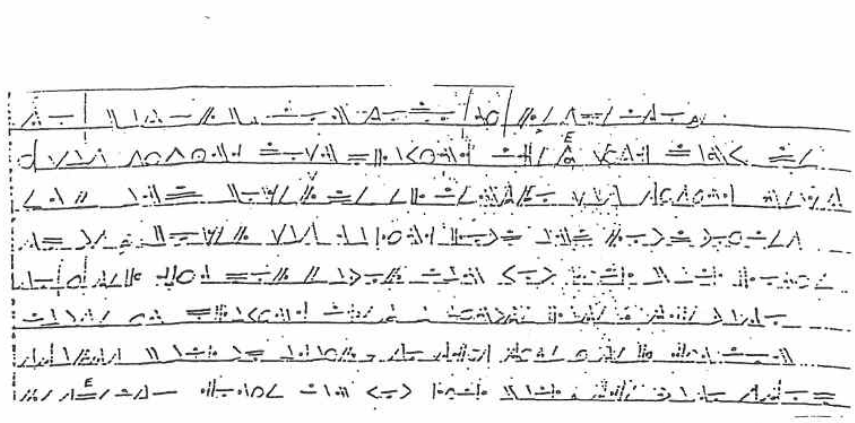
These words, attributed to Laplace in 1801, refer to the accomplishment of Carl Friedrich Gauss in computing the orbit of the newly discovered planetoid *Ceres Ferdinandea* from extremely limited data. Indeed, although Gauss had already achieved some fame among mathematicians, it was his work on the 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?

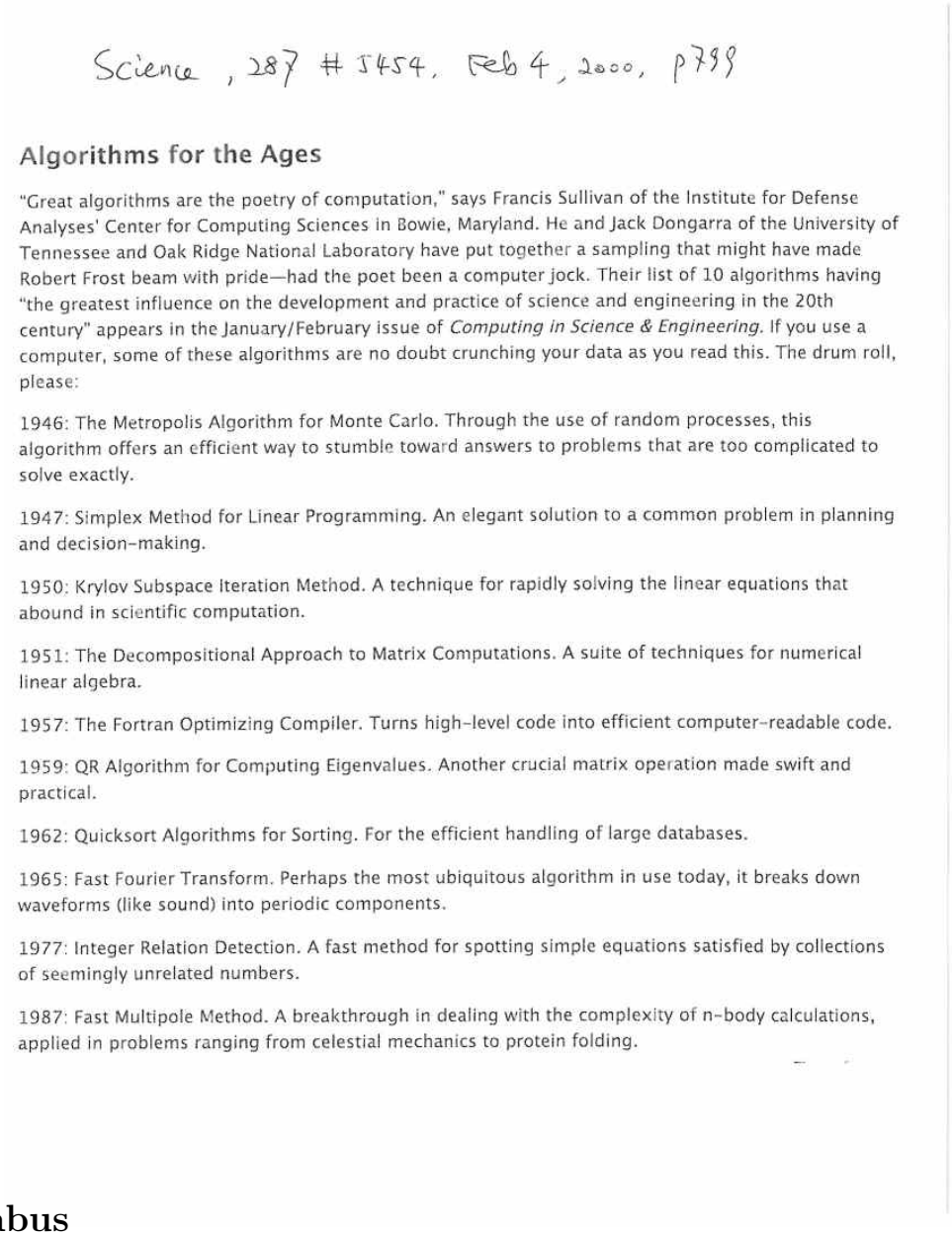
- Not a course on "packages and languages for data analysis". It does not answer questions such as "How to fit a linear mixed model in SAS, SPSS or R?"
- Not a programming course, although programming is *extremely* important and we do homework in R.
- This course is about "numerical methods in statistics".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

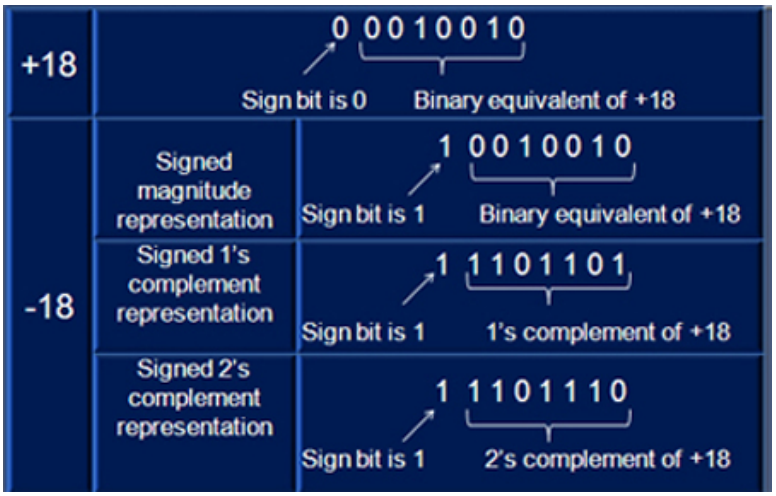
- Computer representation and arithmetic of integers: fixed-point numbers.
- Computer representation and arithmetic of real numbers: floating-point numbers.

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.

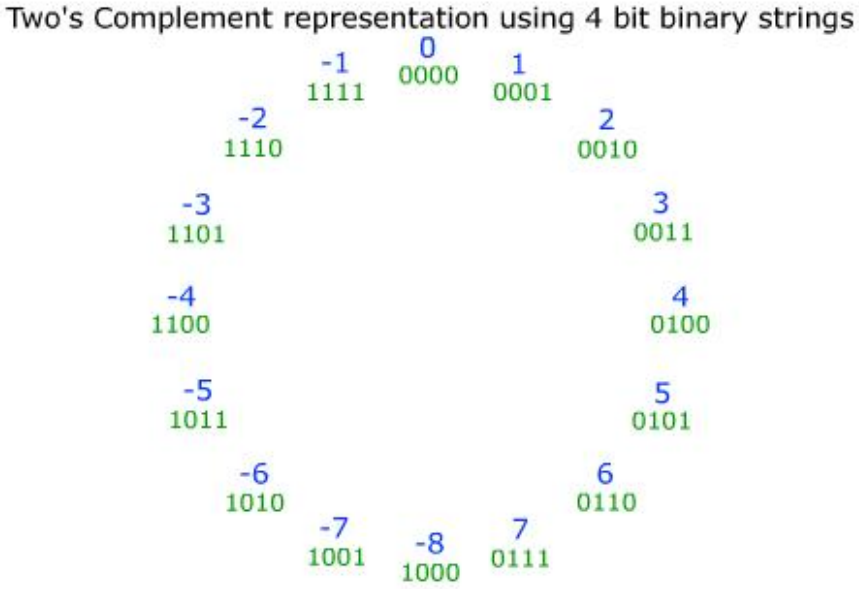


- Range of representable integers by M-bit storage is $[-2^{M-1}, 2^{M-1}-1]$ (don't need to represent 0 anymore so *could* have capacity for 2^{M-1} negative numbers).

- 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 $[0, 2^M - 1]$.

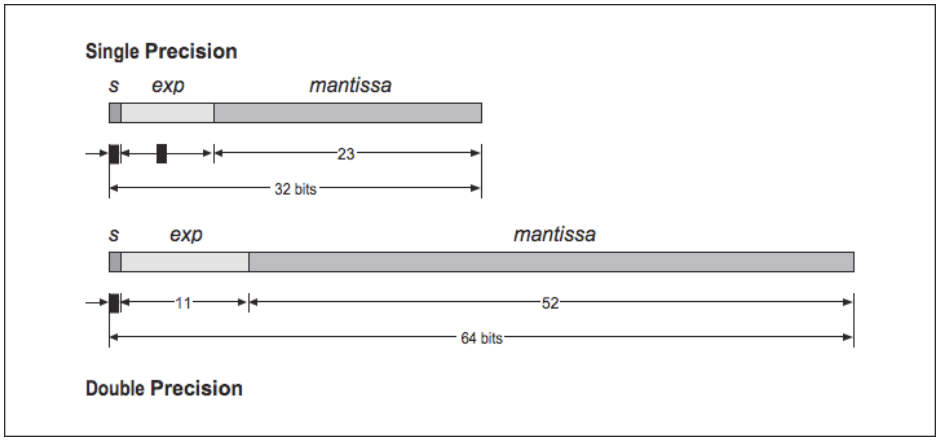


- An integer $-i$ in the interval $[-2^{M-1}, -1]$ would be represented by the same bit pattern by which the nonnegative integer $2^M - i$ is represented, treating the sign bit as a regular numeric bit. For example, (-18) = 11101110, $(2^8) - (18)$=(238)=11101110. Two's complement is subtracting the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

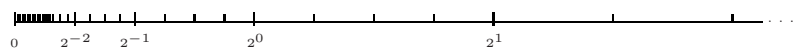


Fig. 2.4. The Floating-Point Number Line, Nonnegative Half

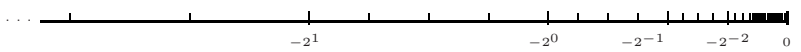


Fig. 2.5. The Floating-Point Number Line, Nonpositive Half

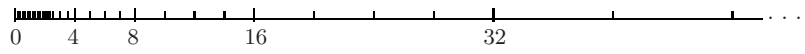


Fig. 2.6. The Floating-Point Number Line, Nonnegative Half; Another View

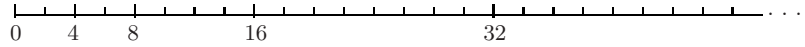


Fig. 2.7. The Fixed-Point Number Line, Nonnegative Half

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

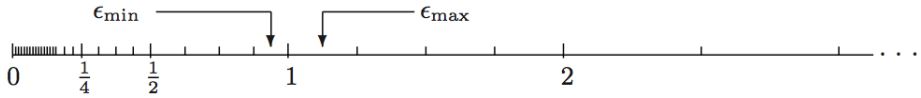


Fig. 2.8. Relative Spacings at 1: "Machine Epsilons"

- 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)?

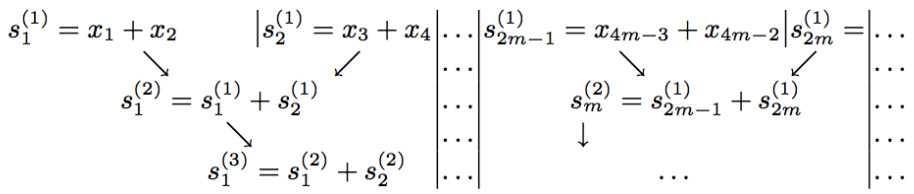
- Know the limit. *Overflow* and *Underflow*. For double precision, $\pm 10^{\pm 308}$. In most situations, underflow is preferred over overflow. Overflow often causes crashes. Underflow yields zeros. E.g., in logistic regression, $p_i = \frac{\exp(\boldsymbol{x}_i^T \boldsymbol{\beta})}{1+\exp(\boldsymbol{x}_i^T \boldsymbol{\beta})} = \frac{1}{1+\exp(-\boldsymbol{x}_i^T \boldsymbol{\beta})}$. 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`

3 Lecture 3, Jan 12

Announcements

- HW1 posted. Due Thu Jan 21 @ 11:59PM. `http://hua-zhou.github.io/teaching/biostatm280-2016winter/biostat_m280_2016_hw1.pdf`
- Quiz 1 Thu Jan 21 in class.
- Teaching assistant (TA): Max Tolkoff `mtolkoff@ucla.edu`, office hours M @ 11A–12P, W @ 2P–3P, at the common area of the 6th floor of Gonda.

Last time

- Computer arithmetics: fixed-point numbers and floating-point numbers.
- Consequences of computer arithmetics: range and precision of single/double precision numbers, cancellations, ...

Today

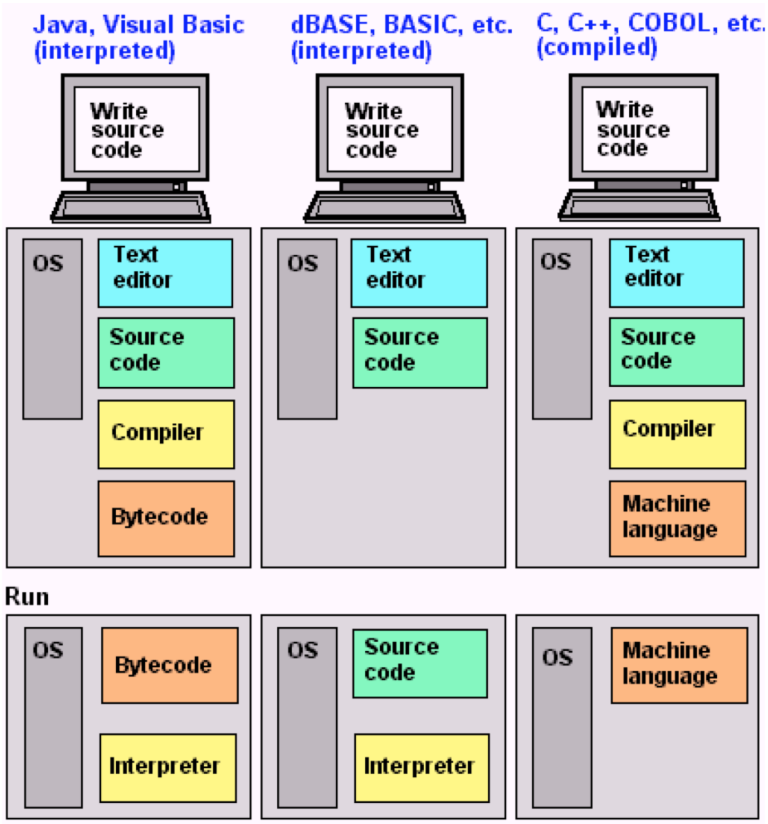
- Computer languages.
- Comparison of R, MATLAB, and JULIA.
- R, RStudio, RMarkdown, version control.

Computer Languages

> **工欲善其事，必先利其器**
>
> *To do a good job, an artisan needs the best tools.*
>
> *The Analects* by Confucius (about 500 BC)

- What features are we looking for in a language?
  - Efficiency (in both run time and memory) for handling big data.
  - IDE support (debugging, profiling).

- Open source.
- Legacy code.
- Tools for generating dynamic report (reproducibility).
- Adaptivity to hardware evolution (parallel and distributed computing).

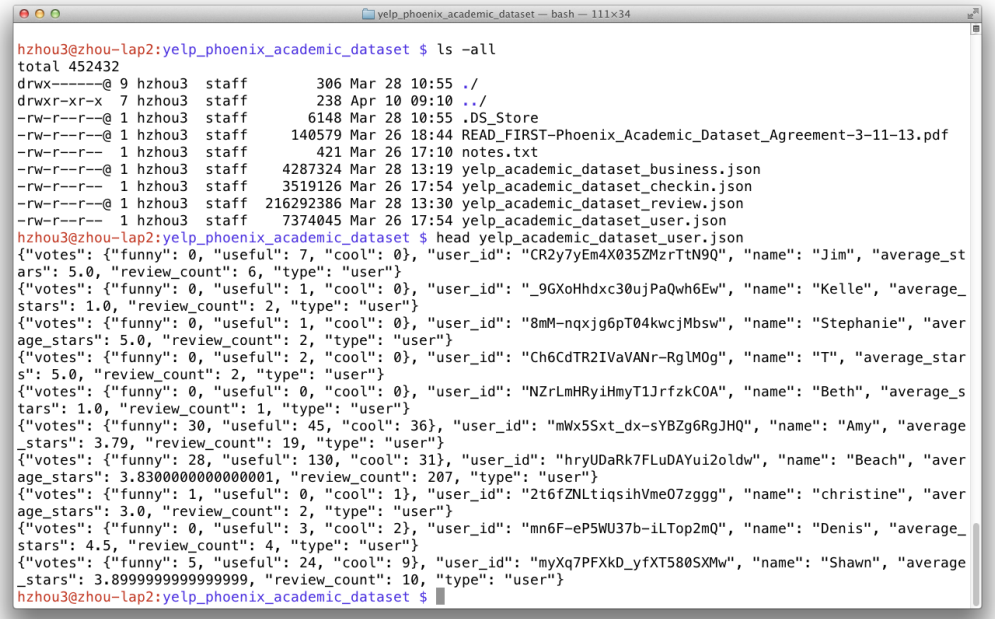


- Types of languages

1. Compiled languages: C/C++, Fortran, ...
   - Directly compiled to machine code that is executed by CPU
   - Pros: fast, memory efficient
   - Cons: longer development time, hard to debug
2. Interpreted language: R, Matlab, SAS IML, JavaScript, BASIC, ...
   - Interpreted by interpreter
   - Pros: fast prototyping

  - Cons: excruciatingly slow for loops

3. Mixed (dynamic) languages: Julia, Python, JAVA, Matlab (JIT), R (JIT), ...
   - Compiled to bytecode and then interpreted by virtual machine
   - Pros: relatively short development time, cross-platform, good at data preprocessing and manipulation, rich libraries and modules
   - Cons: not as fast as compiled language
4. Script languages: shell scripts, Perl, ...
   - Extremely useful for data preprocessing and manipulation
   - E.g., massage the Yelp (`http://www.yelp.com/dataset_challenge`) data before analysis



5. Database languages: SQL, Hadoop. Data analysis never happens if we do not know how to retrieve data from databases.

- Messages
  - To be versatile in the big data era, be proficient in at least one language in each category.

  - To improve efficiency of interpreted languages such as R or MATLAB, avoid loops as much as possible. Aka, vectorize code
    "The only loop you are allowed to have is that for an iterative algorithm."
    For some tasks where looping is necessary, consider coding in C or FORTRAN. It is "convenient" to incorporate compiled code into R or MATLAB. But do this **only after profiling!**
    Success stories: `glmnet` and `lars` packages in R are based on FORTRAN.
  - Modern languages such as Julia are trying to bridge the gap between interpreted and compiled languages. That is to achieve efficiency without vectorizing code.

- Language features of R, MATLAB, and JULIA:

Features	R	MATLAB	JULIA
Open source	☺	☹	☺
IDE	RStudio ☺☺	☺☺☺	☹
Dynamic document	☺☺☺(RMarkdown)	☺☺☺	☺☺(IJulia)
Multi-threading	`parallel` pkg	☺	☺
JIT	`compiler` pkg	☺	☺
Call C/Fortran	wrapper, Rcpp	wrapper	no glue code
Call shared library	wrapper	wrapper	no glue code
Typing	☹	☺☺	☺☺☺
Pass by reference	☹	☹	☺☺☺
Linear algebra	☹	MKL, Arpack	OpenBLAS, Eigpack
Distributed computing	☹	☺	☺☺☺
Sparse linear algebra	☹ (`Matrix` package)	☺☺☺	☺☺☺
Documentation	☹	☺☺☺	☺☺
Profiler	☹	☺☺☺	☺☺☺

- Benchmark code `R-benchmark-25.R` from `http://r.research.att.com/benchmarks/R-benchmark-25.R` covers many commonly used numerical operations used in statistics. We ported (literally) to MATLAB and Julia and report the run times (averaged over 5 runs) here.

Matlab benchmark code:
`http://hua-zhou.github.io/teaching/biostatm280-2016winter/benchmark_matlab.m`
Julia benchmark code:
`http://hua-zhou.github.io/teaching/biostatm280-2016winter/benchmark_julia.jl`

Machine specs: Intel i7 @ 2.6GHz (4 physical cores, 8 threads), 16G RAM, Mac OS 10.11.2.

Test	R 3.2.2	MATLAB R2014a	JULIA 0.4.2
Matrix creation, trans, deformation (2500×2500)	0.77	0.17	**0.14**
Power of matrix (2500×2500, $A.^{1000}$)	0.26	**0.11**	0.21
Quick sort ($n = 7 \times 10^6$)	0.76	**0.24**	0.69
Cross product (2800×2800, A^TA)	10.72	0.35	**0.31**
LS solution ($n = p = 2000$)	1.28	**0.07**	0.09
FFT ($n = 2,400,000$)	0.40	**0.04**	0.64
Eigen-values (600×600)	0.82	**0.31**	0.57
Determinant (2500×2500)	3.76	0.18	**0.17**
Cholesky (3000×3000)	4.23	**0.15**	0.20
Matrix inverse (1600×1600)	3.37	**0.16**	0.21
Fibonacci (vector calc)	0.34	**0.17**	0.68
Hilbert (matrix calc)	0.21	0.07	**0.01**
GCD (recursion)	0.31	0.14	**0.12**
Toeplitz matrix (loops)	0.36	**0.0014**	0.02
Escoufiers (mixed)	0.45	0.40	**0.21**

- A slightly more complicated (or realistic) example taken from Doug Bates's slides `http://www.stat.wisc.edu/~bates/JuliaForRProgrammers.pdf`. The task is to use Gibbs sampler to sample from bivariate density

$$f(x, y) = kx^2 \exp(-xy^2 - y^2 + 2y - 4x), x > 0,$$

using the conditional distributions

$$\begin{aligned} X|Y &\sim \Gamma\left(3, \frac{1}{y^2+4}\right) \\ Y|X &\sim \mathcal{N}\left(\frac{1}{1+x}, \frac{1}{2(1+x)}\right). \end{aligned}$$

Let's sample 10,000 points from this distribution with a thinning of 500.

- How long does R take? 42 seconds.
  `http://hua-zhou.github.io/teaching/biostatm280-2016winter/gibbs_r.html`
- How long does Julia take? 0.43 seconds.
  `http://hua-zhou.github.io/teaching/biostatm280-2016winter/gibbs_julia.html`
- With similar coding efforts, Julia offers $\sim$ 100 fold speed-up! JIT in R didn't kick in. Neither does Matlab, which took about 20 seconds.
- Julia offers the capability of strong typing of variables. This facilitates the optimization by compiler.
- With little extra efforts, we can do parallel and distributed computing using Julia.

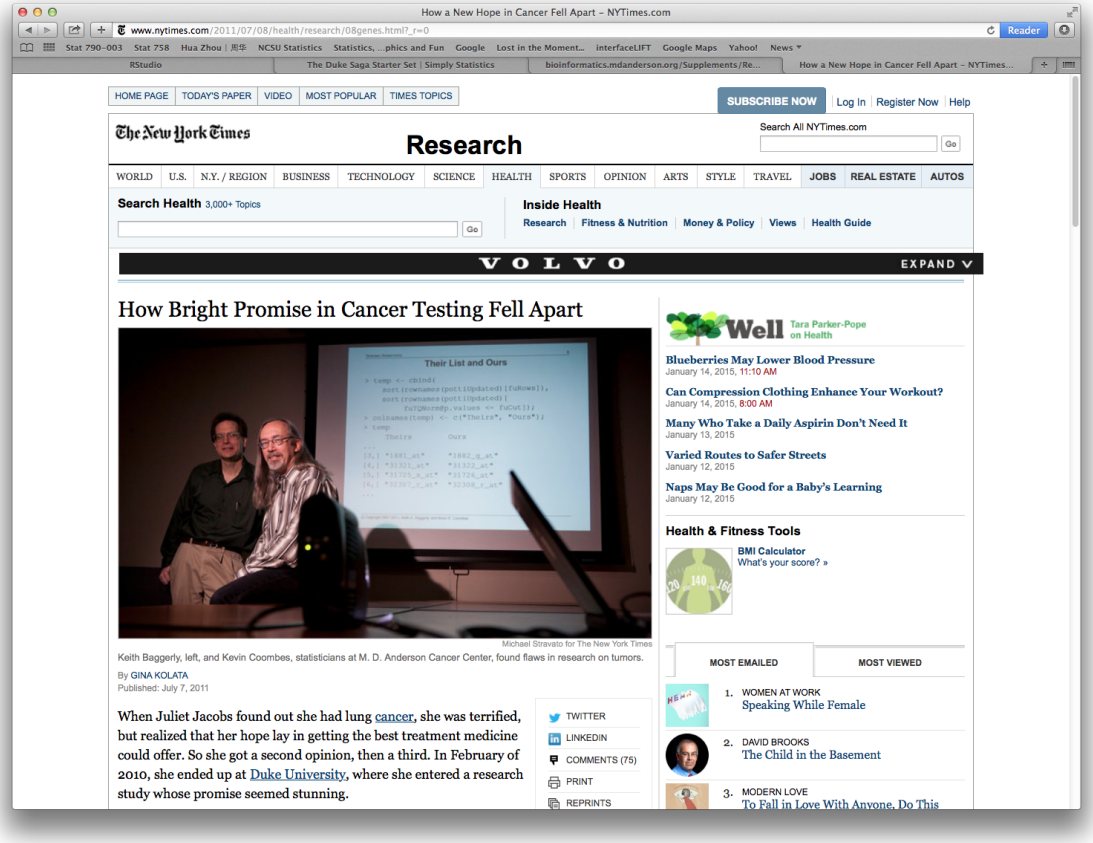
Benchmark of the same example in other languages including Rcpp is available in the blogs by Darren Wilkinson (`http://bit.ly/IWhJ52`) and Dirk Eddelbuettel's (`http://dirk.eddelbuettel.com/blog/2011/07/14/`).

> *"As some of you may know, I have had a (rather late) mid-life crisis and run off with another language called Julia.* ***`http://julialang.org`*** *"*
>
> Doug Bates (on the `knitr` Google Group)

Reproducible research (in computational science)

> *An article about computational result is* **advertising**, **not** *scholarship. The actual scholarship is the full software environment, code and data, that produced the result.*
>
> Buckheit and Donoho (1995)
> also see Claerbout and Karrenbach (1992)

- 3 stories of *not* being reproducible.
  - Duke Potti Scandal.



Potti et al. (2006) Genomic signatures to guide the use of chemotherapeutics, *Nature Medicine*, 12(11):1294–1300.

Baggerly and Coombes (2009) Deriving chemosensitivity from cell lines: Forensic bioinformatics and reproducible research in high-throughput biology, *Ann. Appl. Stat.*, 3(4):1309–1334. `http://projecteuclid.org/euclid.aoas/1267453942`

More information is available at

`http://en.wikipedia.org/wiki/Anil_Potti`

`http://simplystatistics.org/2012/02/27/the-duke-saga-starter-set/`

– Nature Genetics (2013 Impact Factor: 29.648). 20 articles about microarray profiling published in *Nature Genetics* between Jan 2005 and Dec 2006.

nature genetics

ANALYSIS

Repeatability of published microarray gene expression analyses

John P A Ioannidis[1–3], David B Allison[4], Catherine A Ball[5], Issa Coulibaly[4], Xiangqin Cui[4], Aedín C Culhane[6,7], Mario Falchi[8,9], Cesare Furlanello[10], Laurence Game[11], Giuseppe Jurman[10], Jon Mangion[11], Tapan Mehta[4], Michael Nitzberg[5], Grier P Page[4,12], Enrico Petretto[11,13] & Vera van Noort[14]

Given the complexity of microarray-based gene expression studies, guidelines encourage transparent design and public data availability. Several journals require public data deposition and several public databases exist. However, not all data are publicly available, and even when available, it is unknown whether the published results are reproducible by independent scientists. Here we evaluated the replication of data analyses in 18 articles on microarray-based gene expression profiling published in *Nature Genetics* in 2005–2006. One table or figure from each article was independently evaluated by two teams of analysts. We reproduced two analyses in principle

research, the Uniform Guidelines of the International Committee of Medical Journal Editors state that authors should "identify the methods, apparatus and procedures in sufficient detail to allow other workers to reproduce the results"[12]. Making primary data publicly available has many challenges but also many benefits[13]. Public data availability allows other investigators to confirm the results of the original authors, exactly replicate these results in other studies and try alternative analyses to see whether results are robust and to learn new things. Journals such as *Nature Genetics* require public data deposition as a prerequisite for publication for microarray-based research. Yet, the extent to which data are indeed made fully and accurately publicly available and permit con-

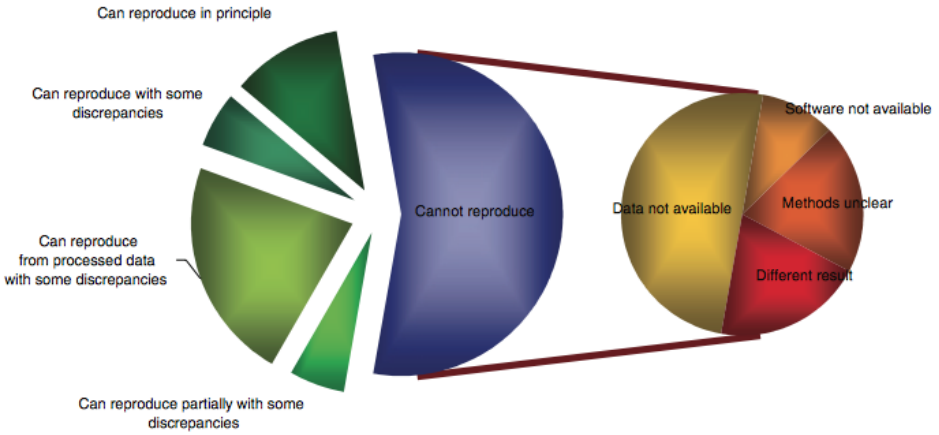
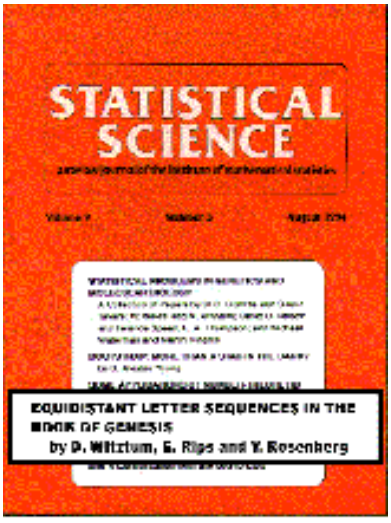
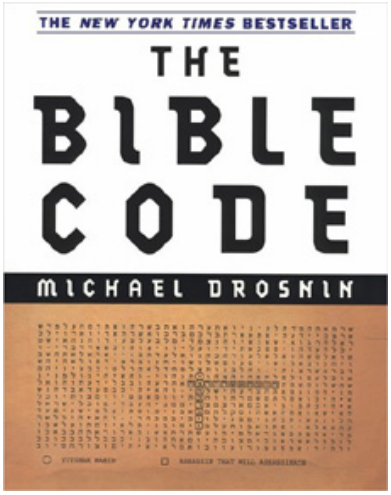


Figure 1 Summary of the efforts to replicate the published analyses.

– Bible code.





Witztum et al. (1994) Equidistant letter sequences in the book of genesis. *Statist. Sci.*, 9(3):429438. `http://projecteuclid.org/euclid.ss/1177010393`

McKay et al. (1999) Solving the Bible code puzzle, *Statist. Sci.*, 14(2):150–173.
`https://www.math.washington.edu/~greenber/BibleCode.html`

- Why reproducible research?
  - Replicability has been a foundation of science. It helps accumulate scientific knowledge.
  - Better work habit boosts quality of research.
  - Greater research impact.
  - Better teamwork. For you, it means better communication with your advisor (Buckheit and Donoho, 1995).
  - ...
- Readings.
  - Buckheit and Donoho (1995) Wavelab and reproducible research, in *Wavelets and Statistics*, volume 103 of *Lecture Notes in Statistics*, page 55–81.

Springer Newt York. `http://statweb.stanford.edu/~donoho/Reports/1995/wavelab.pdf`

Donoho (2010) An invitation to reproducible computational research, *Biostatistics*, 11(3):385-388.

- Peng (2009) Reproducible research and biostatistics, *Biostatistics*, 10(3):405–408.

  Peng (2011) Reproducible research in computational science, *Science*, 334(6060):1226–1227.

  Roger Peng's blogs *Treading a New Path for Reproducible Research*.
  `http://simplystatistics.org/2013/08/21/treading-a-new-path-for-reproducible-`
  `http://simplystatistics.org/2013/08/28/evidence-based-data-analysis-treading`
  `http://simplystatistics.org/2013/09/05/implementing-evidence-based-data-anal`

- *Reproducible research with R and RStudio* by Christopher Gandrud. It covers many useful tools: R, RStudio, LaTeX, Markdown, *knitr*, Github, Linux shell, ...

  This book is nicely reproducible. Git clone the source from `https://github.com/christophergandrud/Rep-Res-Book` and you should be able to compile into a pdf.

- *Reproducibility in Science* at
  `http://ropensci.github.io/reproducibility-guide/`

- How to be reproducible in statistics?

> *When we publish articles containing figures which were generated by computer, we also publish the complete software environment which generates the figures.*
>
> Buckheit and Donoho (1995)

  - For theoretical results, include all detailed proofs.
  - For data analysis or simulation study
    - Describe your computational results with painstaking details.

    - Put your code on your website or in an online supplement (required by many journals, e.g., *Biostatistics*, *JCGS*, ...) that allow replication of entire analysis or simulation study. A good example: `http://stanford.edu/~boyd/papers/admm_distr_stats.html`
    - Create a dynamic version of your simulation study/data analysis.

- What can we do *now*? At least make your homework reproducible!
  - Document everything!
  - Everything is a text file (.csv, .tex, .bib, .Rmd, .R, ...) They aid future proof and are subject to version control.

    Word/Excel are not text files.
  - All files should be human readable. Abundant comments and adopt a good style.
  - Tie your files together.
  - Use a dynamic document generation tool (weaving/knitting text, code, and output together) for documentation.
  - Use a version control system proactively.
  - Print `sessionInfo()` in R.

  For your homework, submit (put in the `master` branch) a final pdf or html report and all files and instructions necessary to reproduce all results.

- Tools for dynamic document/report generation.
  - R: RMarkdown, knitr, Sweave.
  - Matlab: automatic report generator.
  - Python: IPython, Pweave.
  - Julia: IJulia.

  For this course, please write homework report in RMarkdown (or IJulia if you use Julia).

4 Lecture 4, Jan 14

Last time

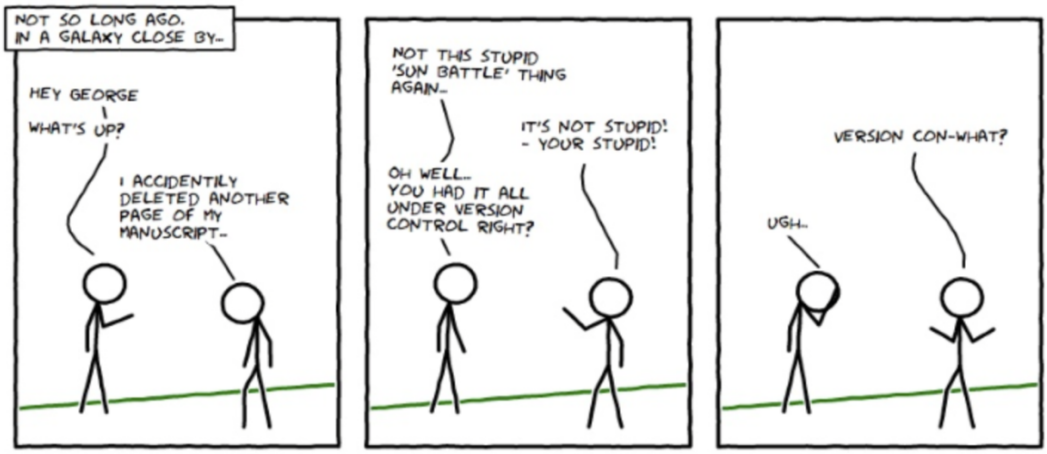
- Computer languages.
- Comparison of R, MATLAB, and JULIA.
- R, RStudio, RMarkdown.

Today

- Version control.

Version control by Git

If it's not in source control, it doesn't exist.



- Collaborative research. Statisticians, as opposed to "closet mathematicians", rarely do things in vacuum.
  - We talk to scientists/clients about their data and questions.
  - We write code (a lot!) together with team members or coauthors.
  - We run code/program on different platforms.
  - We write manuscripts/reports with co-authors.

  - ...

- 4 things distinguish professional programmers from amateurs:
  - *Use a version control system.*
  - Automate repetitive taks.
  - Systematic testing.
  - Use debugging aids rather than print statements.

- Why version control?
  - A centralized repository helps coordinate multi-person projects.
  - Time machine. Keep track of all the changes and revert back easily (reproducible).
  - Storage efficiency.
  - Synchronize files across multiple computers and platforms.
  - `github.com` is becoming a *de facto* central repository for open source development. E.g., all packages in Julia are distributed through `github.com`.
  - Advertise yourself thru `github.com`.

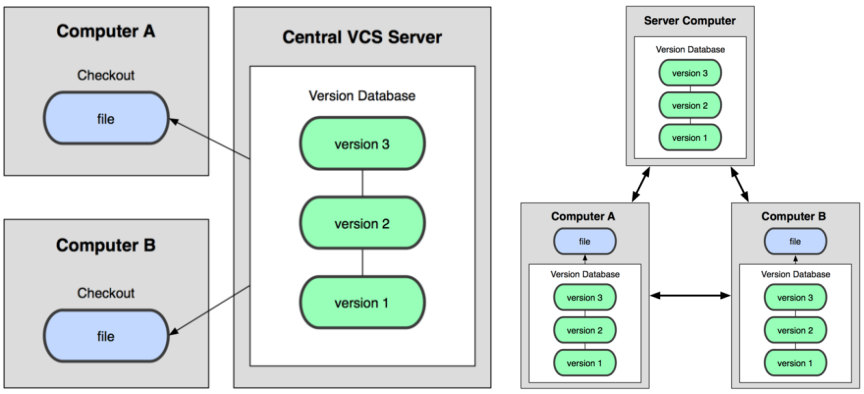
- Available version control tools.
  - Open source: cvs, subversion (aka svn), Git, ...
  - Proprietary: Visual SourceSafe (VSS), ...
  - Dropbox? Mostly for file back and sharing, limited version control (1 month?), ...

  We use Git in this course.

- Why Git?
  - The Eclipse Community Survey in 2014 shows Git is the most widely used source code management tool *now*. Git (33.3%) vs svn (30.7%).
  - History: Initially designed and developed by Linus Torvalds in 2005 for Linux kernel development. "`git`" is the British English slang for "unpleasant person".

> I'm an egotistical bastard, and I name all my projects after myself. First 'Linux', now 'git'.
>
> Linus Torvalds

  - A fundamental difference between svn (centralized version control system, left plot) and Git (distributed version control system, right plot):



  - Advantages of Git.
    * Free and open source.
    * Speed and simple (?) design.
    * Strong support for non-linear development (1000s of parallel branches). Scalable to large projects like the Linux kernel project.
    * Fully distributed. Fast, no internet required, disaster recovery,
  - Be aware that svn is still widely used in IT industry (Apache, GCC, SourceForge, Google Code, ...) and R development. E.g., type
    `svn log -v -l 5 https://svn.r-project.org/R`
    on command line to get a glimpse of what R development core team is doing. Good to master some basic svn commands.

- What do I need to use Git?
  - A Git server enabling multi-person collaboration through a centralized repository.
    * `github.com`: unlimited public repositories, private repositories costs $, academic user can get 5 private repositories for free.

    * `bitbucket.org`: unlimited private repositories for academic account (register for free using your UCLA email).
    * Some institutes and departments have their own git server.

  We use `bitbucket.org` in this course.

- Git client.
    * Linux: installed on many servers. If not, install on CentOS by `yum install git`.
    * Mac: install by `port install git`.
    * Windows: GitHub for Windows (GUI), TortoiseGIT (is this good?)

  Don't rely on GUI. Learn to use Git on command line.

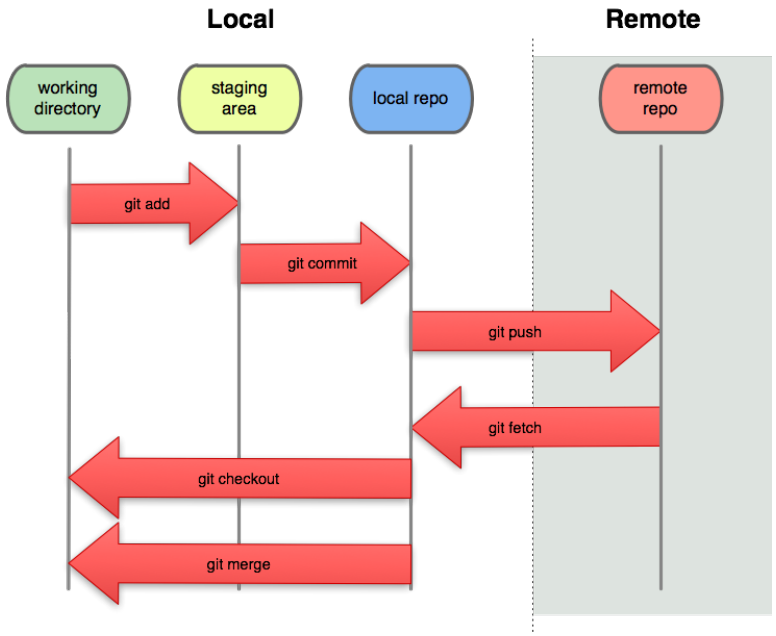
- Life cycle of a (research) project.
  Stage 1:
    - A project (idea) is started on `bitbucket.org`, with directories say `codebase`, `datasets`, `manuscripts`, `talks`, ...
    - Advantage of `bitbucket.org`: privacy of research ideas (free private repositories).
    - Downside of `bitbucket.org`: not as widely known as `github.com`.

  Stage 2:
    - Hopefully, research idea pans out and we want to distribute a standalone software development, e.g., an R package, repository at `github.com`. Read the (free) book *R Packages* (`http://r-pkgs.had.co.nz`) by Hadley Wickham for development of R packages and distribution via `github.com`.
    - This usually inherits from the `codebase` folder and happens when we submit a paper.
    - Challenges: keep all version history. It's doable.

  Stage 3:
    - Active maintenance of the public software repository.

  - At least three branches: `develop`, `master`, `gh-pages`.
    `develop`: main development area.
    `master`: software release.
    `gh-pages`: software webpage.

- Basic workflow of Git.



  - Synchronize local Git directory with remote repository (`git pull`).
  - Modify files in local working directory.
  - Add snapshots of them to staging area (`git add`).
  - Commit: store snapshots permanently to (local) Git repository (`git commit`).
  - Push commits to remote repository (`git push`).

- Basic Git usage.

  - Register for an account on a Git server, e.g., `bitbucket.org`. Fill out your profile, upload your public key to the server, ...
  - Identify yourself at local machine:
    `git config --global user.name "Hua Zhou"`
    `git config --global user.email "huazhou@ucla.edu"`
    Name and email appear in each commit you make.

- Initialize a project:
  - Create a repository, e.g., `biostat-m280-2016-winter`, on the server `bitbucket.org`. Then clone to local machine
    `git clone git@bitbucket.org:username/biostat-m280-2016-winter.git`
  - Alternatively use following commands to initialize a Git directory from a local folder and then push to the Git server
    `git init`
    `git remote add origin git@bitbucket.org:username/biostat-m280-2016-winter`
    `git push -u origin master`
- Edit working directory.

  `git pull` update local Git repository with remote repository (fetch + merge).

  `git status` displays the current status of working directory.

  `git log filename` displays commit logs of a file.

  `git diff` shows differences (by default difference from the most recent commit).

  `git add ...` adds file(s) to the staging area.

  `git commit` commits changes in staging area to Git directory.

  `git push` publishes commits in local Git directory to remote repository.

  Following demo session is on my local Mac machine.

```
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ pwd
/Users/hzhou3/github.ncsu/mglm
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ ls
.DS_Store    .gitignore   datasets/    manuscripts/
.git/        codebase/    literature/  talks/
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ git pull
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 5 (delta 3), reused 5 (delta 3)
Unpacking objects: 100% (5/5), done.
From github.ncsu.edu:hzhou3/vctest
   80be212..b22d29f  master     -> origin/master
Updating 80be212..b22d29f
Fast-forward
 manuscripts/letter-skat-famskat/Letter_to_the_editor.tex | 4 ++--
 1 file changed, 2 insertions(+), 2 deletions(-)
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ echo "hello st790 class" > gitdemo.txt
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ ls
.DS_Store    .gitignore   datasets/    literature/  talks/
.git/        codebase/    gitdemo.txt  manuscripts/
```

```
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ git add gitdemo.txt
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ git status .
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

   new file:   gitdemo.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

   codebase/Example_RNAseq_top100/
   codebase/MGLM/R/.Rhistory
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ git commit -m "git demo for st790 class"
[master ea636ff] git demo for st790 class
 1 file changed, 1 insertion(+)
 create mode 100644 gitdemo.txt
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ git log gitdemo.txt
commit ea636ff5665bc26bf8a79751b75d0e9d67bdb7d1
Author: Hua Zhou <hua_zhou@ncsu.edu>
Date:   Sun Jan 11 17:06:23 2015 -0500

    git demo for st790 class
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ git push
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 301 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To git@github.ncsu.edu:hzhou3/mglm.git
   77145d2..ea636ff  master -> master
```

`git reset --soft HEAD 1` undo the last commit.

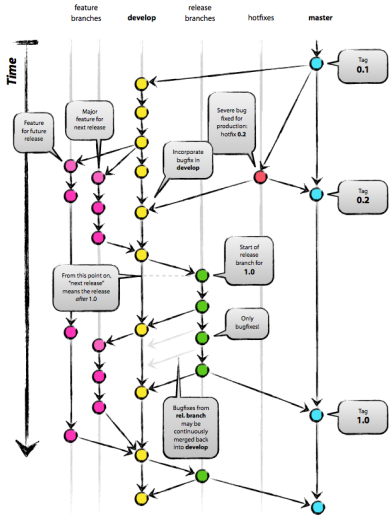
`git checkout filename` go back to the last commit.

`git rm` different from `rm`.

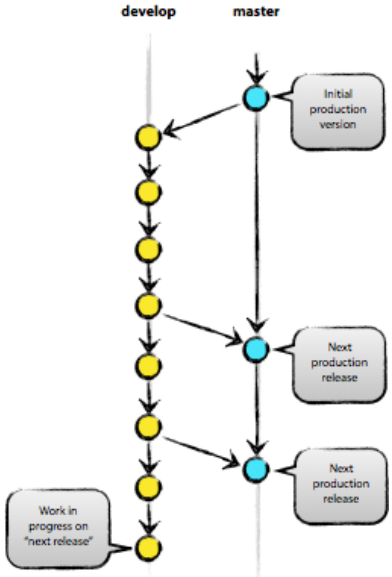
Although `git rm` deletes files from working directory. They are still in Git history and can be retrieved whenever needed. So always be cautious to put large data files or binary files into version control.

```
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ echo "bye st790 class" >> gitdemo.txt
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ cat gitdemo.txt
hello st790 class
bye st790 class
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ git diff gitdemo.txt
diff --git a/gitdemo.txt b/gitdemo.txt
index ece6d4e..2bb77f8 100644
--- a/gitdemo.txt
+++ b/gitdemo.txt
@@ -1 +1,2 @@
 hello st790 class
+bye st790 class
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ git checkout gitdemo.txt
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ cat gitdemo.txt
hello st790 class
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ git rm gitdemo.txt
rm 'gitdemo.txt'
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ git status .
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

   deleted:    gitdemo.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

   codebase/Example_RNAseq_top100/
   codebase/MGLM/R/.Rhistory
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ git commit -m "delete the git demo file for st790"
[master 4b4f9c5] delete the git demo file for st790
 1 file changed, 1 deletion(-)
 delete mode 100644 gitdemo.txt
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ git push
Counting objects: 2, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 238 bytes | 0 bytes/s, done.
Total 2 (delta 1), reused 0 (delta 0)
lTo git@github.ncsu.edu:hzhou3/mglm.git
   ea636ff..4b4f9c5  master -> master
hzhou3@Hua-Zhous-MacBook-Pro:mglm $ ls
.DS_Store    .gitignore   datasets/    manuscripts/
.git/        codebase/    literature/  talks/
```

- Branching in Git.
  - Branches in a project:



– For this course, you need to have two branches: `develop` for your own development and `master` for releases (homework submission). Note `master` is the default branch when you initialize the project; create and switch to `develop` branch immediately after project initialization.



– Commonly used commands:

`git branch branchname` creates a branch.

`git branch` shows all project branches.

`git checkout branchname` switches to a branch.

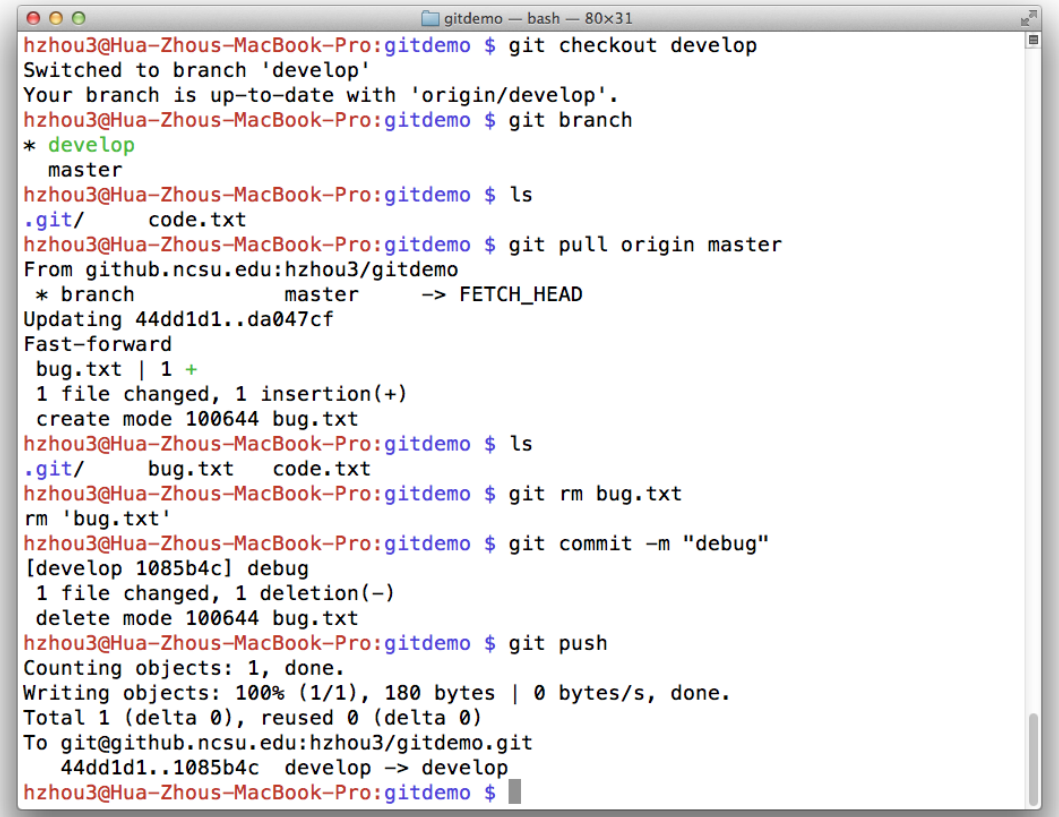
`git tag` shows tags (major landmarks).

`git tag tagname` creates a tag.

– Let's look at a typical branching and merging workflow.

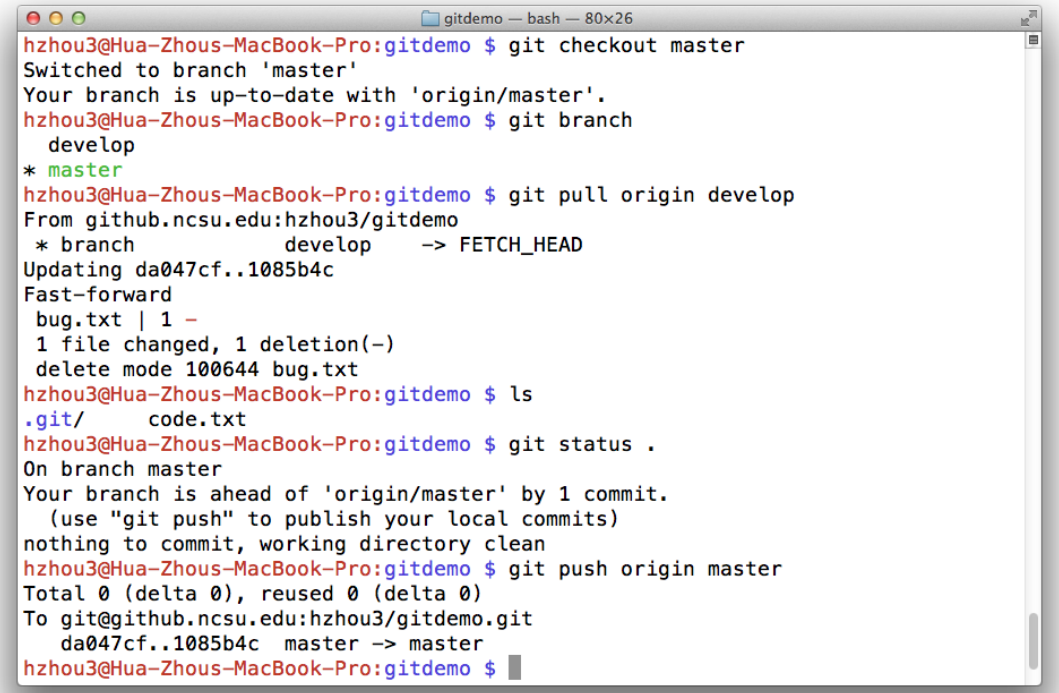
  * Now there is a bug in v0.0.3 ...

How to organize version number of your software? Read blog "R Package Versioning" by Yihui Xie
`http://yihui.name/en/2013/06/r-package-versioning/`

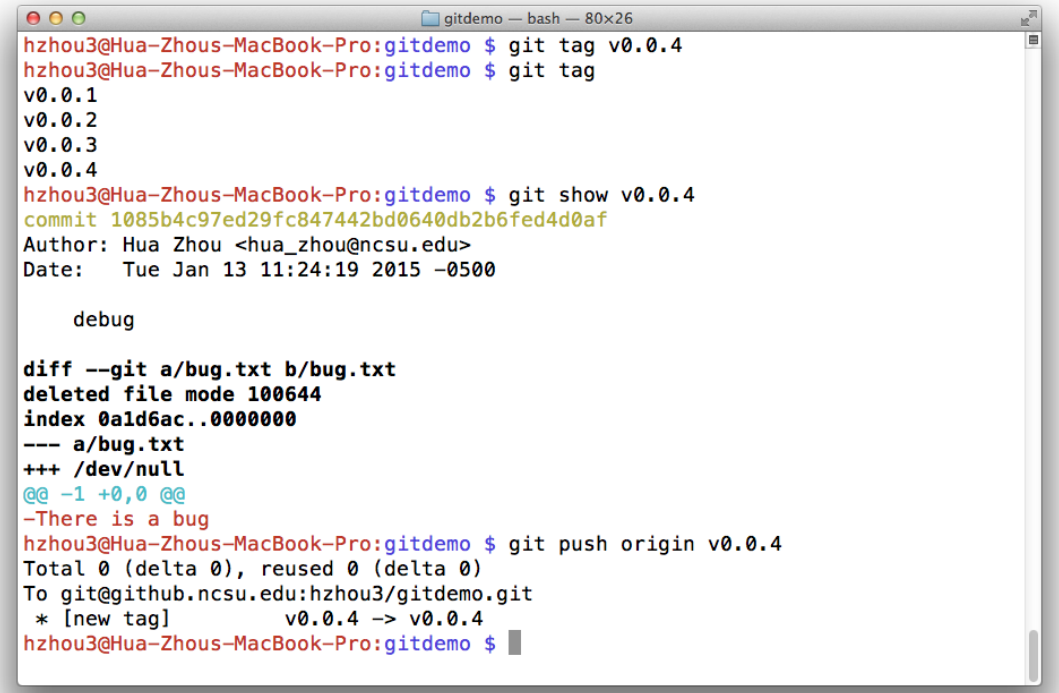


Now 'debug' in develop branch is ahead of master branch.

* Merge bug fix to the `master` branch.



* Tag a new release v0.0.4.



- Further resources:
  - Book *Pro Git*, `http://git-scm.com/book/en/v2`
  - Google
- Some etiquettes of using Git and version control systems in general.
  - Be judicious what to put in repository
    - Not too less: Make sure collaborators or yourself can reproduce everything on other machines.
    - Not too much: No need to put all intermediate files in repository.

    Strictly version control system is for source files only. E.g. only `xxx.tex`, `xxx.bib`, and figure files are necessary to produce a pdf file. Pdf file doesn't need to be version controlled or frequently committed.
  - "Commit early, commit often and don't spare the horses"

– Adding an informative message when you commit is *not* optional. Spending one minute now saves hours later for your collaborators and yourself. Read the following sentence to yourself 3 times:
"Write every commit message like the next person who reads it is an axe-wielding maniac who knows where you live."

5 Lecture 5, Jan 19

Announcements

- HW1 due this Thu @ 11:59PM.
- Quiz 1 this Thu, in class, closed-book.
- Demo of JIT in R. `http://hua-zhou.github.io/teaching/biostatm280-2016winter/jit.html`

Last time

- Version control.

Today

- Version control (cont'd).
- Algorithm: general concepts.
- Numerical linear algebra: introduction, BLAS.

Algorithms

- *Algorithm* is loosely defined as a set of instructions for doing something. Input $\rightarrow$ Output.
- Knuth (2005): (1) finiteness, (2) definiteness, (3) input, (4) output, (5) effectiveness
- Basic unit for measuring efficiency is flop. A *flop* (floating point operation) consists of a floating point addition, subtraction, multiplication, division, or comparison, and the usually accompanying fetch and store. Some books count multiplication followed by an addition (fused multiply-add, FMA) as one flop. This results a factor of 2 difference in flop counts.
- How to measure efficiency of an algorithm? Big O notation. If n is the size of a problem, an algorithm has order $O(f(n))$, where the leading term in the number of flops is $c \cdot f(n)$.

- E.g., matrix-vector multiplication `A %*% b`, where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^n$, takes $2mn$ or $O(mn)$ flops. Matrix-matrix multiplication `A %*% B`, where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n \times p}$, takes $2mnp$ or $O(mnp)$ flops.

- A hierarchy of computational complexity:
  Exponential order $O(b^n)$ (NP-hard="horrible")
  Polynomial order $O(n^q)$ (doable)
  $O(n \log n)$ (fast)
  Linear order $O(n)$ (fast)
  Log order $O(\log n)$ (super fast).

- One goal of this course is to get familiar with the flop counts for some common numerical tasks in statistics.

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

- Compare flops of the following two R expressions:

  ```
  G %*% Xt %*% y
  G %*% (Xt %*% y)
  ```

  where `G` $\in \mathbb{R}^{p \times p}$, `Xt` $\in \mathbb{R}^{p \times n}$, and `y` $\in \mathbb{R}^n$. "Matrix multiplication is expensive."

- Hardware advancement, e.g., CPU clock rate, only affects constant c. Unfortunately, data size n is increasing too and often at a faster rate.

- Classification of data sets by Huber (1994, 1996).

Data Size	Bytes	Storage Mode
Tiny	10^2	Piece of paper
Small	10^4	A few pieces of paper
Medium	10^6 (megabyte)	A floppy disk
Large	10^8	Hard disk
Huge	10^9 (gigabytes)	Hard disk(s)
Massive	10^{12} (terabytes)	RAID storage

- Difference of $O(n^2)$ and $O(n \log n)$ on massive data. Suppose we have a teraflop supercomputer capable of doing 10^{12} flops per second. For a problem of size $n = 10^{12}$, $O(n \log n)$ algorithm takes about $10^{12} \log(10^{12})/10^{12} \approx 27$ seconds. $O(n^2)$ algorithm takes about 10^{12} seconds, which is approximately 31710 years!

- QuickSort and FFT are celebrated algorithms that turn $O(n^2)$ operations into $O(n \log n)$. *Divide-and-conquer* is a powerful technique. Another example is the Strassen's method, which turns $O(n^3)$ matrix multiplication into $O(n^{\log_2 7})$.

Numerical linear algebra

- The first big chunk of this course is numerical linear algebra.

- Topics in numerical algebra: BLAS, solve linear equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, regression computations $\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{X}^T\boldsymbol{y}$, eigen-problems $\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x}$ and generalized eigen-problems $\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{B}\boldsymbol{x}$, singular value decompositions $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, iterative methods, ...

- Most of these numerical linear algebra tasks are implemented in the BLAS and LAPACK libraries. Our major goal is to (1) know the computational cost of each task, (2) be familiar with the BLAS and LAPACK functions (what they do), and (3) do *not* re-invent wheels by implementing these subroutines by yourself.

  All high-level languages (R, MATLAB, JULIA) call BLAS and LAPACK for numerical linear algebra. JULIA offers more flexibility by exposing interfaces to many BLAS/LAPACK subroutines directly. `http://docs.julialang.org/en/release-0.4/stdlib/linalg/?highlight=blas#module-Base.LinAlg.BLAS`

- Please review the following linear algebra facts by yourself.

Linear algebra review

Vector and matrix norms

KL chapter 6. Norm measures the "size".

- Vector norm $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$. (a) $\|\boldsymbol{x}\| \geq 0$, (b) $\|\boldsymbol{x}\| = 0$ if and only if $\boldsymbol{x} = \boldsymbol{0}$, (c) $\|c\boldsymbol{x}\| = c\|\boldsymbol{x}\|$ (homogeneity), (d) $\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$ (triangle inequality).

- ℓ_p norm. $\|\boldsymbol{x}\|_p = (\sum_i |x_i|^p)^{1/p}$, $p \in [1, \infty]$. ℓ_1 is the Manhattan norm. ℓ_2 is the Euclidean norm. ℓ_∞ is the sup norm.
- For matrix norm $\|\cdot\| : \mathbb{R}^{m\times n} \to \mathbb{R}$, we further require (e) $\|\boldsymbol{AB}\| \le \|\boldsymbol{A}\|\|\boldsymbol{B}\|$.
- Frobenius norm $\|\boldsymbol{A}\|_{\mathrm{F}} = (\sum_i \sum_j a_{ij}^2)^{1/2}$. Properties of (e) is checked by Cauchy-Schwartz inequality.
- Induced matrix norm (or operator norm): $\|\boldsymbol{A}\| = \sup_{\boldsymbol{x}\neq\boldsymbol{0}} \frac{\|\boldsymbol{Ax}\|}{\|\boldsymbol{x}\|} = \sup_{\|\boldsymbol{x}\|=1} \|\boldsymbol{Ax}\|$ for any fixed vector norm. To check property (e), let $\boldsymbol{y} = \boldsymbol{Bx}$, then $\|\boldsymbol{AB}\| = \sup_{\boldsymbol{x}\neq\boldsymbol{0}} \frac{\|\boldsymbol{Ay}\|}{\|\boldsymbol{y}\|} \frac{\|\boldsymbol{Bx}\|}{\|\boldsymbol{x}\|} \le \|\boldsymbol{A}\| \sup_{\boldsymbol{x}\neq\boldsymbol{0}} \frac{\|\boldsymbol{Bx}\|}{\|\boldsymbol{x}\|} = \|\boldsymbol{A}\|\|\boldsymbol{B}\|$.
- Matrix-1 norm, $\|\boldsymbol{A}\|_1 = \max_j \sum_i |a_{ij}|$.
  Matrix-2 norm, $\|\boldsymbol{A}\|_2 = \sqrt{\rho(\boldsymbol{A}^\intercal \boldsymbol{A})} = \max_{\|\boldsymbol{u}\|_2=1, \|\boldsymbol{v}\|_2=1} \boldsymbol{u}^\intercal \boldsymbol{A} \boldsymbol{v}$, which reduces to $\rho(\boldsymbol{A})$ if $\boldsymbol{A}$ is symmetric. ρ is the spectral radius of a matrix, the absolute value of the dominant eigenvalue.
  Matrix-∞ norm, $\|\boldsymbol{A}\|_\infty = \max_i \sum_j |a_{ij}|$.
- When $\boldsymbol{A}$ is a column vector, these matrix induced norms reduce to the original vector norm.
- $\rho(\boldsymbol{A}) \le \|\boldsymbol{A}\|$ for any induced matrix norm. For any $\boldsymbol{A}$ and $\epsilon > 0$, there exists an induced matrix norm such that $\|\boldsymbol{A}\| \le \rho(\boldsymbol{A}) + \epsilon$.

Rank

Assume $\boldsymbol{A} \in \mathbb{R}^{m\times n}$.

- $\mathrm{rank}(\boldsymbol{A})$ is the maximum number of linearly independent rows (or columns) of a matrix.
- $\mathrm{rank}(\boldsymbol{A}) \le \min\{m, n\}$.
- A matrix is *full rank* if $\mathrm{rank}(\boldsymbol{A}) = \min\{m, n\}$. It is *full row rank* if $\mathrm{rank}(\boldsymbol{A}) = m$. It is *full column rank* if $\mathrm{rank}(\boldsymbol{A}) = n$.
- A square matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is *singular* if $\mathrm{rank}(\boldsymbol{A}) < n$ and *non-singular* if $\mathrm{rank}(\boldsymbol{A}) = n$.

- $\text{rank}(\boldsymbol{AB}) \leq \min\{\text{rank}(\boldsymbol{A}), \text{rank}(\boldsymbol{B})\}$. "Matrix multiplication cannot increase the rank."
- $\text{rank}(\boldsymbol{A}) = \text{rank}(\boldsymbol{A}^T) = \text{rank}(\boldsymbol{A}^T\boldsymbol{A}) = \text{rank}(\boldsymbol{A}\boldsymbol{A}^T)$.
- $\text{rank}(\boldsymbol{AB}) = \text{rank}(\boldsymbol{A})$ if $\boldsymbol{B}$ has full row rank.
- $\text{rank}(\boldsymbol{AB}) = \text{rank}(\boldsymbol{B})$ if $\boldsymbol{A}$ has full column rank.
- $\text{rank}(\boldsymbol{A} + \boldsymbol{B}) \leq \text{rank}(\boldsymbol{A}) + \text{rank}(\boldsymbol{B})$.

Trace

$\boldsymbol{A} \in \mathbb{R}^{n \times n}$ a square matrix.

- $\text{tr}(\boldsymbol{A}) = \sum_{i=1}^{n} a_{ii}$
- $\text{tr}(\boldsymbol{A} + \boldsymbol{B}) = \text{tr}(\boldsymbol{A}) + \text{tr}(\boldsymbol{B})$
- $\text{tr}(\lambda\boldsymbol{A}) = \lambda\text{tr}(\boldsymbol{A})$ where λ is a scalar
- $\text{tr}(\boldsymbol{A}^\intercal) = \text{tr}(\boldsymbol{A})$
- Invariance under cycle permutation: $\text{tr}(\boldsymbol{AB}) = \text{tr}(\boldsymbol{BA})$. In general, $\text{tr}(\boldsymbol{A}_1 \cdots \boldsymbol{A}_k) = \text{tr}(\boldsymbol{A}_{j+1} \cdots \boldsymbol{A}_k \boldsymbol{A}_1 \cdots \boldsymbol{A}_j)$.

Orthogonality and orthogonalization

- $\boldsymbol{v}_1$ is *orthogonal* to $\boldsymbol{v}_2$, written $\boldsymbol{v}_1 \perp \boldsymbol{v}_2$, if $\langle \boldsymbol{v}_1, \boldsymbol{v}_2 \rangle = \boldsymbol{v}_1^\intercal \boldsymbol{v}_2 = 0$. They are *orthonormal* if $\boldsymbol{v}_1 \perp \boldsymbol{v}_2$ and $\|\boldsymbol{v}_i\|_2 = 1$, $i = 1, 2$.
- Gram-Schmidt transformation orthonormalizes two non-zero vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$.

$$\begin{aligned} \tilde{\boldsymbol{x}}_1 &= \frac{1}{\|\boldsymbol{x}_1\|_2} \boldsymbol{x}_1 \\ \tilde{\boldsymbol{x}}_2 &= \frac{1}{\|\boldsymbol{x}_2 - \langle \tilde{\boldsymbol{x}}_1, \boldsymbol{x}_2 \rangle \tilde{\boldsymbol{x}}_1\|_2} (\boldsymbol{x}_2 - \langle \tilde{\boldsymbol{x}}_1, \boldsymbol{x}_2 \rangle \tilde{\boldsymbol{x}}_1) \end{aligned}$$

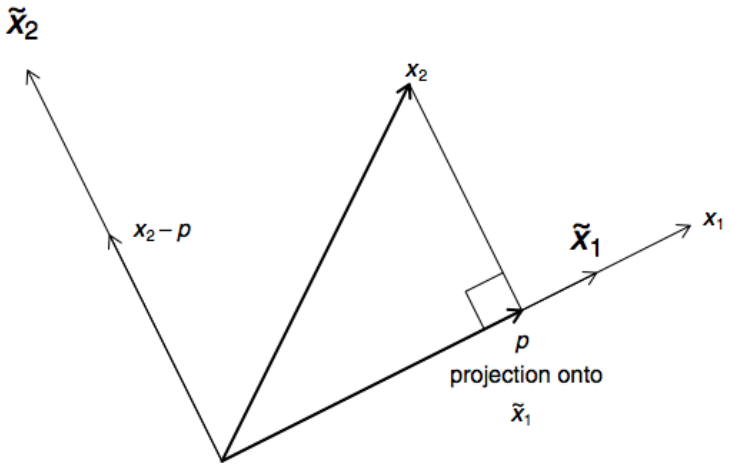


Fig. 2.2. Orthogonalization of x_1 and x_2

- A set of nonzero, mutually orthogonal vectors are linearly independent.
- A real square matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is orthogonal if $\boldsymbol{A}^\intercal \boldsymbol{A} = \boldsymbol{I}_n$, i.e., its rows/columns are orthonormal. Orthogonal matrix is of full rank, thus $\boldsymbol{A}^\intercal = \boldsymbol{A}^{-1}$ and $\boldsymbol{A}\boldsymbol{A}^\intercal = \boldsymbol{I}_n$.

Positive (semi)definite matrix

Assume $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is *symmetric*.

- A real symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is *positive semidefinite* (or *nonnegative definite*) if $\boldsymbol{x}^\intercal \boldsymbol{A}\boldsymbol{x} \geq 0$ for all $\boldsymbol{x}$. Notation: $\boldsymbol{A} \succeq \boldsymbol{0}_{n\times n}$.
- E.g., the *Gramian matrix* $\boldsymbol{X}^\intercal \boldsymbol{X}$ or $\boldsymbol{X}\boldsymbol{X}^T$.
- If inequality is strict for all $\boldsymbol{x} \neq \boldsymbol{0}$, then $\boldsymbol{A}$ is *positive definite*. Notation: $\boldsymbol{A} \succ \boldsymbol{0}_{n\times n}$.
- $\boldsymbol{A} \succeq \boldsymbol{B}$ means $\boldsymbol{A} - \boldsymbol{B} \succeq \boldsymbol{0}_{n\times n}$.
- If $\boldsymbol{A} \succeq \boldsymbol{B}$, then $\det(\boldsymbol{A}) \geq \det(\boldsymbol{B})$ with equality if and only if $\boldsymbol{A} = \boldsymbol{B}$.
- $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is positive semidefinite if and only if $\boldsymbol{A}$ is a covariance matrix of a random vector in $\mathbb{R}^n$.

Matrix inverses

Assume $\boldsymbol{A} \in \mathbb{R}^{m \times n}$.

- The *Moore-Penrose inverse* of $\boldsymbol{A}$ is a matrix $\boldsymbol{A}^+ \in \mathbb{R}^{n \times m}$ with following properties

  (a) $\boldsymbol{A}\boldsymbol{A}^+\boldsymbol{A} = \boldsymbol{A}$. (*Generalized inverse*, g_1 *inverse*, or *inner pseudo-inverse*)

  (b) $\boldsymbol{A}^+\boldsymbol{A}\boldsymbol{A}^+ = \boldsymbol{A}^+$. (*Outer pseudo-inverse*. Any g_1 inverse that satisfies this condition is called a g_2 *inverse*, or *reflexive generalized inverse* and is denoted by $\boldsymbol{A}^*$.)

  (c) $\boldsymbol{A}^+\boldsymbol{A}$ is symmetric.

  (d) $\boldsymbol{A}\boldsymbol{A}^+$ is symmetric.

- $\boldsymbol{A}^+$ exists and is unique for any matrix $\boldsymbol{A}$.

- *Generalized inverse* (or g_1 *inverse*, denoted by $\boldsymbol{A}^-$ or $\boldsymbol{A}^g$): property (a).

- g_2 *inverse* (denoted by $\boldsymbol{A}^*$): properties (a)+(b).

- *Moore-Penrose inverse* (denoted by $\boldsymbol{A}^+$): properties (a)+(b)+(c)+(d).

- If $\boldsymbol{A}$ is square and full rank, then the generalized inverse is unique and denoted by $\boldsymbol{A}^{-1}$ (inverse).

- In practice, the Moore-Penrose inverse $\boldsymbol{A}^+$ is easily computed from the singular value decomposition (SVD) of $\boldsymbol{A}$.

- $(\boldsymbol{A}^-)^T$ is a generalized inverse of $\boldsymbol{A}^T$.

- $\mathcal{C}(\boldsymbol{A}) = \mathcal{C}(\boldsymbol{A}\boldsymbol{A}^-)$ and $\mathcal{C}(\boldsymbol{A}^T) = \mathcal{C}((\boldsymbol{A}^-\boldsymbol{A})^T)$.
  $\text{rank}(\boldsymbol{A}) = \text{rank}(\boldsymbol{A}\boldsymbol{A}^-) = \text{rank}(\boldsymbol{A}^-\boldsymbol{A})$.
  "Multiplication by generalized inverse does not change rank."

- $\text{rank}(\boldsymbol{A}^-) \geq \text{rank}(\boldsymbol{A})$. "Generalized inverse has equal or a larger rank than the original matrix."

System of linear equations

$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^m$.

- When is there a solution? The following statements are equivalent.

  1. The linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ has a solution (*consistent*)
  2. $\boldsymbol{b} \in \mathcal{C}(\boldsymbol{A})$.
  3. $\text{rank}((\boldsymbol{A}, \boldsymbol{b})) = \text{rank}(\boldsymbol{A})$.
  4. $\boldsymbol{A}\boldsymbol{A}^{-}\boldsymbol{b} = \boldsymbol{b}$.

  The last equivalence gives intuition why $\boldsymbol{A}^{-}$ is called an inverse.

- What are the solutions to a homogeneous system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$?
  $\mathcal{N}(\boldsymbol{A}) = \mathcal{C}(\boldsymbol{I}_n - \boldsymbol{A}^{-}\boldsymbol{A})$.

- If $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is consistent, then $\tilde{\boldsymbol{x}}$ is a solution to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ if and only if

  $$\tilde{\boldsymbol{x}} = \boldsymbol{A}^{-}\boldsymbol{b} + (\boldsymbol{I}_n - \boldsymbol{A}^{-}\boldsymbol{A})\boldsymbol{q}$$

  for some $\boldsymbol{q} \in \mathbb{R}^n$.
  Interpretation: "a specific solution" + "a vector in the null space of $\boldsymbol{A}$".

- $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is consistent for *all* $\boldsymbol{b}$ if and only if $\boldsymbol{A}$ has full row rank.

- If a system is consistent, its solution is unique if and only if $\boldsymbol{A}$ has full column rank.

- If $\boldsymbol{A}$ has full row and column rank, then $\boldsymbol{A}$ is non-singular and the unique solution is $\boldsymbol{A}^{-1}\boldsymbol{b}$.

Gramian matrix $\boldsymbol{A}^{\intercal}\boldsymbol{A}$

- $\boldsymbol{A}^{\intercal}\boldsymbol{A}$ is symmetric and positive semidefinite.

- $\text{rank}(\boldsymbol{A}) = \text{rank}(\boldsymbol{A}^{\intercal}) = \text{rank}(\boldsymbol{A}^{\intercal}\boldsymbol{A}) = \text{rank}(\boldsymbol{A}\boldsymbol{A}^{\intercal})$.

- $\boldsymbol{A}^{\intercal}\boldsymbol{A} = \boldsymbol{0}$ if and only if $\boldsymbol{A} = \boldsymbol{0}$.

- $\boldsymbol{B}\boldsymbol{A}^{\intercal}\boldsymbol{A} = \boldsymbol{C}\boldsymbol{A}^{\intercal}\boldsymbol{A}$ if and only if $\boldsymbol{B}\boldsymbol{A}^{\intercal} = \boldsymbol{C}\boldsymbol{A}^{\intercal}$.

- $\boldsymbol{A}^\intercal\boldsymbol{A}\boldsymbol{B} = \boldsymbol{A}^\intercal\boldsymbol{A}\boldsymbol{C}$ if and only if $\boldsymbol{A}\boldsymbol{B} = \boldsymbol{A}\boldsymbol{C}$.
- For any generalized inverse $(\boldsymbol{A}^\intercal\boldsymbol{A})^-$, $[(\boldsymbol{A}^\intercal\boldsymbol{A})^-]^\intercal$ is also a generalized inverse of $\boldsymbol{A}^\intercal\boldsymbol{A}$. Note $(\boldsymbol{A}^\intercal\boldsymbol{A})^-$ is not necessarily symmetric.
- $(\boldsymbol{A}^\intercal\boldsymbol{A})^-\boldsymbol{A}^\intercal$ is a generalized inverse of $\boldsymbol{A}$.
- $\boldsymbol{A}\boldsymbol{A}^+ = \boldsymbol{A}(\boldsymbol{A}^\intercal\boldsymbol{A})^-\boldsymbol{A}^\intercal$, where $\boldsymbol{A}^+$ is the Moore-Penrose inverse of $\boldsymbol{A}$.
- $\boldsymbol{P}_{\boldsymbol{A}} = \boldsymbol{A}(\boldsymbol{A}^\intercal\boldsymbol{A})^-\boldsymbol{A}^\intercal$ is symmetric, idempotent, invariant to the choice of generalized inverse $(\boldsymbol{A}^\intercal\boldsymbol{A})^-$, and projects onto $\mathcal{C}(\boldsymbol{A})$.

Idempotent matrix and projection

Assume $\boldsymbol{P} \in \mathbb{R}^{n\times n}$.

- A matrix $\boldsymbol{P} \in \mathbb{R}^{n\times n}$ is *idempotent* if and only if $\boldsymbol{P}^2 = \boldsymbol{P}$.
- A matrix $\boldsymbol{P}$ is a *projection* on a vector space $\mathcal{V}$ if (a) $\boldsymbol{P}$ is idempotent, (b) $\boldsymbol{P}\boldsymbol{x} \in \mathcal{V}$ for all $\boldsymbol{x}$, and (c) $\boldsymbol{P}\boldsymbol{z} = \boldsymbol{z}$ for all $\boldsymbol{z} \in \mathcal{V}$.
- An idempotent matrix $\boldsymbol{P}$ is a projection onto $\mathcal{C}(\boldsymbol{P})$.
- For a *general* matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, the matrices $\boldsymbol{A}\boldsymbol{A}^-$ are projections onto $\mathcal{C}(\boldsymbol{A})$ and $\boldsymbol{I}_n - \boldsymbol{A}^-\boldsymbol{A}$ are projections onto $\mathcal{N}(\boldsymbol{A})$.

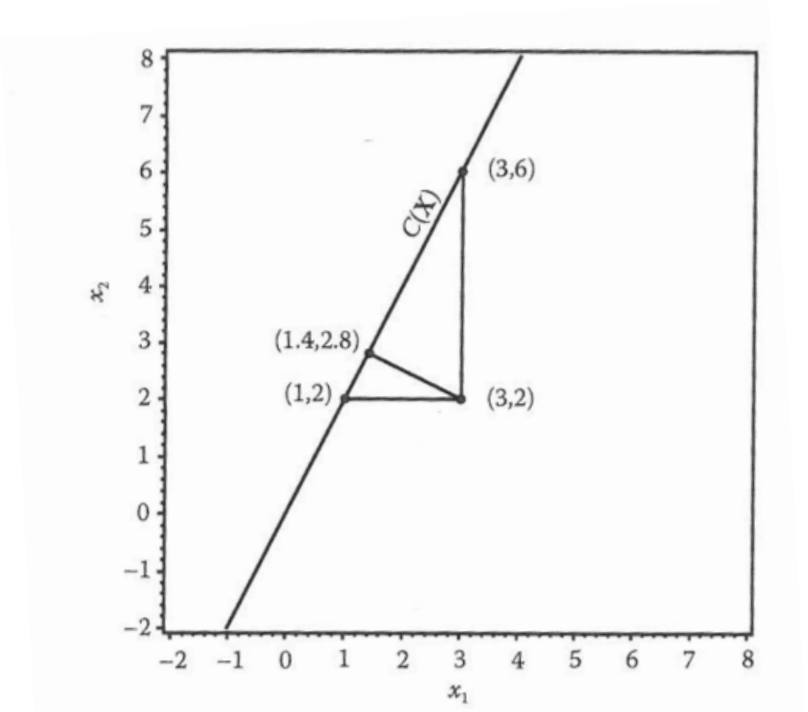


Figure A.1: Three projections onto a column space.

Symmetric idempotent matrix and orthogonal projection

Assume $\boldsymbol{A} \in \mathbb{R}^{n \times n}$.

- A symmetric, idempotent matrix is called an *orthogonal projection*.
- An orthogonal projection $\boldsymbol{P}$ satisfies $\boldsymbol{y} - \boldsymbol{P}\boldsymbol{y} \perp \boldsymbol{v}$ for all $\boldsymbol{v} \in \mathcal{C}(\boldsymbol{P})$.
- The orthogonal projection onto a vector space is unique.
- If a symmetric, idempotent matrix $\boldsymbol{P}$ projects onto $\mathcal{V}$, then $\boldsymbol{I} - \boldsymbol{P}$ projects onto the orthogonal complement $\mathcal{V}^{\perp}$.
- Pythagorean theorem: For $\boldsymbol{P}$ an orthogonal projection,

$$\|\boldsymbol{y}\|_2^2 = \|\boldsymbol{P}\boldsymbol{y}\|_2^2 + \|(\boldsymbol{I} - \boldsymbol{P})\boldsymbol{y}\|_2^2.$$

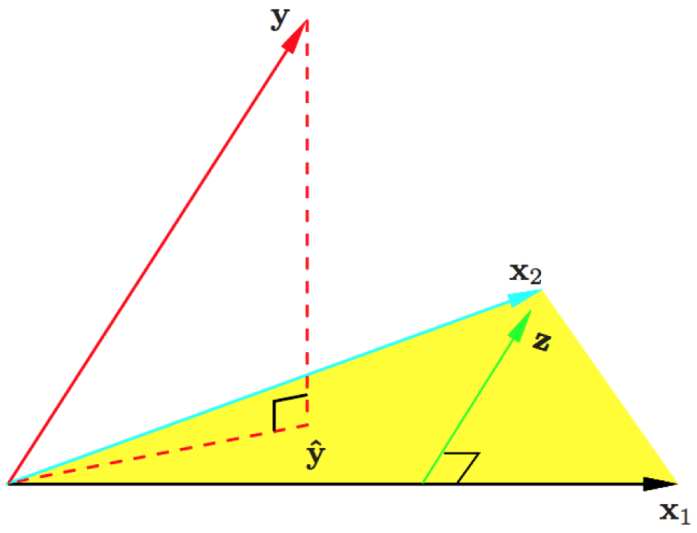
- Many books use the term "projection" in the sense of of orthogonal projection.

Method of least squares

- Goal: Approximate $\boldsymbol{y} \in \mathbb{R}^n$ by a linear combination of columns of $\boldsymbol{X} \in \mathbb{R}^{n \times p}$.
- Least squares criterion: $\min Q(\boldsymbol{b}) = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{b}\|_2^2$.
- Any solution to the normal equation $\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{b} = \boldsymbol{X}^T\boldsymbol{y}$ (always consistent) is a minimizer of the least squares criterion $Q(\boldsymbol{b})$.
- Solutions to the normal equation:

  $$\hat{\boldsymbol{b}} = (\boldsymbol{X}^T\boldsymbol{X})^-\boldsymbol{X}^T\boldsymbol{y} + (\boldsymbol{I}_p - (\boldsymbol{X}^T\boldsymbol{X})^-\boldsymbol{X}^T\boldsymbol{X})\boldsymbol{q},$$

  where $\boldsymbol{q} \in \mathbb{R}^q$ is arbitrary.
- $(\boldsymbol{X}^T\boldsymbol{X})^-\boldsymbol{X}^T$ is a generalized inverse of $\boldsymbol{X}$. Therefore the least squares solution applies even when the system $\boldsymbol{X}\boldsymbol{b} = \boldsymbol{y}$ is consistent.
- Least squares solution is unique if and only if $\boldsymbol{X}$ has full column rank.
- $\boldsymbol{P}_{\boldsymbol{X}} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^-\boldsymbol{X}^T$ is *the* orthogonal projection onto $\mathcal{C}(\boldsymbol{X})$.
- Geometry: The fitted value from the least squares solution $\hat{\boldsymbol{y}} = \boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{y}$ is the orthogonal projection of the response vector $\boldsymbol{y}$ onto the column space $\mathcal{C}(\boldsymbol{X})$.



- $\boldsymbol{I}_n - \boldsymbol{P}_{\boldsymbol{X}}$ is the orthogonal projection onto $\mathcal{N}(\boldsymbol{X}^T)$.

- Decomposition of $\boldsymbol{y}$:

$$\boldsymbol{y} = \boldsymbol{P_X y} + (\boldsymbol{I}_n - \boldsymbol{P_X})\boldsymbol{y} = \hat{\boldsymbol{y}} + \hat{\boldsymbol{e}},$$

  where $\hat{\boldsymbol{y}} \perp \hat{\boldsymbol{e}}$ and

$$\|\boldsymbol{y}\|_2^2 = \|\hat{\boldsymbol{y}}\|_2^2 + \|\hat{\boldsymbol{e}}\|_2^2.$$

Eigenvalues and eigenvectors

KL chapter 8. Assume $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ a square matrix.

- *Eigenvalues* are defined as roots of the characteristic equation $\det(\lambda \boldsymbol{I}_n - \boldsymbol{A}) = 0$.
- If λ is an eigenvalue of $\boldsymbol{A}$, then there exist non-zero $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ such that $\boldsymbol{Ax} = \lambda \boldsymbol{x}$ and $\boldsymbol{y}^T\boldsymbol{A} = \lambda \boldsymbol{y}^T$. $\boldsymbol{x}$ and $\boldsymbol{y}$ are called the (column) *eigenvector* and *row eigenvector* of $\boldsymbol{A}$ associated with the eigenvalue λ.
- $\boldsymbol{A}$ is singular if and only if it has at least one 0 eigenvalue.
- Eigenvectors associated with distinct eigenvalues are linearly independent.
- Eigenvalues of an upper or lower triangular matrix are its diagonal entries: $\lambda_i = a_{ii}$.
- Eigenvalues of an idempotent matrix are either 0 or 1.
- Eigenvalues of an orthogonal matrix have complex modulus 1.
- In most statistical applications, we deal with eigenvalues/eigenvectors of symmetric matrices.
  The eigenvalues and eigenvectors of a real *symmetric* matrix are real.
- Eigenvectors associated with distinct eigenvalues of a symmetry matrix are orthogonal.
- Eigen-decompostion of a symmetric matrix: $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\intercal$, where
  - $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$
  - columns of $\boldsymbol{U}$ are the eigenvectors which are (or can be chosen to be) mutually orthonormal

- A real symmetric matrix is positive semidefinite (positive definite) if and only if all eigenvalues are nonnegative (positive).
- *Spectral radius* $\rho(\boldsymbol{A}) = \max_i |\lambda_i|$.
- $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ a square matrix (not required to be symmetric), then $\mathrm{tr}(\boldsymbol{A}) = \sum_i \lambda_i$ and $|\boldsymbol{A}| = \prod_i \lambda_i$.

Singular value decomposition

KL chapter 9. Assume $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $p = \min\{m, n\}$.

- Singular value decomposition (SVD): $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, where
  - $\boldsymbol{U} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m) \in \mathbb{R}^{m\times m}$ is orthogonal
  - $\boldsymbol{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n) \in \mathbb{R}^{n\times n}$ is orthgonal
  - $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \ldots, \sigma_p) \in \mathbb{R}^{m\times n}$, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$.

  σ_i are called the *singular values*, $\boldsymbol{u}_i$ are the *left singular vectors*, and $\boldsymbol{v}_i$ are the *right singular vectors*.
- $\boldsymbol{A}\boldsymbol{v}_i = \sigma_i \boldsymbol{u}_i$ and $\boldsymbol{A}^T\boldsymbol{u}_i = \sigma_i \boldsymbol{v}_i$ for $i = 1, \ldots, p$.
- Thin SVD. Assume $m \geq n$. $\boldsymbol{A}$ can be factored as $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, where
  - $\boldsymbol{U} \in \mathbb{R}^{m\times n}$, $\boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I}_n$
  - $\boldsymbol{V} \in \mathbb{R}^{n\times n}$, $\boldsymbol{V}^\top\boldsymbol{V} = \boldsymbol{I}_n$
  - $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$
- Relation to eigen-decomposition. Using thin SVD,

$$\begin{aligned} \boldsymbol{A}^\top\boldsymbol{A} &= \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^\top \\ \boldsymbol{A}\boldsymbol{A}^\top &= \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^\top = \boldsymbol{U}\boldsymbol{\Sigma}^2\boldsymbol{U}^\top. \end{aligned}$$

- Another relation to eigen-decomposition. Using thin SVD,

$$\begin{pmatrix} \boldsymbol{0}_{n\times n} & \boldsymbol{A}^\top \\ \boldsymbol{A} & \boldsymbol{0}_{m\times m} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \boldsymbol{V} & \boldsymbol{V} \\ \boldsymbol{U} & -\boldsymbol{U} \end{pmatrix} \begin{pmatrix} \boldsymbol{\Sigma} & \boldsymbol{0}_{n\times n} \\ \boldsymbol{0}_{n\times n} & -\boldsymbol{\Sigma} \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} \boldsymbol{V}^\top & \boldsymbol{U}^\top \\ \boldsymbol{V}^\top & -\boldsymbol{U}^\top \end{pmatrix}.$$

  Hence any symmetric eigen-solver can produce the SVD of a matrix $\boldsymbol{A}$ without forming $\boldsymbol{A}\boldsymbol{A}^\top$ or $\boldsymbol{A}^\top\boldsymbol{A}$.

- Yet another relation to eigen-decomposition: If the eigendecomposition of a real symmetric matrix is $\boldsymbol{A} = \boldsymbol{W}\boldsymbol{\Lambda}\boldsymbol{W}^T = \boldsymbol{W}\mathrm{diag}(\lambda_1, \ldots, \lambda_n)\boldsymbol{W}^T$, then

$$\boldsymbol{A} = \boldsymbol{W}\boldsymbol{\Lambda}\boldsymbol{W}^T = \boldsymbol{W} \begin{pmatrix} |\lambda_1| & & \\ & \ddots & \\ & & |\lambda_n| \end{pmatrix} \begin{pmatrix} \mathrm{sgn}(\lambda_1) & & \\ & \ddots & \\ & & \mathrm{sgn}(\lambda_n) \end{pmatrix} \boldsymbol{W}^T$$

  is the SVD of $\boldsymbol{A}$.

- Relation to the Moore-Penrose (MP) inverse: Using thin SVD,

$$\boldsymbol{A}^+ = \boldsymbol{V}\boldsymbol{\Sigma}^+\boldsymbol{U}^\intercal,$$

  where $\boldsymbol{\Sigma}^+ = \mathrm{diag}(\sigma_1^{-1}, \ldots, \sigma_r^{-1}, 0, \ldots, 0)$, $r = \mathrm{rank}(\boldsymbol{A})$.

- Denote $\sigma(\boldsymbol{A}) = (\sigma_1, \ldots, \sigma_p)$. Then
  - $\mathrm{rank}(\boldsymbol{A}) = \#$ nonzero singular values $= \|\sigma(\boldsymbol{A})\|_0$
  - $\boldsymbol{A} = \boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r^T = \sum_{i=1}^r \sigma_i \boldsymbol{u}_i\boldsymbol{v}_i^T$
  - $\|\boldsymbol{A}\|_{\mathrm{F}} = (\sum_{i=1}^p \sigma_i^2)^{1/2} = \|\sigma(\boldsymbol{A})\|_2$
  - $\|\boldsymbol{A}\|_2 = \sigma_1 = \|\sigma(\boldsymbol{A})\|_\infty$

- Assume $\mathrm{rank}(\boldsymbol{A}) = r$ and partition $\boldsymbol{U} = (\boldsymbol{U}_r, \tilde{\boldsymbol{U}}_r) \in \mathbb{R}^{m\times m}$ and $\boldsymbol{V} = (\boldsymbol{V}_r, \tilde{\boldsymbol{V}}_r) \in \mathbb{R}^{n\times n}$, then
  - $\mathcal{C}(\boldsymbol{A}) = \mathrm{span}\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r\}$, $\mathcal{N}(\boldsymbol{A}^T) = \mathrm{span}\{\boldsymbol{u}_{r+1}, \ldots, \boldsymbol{u}_m\}$
  - $\mathcal{N}(\boldsymbol{A}) = \mathrm{span}\{\boldsymbol{v}_{r+1}, \ldots, \boldsymbol{v}_n\}$, $\mathcal{C}(\boldsymbol{A}^T) = \mathrm{span}\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r\}$
  - $\boldsymbol{U}_r\boldsymbol{U}_r^T$ is the orthogonal projection onto $\mathcal{C}(\boldsymbol{A})$
  - $\tilde{\boldsymbol{U}}_r\tilde{\boldsymbol{U}}_r^T$ is the orthogonal projection onto $\mathcal{N}(\boldsymbol{A}^T)$
  - $\boldsymbol{V}_r\boldsymbol{V}_r^T$ is the orthogonal projection onto $\mathcal{C}(\boldsymbol{A}^T)$
  - $\tilde{\boldsymbol{V}}_r\tilde{\boldsymbol{V}}_r^T$ is the orthogonal projection onto $\mathcal{N}(\boldsymbol{A})$

Preliminaries of numerical linear algebra

Numerical linear algebra concerns how matrix/vector computations are done in computer. We first look at some basic linear algebra operations.

Flop counts of some basic linear algebra subroutines (BLAS)

See `http://www.netlib.org/blas/` for a complete listing of BLAS functions.

Level	Example Operation	Name	Dimension	Flops
1	$\alpha \leftarrow \boldsymbol{x}^T\boldsymbol{y}$	dot product	$\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$	$2n$
	$\boldsymbol{y} \leftarrow \boldsymbol{y} + a\boldsymbol{x}$	saxpy	$a \in \mathbb{R}$, $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$	$2n$
2	$\boldsymbol{y} \leftarrow \boldsymbol{y} + \boldsymbol{A}\boldsymbol{x}$	gaxpy	$\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{y} \in \mathbb{R}^m$	$2mn$
	$\boldsymbol{A} \leftarrow \boldsymbol{A} + \boldsymbol{y}\boldsymbol{x}^T$	rank one update	$\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{y} \in \mathbb{R}^m$	$2mn$
3	$\boldsymbol{C} \leftarrow \boldsymbol{C} + \boldsymbol{A}\boldsymbol{B}$	matrix multiplication	$\boldsymbol{A} \in \mathbb{R}^{m\times p}$, $\boldsymbol{B} \in \mathbb{R}^{p\times n}$, $\boldsymbol{C} \in \mathbb{R}^{m\times n}$	$2mnp$
	$\boldsymbol{A} \leftarrow \boldsymbol{A}\boldsymbol{D}$	column scaling	$\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{D} = \text{diag}(d_1, \ldots, d_n)$	mn
	$\boldsymbol{A} \leftarrow \boldsymbol{D}\boldsymbol{A}$	row scaling	$\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{D} = \text{diag}(d_1, \ldots, d_m)$	mn

- Go over the "mxmult" session in R.
  `http://hua-zhou.github.io/teaching/biostatm280-2016winter/matmult.html`
  Different ways to compute $\boldsymbol{X}^T\boldsymbol{W}^{-1}\boldsymbol{X}$, such as in the weighted least squares. $\boldsymbol{X} \in \mathbb{R}^{n\times p}$, $\boldsymbol{W} = \text{diag}(w_1, \ldots, w_n) \in \mathbb{R}^{n\times n}$.

  1. `t(X) %*% solve(W) %*% X`: $O(n^3 + n^2p + np^2)$ flops, why need to do the expensive matrix inversion?
  2. `t(X) %*% diag(1 / w) %*% X`: $O(n^2p + np^2)$ flops, why need to save a diagonal matrix and do an extra matrix multiplication?
  3. `(t(X) / w) %*% X`: wrong! $\boldsymbol{w}$ recycled incorrectly
  4. `t(X) %*% (X / w)`: $O(np^2 + np) = O(np^2)$ flops
  5. `crossprod(X, X / w)`: same as 4, skip the transpose operation

- Another example: Fisher information matrix of a generalized linear model (GLM): $\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X}$, where $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ and $\boldsymbol{W} = \text{diag}(w_1, \ldots, w_n)$ are the observation weights.

- Bottom line: Always be flop-aware when writing code.

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

- But, for high-performance matrix commutations, it is *not* enough to minimize flops. Pipelining, effective use of memory hierarchy, data layout in memory, ... play important role too.

Vector computer

- Most modern computers are vector machines, which perform vector calculations (saxpy, inner product) fast.
- Vector processing by *pipelining*. E.g., vector addition $\boldsymbol{z} = \boldsymbol{x} + \boldsymbol{y}$

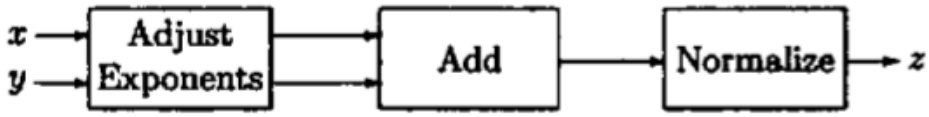


FIG. 1.4.1 *A 3-Cycle Adder*

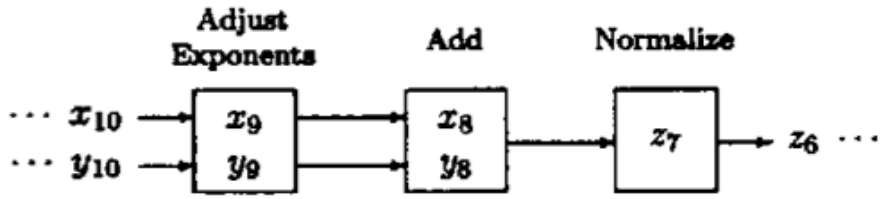


FIG. 1.4.2 *Pipelined Addition*

- For example, for Intel Sandy Bridge/Ivy Bridge:
  - 8 DP FLOPs/cycle: 4-wide AVX addition + 4-wide AVX multiplication
  - 16 SP FLOPs/cycle: 8-wide AVX addition + 8-wide AVX multiplication

  and for Intel Haswell/Broadwell/Skylake:
  - 16 DP FLOPs/cycle: two 4-wide FMA (fused multiply-add) instructions
  - 32 SP FLOPs/cycle: two 8-wide FMA (fused multiply-add) instructions

- One implication of the pipelining technology is that we need to ship vectors to the pipeline fast enough to keep the arithmetic units (ALU) busy and maintain high throughput.

6 Lecture 6, Jan 21

Announcements

- HW1 due today @ 11:59PM. Tag and push to the `master` branch.
- HW2 posted. Due Tue Feb 2 @ 11:59PM.
- Quiz 1 at end of today's class (2:35PM–2:50PM).

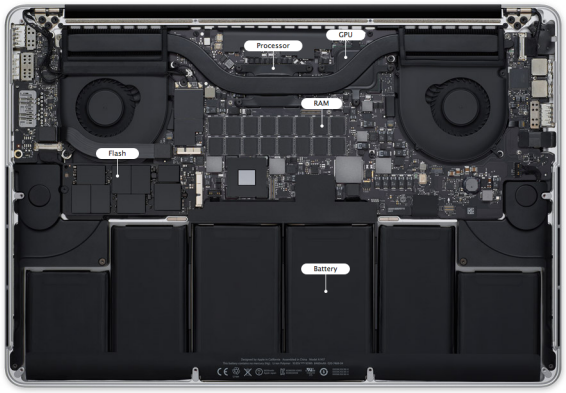
Last time

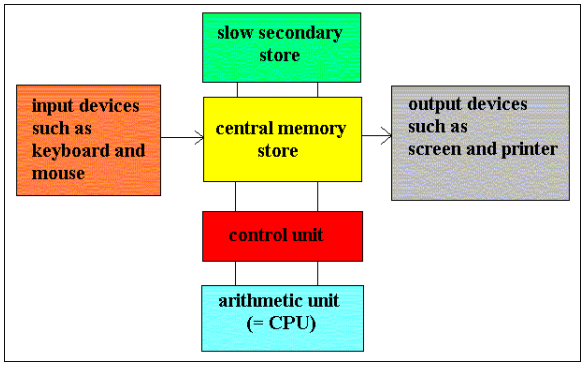
- Version control with Git.
- Algorithm, flop, computational complexity.
- Numerical linear algebra: introduction, BLAS.

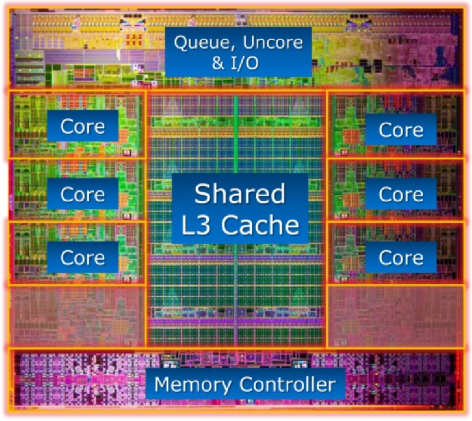
Today

- BLAS (con'td).
- Triangular systems.
- Gaussian elimination and LU decomposition.

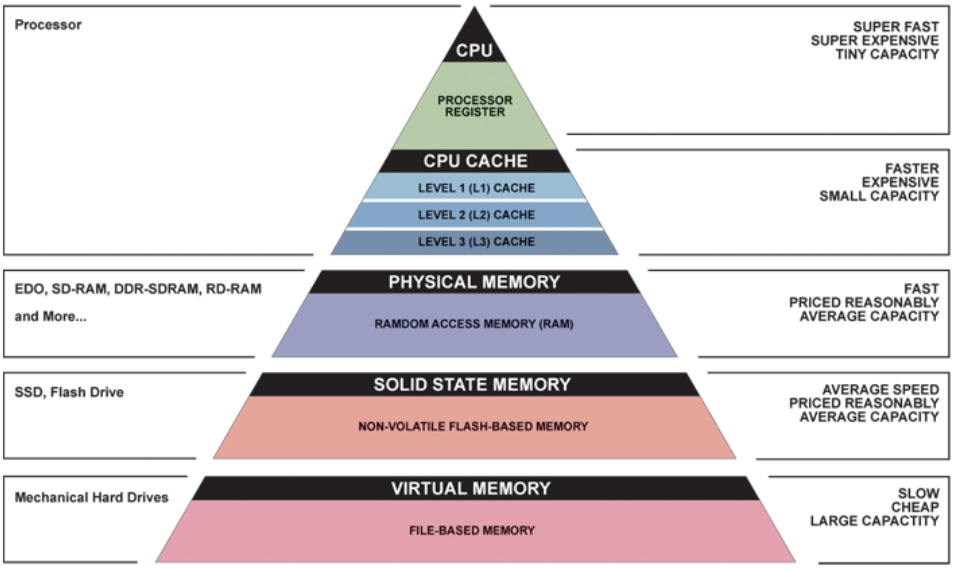
Computer architecture and high-level BLAS







- Memory hierarchy:



▲ Simplified Computer Memory Hierarchy
Illustration: Ryan J. Leng

Upper the hierarchy, faster the memory accessing speed, and more expensive the memory units.

> Key to high performance is effective use of memory hierarchy. True on all architectures.

- Can we keep the super fast arithmetic units busy with enough deliveries of matrix data and ship the results to memory fast enough to avoid backlog? Answer: use high-level BLAS as much as possible.

- Why high-level BLAS?

BLAS	Dimension	Mem Refs	Flops	Ratio
Level 1: $\boldsymbol{y} \leftarrow \boldsymbol{y} + a\boldsymbol{x}$	$\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$	$3n$	$2n$	3:2
Level 2: $\boldsymbol{y} \leftarrow \boldsymbol{y} + \boldsymbol{A}\boldsymbol{x}$	$\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n, \boldsymbol{A} \in \mathbb{R}^{n \times n}$	n^2	$2n^2$	1:2
Level 3: $\boldsymbol{C} \leftarrow \boldsymbol{C} + \boldsymbol{A}\boldsymbol{B}$	$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C} \in \mathbb{R}^{n \times n}$	$4n^2$	$2n^3$	2:n

- BLAS 1 tend to be *memory bandwidth-limited*. E.g., Xeon X5650 CPU has a theoretical throughput of 128 DP GFLOPS but a max memory bandwidth of 32GB/s.

- Higher level BLAS (3 or 2) make more effective use of arithmetic logic units (ALU) by keeping them busy.

- Message: Although we state many algorithms (solving linear equations, least squares, eigen-decomposition, SVD, ...) in terms of inner product and saxpy, the actual implementation in BLAS and LAPACK may be quite different.

- A distinction between LAPACK and LINPACK (older version of R uses LINPACK) is that LAPACK makes use of higher level BLAS as much as possible (usually by smart partitioning) to increase the so-called *level-3 fraction*.

Effect of data layout

- Data layout in memory effects execution speed too. It is much faster to move chunks of data in memory than retrieving/writing scattered data.

- Storage mode: column-major (Fortran, Matlab, R, Julia) vs row-major (C/C++).

  When you call BLAS from C/C++, use the CBLAS library instead of the traditional BLAS library implemented in Fortran.

- Take matrix multiplication as an example. Assume the storage is column-major, such as in Fortran. $\boldsymbol{C} \leftarrow \boldsymbol{C} + \boldsymbol{A}\boldsymbol{B}$, where $\boldsymbol{A} \in \mathbb{R}^{m \times p}$, $\boldsymbol{B} \in \mathbb{R}^{p \times n}$, $\boldsymbol{C} \in \mathbb{R}^{m \times n}$. There are 6 variants of the algorithms according to the order in the triple loops. We pay attention to the innermost loop, where the vector calculation occurs,

***jki* or *kji*:**

for $i = 1{:}m$
  $C(i,j) = C(i,j) + A(i,k)B(k,j)$
end

***ikj* or *kij*:**

for $j = 1{:}n$
  $C(i,j) = C(i,j) + A(i,k)B(k,j)$
end

***ijk* or *jik*:**

for $k = 1{:}p$
  $C(i,j) = C(i,j) + A(i,k)B(k,j)$
end

and the associated *stride* when accessing the three matrices in memory (assuming column-major storage)

Variant	A Stride	B Stride	C Stride
jki or *kji*	Unit	0	Unit
ikj or *kij*	0	Non-Unit	Non-Unit
ijk or *jik*	Non-Unit	Unit	0

Apparently the variants jki or kji are preferred.

- Message: data storage mode effects algorithm implementation too.
- A numerical experiment in JULIA: `http://hua-zhou.github.io/teaching/biostatm280-2016winter/matmul_loop.html`.

Solving linear equations

We consider algorithms for solving linear equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, a ubiquitous task in statistics. Idea: turning original problem into an "easy" one, e.g., triangular system.

Triangular system

- *Forward substitution* to solve $\boldsymbol{L}\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{L} \in \mathbb{R}^{n \times n}$ is lower triangular

$$\begin{bmatrix} a_{11} & 0 & ... & 0 \\ a_{21} & a_{22} & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & ... & a_{mm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

$$\begin{aligned} x_1 &= b_1/a_{11} \\ x_2 &= (b_2 - a_{21}x_1)/a_{22} \\ x_3 &= (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33} \\ &\vdots \\ x_m &= b_m - a_{m1}x_1 - a_{m2}x_2 - ... - a_{m,m-1}x_{m-1})/a_{mm} \end{aligned}$$

n^2 flops ($n^2/2$ multiplications/divisions and $n^2/2$ additions/subtractions) and $\boldsymbol{L}$ is accessed by row.

- *Back substitution* to solve $\boldsymbol{Ux} = \boldsymbol{b}$ where $\boldsymbol{U} \in \mathbb{R}^{n \times n}$ is upper triangular

$$\begin{bmatrix} a_{11} & ... & a_{1,m-1} & a_{1m} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & ... & a_{m-1,m-1} & a_{m-1,m} \\ 0 & ... & 0 & a_{mm} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_{m-1} \\ x_m \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_{m-1} \\ b_m \end{bmatrix}$$

$$\begin{aligned} x_m &= b_m/a_{mm} \\ x_{m-1} &= (b_{m-1} - a_{m-1,m}x_m)/a_{m-1,m-1} \\ x_{m-2} &= (b_{m-2} - a_{m-2,m-1}x_{m-1} - a_{m-2,m}x_m)/a_{m-2,m-2} \\ &\vdots \\ x_1 &= b_1 - a_{12}x_2 - a_{13}x_3 - ... - a_{1m}x_m)/a_{11} \end{aligned}$$

n^2 flops ($n^2/2$ multiplications/divisions and $n^2/2$ additions/subtractions) and $\boldsymbol{U}$ is accessed by row.

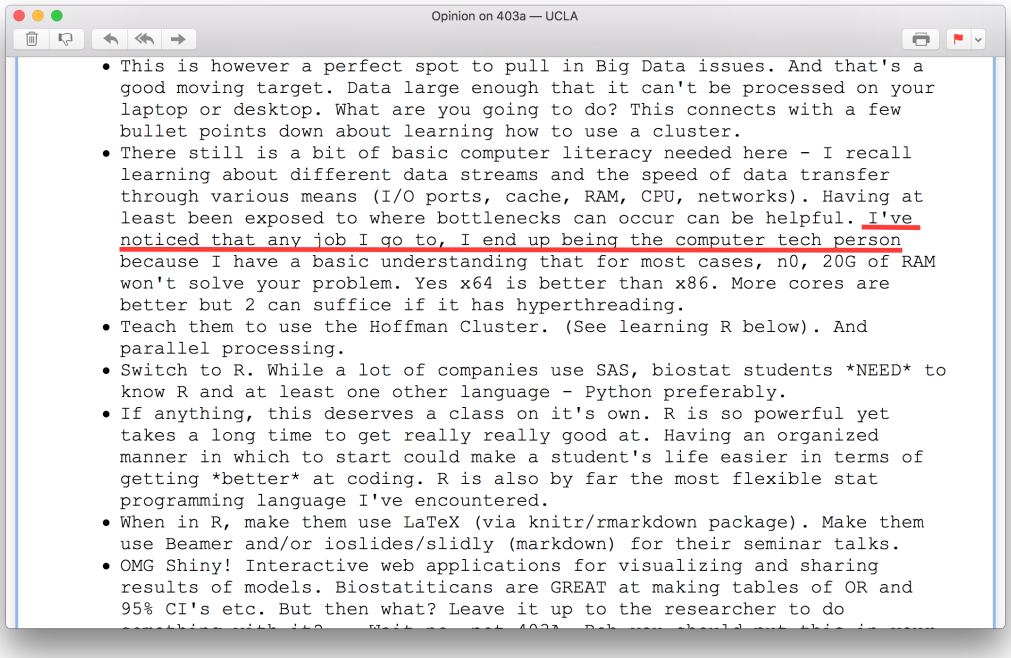
- Column version: reverse the order of looping.
- BLAS level 2 function: `?trsv` (triangular solve with one right hand side)
- BLAS level 3 function: `?trsm` (matrix triangular solve, i.e., multiple right hand sides)

- In R, `forwardsolve()` and `backsolve()` (wrappers of `dtrsm`).
- In JULIA, `A \ b` uses forward or backward substitution when $\boldsymbol{A}$ is a triangular matrix. Or we can simply call BLAS functions directly.
- Eigenvalues of $\boldsymbol{L}$ are diagonal entries $\lambda_i = \ell_{ii}$. $\det(\boldsymbol{L}) = \prod_i \ell_{ii}$.
- A *unit triangular matrix* is a triangular matrix with all diagonal entries being 1.
- The algebra of triangular matrices (HW2)
  - The product of two upper (lower) triangular matrices is upper (lower) triangular.
  - The inverse of an upper (lower) triangular matrix is upper (lower) triangular.
  - The product of two unit upper (lower) triangular matrices is unit upper (lower) triangular.
  - The inverse of a unit upper (lower) triangular matrix is unit upper (lower) triangular.

7 Lecture 7, Jan 26

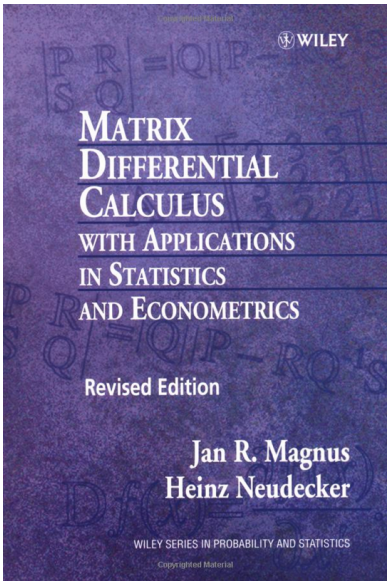
Announcements

- Quiz 1 returned: 7.73 ± 1.39. Statistical culture, future of statistics, computer literacy, ... An email from Jeffrey Robbins (Biostatistics alumna)

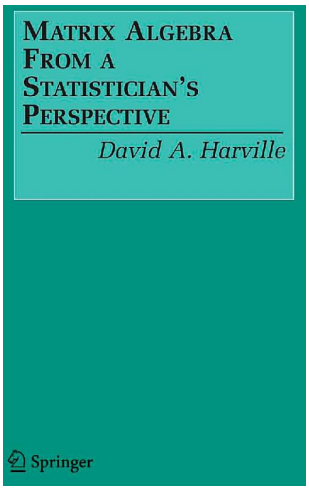


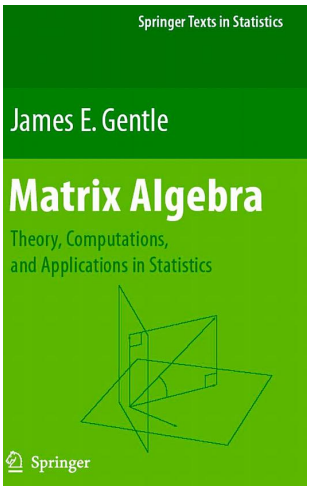
- HW2 due next Tuesday. FAQs at `http://hua-zhou.github.io/teaching/biostatm280-2016winter/biostatm280winter2016/2016/01/26/hw2-hints.html`

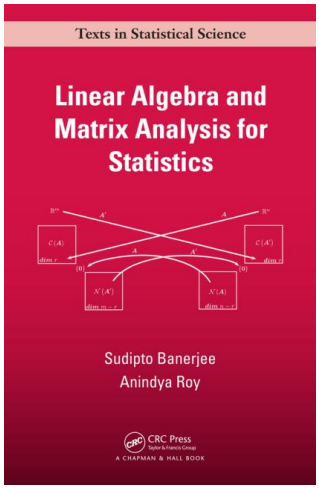
- Any good reference books for linear algebra? Magnus and Neudecker (1999), Horn and Johnson (1985), Harville (1997), Gentle (2007), Banerjee and Roy (2014).


WILEY
Matrix Differential Calculus with Applications in Statistics and Econometrics
Revised Edition
Jan R. Magnus
Heinz Neudecker
WILEY SERIES IN PROBABILITY AND STATISTICS


MATRIX ANALYSIS
ROGER A. HORN
AND
CHARLES R. JOHNSON


Matrix Algebra From a Statistician's Perspective
David A. Harville
Springer


Springer Texts in Statistics
James E. Gentle
Matrix Algebra
Theory, Computations, and Applications in Statistics
Springer


Texts in Statistical Science
Linear Algebra and Matrix Analysis for Statistics
Sudipto Banerjee
Anindya Roy
CRC Press
A CHAPMAN & HALL BOOK

Last time

- Memory hierarchy, high-level BLAS, effect of data layout. A few messages from the matrix multiplication example (in Julia): `http://hua-zhou.github.io/teaching/biostatm280-2016winter/matmul_loop.html`

    1. Order of looping matters. Access a contiguous chunk of data in memory is much more efficient than accessing scattered data.
    2. Don't re-invent wheels and use BLAS/LAPACK whenever possible.
    3. Be alert to unnecessary allocation of intermediate variables. Memory transactions and garbage collection are *expensive*.

- Solving triangular systems. n^2 flops for the backward or forward substitution algorithms.

Today

- Gaussian elimination and LU decomposition.
- Cholesky decomposition.
- Reading: *The decompositional approach to matrix computation*, by G. W. Stewart.
  `http://hua-zhou.github.io/teaching/biostatm280-2016winter/readings/decomp.pdf`

Gaussian elimination and LU decomposition

Given a system of linear algebraic equations

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Step 1: Each row times a_{11}/a_{k1},
then use row one to subtract other rows.

$$\Rightarrow \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & \tilde{a}_{22} & \cdots & \tilde{a}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \tilde{a}_{n2} & \cdots & \tilde{a}_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \tilde{b}_2 \\ \vdots \\ \tilde{b}_n \end{bmatrix}$$

Step 2: The second row and down multiply by $\tilde{a}_{22}/\tilde{a}_{k2}$,
then use row two to subtract every row below.

$$\Rightarrow \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & \tilde{a}_{22} & \tilde{a}_{23} & \cdots & \tilde{a}_{2n} \\ 0 & 0 & \tilde{a}_{33} & \cdots & \tilde{a}_{3n} \\ \vdots & \vdots & \cdots & \ddots & \vdots \\ 0 & 0 & \tilde{a}_{n3} & \cdots & \tilde{a}_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \tilde{b}_2 \\ \tilde{b}_3 \\ \vdots \\ \tilde{b}_n \end{bmatrix}$$

Step 3: Similar to the previous two steps, repeat until all
elements in the lower triangle of the matrix A
become zeros.

$$\Rightarrow \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & \tilde{a}_{22} & \cdots & \tilde{a}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \tilde{a}_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \tilde{b}_2 \\ \vdots \\ \tilde{b}_n \end{bmatrix}$$

- History: It is one by-product of Gauss's efforts to re-discover the dwarf planet Ceres using method of least squares. No linear algebra in 1801!
- Solve $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ for a general matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$.
- A toy example: `https://en.wikipedia.org/wiki/Gaussian_elimination#Example_of_the_algorithm`.

- Idea: a series of *elementary operations* that turn $\boldsymbol{A}$ into a triangular system.
- It turns out Gaussian elimination not only transforms $\boldsymbol{A}$ into a triangular system,
- We consider the square $\boldsymbol{A}$ case first.
- *Elementary operator matrix* $\boldsymbol{E}_{jk}(c)$ is the identity with the 0 in position (j,k) replaced by c. For any vector $\boldsymbol{x}$,

$$\boldsymbol{E}_{jk}(c)\boldsymbol{x} = (x_1, \ldots, x_{j-1}, x_j + cx_k, x_{j+1}, \ldots, x_n)^\mathsf{T}.$$

  Applying $\boldsymbol{E}_{jk}(c)$ to both sides of the system $\boldsymbol{Ax} = \boldsymbol{b}$ replaces the j-th equation $\boldsymbol{a}_{j*}^\mathsf{T}\boldsymbol{x} = b_j$ by $\boldsymbol{a}_{j*}^\mathsf{T}\boldsymbol{x} + c\boldsymbol{a}_{k*}^\mathsf{T}\boldsymbol{x} = b_j + cb_k$. For $j > k$, $\boldsymbol{E}_{jk}(c) = \boldsymbol{I} + c\boldsymbol{e}_j\boldsymbol{e}_k^\mathsf{T}$ is unit lower triangular and full rank. $\boldsymbol{E}_{jk}^{-1}(c) = \boldsymbol{E}_{jk}(-c)$.

- Zeroing the first column

$$\begin{aligned}
\boldsymbol{E}_{21}(c_2^{(1)})\boldsymbol{Ax} &= \boldsymbol{E}_{21}(c_2^{(1)})\boldsymbol{b} \\
\boldsymbol{E}_{31}(c_3^{(1)})\boldsymbol{E}_{21}(c_2^{(1)})\boldsymbol{Ax} &= \boldsymbol{E}_{31}(c_3^{(1)})\boldsymbol{E}_{21}(c_2^{(1)})\boldsymbol{b} \\
&\vdots \\
\boldsymbol{E}_{n1}(c_n^{(1)})\cdots\boldsymbol{E}_{31}(c_3^{(1)})\boldsymbol{E}_{21}(c_2^{(1)})\boldsymbol{Ax} &= \boldsymbol{E}_{n1}(c_n^{(1)})\cdots\boldsymbol{E}_{31}(c_3^{(1)})\boldsymbol{E}_{21}(c_2^{(1)})\boldsymbol{b}
\end{aligned}$$

  where $c_i^{(1)} = -a_{i1}/a_{11}$. Denote $\boldsymbol{M}_1 = \boldsymbol{E}_{n1}(c_n^{(1)})\cdots\boldsymbol{E}_{31}(c_3^{(1)})\boldsymbol{E}_{21}(c_2^{(1)})$.

- Then zero the k-th column for $k = 2, \ldots, n-1$ sequentially. This results in a transformed linear system $\boldsymbol{Ux} = \tilde{\boldsymbol{b}}$, where $\boldsymbol{U} = \boldsymbol{M}_{n-1}\cdots\boldsymbol{M}_1\boldsymbol{A}$ is upper triangular and $\tilde{\boldsymbol{b}} = \boldsymbol{M}_{n-1}\cdots\boldsymbol{M}_1\boldsymbol{b}$. $\boldsymbol{M}_k$ has the shape

$$\boldsymbol{M}_k = \boldsymbol{E}_{n,k}^{(k)}\cdots\boldsymbol{E}_{k+1,k}^{(k)} = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & c_{k+1}^{(k)} & 1 & \\ & & \vdots & & \ddots \\ & & c_n^{(k)} & & 1 \end{pmatrix},$$

  where $c_i^{(k)} = -\tilde{a}_{ik}^{(k-1)}/\tilde{a}_{kk}^{(k-1)}$. $\boldsymbol{M}_k$ is unit lower triangular and full rank. $\boldsymbol{M}_k$ are called the *Gauss transformations*.

- Let $\boldsymbol{L} = \boldsymbol{M}_1^{-1} \cdots \boldsymbol{M}_{n-1}^{-1}$. We have the decomposition

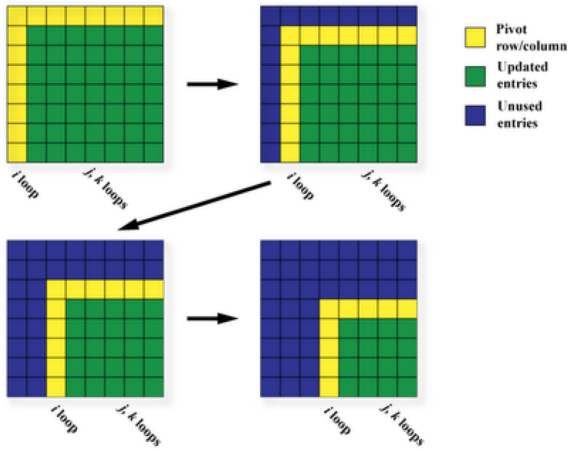
$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}.$$

  $\boldsymbol{M}_k$ is unit upper triangular, so $\boldsymbol{M}_k^{-1}$ and thus $\boldsymbol{L}$ is unit lower triangular.

- Where is $\boldsymbol{L}$? Note $\boldsymbol{M}_k = \boldsymbol{I} + (0, \ldots, 0, c_{k+1}^{(k)}, \ldots, c_n^{(k)})^\mathsf{T} \boldsymbol{e}_k^\mathsf{T}$. By Sherman-Morrison, $\boldsymbol{M}_k^{-1} = \boldsymbol{I} - (0, \ldots, 0, c_{k+1}^{(k)}, \ldots, c_n^{(k)})^\mathsf{T} \boldsymbol{e}_k^\mathsf{T}$. So the entries of $\boldsymbol{L}$ are simply $\ell_{ik} = -c_i^{(k)}$, $i > k$, the negative of multipliers in GE.

- The whole LU procedure is done in place, i.e., $\boldsymbol{A}$ is overwritten by $\boldsymbol{L}$ and $\boldsymbol{U}$.

- Implementation: outer product LU (kij loop), block outer product LU (higher level-3 fraction), Crout's algorithm (jki loop), ...



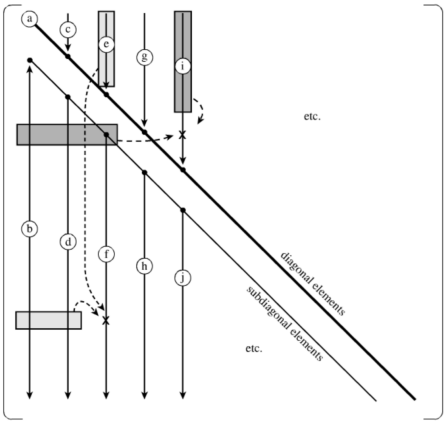


Figure 2.3.1. Crout's algorithm for *LU* decomposition of a matrix. Elements of the original matrix are modified in the order indicated by lower case letters: a, b, c, etc. Shaded boxes show the previously modified elements that are used in modifying two typical elements, each indicated by an "x".

- Exercise: work out the LU decomposition of a 3-by-3 example on paper. `https://en.wikipedia.org/wiki/Gaussian_elimination#Example_of_the_algorithm`

- LU decomposition exists if the principal sub-matrix $A(1:k, 1:k)$ is non-singular for $k = 1, \ldots, n-1$. If the LU decomposition exists and $\boldsymbol{A}$ is non-singular, then the LU decomposition is unique and $\det(\boldsymbol{A}) = \prod_{i=1}^n u_{ii}$.

- This *LU decomposition* costs $2(n-1)^2 + 2(n-2)^2 + \cdots + 2 \cdot 1^2 \approx \frac{2}{3}n^3$ flops ($n^3/3$ multiplications and $n^3/3$ additions).

- Given LU, one right hand side costs $2n^2$ flops: one forward substitution to solve $\boldsymbol{L}\boldsymbol{y} = \boldsymbol{b}$ and then one backward substitution to solve $\boldsymbol{U}\boldsymbol{x} = \boldsymbol{y}$.
- For *matrix inversion*, there are n right hand sides $\boldsymbol{e}_i$. However, taking advantage of zeros reduces $2n^3$ flops to $\frac{4}{3}n^3$. So matrix inversion costs $\frac{2}{3}n^3 + \frac{4}{3}n^3 = 2n^3$ flops in total.
- *No inversion* mentality: Whenever we see matrix inverse, we should think in terms of solving linear equations. We do *not* compute matrix inverse unless (i) it is necessary to compute standard errors, (2) number of right hand sides is much larger than n, (3) n is small.

Pivoting for LU

- What if we encounter a *pivot* $\tilde{a}_{kk}^{(k-1)}$ being 0 or close to 0 due to underflow?
- Think about $\boldsymbol{A} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. Does it have a solution for arbitrary $\boldsymbol{b}$? Does GE work?
- Work on the example

$$\begin{aligned} 0.0001x_1 + x_2 &= 1 \\ x_1 + x_2 &= 2, \end{aligned}$$

  which has solution $x_1 = 1.0001$ and $x_2 = 0.9999$. Suppose we have 3 digits of precision. After first step of elimination, we have (catastrophic cancellation happens)

$$\begin{aligned} 0.0001x_1 + x_2 &= 1 \\ -10,000x_2 &= -10,000 \end{aligned}$$

  and the solution by back substitution is $x_2 = 1.000$ and $x_1 = 0.000$.
- Message: zero or very small pivots cause trouble.
  Solution: *pivoting*.
- *Partial pivoting*: at the k-th stage the equation with $\max_{i=k}^{n} |\tilde{a}_{ik}^{(k-1)}|$ is moved into the k-th row. Thus we have $\boldsymbol{M}_{n-1}\boldsymbol{P}_{n-1} \cdots \boldsymbol{M}_1\boldsymbol{P}_1\boldsymbol{A} = \boldsymbol{U}$.

- With partial pivoting, it can be shown that

$$\boldsymbol{PA} = \boldsymbol{LU},$$

  where $\boldsymbol{P} = \boldsymbol{P}_{n-1} \cdots \boldsymbol{P}_1$, $\boldsymbol{L}$ is unit lower triangular with $|\ell_{ij}| \leq 1$, and $\boldsymbol{U}$ is upper triangular.

- $\det(\boldsymbol{P}) \det(\boldsymbol{A}) = \det(\boldsymbol{U}) = \prod_{i=1}^{n} u_{ii}$.

- To solve $\boldsymbol{Ax} = \boldsymbol{b}$, we solve two triangular systems

$$\boldsymbol{Ly} = \boldsymbol{Pb} \quad \text{and} \quad \boldsymbol{Ux} = \boldsymbol{y},$$

  costing $2n^2$ flops.

- *Complete pivoting*: Do both row and column interchanges so that the largest entry in the sub matrix $\boldsymbol{A}(k:n, k:n)$ is permuted to the (k,k)-th entry. This yields the decomposition $\boldsymbol{PAQ} = \boldsymbol{LU}$, where $|\ell_{ij}| \leq 1$.

- Warning: In the actual LAPACK implementation, we do not really need to interchange rows/columns for pivoting. Just keep track of the indices we have interchanged.

- Gaussian elimination with partial pivoting is one of the most commonly used methods for solving *general* linear systems. Complete pivoting is stable but costs more computation. Partial pivoting is stable most of times.

- LAPACK: `?GETRF` does $\boldsymbol{PA} = \boldsymbol{LU}$ (LU decomposition with partial pivoting) in place.

- In R, `solve()` implicitly performs LU decomposition (wrapper of LAPACK routine `DGESV`). `solve()` allows specifying a single or multiple right hand sides. If none, it computes the matrix inverse. The `matrix` package contains `lu()` function that outputs `L`, `U`, and `pivot`.

- In JULIA, `lu()` and `lufact()` perform LU decomposition with partial pivoting, or call LAPACK function directly using `getrf!()`.

Cholesky decomposition (symmetric LU)

Reference: KL 7.7.

Archives de l'Ecole polytechnique (Fonds A. Cholesky)

- A basic tenet in numerical analysis:

  The structure should be exploited whenever solving a problem.

  Common structures include: symmetry, definiteness, sparsity, Kronecker product, low rank, ...

- LU decomposition (GE) is *not* used in statistics so often because most of time statisticians deal with positive (semi)definite matrix.

- Consider solving the normal equation $\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{X}^T\boldsymbol{y}$ for linear regression. The coefficient matrix $\boldsymbol{X}^T\boldsymbol{X}$ is symmetric and positive semidefinite. How to exploit this structure?

- Theorem: Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ be symmetric and positive definite. Then $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^\intercal$, where $\boldsymbol{L}$ is lower triangular with positive diagonal entries and is unique.

Proof by induction. If $n = 1$, then $\ell = \sqrt{a}$. For $n > 1$, the block equation

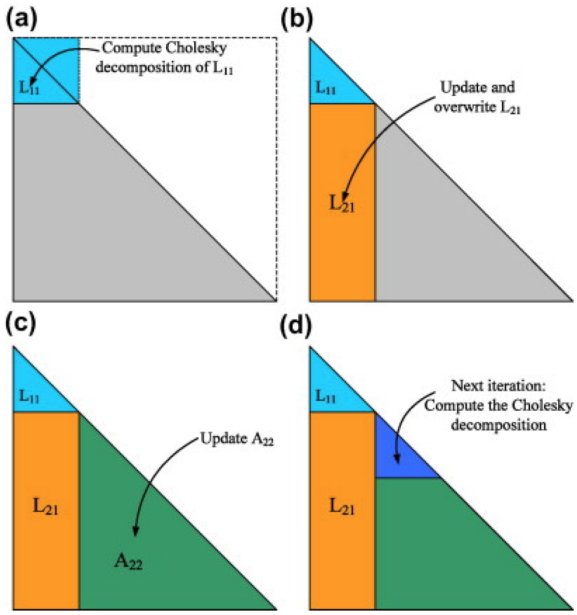
$$\begin{pmatrix} a_{11} & \boldsymbol{a}^\mathsf{T} \\ \boldsymbol{a} & \boldsymbol{A}_{22} \end{pmatrix} = \begin{pmatrix} \ell_{11} & \boldsymbol{0}_{n-1}^\mathsf{T} \\ \boldsymbol{l} & \boldsymbol{L}_{22} \end{pmatrix} \begin{pmatrix} \ell_{11} & \boldsymbol{l}^\mathsf{T} \\ \boldsymbol{0}_{n-1} & \boldsymbol{L}_{22}^\mathsf{T} \end{pmatrix}.$$

has solution

$$\begin{aligned} \ell_{11} &= \sqrt{a_{11}} \\ \boldsymbol{l} &= \ell_{11}^{-1}\boldsymbol{a} \\ \boldsymbol{L}_{22}\boldsymbol{L}_{22}^\mathsf{T} &= \boldsymbol{A}_{22} - \boldsymbol{l}\boldsymbol{l}^\mathsf{T} = \boldsymbol{A}_{22} - a_{11}^{-1}\boldsymbol{a}\boldsymbol{a}^\mathsf{T}. \end{aligned}$$

Now $a_{11} > 0$ (why?), so ℓ_{11} and $\boldsymbol{l}$ are uniquely determined. $\boldsymbol{A}_{22} - a_{11}^{-1}\boldsymbol{a}\boldsymbol{a}^\mathsf{T}$ is positive definite because $\boldsymbol{A}$ is positive definite (why?). By induction hypothesis, $\boldsymbol{L}_{22}$ exists and is unique. □

- The constructive proof completely specifies the algorithm.



- Exercise: work out the Cholesky decomposition of a 3-by-3 example on paper. `https://en.wikipedia.org/wiki/Cholesky_decomposition#Example`
- Computational cost: $\frac{1}{2}[2(n-1)^2 + 2(n-2)^2 + \cdots + 2 \cdot 1^2] \approx \frac{1}{3}n^3$ (half the cost of LU decomposition due to symmetry) plus n square roots.

- Avoid square roots: $\boldsymbol{LDL}^\intercal$ decomposition.
- In general Cholesky decomposition is very stable. Failure of the decomposition simply means $\boldsymbol{A}$ is not positive definite. It is an efficient way to test positive definiteness.

  If zero pivots $\tilde{a}_{ii} = 0$ are encountered, we can still continue the algorithm by setting $\ell_{ii} = 0$ and $\boldsymbol{l} = \boldsymbol{0}$. A better alternative is to use Cholesky decomposition with symmetric pivoting.

Cholesky with symmetric pivoting

Assume $\boldsymbol{A} \succeq \boldsymbol{0}_{n\times n}$ (p.s.d.)

- When $\boldsymbol{A}$ does not have full rank, e.g., $\boldsymbol{X}^T\boldsymbol{X}$ with a non-full column rank $\boldsymbol{X}$, we encounter $\tilde{a}_{kk} = 0$ during the procedure.
- *Symmetric pivoting.* At each stage k, we permute both row and column such that $\max_{i=k}^n \tilde{a}_{kk}$ becomes the pivot. If we encounter $\max_{i=k}^n \tilde{a}_{kk} = 0$, then $A(k : n, k : n) = \boldsymbol{0}$ (why?) and the algorithm terminates.
- With symmetric pivoting: $\boldsymbol{PAP}^T = \boldsymbol{LL}^T$, where $\boldsymbol{P}$ is a permutation matrix and $\boldsymbol{L} \in \mathbb{R}^{n\times r}$, $r = \text{rank}(\boldsymbol{A})$.
- In R, `chol()` is a wrapper function for LAPACK routines `DPOTRF` (p.d. no pivoting) and `DPSTRF` (p.s.d. with pivoting).
  - Option `pivot = FALSE` calls `DPOTRF`. It does $\boldsymbol{A} = \boldsymbol{R}^T\boldsymbol{R}$ and gives error message if $\boldsymbol{A}$ is not full rank.
  - Option `pivot = TRUE` calls `DPSTRF`. It does symmetric pivoting $\boldsymbol{PAP}^T = \boldsymbol{R}^T\boldsymbol{R}$ and yields `rank` and `pivot`.
  - Option `tol` passes the tolerance to LAPACK for deciding zero pivots. Default is $n \cdot$ machine epsilon $\cdot \max(\text{diag}(\boldsymbol{A}))$.
- In Julia, `chol()`, `cholfact()`, `ldlfact()`, or call LAPACK functions directly.

Applications of Cholesky decomposition

There are numerous applications of Cholesky decomposition.

- *No inversion* mentality: Whenever we see matrix inverse, we should think in terms of solving linear equations. If the matrix is positive (semi)definite, Cholesky decomposition applies.
- Example: multivariate normal density $N_n(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, $\boldsymbol{\Sigma}$ is p.d.

$$-\frac{n}{2}\ln(2\pi) - \frac{1}{2}\ln\det\boldsymbol{\Sigma} - \frac{1}{2}(\boldsymbol{y}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{y}-\boldsymbol{\mu}).$$

  - Method 1: (a) compute explicit inverse $\boldsymbol{\Sigma}^{-1}$ ($2n^3$ flops), (b) compute quadratic form ($2n^2 + 2n$ flops), (c) compute determinant ($2n^3/3$ flops).
  - Method 2: (a) Cholesky decomposition $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^T$ ($n^3/3$ flops), (b) Solve $\boldsymbol{L}\boldsymbol{x} = \boldsymbol{y} - \boldsymbol{\mu}$ by forward substitutions (n^2 flops), (c) compute quadratic form $\boldsymbol{x}^T\boldsymbol{x}$ ($2n$ flops), and (d) compute determinant from Cholesky factor (n flops).

  Which method is better?

- Compute Moore-Penrose inverse $\boldsymbol{A}^+$.
- Cholesky decomposition is *one* approach to solve linear regression.

References

Baggerly, K. A. and Coombes, K. R. (2009). Deriving chemosensitivity from cell lines: Forensic bioinformatics and reproducible research in high-throughput biology. *Ann. Appl. Stat.*, 3(4):1309–1334.

Banerjee, S. and Roy, A. (2014). *Linear algebra and matrix analysis for statistics*. Chapman & Hall/CRC Texts in Statistical Science Series. CRC Press, Boca Raton, FL.

Buckheit, J. and Donoho, D. (1995). Wavelab and reproducible research. In Antoniadis, A. and Oppenheim, G., editors, *Wavelets and Statistics*, volume 103 of *Lecture Notes in Statistics*, pages 55–81. Springer New York.

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

Donoho, D. L. (2010). An invitation to reproducible computational research. *Biostatistics*, 11(3):385–388.

Gentle, J. E. (2007). *Matrix Algebra*. Springer Texts in Statistics. Springer, New York. Theory, computations, and applications in statistics.

Harville, D. A. (1997). *Matrix Algebra From a Statistician's Perspectives*. Springer-Verlag, New York.

Horn, R. A. and Johnson, C. R. (1985). *Matrix Analysis*. Cambridge University Press, Cambridge.

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.

Magnus, J. R. and Neudecker, H. (1999). *Matrix Differential Calculus with Applications in Statistics and Econometrics*. Wiley Series in Probability and Statistics. John Wiley & Sons Ltd., Chichester.

McKay, B., Bar-Natan, D., Bar-Hillel, M., and Kalai, G. (1999). Solving the bible code puzzle. *Statist. Sci.*, 14(2):150–173.

Peng, R. D. (2009). Reproducible research and biostatistics. *Biostatistics*, 10(3):405–408.

Peng, R. D. (2011). Reproducible research in computational science. *Science*, 334(6060):1226–1227.

Potti, A., Dressman, H. K., Bild, A., and Riedel, R. F. (2006). Genomic signatures to guide the use of chemotherapeutics. *Nature medicine*, 12(11):1294–1300.

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.

Witztum, D., Rips, E., and Rosenberg, Y. (1994). Equidistant letter sequences in the book of genesis. *Statist. Sci.*, 9(3):429–438.