A brief interlude...

melting DNA: salts (Na^+ and K^+)

higher salt concentrations increase melting temperature

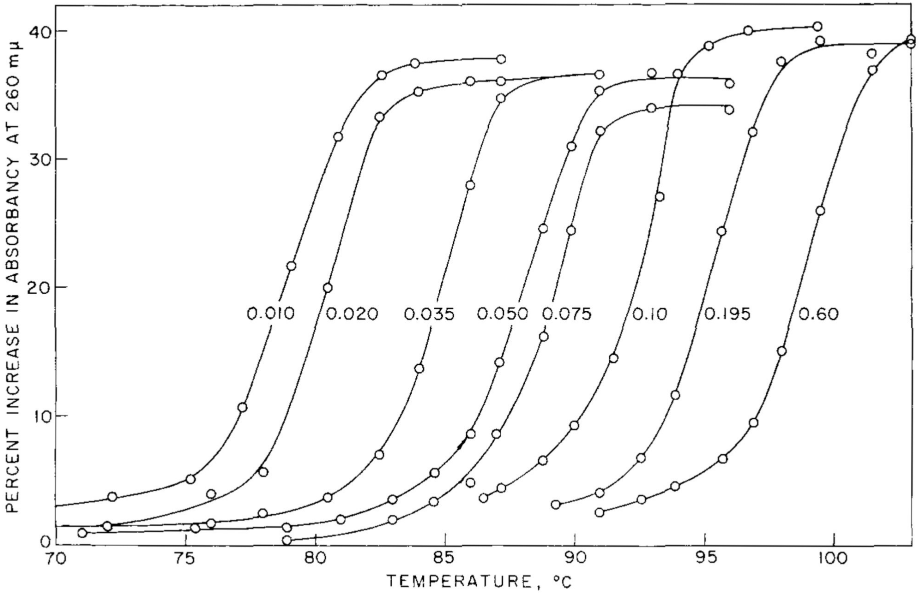

Fig. 1. Temperature dependence of the absorbancy of *Ps. aeruginosa* DNA at salt concentration between 0.01 and 0.6*M*. The per cent increase is relative to the absorbancy at 25°C.

Schildkraut and Lifson (1965; https://doi.org/10.1002/bip.360030207)

melting DNA: composition (%GC)

higher %GC increase melting temperature

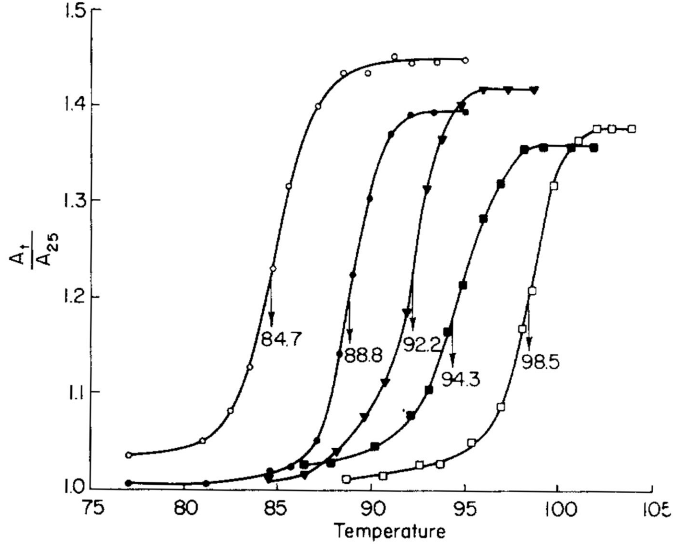

FIG. 2. Thermal denaturation curves of DNA isolated from various bacteria. All samples at 20 μg/ml in SSC. Relative absorbance (corrected for thermal expansion) measured at the elevated temperatures. The midpoint of each transition (T_m) is indicated at the arrow. DNA isolated from *Proteus vulgaris* (open circles); *Bacillus licheniformis* (filled circles); *Klebsiella pneumoniae* (inverted triangles); *Pseudomonas fluorescens* (filled squares); *Sarcina lutea* (open squares).

Mandel and Marmur (1968; https://doi.org/10.1016/0076-6879(67)12133-2)

melting DNA: polyols (e.g. betaine)

higher betaine concentration decreases melting temperature

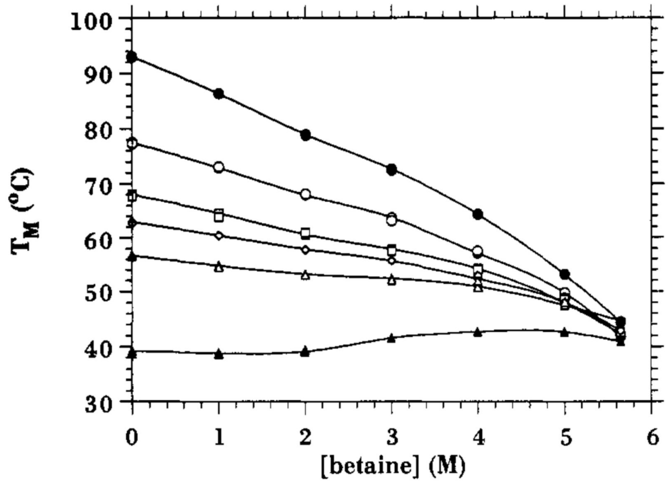

FIGURE 2: Variation of T_m with betaine concentration for DNAs of varying base composition. Melting was monitored by the UV absorbance at 260 nm. Buffers and heating rates were as in Figure 1A. (Open circles) *M. lysodeikticus* DNA (72% GC); (open squares) *E. coli* DNA (50% GC); (open diamonds) calf thymus DNA (42% GC); (open triangles) *Cl. perfringens* DNA (26% GC); (filled triangles) poly(dA-dT); (filled circles) poly(dG-dC).

Rees et al. (1993; https://doi.org/10.1021/bi00052a019)

melting DNA: empirical formulae

empirical formulae (usually) use some combination of:

salt concentration, %GC, DNA length, DNA concentration, formamide concentration [e.g. Howell et al. (1979), Wetmur (1991)]

von Ahsen et al. (2001) revised the most common formulae using regression on 475 oligos

melting DNA: vertical stacking

vertical stacking is the greatest contributor to helix stability

nearest–neighbor interactions predict helix stability

ratio of $\Delta H/\Delta S$ (Borer et al. 1974)

E (Khandelwal and Bhyravabhotla 2010)

can be used to estimate T_m (with some ‘corrections’)

melting DNA: ΔH/ΔS

Table 1. Comparison of published NN free energy parameters at 37°C

Sequence	Parameter, kcal/mol							
	Gotoh (ref. 7)	Vologodskii (ref. 10)	Breslauer (ref. 12)	Blake (ref. 17)	Benight (ref. 18)	SantaLucia (ref. 20)	Sugimoto (ref. 21)	Unified (ref. 22)
AA/TT	−0.43	−0.89	(−1.66)	−0.67	−0.93	−1.02	−1.20	−1.00
AT/TA	−0.27	−0.81	−1.19	−0.62	−0.83	−0.73	−0.90	−0.88
TA/AT	−0.22	−0.76	−0.76	−0.70	−0.70	−0.60	−0.90	−0.58
CA/GT	−0.97	−1.37	−1.80	−1.19	−1.26	−1.38	−1.70	−1.45
GT/CA	−0.98	−1.35	−1.13	−1.28	−1.52	−1.43	−1.50	−1.44
CT/GA	−0.83	−1.16	−1.35	−1.17	−1.03	−1.16	−1.50	−1.28
GA/CT	−0.93	−1.25	−1.41	−1.12	−1.56	−1.46	−1.50	−1.30
CG/GC	−1.70	−1.99	(−3.28)	−1.87	(−1.65)	−2.09	(−2.80)	−2.17
GC/CG	−1.64	−1.96	(−2.82)	−1.85	−2.44	−2.28	−2.30	−2.24
GG/CC	−1.22	−1.64	(−2.75)	−1.55	−1.67	−1.77	−2.10	−1.84
Average	−0.92	−1.32	−1.82	−1.20	−1.36	−1.39	−1.64	−1.42
Init. w/term. G·C*	NA	NA	(+2.60)	NA	NA	0.91	(+1.70)	0.98
Init. w/term. A·T*	NA	NA	(+2.60)	NA	NA	1.11†	(+1.70)	1.03
Sodium concentration, M	0.0195	0.195	1.0	0.075	0.115	1.0	1.0	1.0
Rank of stacking matrix	8	8	11	8	9	10	11	12

SantaLucia (1998; https://doi.org/10.1073/pnas.95.4.1460)

melting DNA: nearest–neighbor algorithms

Borer et al. (1974), Rychlik et al. (1990), Khandelwal and Bhyravabhotla (2010)

start at the 5' end; for each base:

sum the lookup values for the base and its 3' neighbor

correct the sum:

ratio of $\Delta H/\Delta S$ or oligo length

salt concentration

DNA concentration

And now back to our regularly scheduled programming...

python

created by Guido van Rossum

named for *Monty Python’s Flying Circus*

v1.0 (1991), v2.0 (2000), v3.0 (2008)

versions 2 and 3 are incompatible

has a small core and a highly modular standard library

tries to enforce simplicity and code readability

“There should be one—and preferably only one—obvious way to do it”

[Tim Peters, on the subject on Python programming]

“Do I contradict myself? Very well then I contradict myself, (I am large, I contain multitudes.)”

[Walt Whitman, not on the subject on Python programming]

python3 (and 99 bottles of beer)

```python
#!/usr/bin/env python3
for quant in range(99, 0, -1):
   if quant > 1:
      print(quant, "bottles of beer on the wall,", quant, "bottles of beer.")
      if quant > 2:
         suffix = str(quant - 1) + " bottles of beer on the wall."
      else:
         suffix = "1 bottle of beer on the wall."
   elif quant == 1:
      print("1 bottle of beer on the wall, 1 bottle of beer.")
      suffix = "no more beer on the wall!"
   print("Take one down, pass it around,", suffix)
   print("--")
```

python3 basics

primarily procedural (some object oriented and functional features)

block symbols are used sparingly and inconsistently (e.g. {}, [], (), :)

indendentation plus colons delimit blocks

- indendentation type is significant (cannot mix tabs and spaces)

variables are ‘strongly’ and dynamically–typed at time of use

- i.e. Python crashes when types collide
- type annotations are gradually being added to the language

complicated variables are passed as references

variables are garbage–collected in the background

python3 basics: interfaces

one–liners are ‘allowed’

    -c 'import sys; [sys.stdout.write(...) for line in sys.stdin]'

an interactive shell for testing or learning

    variable values are printed if operators are omitted

    ^d to exit

scripts

    #!/usr/bin/env python3

python3 basics: select interpreters

CPython is the ‘standard’ reference implementation

- C89 (with some C99)
- compiles Python to bytecode
- runs bytecode on a virtual machine

PyPy has a just–in–time compiler (faster than CPython)

Stackless Python is microthreaded for parallel processing

MicroPython and CircuitPython are for microcontrollers

python3 basics: packages

multiple package managers available (always a bad sign)

pip (Pip Installs Packages)

- standard with Python distributions
- ‘works’, but version conflicts often cause problems
- best used with virtual environments

Conda/Anaconda

- Python centric, but works with multiple languages
- uses virtual environments to isolate installed packages

apt

python3 basics: syntax

lines end with new line symbols (or semicolons;)

indendentation type and amount is significant

comments start with an octothorpe #

single and double quotes are processed the same way

slashes escape special characters \ (including quotes)

indices start from zero

python3 basics: variables

bool: a Boolean value (True, False)

int: a single integer number

float: a single decimal number

str: a string of characters

    size given by len(str)

list/tuple: arrays of Booleans, numbers, strings, or objects

    size given by len(array)

    list is mutable, tuple is immutable

dict: hashes of Booleans, numbers, strings, or objects

    size given by len(hash)

python3 basics: list/tuple tricks…

to create a list or tuple explicitly, use [] or (,)

e.g. x = [] ### empty mutable list

e.g. x = [0, 3, 5] ### mutable list

e.g. x = (0, 3, 5) ### immutable tuple

e.g. x = (0,) ### immutable tuple

python3 basics: …list/tuple tricks…

to convert a string into a list, use .split()

e.g. 'this is a string'.split(' ') ### [this] [is] [a] [string]

e.g. re.split('i|s', 'this is a string') ### [th] [ ] [ ] [ ] [a] [tr] [ng]

to convert a list or tuple into a string, use .join()

e.g. ' '.join(('x', 'y', 'z')) ### x y z

e.g. ' | '.join(['x', 'y', 'z']) ### x | y | z

to access an element of an list or tuple, use index numbers

e.g. x[0] ### element 0 only

python3 basics: …list/tuple tricks…

to access a subset (slice) of a list or tuple, use index numbers

e.g. x[1:3] ### elements 1 and 2 only

e.g. x[0:-1] ### everything but the last element

to add an element to a list, use .append()

e.g. x.append(y)

to remove element(s) from a list, use .remove()

e.g. x.remove(y)

to sort a list (one data type only), use .sort()

e.g. x.sort()

python3 basics: …list/tuple tricks

to sort a list copy (one data type only), use sorted()

e.g. x = sorted(y)

to reverse an list order, use .reverse()

e.g. x.reverse()

to iterate an list or tuple, use for

e.g. for y in x: ### element (y)

e.g. for k, y in enumerate(x): ### index (k), element (y)

to convert an immutable tuple to a mutable list, use list()

e.g. x = list(y)