Evaluating the Global CpG Methylation Status of Native DNA Utilizing a Bipartite Split-Luciferase Sensor

Ahmed H. Badran, Jennifer L. Furman, Andrew S. Ma, Troy J. Comi, Jason R. Porter and Indraneel Ghosh*

Department of Chemistry and Biochemistry, University of Arizona, Tucson, AZ 85721
* To whom correspondence should be addressed. E-mail: ghosh@email.arizona.edu

---

Supplementary Information

Table of Contents:

Plasmid	DNA Binding Domain	Source	I.M.A.G.E. No.
pOTB7 MeCP2	flMeCP2 (1-486)	Open Biosystems	3956518
pUC57 MBD1	MBD1 (1-69)	Genscript	-
pCMV-SPORT6 MBD2	flMBD2 (1-411)	Open Biosystems	5496721
pUC57 MBD3	MBD3 (1-72)	Genscript	-
pOTB7 MBD4	flMBD4 (1-580)	Open Biosystems	3534047
pCMV-SPORT6 ZBTB4	ZBTB4 (268-460)	Open Biosystems	6175382
pBluescriptR ZBTB33	ZBTB33 (455-639)	Open Biosystems	4828271
PAD1258	ZBTB38 (418-612)	Sasai et al. (2010)	-
pCR-BluntII-TOPO UHRF1	flUHRF1 (1-793)	Open Biosystems	40053955
pCMV-SPORT6 UHRF2	UHRF2 (405-663)	Open Biosystems	6500979
pMal E2C	E2C (1 – 172)	Tan et al. (2004)	-

Table S1. Summary of Starting DNA-Binding Domain Constructs. The residues of each protein used in each construct are shown in parenthesis. Full-length proteins are designated “fl”.

Plasmid	Fusion Protein	DNA Binding Domain	Fluc Fragment
pcDNA3.1(+) MeCP2-PWNFluc	MeCP2-NFluc	MeCP2 (77-167)	NFluc (2-416)
pcDNA3.1(+) MBD1-PWNFluc	MBD1-NFluc	MBD1 (1-69)	NFluc (2-416)
pcDNA3.1(+) MBD2-PWNFluc	MBD2-NFluc	MBD2 (147-215)	NFluc (2-416)
pcDNA3.1(+) MBD3-PWNFluc	MBD3-NFluc	MBD3 (1-72)	NFluc (2-416)
pcDNA3.1(+) MBD4-PWNFluc	MBD4-NFluc	MBD4 (76- 148)	NFluc (2-416)
pcDNA3.1(+) ZBTB4-PWNFluc	ZBTB4-NFluc	ZBTB4 (268-460)	NFluc (2-416)
pcDNA3.1(+) ZBTB33-PWNFluc	ZBTB33-NFluc	ZBTB33 (455-639)	NFluc (2-416)
pcDNA3.1(+) ZBTB38-PWNFluc	ZBTB38-NFluc	ZBTB38 (418-612)	NFluc (2-416)
pcDNA3.1(+) UHRF1-PWNFluc	UHRF1-NFluc	UHRF1 (414-617)	NFluc (2-416)
pcDNA3.1(+) UHRF2-PWNFluc	UHRF2-NFluc	UHRF2 (443-647)	NFluc (2-416)
pcDNA3.1(+) PWCFluc-Zif268	PWCFluc-Zif268	Zif268 (335 - 423)	CFluc (398-550)
pDNC	CFluc-Zif268	Zif268 (335 - 423)	CFluc (398-550)
pDNC MBD1-NFluc	MBD1-NFluc	MBD1 (1 – 69)	NFluc (2-416)
pDNC CFluc-E2C	CFluc-E2C	E2C (1 – 172)	CFluc (398-550)
pDNC CFluc-MBD1	CFluc-MBD1	MBD1 (1 – 69)	CFluc (398-550)

Table S2. Summary of Synthesized Fusion Constructs. The residues of each protein used in each construct are shown in parenthesis.

Target	Sequence
C2E(m)	5’-GCGTA^{m}CGTACACTGCGGCTCCGGCCCCTACCG-3’ 3’-CGCATGC^{m}ATGTGACGCCGAGGCCGGGGATGGC-5’
C2Z(u)	5’-GCGTACGTACGCCCACGCCACCG-3’ 3’-CGCATGCATGCGGGTGCGGTGGC-5’
C2Z(tm)	5’-GCGTA^{m}CGTACGCCCACGCCACCG-3’ 3’-CGCATGCATGCGGGTGCGGTGGC-5’
C2Z(bm)	5’-GCGTACGTACGCCCACGCCACCG-3’ 3’-CGCATGC^{m}ATGCGGGTGCGGTGGC-5’
C2Z(m)	5’-GCGTA^{m}CGTACGCCCACGCCACCG-3’ 3’-CGCATGC^{m}ATGCGGGTGCGGTGGC-5’
C6C(m)	5’-GATCA^{m}CGATGGTA^{m}CGACTAG-3’ 3’-CTAGTGC^{m}TACCATGC^{m}TGATC-5’
C21C(m)	5’-GCCTA^{m}CGACTATCACCGCGGGTGATAGT^{m}CGTAGGC-3’ 3’-CGGATGC^{m}TGATAGTGGCGCCCACTATCAGC^{m}ATCCG-5’

Table S3. Summary of Target Oligonucleotides. The appropriate zinc finger sites are highlighted in blue, methylated CpG dinucleotides in red, and unmethylated CpG dinucleotides in green. All oligos were synthesized by Integrated DNA Technologies and HPLC purified.

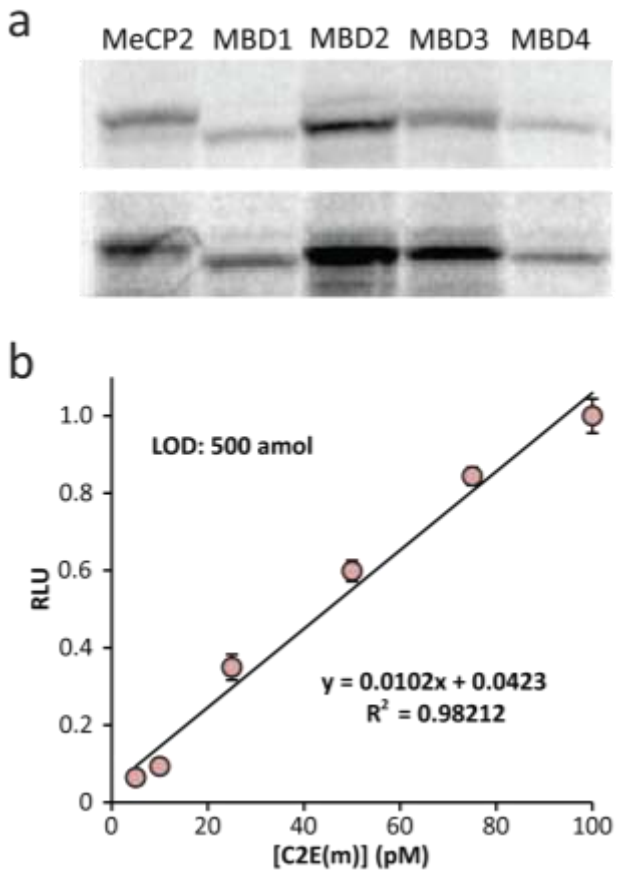

Figure S1. Interrogation of MBD-Fluc Translation Efficiency and MBD1/E2C Limit of Detection. a) ^{35}S-methionine labeled SDS-PAGE gel of *in vitro* translated MBD-NFluc fusion proteins. Two trials of the same experiment are shown. b) Relative luminescence in the presence of decreasing concentrations of the target, C2E(m), using the sensor comprising MBD1-NFluc and CFluc-E2C, shows a limit of detection (LOD) of 500 amols.

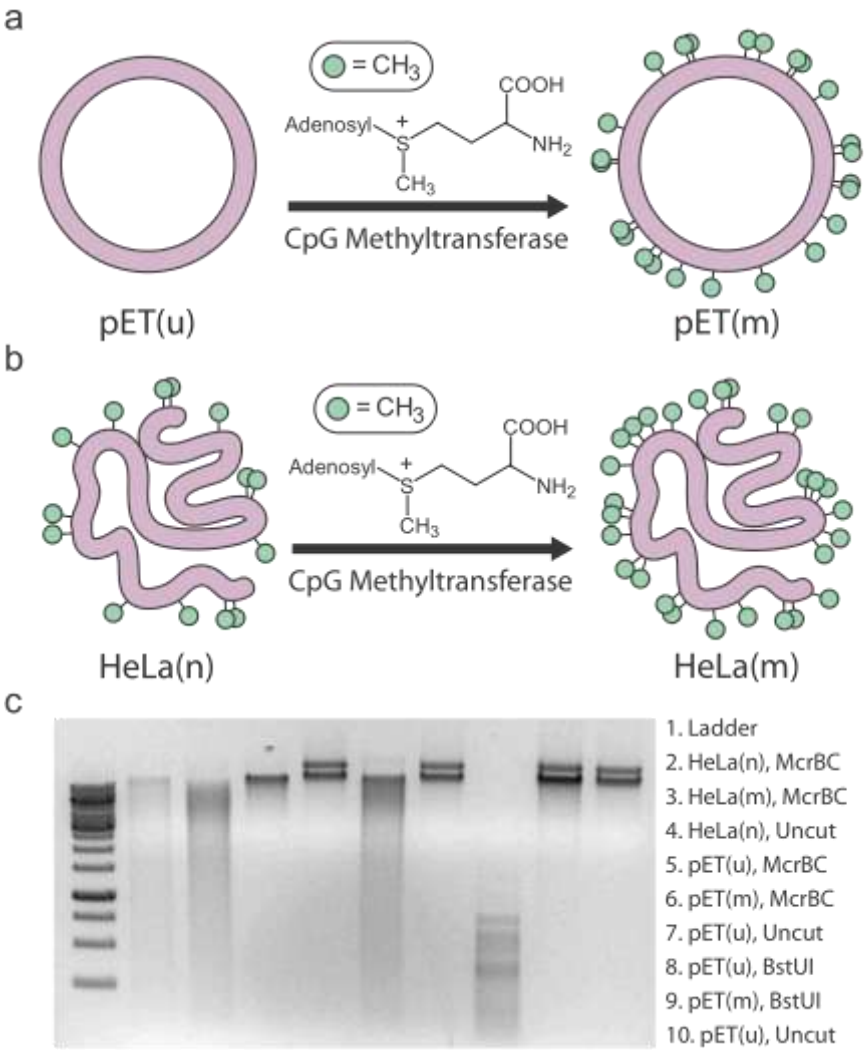

Figure S2. Exogenous Methylation of DNA. (a) Plasmid pET(u) was treated with a CpG methyltransferase to yield a fully methylated plasmid, pET(m). (b) Natively-methylated HeLa genomic DNA HeLa(n) was treated with a CpG methyltransferase to yield the fully methylated counterpart, HeLa(m). (c) HeLa(n) and HeLa(m) were digested with McrBC endonuclease, while pET(u) and pET(m) were digested with both BstUI and McrBC endonucleases. BstUI cleaves the symmetric 5'-CGCG-3' site only in the absence of methylation, while McrBC cleaves between two distant (G/A)mC sites separated by up to 3000 basepairs. BstUI fully degrades unmodified pET(u), while the fully methylated pET(m) is not digested. McrBC degrades HeLa(n), HeLa(m) and pET(m), indicative of CpG methylation.

Figure S3. Evaluation of Global HeLa Methylation. Genomic DNA extracted from 5-aza-2'deoxycytidine- or vehicle-treated HeLa cells was digested with McrBC endonuclease. McrBC degrades vehicle-treated DNA the most, followed by 0.10 µM 5-aza-2'deoxycytidine and 1.0 µM 5-aza-2'deoxycytidine, indicating decreased genomic DNA methylation with the addition of 5-aza-2'deoxycytidine.

Supplementary Protocol - CGContentAnalyzer Code (Java):

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.TreeMap;

public class CGContentAnalyzer {
	private TreeMap<Integer, Integer> histogram;
	private String sequence;

	public CGContentAnalyzer(String fileName) {
		StringBuilder temporarySequence = new StringBuilder();
		try {
			histogram = new TreeMap<Integer, Integer>();
			BufferedReader reader = new BufferedReader(new FileReader(fileName));
			String nextLine = reader.readLine();
			while (nextLine != null) {
				temporarySequence.append(nextLine);
				nextLine = reader.readLine();
			}
			sequence = sequenceBuilder(temporarySequence.toString());
			reader.close();
		} catch (FileNotFoundException fnfe) {
			fnfe.printStackTrace();
		} catch (IOException ioe) {
			ioe.printStackTrace();
		}
	}

	private String sequenceBuilder(String sequence) {
		StringBuilder result = new StringBuilder();
		sequence = sequence.toUpperCase();
		for (int index = 0; index < sequence.length(); index++)
			if (isValidBase(sequence.charAt(index)) == true)
				result.append(sequence.charAt(index));
		return result.toString();
	}

	private boolean isValidBase(char possibleBase) {
		return possibleBase == 'A' || possibleBase == 'G'
				|| possibleBase == 'C' || possibleBase == 'T';
	}

	public void run() {
		int CGcounter = 0;
		int temporaryCounter = 0;
		int CGindex = 0;
```

```
                // find the first occurrence of a CG pair
                while ( CGindex < sequence.length() -1 && "CG".equals(sequence.substring(CGindex, CGindex +
2)) == false) {
                        CGindex++;
                }
                // rearrange circular plasmid to start indexing at a CG pair
                sequence = sequence.substring(CGindex) + sequence.substring(0, CGindex);
                System.out.println("Imported " + sequence.length() + " nucleotides");
                for (int index = 0; index < sequence.length() - 1; index++) {
                        // if a CG pair is found, the length between this and the last is
                        // added to the histogram
                        if ("CG".equals(sequence.substring(index, index + 2))) {
                                index++;
                                CGcounter++;
                                if (histogram.containsKey(temporaryCounter)) {
                                        histogram.put(temporaryCounter, histogram
                                                        .get(temporaryCounter) + 1);
                                } else {
                                        histogram.put(temporaryCounter, 1);
                                }
                                temporaryCounter = 0;
                        } else
                                temporaryCounter++;
                }
                // boundary condition to count the distance from the last CG pair
                temporaryCounter++;
                if (histogram.containsKey(temporaryCounter)) {
                        histogram
                                        .put(temporaryCounter, histogram.get(temporaryCounter) + 1);
                } else {
                        histogram.put(temporaryCounter, 1);
                }
                // remove the first CG pair encountered at onset
                histogram.put(0, histogram.get(0) - 1);
                try {
                        BufferedWriter writer = new BufferedWriter(new FileWriter(
                                        "results.txt"));
                        writer.write("Number of CG's: " + CGcounter);
                        writer.newLine();
                        writer.write("Length" + "\t" + "Occurences");
                        writer.newLine();
                        for (int key : histogram.keySet()) {
                                writer.write(key + "\t" + histogram.get(key));
                                writer.newLine();
                        }
                        writer.close();
                } catch (FileNotFoundException fnfe) {
                        fnfe.printStackTrace();
                } catch (IOException ioe) {
                        ioe.printStackTrace();
                }
        }
```

}

mCpG Distance Separation Calculation by CGContentAnalyzer. The function for determining the occurrences of CG in a plasmid was written in Java 6 and utilized via a simple consol interface. The function begins by constructing a CGContentAnalyzer object using the supplied plasmid file name. A histogram was stored as a TreeMap with the key representing the distance between CG pairs and the value of the number of occurrences at this length. Data was read into a StringBuilder, instead of a String, to decrease runtime due to multiple concatenation events. The temporary sequence is then converted to a String representing a sequence containing only A, C, G, T characters. This allows the user to analyze sequences containing line breaks and numbering of base pairs. The run method is then used to analyze the sequence generated in the constructor. The first step iterates over the sequence to find the first CG pair and wraps the sequence to ensure the start of the sequence is a CG pair. This is a valid operation as the sequences were circular plasmids. Next, the method iterates through the entire sequence detecting CG pairs. When a CG pair is found, the number of bases iterated since the last CG pair is recorded in the histogram, otherwise the counter for number of bases since the last CG event is incremented. After the sequence has been scanned completely, the final counter is added to the histogram to count the final CG separation. Next the histogram value at 0 is decremented to prevent counting the first CG twice. Finally the results are written to a text file for further analysis.