Published in final edited form as:

Nat Protoc. 2021 April ; 16(4): 2190–2212. doi:10.1038/s41596-021-00496-3.

A streamlined solution for processing, elucidating and quality control of cyclobutane pyrimidine dimer sequencing data

Quanhu Sheng^{1,4}, Hui Yu^{2,4}, Mingrui Duan², Scott Ness², Jiapeng He², Huining Kang², Limin Jiang², John J. Wyrick³, Peng Mao^{2,✉}, Yan Guo^{2,✉}

¹Department of Biostatistics, Vanderbilt University Medical Center, Nashville, TN, USA.

²Comprehensive Cancer Center, Department of Internal Medicine, University of New Mexico, Albuquerque, NM, USA.

³School of Molecular Biosciences, Washington State University, Pullman, WA, USA.

⁴These authors contributed equally: Quanhu Sheng, Hui Yu.

Abstract

UV radiation may lead to melanoma and nonmelanoma skin cancers by causing helix-distorting DNA damage such as cyclobutane pyrimidine dimers (CPDs). These DNA lesions, if located in important genes and not repaired promptly, are mutagenic and may eventually result in carcinogenesis. Examining CPD formation and repair processes across the genome can shed light on the mutagenesis mechanisms associated with UV damage in relevant cancers. We recently developed CPD-Seq, a high-throughput and single-nucleotide resolution sequencing technique that can specifically capture UV-induced CPD lesions across the genome. This novel technique has been increasingly used in studies of UV damage and can be adapted to sequence other clinically relevant DNA lesions. Although the library preparation protocol has been established, a systematic protocol to analyze CPD-Seq data has not been described yet. To streamline the various general or specific analysis steps, we developed a protocol named CPDSeqr to assist researchers with CPD-Seq data processing. CPDSeqr can accommodate both a single- and multiple-sample experimental design, and it allows both genome-wide analyses and regional scrutiny (such as of suspected UV damage hotspots). The runtime of CPDSeqr scales with raw data size and takes roughly 4 h per sample with the possibility of acceleration by parallel computing. Various guiding graphics are generated to help diagnose the performance of the experiment and inform regional enrichment of CPD formation. UV damage comparison analyses are set forth in three analysis scenarios, and the resulting HTML pages report damage directional trends and statistical significance. CPDSeqr can be accessed at <https://github.com/shengqh/cpdseqr>.

Reprints and permissions information is available at www.nature.com/reprints.

✉ Correspondence and requests for materials should be addressed to P.M. or Y.G. pmao@salud.unm.edu; yanguo1978@gmail.com. Author contributions Q.S., H.Y. and L.J. developed code for the protocol. M.D. and J.H. performed protocol testing. H.K. provided statistical support. J.J.W. and P.M. provided knowledge support for CPD-Seq. Y.G., P.M., H.Y. and S.N. wrote the manuscript. Y.G. and P.M. supervised the project.

Competing interests

The authors declare no competing interests.

Supplementary information The online version contains supplementary material available at <https://doi.org/10.1038/s41596-021-00496-3>.

Introduction

The incidence rate of skin cancer has increased over recent decades. There are 2–3 million new skin cancer cases every year worldwide, and it is one of the most common cancers in the United States¹. Skin cancer has a mutagenesis process that is ascribed primarily to DNA damage induced by prolonged exposure to ultraviolet (UV) light. In addition, UV signature mutations are identified in cutaneous T cell lymphomas and other cancer types, suggesting that UV radiation is associated with DNA mutations in multiple types of cancer.

A major lesion caused by the most harmful UV light (UVB and UVC) is the cyclobutane pyrimidine dimer (CPD)². CPDs disrupt the DNA double-helical structure and block DNA and RNA polymerases, which may eventually result in mutagenesis or cell death if not repaired properly. Genome sequencing of human melanomas has shown that predominant UV signature mutations (i.e., C→T transitions) are almost entirely associated with the CPD-forming dipyrimidine sequence, highlighting the important contribution of UV damage to melanoma mutations. Therefore, investigating CPD formation and repair processes across the entire genome is essential for understanding the mutational mechanisms in skin cancers. To meet the need for research on UV damage repair and mutagenesis, we previously developed a new high-throughput CPD sequencing method called ‘CPD sequencing’ (CPD-Seq)^{3,4}. Although CPD-Seq does not directly measure mutations, data generated by CPD-Seq can provide important insights into the mechanism for DNA mutations observed in melanoma. CPD-Seq was first published in 2016 to study UV damage in yeast³. The yeast CPD-Seq data indicate that the chromosome landscape, particularly DNA packaging by nucleosomes and DNA binding by transcription factors (TFs), plays a profound role in affecting the local DNA sensitivity to UV light³. In 2018, CPD-Seq was adapted to study mutations in human melanoma. In a study by Mao et al., the authors found that CPD lesions are strikingly elevated at active binding sites of E26 transformation-specific (ETS) transcription factor, an important oncogenic TF family⁵. Furthermore, ETS-induced CPD hotspots are highly correlated with recurrent mutations at ETS binding sites in melanoma⁵. This finding was soon confirmed by another research group, who utilized CPD-Seq to show similar UV-induced CPD hotspots near ETS binding sites in a melanoma cell line^{6,7}. The CPD-Seq technique has been improved to detect lesions induced by two major harmful UV types, UVC and UVB⁸. CPD-Seq data have also been used to investigate novel mutational signatures associated with nucleosome structure. For example, Brown et al. found an ~10-bp oscillation pattern of melanoma mutations across the 147 bp of DNA that constitutes the nucleosome core particle, which suggests that the structural conformations of DNA modulate CPD formation within nucleosomes with positional bias⁹, with periodic mutation peaks occurring at DNA positions facing away from the histone core. This nucleosome-associated mutation pattern resonates with CPD-Seq data generated in human cells irradiated by UV light, because CPD formation is also regulated similarly by the nucleosome rotational setting, displaying ~10-bp periodicity on the nucleosome surface⁹. In addition to the unique CPD formation pattern on nucleosomes, new CPD-Seq data revealed that repair of UV damage in nucleosomes is asymmetric, with elevated repair activity on the 5′ end of both DNA strands¹⁰, which is probably caused by left-handed wrapping around the histone octamer. Duan et al. used CPD-Seq to find that yeast Rad26, a homolog of human

Cockayne syndrome group B protein, is not uniformly required for transcription-coupled nucleotide excision repair¹¹.

In summary, a number of CPD-Seq datasets have provided genome-wide insights into DNA damage distribution and repair kinetics, which has important implications in illuminating cancer mutagenesis. CPD-Seq uses DNA repair enzymes to cleave the damage site and create a new 3'-OH group for sequencing adaptor ligation. The principle of CPD-Seq can be applied for sequencing other types of DNA damage, as long as a damage-specific enzyme is available. Indeed, CPD-Seq has been adapted for mapping alkylation damage—a parallel method, *N*-methylpurine sequencing, has been developed for precisely mapping mutagenic alkylation damage such as 3-methyladenine¹². In addition, another sequencing technology named 'GLOE-Seq'¹² was recently developed by Sriramachandran et al. to map single-strand breaks and DNA base lesions at the genome scale. GLOE-Seq is similar to CPD-Seq in utilizing DNA repair enzymes to generate damage-associated 3'-OHs for adaptor DNA ligation, and thus uses similar data analysis methods. With the large amount of data being generated by CPD-Seq and its sibling techniques, such as *N*-methylpurine sequencing and GLOE-Seq, it is imperative to develop a bioinformatics method to streamline numerous general or specific data analysis steps.

Through working with CPD-Seq data on several projects, we developed a CPD-Seq data processing and analysis protocol. This protocol is composed of components of quality control (QC), data processing and UV damage analysis, which are delivered through a combination of mature bioinformatics programs and self-written Python, R and Unix shell scripts. We rely on current public CPD-Seq data to establish expected ranges for QC measures; these expected ranges may undergo appreciable changes as we incorporate newly accrued CPD-Seq data. Because most CPD-Seq studies are currently conducted in human cells or yeast, CPDSeqr is tailored to human and yeast genomes. To address the special QC requirement of CPD-Seq data, we designed innovative metrics and procedures that take into account background dinucleotide composition in the reference genome. Disparity and disproportionality in libraries of different samples present another challenge to QC and damage analysis, and thus we leveraged the trimmed mean of M-values (TMM) method¹³ to alleviate this problem. Future CPD-Seq experiments may comprise spiked-in UV-damaged plasmid DNA for aiding in library size normalization, and our protocol will evolve accordingly to take advantage of the envisioned improvement in experimental design.

Because of the enormous volume of sequencing data and the requirement for external programs, CPDSeqr is designed to run in the Linux environment. In some steps of the protocol, we have adopted established programs such as Bowtie2¹⁴ for read alignment and deepTools¹⁵ for GC bias correction. In theory, alternative programs such as BWA¹⁶ can be used for read alignment. In practice, however, because of potential incompatibility in data formats or argument settings, such substitutions may necessitate additional code modifications for the protocol to function correctly. The protocol components regarding special QC and UV damage analysis are specifically tailored for CPD-Seq data; to the best of our knowledge, no existent alternative solutions are readily available for these analytical purposes.

CPD sequencing

CPD-Seq is specifically designed to capture UV-induced DNA damage. The detailed library preparation techniques have been provided in our original yeast CPD-Seq article³. In CPD-Seq, human or yeast cells are exposed to UV light to induce damage in the genomic DNA. Genomic DNA is purified and sonicated to short fragments of ~400 bp. After end repair and dA-tailing, DNA fragments are ligated with the first adaptor DNA (Fig. 1, green). Afterward, all free 3'-OH groups are blocked by terminal transferase and dideoxyATP (Fig. 1, 'dd') to prevent nonspecific ligation with the second adaptor DNA. DNA is then digested by repair enzymes T4 endonuclease V and APE1 to generate a new ligatable 3'-OH group on the 5' side of the CPD damage. DNA fragments are denatured and ligated to the second adaptor, which is a double-stranded DNA with an overhang containing six random nucleotides (Fig. 1, red). After purification with streptavidin beads (one strand of the second adaptor has a biotin label) and second strand synthesis, the resultant double-stranded CPD-Seq library is briefly amplified by PCR by using primers complementary to the two adaptors and sequenced with the second adaptor primer (Fig. 1). CPD formation is known to be modulated by DNA sequence constituents. For example, CPDs are usually formed between two consecutive pyrimidines such as TT, TC, CT and CC, with CPD formation being highest at TT sequences. Therefore, DNA with frequent dipyrimidine oligonucleotides is more prone to CPD formation. Naked DNA data, generated by a CPD-Seq experiment where all proteins are removed from genomic DNA and the naked DNA is exposed to UV light, can be used as an important control to account for the impact of DNA sequence on UV damage. By normalizing cellular CPD-Seq data to naked DNA data, we can assess how various chromatin features, including nucleosomes and TF binding, affect CPD formation⁹. In this protocol, we utilize previously generated human⁵ and yeast³ naked CPD-Seq data for normalization purposes.

Previous studies used UVB⁸ or UVC³⁻⁵ radiation to induce CPD damage. Single-end sequencing is sufficient to capture the CPD damage. Pair-end sequencing has also been used in recent CPD-Seq studies⁶, which can provide more accurate alignment than single-end sequencing because of the pair-end design. However, only read 1 was used to count CPD damage. Our protocol is compatible with both sequencing strategies and can determine during alignment if the CPD-Seq data is pair-end. Read 2 in the pair-end design contributes to read alignment but is ignored in CPD damage analysis.

Overview of the protocol

The comprehensive bioinformatics analysis protocol CPDSequer is organized into three major sections: QC, raw data processing and CPD damage analysis. CPD-Seq has been successfully implemented with both Illumina⁸ and Ion Torrent¹¹ platforms, which use the same FASTQ format for storing raw data. Our protocol is designed to analyze FASTQ files and thus can be applied to data generated by different sequencing technologies (e.g., Illumina, Ion Torrent and others). The whole protocol comprises 17 steps as illustrated in Fig. 2.

QC is an important step in genomic data analysis to ensure robust and reproducible results. Many previously established QC protocols and techniques¹⁷⁻¹⁹ are applicable to CPD-Seq

datasets. Our protocol focuses on several unique aspects of CPD-Seq QC to evaluate the experiment's success, including UV radiation damage, CPD-Seq library preparation and sequencing efficiency. Raw data processing involves alignment of sequencing reads to the reference genome, followed by identification and counting of CPD-associated dinucleotides. After raw data processing, our protocol conducts a series of statistical and bioinformatics analyses, ranging from standard to tailored analysis strategies, to address important questions related to specific projects. A series of resources have been pre-compiled for the protocol, including background dinucleotide counts for the whole genome (e.g., human GRCh38 and GRCh37 and yeast sacCer3) and 113 browser extensible data (BED) files of genomic regions in the human or yeast genome (e.g., promoters, 3' untranslated regions and nucleosomes). The runtime of the protocol is estimated by using a Linux workstation with Ubuntu operating system, with Intel Xeon CPU E5-2650 V4 at 2.20 GHz and 32 GB of memory. Different modules of the protocol are demonstrated on several human CPD-Seq samples (See Data availability), which have ~30 million raw reads.

Materials

Equipment

Genome files

- *Sequencing raw data in FASTQ format.* FASTQ format is the standard format for storing raw high-throughput sequencing data. CPD-Seq FASTQ data can be obtained from databases such as Gene Expression Omnibus or generated by users.
- *Genome reference in FASTA format.* FASTA format is the standard format for storing reference genome sequences. Reference genome sequences for common species such as yeast and humans can be downloaded from many public sources, including the NCBI or Ensembl.
- *Indexed genome reference file.* Indexing a large reference genome FASTA file allows quicker access to a specific position in the genome. Bowtie2 genome index files for common species such as yeast and humans can be downloaded from the Bowtie2 portal (<http://bowtie-bio.sourceforge.net/bowtie2/index.shtml>).
- *(Optional) genome reference in TwoBit (2bit) format.* TwoBit genome files are necessary in the optional step of correcting GC bias. TwoBit genome files for GRCh38, GRCh37 and sacCer3 can be found within the species-specific directories at <http://hgdownload.cse.ucsc.edu/gbdb/>. Search for '.2bit' at the end of the file name. Otherwise, FASTA files can be converted to 2bit by using the University of California, Santa Cruz program called faToTwoBit available for different platforms at <http://hgdownload.cse.ucsc.edu/admin/exe/>.

Hardware

- *Linux workstation.* Recommended specs: Ubuntu or other compatible operating system, Intel or AMD 8–12 core CPU at 3.0 GHz, 32 GB of memory and a 1-TB hard drive.

Software

- *Bowtie2* (v2.4.1)¹⁴. Bowtie2 is an alignment tool based on Burrows Wheeler Algorithm. It is used to index a reference genome FASTA file and align raw CPD-Seq data to an indexed reference genome. Sequence alignment map (SAM) files¹⁶ are generated with the use of Bowtie2.
- *SAMtools*¹⁶. SAMtools is a sequencing data processing package that provides various sequencing data utilities. It is used to convert SAM files to binary alignment map (BAM) files and sort and index BAM files.
- *Tabix*. Tabix can index position-sorted files in tab-delimited format. Tabix is used to index the intermediate files during raw data processing for faster access.
- *Python 3.7*. The main CPD-Seq data processing script is written in Python 3.7.
- *R version 3.5 or above*. R is used to generate the QC report, CPD damage analysis report and other reporting figures. Users do not need to call R procedures directly. All R procedures are called through Python scripts.
- *Pandoc v2.10.1*. Pandoc provides support to document format conversion, a necessary step in our QC and CPD damage analysis modules.
- *deepTools* v2¹⁵. Two utilities, computeGCBias and correctGCBias, are embedded in deepTools, and they are used to compute and correct GC-content bias.
- *Je*²⁰. Je is a versatile suite to handle multiplexed next-generation sequencing libraries with unique molecular identifiers. It provides a solution to demultiplex a FASTQ file of mixed samples.
- *R and Python scripts*. Twenty-two Python scripts and eight R scripts were written for this protocol. They are available at <https://github.com/shengqh/cpdseqr>.

Resource files

- We have prepared 113 files in BED format and nine other resource files to accompany this protocol and deposited them at the corresponding GitHub repository and our own project webpage (<https://cqsweb.app.vumc.org/Data/cpdseqr/>). The direct download link, file names and description of these resource files are in Supplementary Data 1. Common and large resource files such as the human genome reference FASTA files are not provided, because they can be downloaded easily from multiple sources.

Equipment setup

The protocol CPDSeqr requires several software tools to be installed under the Linux operating system. During software installation, the ‘sudo’ command implies that the user must have a super user privilege. The commands used for installation of required software such as pip may require installation as well. If the user has trouble installing Python or other Linux-based software, a system administrator should be contacted. To install required software, follow the steps below.

1. Install Bowtie2 in Ubuntu with the following command:

```
sudo apt-get install -y bowtie2
```

2. Install Tabix in Ubuntu with the following command:

```
sudo apt-get install -y tabix
```

3. Install SAMtools in Ubuntu with the following command:

```
wget https://github.com/samtools/samtools/releases/download/1.10/samtools-1.10.tar.bz2
tar -jxvf samtools-1.10.tar.bz2
cd samtools-1.10./configure
make
sudo make install
```

If a newer version of SAMtools becomes available, the wget command may fail. Please visit <http://www.htslib.org/download/> for updated installation commands.

4. Install Pandoc

```
PANDOC_VERSION="2.7.3"
cd /opt
wget
https://github.com/jgm/pandoc/releases/download/${PANDOC_VERSION}/pandoc-${PANDOC_VERSION}-linux.tar.gz
tar -xzvf pandoc-${PANDOC_VERSION}-linux.tar.gz
sudo rm pandoc-${PANDOC_VERSION}-linux.tar.gz
echo "export PATH=/opt/pandoc-${PANDOC_VERSION}/bin:${PATH}"
» ~/.bashrc
```

5. Install R packages in R

```
install.packages(c("knitr", "rmarkdown", "data.table", "R.utils",
"ggplot2", "reshape2"))
if (!requireNamespace("BiocManager", quietly = TRUE))
install.packages("BiocManager") BiocManager::install("edgeR")
```

6. Install deepTools (optional)

```
sudo pip3 install deeptools
```

7. Install Je (optional)

```
cd /opt
sudo wget https://github.com/gbcs-embl/Je/raw/master/dist/je_1.2.tar.gz
sudo tar -xzvf je_1.2.tar.gz
```



```
sudo rm je_1.2.tar.gz
echo "export PATH=/opt/je_1.2:\$PATH" » ~/.bashrc
```

8. Install Python 3.7 in Ubuntu. Most factory versions of Ubuntu 18.04 and later come with Python pre-installed. To check if Python is installed and the Python version, use the following command:

```
python -version
```

If Python is not installed or the version is lower than 3.7, use the following commands in sequential order to install Python 3.7:

```
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository ppa:deadsnakes/ppa
sudo apt update
sudo apt install python3.7
```

9. Install CPDSeqr in Ubuntu with the following command:

```
sudo pip3 install git+git://github.com/shenggh/cpdseqr.git
```

Procedure

De-multiplexing of raw data ● Timing ~10 min

1. If raw sequencing data in FASTQ format is multiplexed, they need to be first de-multiplexed on the basis of barcode sequence. Otherwise, skip to the next step. There are two ways to de-multiplex a multiplexed FASTQ file: using a command from CPDSeqr (option A) or using Je (option B).

A. Using a command from CPDSeqr to de-multiplex a multiplexed FASTQ file

- i. Run the following command:

```
cpdseqr demultiplex [-h] -i [INPUT] -o [OUTPUT] -b [BARCODEFILE]
```

The `-h` parameter displays the help information for the current command, and it serves the same purpose for all downstream CPDSeqr functions (Steps 6–17). The `[INPUT]` parameter is the name for the input FASTQ file to be de-multiplexed. The `[OUTPUT]` parameter denotes the name of the output directory. If the specified directory does not exist, it will be created on the basis of the given name. The `[BARCODEFILE]` parameter denotes the file that contains the barcodes for de-multiplexing. The file is expected to be tab delimited, with the first column denoting the barcode sequence and the second

column the sample ID. Example rows of [BARCODEFILE] are shown below.

```
ATCGACTSample1
CGTGTTCSample2
```

The de-multiplexing should produce N individual FASTQ files, where N corresponds to the number of samples included in [BARCODEFILE].

B. Using Je to de-multiplex a multiplexed FASTQ file

i. Run the following command:

```
je demultiplex F1=[INPUT] BF=[JE_BARCODEFILE]
O=[OUTPUT]
```

The [INPUT] parameter is the name for the input FASTQ file to be de-multiplexed. The [OUTPUT] parameter denotes the name of the output directory. The [JE_BARCODEFILE] parameter denotes a barcode file that contains three tab-separated columns indicating sample name, barcode and output file name. Example rows of [JE_BARCODEFILE] are shown below.

```
ControlATCGCGATControl.fastq.gz
UVtreatGAACTGATUVtreat.fastq.gz
```

QC on raw FASTQ data ● Timing 3–4 h per sample

2. Even though the purpose of CPD-Seq and conventional DNA sequencing are quite different, common raw data QC procedures can be useful for identifying potential sequencing problems. Without going into detail, we recommend previously established methods such as FASTQC²¹ and QC3²² for this general QC step. Users should examine such primary QC parameters as total sequenced reads, base quality by cycles, nucleotide distribution by cycle, per sequence GC content, per sequence base quality score, sequence duplication level, etc., making sure that each sample reaches modern standards on most QC parameters. Although the absolute threshold values may vary depending on the specific study design and goals, a rule of thumb is that significant deviation from the mass samples signifies an outlier sample, which may need to be discarded.
3. Generate the indexes for reference genome FASTA by using Bowtie2 with the following command:

```
bowtie2-build [--threads [THREADS]][INPUT] [OUTPUT]
```

This step needs to be implemented only once per reference genome. If indexes for the reference genome of interest already exist, skip to the next step. Otherwise, exert the command as shown above. The [INPUT] parameter is the reference genome FASTA file. The [OUTPUT] parameter is the basename (prefix) for the result indexes. The [THREADS] parameter is an integer denoting the number of parallel threads used for building the indexes. Users should adjust this number on the basis of their computer capability. The recommended number is 8. The [THREADS] parameter in the following steps (Steps 4 and 5) is interpreted in the same way as here in Step 3.

4. Align CPD-Seq raw FASTQ data and post-process the alignment.

A. Single-end sequencing data

i. Use the following command:

```
bowtie2 -p [THREADS] -x [INDEX] -U [FASTQ] -S [SAM]
| samtools sort -o [OUTPUT] -T [TEMP_PREFIX] [-@
[THREADS]] [-m [MAX_MEMORY]] -
```

B. Pair-end sequencing data

i. Use the following command:

```
bowtie2 -p [THREADS] -x [INDEX] -1 [FASTQ1] -2
[FASTQ2] -S [SAM] | samtools sort -o [OUTPUT] -T
[TEMP_PREFIX] [-@ [THREADS]] [-m [MAX_MEMORY]] -
```

The command lines above resort to a Shell pipe to serialize the alignment and the post-processing actions.

The alignment process is achieved via the bowtie2 command. The [INDEX] parameter is the prefix of index files generated from Step 3. In the scenario of single-end sequencing, the [FASTQ] parameter is the raw sequencing FASTQ file. In the scenario of pair-end sequencing, the [FASTQ1] parameter is the read 1 FASTQ file, and the [FASTQ2] parameter is the read 2 FASTQ file. The [SAM] parameter is the name of the output SAM file. The user can elect to ignore the read 2 from pair-end sequencing by using only read 1 for alignment. However, pair-end alignment is more accurate than single-end alignment. Thus, we recommend using both reads in the pair for alignment. In the later steps, only alignment from read 1 will be used to quantify CPD damage.

The post-processing of alignment is achieved via a Samtools command. The [OUTPUT] parameter denotes the name of the output BAM file, and it must have the correct suffix of 'bam'. The [TEMP_PREFIX] is a string used as the prefix

of the temporary files. The [MAX_MEMORY] parameter indicates approximately the maximum required memory that can be assigned per thread (e.g., 4 GB). This step produces an alignment BAM file (Fig. 2), where the alignments are sorted by chromosome and then by position.

5. (Optional) Correct GC content bias by using the following deepTools commands:

```
computeGCBias -b [IN_BAM] -g [GENOME] --effectiveGenomeSize
[GENOME_SIZE] -GcbiasFrequenciesFile [TEXT_OUT] [-p [THREADS]]
correctGCBias -b [IN_BAM] -o [OUT_BAM] -g [GENOME]
--effectiveGenome-Size [GENOME_SIZE] -GcbiasFrequenciesFile
[TEXT_OUT] [-p [THREADS]]
```

Parameters involved in the computeGCBias command are as follows. The [GENOME] parameter designates the reference genome file in 2bit format. The [IN_BAM] parameter designates the sorted BAM file as output in Step 4. The [GENOME_SIZE] parameter indicates the size of the mappable genome. The [TEXT_OUT] parameter gives the file name to save the file containing the observed and expected read frequencies per %GC-content.

In the correctGCBias command line, the [TEXT_OUT] parameter should inherit the same value as that assigned in the computeGCBias command line, meaning that correctGCBias takes the output file of computeGCBias as a primary input. The [OUT_BAM] is the output file name for the GC bias-corrected BAM file, and it must have a suffix of '.bam'.

6. Count CPD-associated dinucleotides with the following command:

```
cpdseq bam2dinucleotide [-h] -i [INPUT] -g [FASTA] [-q [MAPPING_
QUALITY]] [-m [MIN_COVERAGE]] [-u] [-t] -o [OUTPUT]
```

The [INPUT] parameter is the sorted and indexed BAM file from Step 4 or Step 5. The [FASTA] parameter is the reference genome FASTA file. The [MAPPING_QUALITY] parameter is an integer of mapping quality score threshold, preset at default 20. The [MIN_COVERAGE] parameter is an integer of coverage threshold, preset at default 1. Any read with the mapping quality or the coverage not satisfying the minimum threshold is excluded from the downstream analysis. The [-u] parameter designates that only the reads uniquely mapped to the genome will be used. The [-t] parameter allows the user to run a brief testing of the functionality by using only the first 1,000,000 reads in the BAM file. The [OUTPUT] parameter is the prefix for the output files. This step completes raw data processing and generates important output files that will be required in several steps in the following workflow (Fig. 2). Two major output files from this step are [OUTPUT].bed.bgz and [OUTPUT].count, which we refer to as the dinucleotide BED file and the dinucleotide Count file, respectively (for more information, see Anticipated results). A dinucleotide

BED file conforms to the BED format, and it records the specific genomic locations of individual dinucleotides where a CPD has occurred. A file with suffix '.tbi' is generated by Tabix as an index of the dinucleotide BED file. A dinucleotide Count file summarizes the total number of CPD-occurring reads ('read count') or genomic locations ('site count') for 16 dinucleotide types, stratified for chromosomes. Site count designates the number of unique genomic positions that return a nonzero read count. Read count designates the sum of all read counts. The dinucleotide Count file and dinucleotide BED file (as well as the Tabix index file) should always be stored in the same folder.

7. (Optional) Generate text files to provide information on the expected lesion along chromosomes. Generate a binwise dinucleotide site summary file based on the reference genome

```
cpdseger fasta2bincount -i [INPUT_FA] [-b [BLOCK]] -o [OUTPUT]
```

Generate a binwise dinucleotide read-count summary file based on CPD read count information

```
cpdseger dinucleotide2bincount -i [INPUT_DI] [-g [GENOME]]
[-b [BLOCK]] -o [OUTPUT]
```

The [INPUT_FA] parameter entailed by fasta2bincount is a FASTA file for the reference genome. The [INPUT_DI] parameter entailed by dinucleotide2bincount is a dinucleotide BED file output from bam2dinucleotide (Step 6). The [BLOCK] takes an integer value designating the size of continuous bins along the chromosome, defaulted at 100,000 bp. The [GENOME] parameter specifies a reference genome, taking values hg38 (default) or hg19. It can also be chromosome length file in which the first column is chromosome name and the second column is chromosome length. The [OUTPUT] parameter indicates the output text file name.

8. (Optional) Subtract short tandem repeat regions or narrow down to the genomic regions of interest.

```
cpdseger filter -i [INPUT] -c [COORDINATE] -o [OUTPUT] [-m [MODE]]
```

The [INPUT] parameter indicates a dinucleotide BED file output from the bam2dinucleotide command (Step 6). The [COORDINATE] parameter specifies a BED-formatted file for genomic locations of suspicious repeat regions. We have included the URLs to our published resource of very short tandem repeats²³ in Supplementary Data 1. Users can download the human or yeast Mono-Nucleotide Repeats or Di-Nucleotide Repeats and subtract those regions. The [OUTPUT] specifies the prefix of the output files. This optional command results in a bulk of files in the same format as those of the output files of Step 6.

The [MODE] parameter indicates the operation mode. The default value is 'subtract', which is recommended for common practices such as subtracting repeat regions. Alternatively, the user can set it as 'intersect', whereby only the reads observed in the user-defined genomic regions are retained and summarized.

? TROUBLESHOOTING

QC based on dinucleotide count results ● Timing ~5 min (without Step 9) or ~10 min (with Step 9) per sample

9. (Optional) Estimate sample-wise normalization factors to adjust for size and composition discrepancy across sample libraries.

```
cpdseger size_factor -i [INPUT] -o [OUTPUT] [--calc_type
[CALCULATION_TYPE] }
```

This step is applicable to the GRCh37 and GRCh38 genomes only. The [INPUT] parameter expects a tab-delimited file that lists one or multiple dinucleotide BED files (generated in Step 6) labeled with sample names. Example rows of the [INPUT] file are shown below.

```
UVrep1.bed.bgzUVrep1
UVrep2.bed.bgzUVrep2
Contr11.bed.bgzCtrl1
Contr12.bed.bgzCtrl2
```

The [OUTPUT] parameter specifies the prefix of the output text file in which the inferred normalization factors are written. The [CALCULATION_TYPE] parameter can be set to either 'site_union' or 'chrom_dinucleotide', allowing two alternative ways to delimit the set of presumably invariant entities as the basis of TMM inference. The parameter value of 'site_union' designates the use of the union set of all reads-called dinucleotide sites across all samples, whereas the alternative value of 'chrom_dinucleotide' refers to the 368 combinations formed between 23 chromosome and 16 dinucleotide classes.

10. Perform QC based on dinucleotide count results by using the following command:

```
cpdseger qc [-h] -i [INPUT] -o [OUTPUT] [-n [NAME]] [--count_type
[COUNT_TYPE]] [-g [GENOME]] [-s [SIZE_FACTOR_FILE]]
```

This step achieves versatile QC diagnostics upon CPD read count information (the dinucleotide BED/Count files). QC at the current step provides information regarding CPD-Seq experiment success specifically, which is fundamentally different from the previous Step 2 of general QC. The [INPUT] parameter designates a tab-delimited text file describing a list of dinucleotide BED files (generated in Step 6), following the same format requirement as in Step 9.

This file can have only one row or multiple rows (see more explanations in Anticipated results).

The [NAME] parameter denotes the project name when multiple samples are fed to this step, and it is used to label the figure and tables in the QC report. The [OUTPUT] parameter is the prefix for the output HTML report. Users can assign the same value to both [NAME] and [OUTPUT], but this is not required. The [GENOME] parameter designates the target reference genome, which has a predefined scope of {hg38, hg19, or sccCer3} and is preset at a default value of 'hg38'. The [COUNT_TYPE] parameter allows a choice between read count and site count for quantitative comparison (see Step 6 above), assuming 'rCnt' (read count, default) or 'sCnt' (site count). This parameter works the same way for Steps 10 and 13–17.

The optional parameter [SIZE_FACTOR_FILE] designates a headed, tab-delimited file containing sample-wise normalization factors. This file can be output from Step 10 or compiled by the user. Using an actual normalization factor file generated in Step 9 enforces TMM normalization of library sizes, and thus the Count-Per-Million (CPM) values shown in the report page and used in the following damage analysis steps would have been subjected to library size adjustment. Alternatively, the user can compile a normalization factor file ab initio, in which the sample-specific normalization factors are obtained in a custom way that is not TMM. If normalization factors are all set to 1, this is in effect equivalent to not setting the [SIZE_FACTOR_FILE]—the library size normalization will be turned off, and CPM values will be derived on the basis of raw library sizes. The normalization factor file contains three columns, with the first column for sample name, the second column for normalization factor and the third column for raw library size. Example rows of a normalization factor file are shown below:

```
SampleSizeFactorLibrarySize
UVBrep10.5568047595
UVBrep20.9041458023
Control2.0118664590
```

CPD damage analysis ● Timing ~10 min per analysis

▲**CRITICAL** Steps 11–17 describe CPD damage analysis. There is no particular order for performing these steps, which provides a flexible outline for researchers to choose and arrange steps on the basis of their specific endpoints.

11. To generate a genome-wide UV damage distribution map, use the following command:

```
cpdseger fig_genome [-h] -i [INPUT] [-b [BLOCK]] [-g [GENOME]] [-n  
[NORMALIZATION]] -o [OUTPUT]
```

The [INPUT] parameter denotes a file that describes a list of dinucleotide BED files; this file follows the same format requirement as the [INPUT] parameter in Step 9. The [BLOCK] parameter, at default value 100,000 bp, denotes the block size parameter entailed in drawing a genome-wide CPD distribution. The optional [GENOME] parameter lets the user select a version of the human reference genome. The choices are ‘hg38’ (default) and ‘hg19’, representing GRCh38 and GRCh37, respectively, or a tab-delimited file (intended for sacCer3) in which the first column is chromosome name and the second column is chromosome length. The [OUTPUT] parameter is the output file name.

The [NORMALIZATION] parameter takes one of three values: {None, Total, LocalGC}. When ‘None’ is specified, the command plots the raw summed read counts along the chromosome in (default) 100,000-bp running bins; when ‘Total’ is specified, the command plots the summed read counts divided by the total read counts; when ‘LocalGC’ is specified, the command plots specific quotient values along the same running bins, where the quotient values are obtained by dividing the original count numbers with the GC-associated count numbers in each bin.

12. Previous studies have found positional bias for CPD damage in certain types of genomic regions, such as the oscillating pattern in nucleosomes. To identify similar patterns in a specific genomic region type, generate a dinucleotide aggregate figure with the following command:

```
cpdseger fig_position [-h] -i [INPUT] -c [COORDINATE_FILE] [-b
[BACKGROUND_FILE]] [--space] [--add_chr] [-t] -o [OUTPUT]
```

The [INPUT] parameter denotes the input dinucleotide BED list file that follows the same format requirement as in Step 9. The [COORDINATE_FILE] parameter is the BED file for the specific regions that the user designates to plot on the position aggregate figure. The [-t] parameter allows the user to run a brief test with only the first 10,000 lines in the genomic region file. The [--space] parameter allows the user to use a space instead of the default tab as the delimiter in the genomic region file. The [--add_chr] parameter allows the user to add the string ‘chr’ to the chromosome name in the genomic region file. The [--space] and [--add_chr] parameters are always in companion with the [COORDINATE_FILE] parameter in Steps 12, 14, 15 and 17. The [OUTPUT] parameter specifies the prefix of the output figure name.

The optional [BACKGROUND_FILE] parameter specifies the file name for the dinucleotide BED file generated by the command bam2dinucleotide (Step 6) for a naked DNA CPD-Seq experiment; designating such a dinucleotide BED file would normalize the results on the basis of the naked DNA CPD-Seq data (see Introduction, CPD sequencing). The file name must end with the ‘bed.bgz’ suffix, and it should be accompanied by a dinucleotide Count file (with the ‘.count’ suffix) and a Tabix index file (with the ‘bed.bgz.tbi’) in the same directory. We have prepared a naked DNA dinucleotide BED file along with companion files for humans and yeast, respectively, for use at this step directly

(Supplementary Data 1). However, users can supply their own dinucleotide BED/Count files from a naked DNA CPD-Seq experiment for normalization.

? TROUBLESHOOTING

13. To compare genome-wide UV radiation damage of one or multiple samples against the reference genome background, use the following command:

```
cpdseger uv_comp_genome [-h] -i [INPUT] -o [OUTPUT] [-g [GENOME]]
[--count_type [COUNT_TYPE]] [-s [SIZE_FACTOR_FILE]]
```

The [GENOME] parameter indicates which reference genome is used. The options are 'hg19' (GRCh37), 'hg38' (GRCh38, default) and 'saccer3'. The [OUTPUT] parameter is the prefix for the output file, and it works the same way for Steps 13–17. The [INPUT] parameter expects a tab-delimited file that lists one or multiple dinucleotide Count files (generated in Step 6) labeled with sample names. When multiple Count files are supplied, they are regarded as coming from one homogeneous group. The comparison is made between the sample(s) and the reference genome. Example rows of the [INPUT] file are shown below.

```
UVrep1.countUVrep1
UVrep2.countUVrep2
```

The optional [SIZE_FACTOR_FILE] parameter designates the normalization factor file generated in Step 9 (or compiled ab initio), following the format requirement for the [SIZE_FACTOR_FILE] parameter of Step 10. Given the normalization factor file, the underlying statistical analysis is performed on library-size-normalized CPM values rather than raw read count numbers. If the normalization file is one output from the above Step 9, the library size normalization method is TMM. Alternatively, the user can resort to a custom method to obtain sample-wise normalization factors and format such normalization factors in the required format to generate a normalization factor file. The [SIZE_FACTOR_FILE] parameter is applicable to four CPD damage analysis commands (uv_comp_genome, uv_comp_genome_region, uv_comp_groups, and uv_comp_groups_region) in Steps 13, 14, 16 and 17, with the same analysis purpose.

14. To compare UV damage of one or multiple samples against the reference genome background with respect to a specific type of region, use the following command:

```
cpdseger uv_comp_genome_region [-h] -i [INPUT] -o [OUTPUT] -c
[COORDINATE_FILE] -f [FASTA] [--add_chr] [--space] [--count_type
[COUNT_TYPE]] [-s [SIZE_FACTOR_FILE]]
```

The [COORDINATE_FILE] parameter is a genomic region file in BED format with at least three columns (chromosome, start and end) for the specific regions (e.g., nucleosomes and promoters). This file can be in either raw BED format or its gzipped format. The [FASTA] parameter is the reference genome FASTA file. The underlying statistical analysis relies on a background dinucleotide density of the specific regions, and the program will combine the reference and genomic region files to infer such necessary information. The [INPUT] parameter expects a tab-delimited file that lists one or multiple dinucleotide BED files (generated in Step 6), following the same format requirement as in Step 9. Multiple files represent repetitive samples from one group; one sample per group is allowed. The comparison is made between the sample(s) and the reference genome, taking into account only the user-specified regions.

? TROUBLESHOOTING

15. To compare UV damage between two regions for one or multiple samples, use the following command:

```
cpdseger uv_comp_regions [-h] -i [INPUT] -o [OUTPUT] -c1
[COORDINATE_FILE1] -c2 [COORDINATE_FILE2] -f [FASTA] [--add_chr]
[--space] [--count_type [COUNT_TYPE]]
```

The [INPUT] parameter expects a tab-delimited file that lists one or multiple dinucleotide BED files (generated in Step 6), following the same format requirement as in Step 9. Multiple files represent repetitive samples from one group; one sample per group is allowed. The comparison is made between two types of genomic regions. The [COORDINATE_FILE1] and [COORDINATE_FILE2] parameters specify the two types of genomic regions in BED format for comparison, following the same format requirement as in Step 14. The underlying statistical analysis relies on the background dinucleotide density of the two types of genomic regions, and the program will combine the reference genome file ([FASTA]) and these two genomic region files ([COORDINATE_FILE1] and [COORDINATE_FILE2]) to infer such necessary information.

? TROUBLESHOOTING

16. To compare genome-wide UV damage between two groups (e.g., case versus control) of samples, use the following command:

```
cpdseger uv_comp_groups [-h] -i1 [INPUT1] -i2 [INPUT2] -o [OUTPUT]
[--count_type [COUNT_TYPE]] [-s [SIZE_FACTOR_FILE]]
```

The [INPUT1] and [INPUT2] parameters each expect a tab-delimited file that lists one or multiple dinucleotide Count files (generated in Step 6), following the same format requirement as in Step 13. A single sample per group is allowed. The comparison is made between the two groups of samples.

17. To compare UV damage between two groups (e.g., case versus control) of samples within a type of region, use the following command:

```
cpdseq uv_comp_groups_region [-h] -i1 [INPUT1] -i2 [INPUT2] -o
[OUTPUT] -c [COORDINATE_FILE] [--add_chr] [--space] [--count_type
[COUNT_TYPE]] [-s [SIZE_FACTOR_FILE]]
```

The [INPUT1] and [INPUT2] parameters each expect a tab-delimited file that lists one or multiple dinucleotide BED files (generated in Step 6), following the same format requirement as in Step 9. A single sample per group is allowed. The comparison is made between the two groups of samples, but only the user-specified regions are taken into account.

? TROUBLESHOOTING

Troubleshooting

Steps 8, 12, 14, 15 and 17

Users must be aware of the format of chromosome names in their initial genome FASTA file, which is carried over to the alignment BAM file. If the BAM file designates a chromosome as '1', '2', etc., ensure that the chromosome names in the genomic region file (the [COORDINATE_FILE] parameter) as required in Steps 12, 14, 15 and 17 do not have a 'chr' prefix, and do not apply the '--add_chr' option to these steps. On the contrary, if the chromosome names in the BAM file appear in the form of 'chr1', 'chr2', etc., ensure that the chromosome names in the genomic region file are in the same format or apply the '--add_chr' option to accompany the [COORDINATE_FILE] parameter.

A genomic region file in BED format as expected in Steps 12, 14, 15 and 17 can have one or zero header lines. If a header line is present, it must be prefixed with a hash sign (#).

Timing

The initial installation step takes ~10 min. All software requires one-time installation and does not need to be re-installed for additional CPD-Seq data analysis sessions.

Step 1, demultiplexing of raw data: ~20 min

Step 2, performing general QC on raw sequencing FASTQ data: ~20 min per sample

Step 3, building the index: ~1 h for humans and ~10 min for yeast

Step 4, read alignment and post-processing: ~2–3 h per sample

Step 5, correcting GC bias: ~1–2 h per sample

Step 6, counting reads to quantify CPD damage: ~10 min per sample

Step 7, examining the expected lesion: ~5 min per sample

Step 8, subtracting or intersecting: ~5 s per sample

Step 9, estimating normalization factors: ~5 min ('chrom_dinucleotide') or ~30 min ('site_union') per sample

Step 10, QC: ~5 min per sample

Steps 11–17: various CPD damage analyses: ~10 min per step

Steps 2–10's runtime scales with the initial FASTQ file size, and in these steps multiple samples can be processed in parallel to conserve the overall computational time of a project. Steps 11–17's runtime scales with the number of samples included in a study.

Anticipated results

An example dataset together with the corresponding testing code is available at <https://cqsweb.app.vumc.org/Data/cpdseq/>. Primary output files of several major steps in the protocol are indicated in the workflow chart (Fig. 2).

Step 1

Step 1 demultiplexes multiple samples out of a single input FASTQ file of mixed samples. Multiple FASTQ files each corresponding to an individual sample are generated.

Step 2

Step 2 is a QC step on raw sequencing FASTQ data. Because this step is not unique to CPD-Seq data, we refer to other established QC protocols^{17,18,21,22} to ensure that sequencing reads have high quality for subsequent alignment. The acceptable quality level for raw sequencing data varies on the basis of study design. In general, a total read count of 20–30 million is expected for human samples, and 5–10 million is expected for yeast samples. Furthermore, one should expect that UV-damaged samples have evidently higher read count than control samples. The GC content is dependent on species. Although most concepts of raw sequencing data QC can be applied to CPD-Seq raw data, domain knowledge of CPD-Seq is required to make proper QC assessment.

Steps 3 and 4

Steps 3 and 4 are standard sequencing data processing steps, including building a reference index, alignment, sorting and indexing. For Step 3, a total of six files will be created. More details for creating index files can be found in the Bowtie2 manual (<http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml>). For Step 4, the result for each sample is a sorted and indexed BAM file together with the corresponding index files.

Step 5

Step 5 is optional, and it resorts to the computeGCBias and correctGCBias utilities from deepTools to correct the GC content bias commonly present in high-throughput sequencing data. The computeGCBias command produces a text file documenting the inferred GC bias frequency statistics, and this text file must be supplied as an indispensable input to the

ensuing correct GCBias command. Ultimately, Step 5 generates for each sample a sorted and indexed BAM file together with the corresponding index files.

Step 6

Step 6 counts the number of CPD-associated dinucleotides from the BAM file resulting from Steps 4 or 5. This step generates four files: [OUTPUT].bed.bgz, [OUTPUT].bed.bgz.tbi, [OUTPUT].count and [OUTPUT].log.

[OUTPUT].bed.bgz

We call this bgzip compressed file a dinucleotide BED file. It records read counts of dinucleotides at specific genomic positions. There are six columns in this file. The first column is the chromosome name, the second column is the starting genomic position for the dinucleotide, the third column is the end genomic position for the dinucleotide, the fourth column is the dinucleotide type, the fifth column is the read count number after filtering by the [MAPPING_QUALITY] parameter and the sixth column denotes the strand of the CPD damage.

[OUTPUT].bed.bgz.tbi

This is the Tabix index file for [OUTPUT].bed.bgz.

[OUTPUT].count

We call this text file a dinucleotide Count file. It contains the dinucleotide read counts and site counts summarized by chromosome. There are four columns in this file. The first column denotes the chromosome name, the second column denotes dinucleotide type, the third column denotes the number of reads that aligned adjacent to this type of dinucleotide and the fourth column denotes the number of unique sites of this dinucleotide found within this chromosome.

[OUTPUT].log

This text log file contains command running logs for debugging purposes.

Step 7

Step 7 is optional, and it provides information on expected lesions along chromosomes. CPD formation is known to be dependent on DNA sequence content, and intrinsic dipyrimidine constitution especially predisposes a site to CPD damage formation. Therefore, we conjecture that the expected lesion intensity along chromosomes may be represented as the fraction of CPD-prone dinucleotide sites (such as dipyrimidine sites) in running bins in the reference genome. fasta2bincount sums up the instances of each of the 16 dinucleotide classes in running bins in the reference genome. The output file can be processed to derive the running intensity of expected lesions in the reference genome; for example, users can sum up the four columns for TT, TC, CC and CT dinucleotides and divide these sums with the row totals. Exact integer numbers for all 16 dinucleotide classes are retained in the output to allow for maximal flexibility in the definition of expected lesions.

To enable the user to align the actual, sample-specific CPD-Seq data with expected lesion data in a comparable data format, we developed `dinucleotide2bincount` as well. We recommend that users take advantage of the text files output from `fasta2bincount` and `dinucleotide2bincount` to investigate expected versus observed lesions or related scientific questions. Like `fasta2bincount`, `dinucleotide2bincount` outputs sums count numbers with respect to 16 dinucleotide classes for continuous bins along chromosomes. The fundamental difference is that `fasta2bincount` summarizes the numbers of dinucleotide instances according to the static composition of the reference genome, whereas `dinucleotide2bincount` summarizes the distinct dinucleotide sites with called reads as well as the total read counts associated with these sites. The format of the `dinucleotide2bincount` output file is very similar to that of the `fasta2bincount` output file, except that it possesses 32 columns rather than 16 columns. Half of the 32 columns are dedicated for summed read counts, and the other half are for distinct site counts.

Step 8

Step 8 restricts the census of the CPD damage to a subsection of the whole genome as defined by the user. In a ‘subtract’ context, genomic regions supplied by the user are subtracted from the analysis. In an ‘intersect’ context, only the reads observed in the user-designated genomic regions are considered. Step 8 generates a bundle of output files in the same format as in Step 6.

Step 9

Step 9 estimates normalization factors to adjust for size and composition discrepancy across sample libraries. It performs the TMM normalization algorithm on raw read count numbers to infer sample-specific normalization factors, which can be multiplied with raw library sizes to quantify effective library sizes. A tab-delimited text file with a prefix specified by the user is generated to store the inferred normalization factors. There are three columns in this output file, which contain sample names, normalization factors and raw library sizes, respectively. This normalization factor file will be required in the following QC (Step 10) to report CPM summaries, and it is also needed in the various CPD damage analyses (Steps 13–17), if the user would like to perform the analysis on CPM values rather than raw count values.

Step 10

Step 10 is a CPD-Seq-specific QC step. The command automatically detects if the input involves only one sample or multiple samples, and the content of the resulting HTML report page is contingent on the detected scenario. The report page for a single-sample input has more in-depth demonstration of the individual QC measures (Fig. 3a–c), whereas the report page for multi-sample input comprises CPM information as well as graphs revealing inter-sample relationships (Fig. 3d–h). Several CPD-Seq data-specific quality metrics are captured at this step, including overall CPD efficiency (Fig. 3a,d), contrast of di-thymine to di-adenosine (Fig. 3a,d), and strand symmetry (Fig. 3c). Distribution of the read count or CPM statistics are displayed in barplots (Fig. 3e). When multiple samples are fed to the command, principal component analysis, clustering analysis and Pearson correlation

analysis are called on to produce a scatter plot (Fig. 3f), a heatmap (Fig. 3g) and a complex sample-correlation plot (Fig. 3h).

First, Step 10 computes two major QC measures: (i) overall experiment efficiency, which is computed as the ratio of the read count proportion for CPD-associated dinucleotides (i.e., TT, TC, CC and CT) to the baseline proportion based on the dinucleotide composition within the reference genome (Eq. 1); and (ii) enrichment contrast of adjacent thymines relative to adjacent adenosines, which is computed as the ratio of reads mapped to TT dinucleotides to reads mapped to AA dinucleotides (Eq. 2). A successful CPD experiment should enrich reads to dipyrimidine positions, and there should be a substantial contrast between TT and AA reads because TT dinucleotides are the most preferential sequence for CPD formation, whereas the complementary sequence (AA) is unlikely to be damaged. The combination of high efficiency and high contrast indicates an effective performance of the experiment for specific UV-induced CPD lesions (Fig. 3a). To assist with an objective evaluation of the calculated QC measures, we provide expected ranges for efficiency and contrast, respectively, which were based on currently available CPD-Seq data consisting of 10 human samples^{5,6,8} and 5 yeast samples³. Analysis of these available data suggests that the expected range for overall efficiency in human CPD-Seq experiments is between 2.1 and 2.9, and the contrast between TT and AA reads is between 7.7 and 23.8. We expect to see more published CPD-Seq data and will accordingly expand the reference samples to refine the expected ranges for both overall efficiency and contrast.

$$Efficiency = \frac{(Summed\ read\ counts\ for\ \{TT, TC, CC, CT\}) / (Total\ read\ counts)}{(\#\ Genome\ incidences\ of\ \{TT, TC, CC, CT\}) / (Genome\ size)} \quad (1)$$

$$Contrast = \frac{Read\ count\ for\ TT}{Read\ count\ for\ AA} \quad (2)$$

Second, for an overview of the distribution of read counts, we plot the read count numbers in grouped barplots at three levels: >0, >5 and >10 (Fig. 3b). The read counts over the three successive levels decline nonlinearly, so the *y* axis is rendered at a log scale. The read counts for four dipyrimidines are displayed separately. Theoretically, the TT bars should be appreciably taller than the bars of other dipyrimidines.

Lastly, we provide a symmetry measure that reflects how essential QC metrics are balanced between the forward strand and the reverse strand of the chromosomes (Fig. 3c). QC metrics are calculated for the forward and the reverse strands, respectively, and each pair of QC quantities are scaled to a sum of 1. The scaled QC metrics are visualized in a stacked barplot overlaid with a central line that reflects the perfect symmetry of 0.5:0.5. Caution is raised if the distribution of paired QC metrics deviates obviously from the central line.

Diagnosing multiple samples simultaneously enables the user to visually identify possible outlier samples that differ substantially from the master group on major QC metrics. In the multi-sample QC report page, we plot efficiency/contrast measures for all samples of the same study in one plot (Fig. 3d), highlighting noteworthy data points farther than 1.5 times the inter-quantile range from the lower/upper quantiles. Read count and CPM frequency bars

for multiple samples are plotted side by side to enable direct visual scrutiny (Fig. 3e). If the user has supplied reasonable normalization factors such as those inferred in Step 9, the CPM values have been subjected to library size and content normalization. Finally, common multi-sample explorative graphics, including a principal component analysis plot (Fig. 3f), a heatmap plot with clustering dendrograms (Fig. 3g) and a cross-sample pairwise scatter plot (Fig. 3h), are provided for users' visual diagnostics.

Step 11

From the dinucleotide BED file generated in Step 6, Step 11 extracts dipyrimidine reads (i.e., TT, TC, CT and CC reads) and uses them as the input to visualize the CPD damage along the chromosomes in running blocks. To reduce runtime, the default genome block is set at 100,000 bp. All CPD damage within the 100,000-bp block size is combined for plotting purposes.

This step generates two major output files (file names may vary, reflecting specific parameter setting): `cpd.report_ReadCount.pdf` and `cpd.report_Sitecount.pdf`.

`cpd.report_ReadCount.pdf`

This is the genome-wide CPD damage distribution figure based on the read count of CPD damage. The darker color of the vertical line indicates more CPD damage within the block.

`cpd.report_Sitecount.pdf`

This is the genome-wide CPD damage distribution figure based on the site count of CPD damage. The darker color of the vertical line indicates that more CPD damage sites were identified within the block.

Several other output files will also be generated for debugging and reproducibility purposes, including a dynamically generated R code for generating figures, chromosome length information file, QC report configuration file, QC report log file and one dinucleotide block counting file for each sample. An example output can be found in Fig. 4. For a successful CPD-Seq experiment, the UV light-treated sample should exhibit substantially higher CPD damage levels across the whole genome than a control sample after normalization and correction for local GC content, as indicated by the darker color in the UV-irradiated sample in Fig. 4.

If users believe that the GC content affects the damage distribution in a dataset, they can also perform Step 5 to correct GC bias before counting the number of CPD-associated dinucleotides from the BAM file (Step 6).

Step 12

Step 12 is used to generate a CPD damage aggregate figure based on a specific region. As in Step 11, reads called at dipyrimidine sites are used as the input to visualize CPD damage in a specific genomic region. A previous CPD study⁹ has found CPD damage positional bias within nucleosome regions after normalizing to naked DNA CPD-Seq data. Users can supply pre-compiled (Supplementary Data 1) or customized naked DNA CPD-Seq

dinucleotide BED files for a recommended normalization. There are four major output files for this step: `cpd_position.txt_ReadCount.pdf`, `cpd_position.txt_ReadNormalizedCount.pdf`, `cpd_position.txt_SiteCount.pdf` and `cpd_position.txt_SiteNormalizedCount.pdf`.

`cpd_position.txt_ReadCount.pdf`

This file contains the CPD damage positional aggregate figures for all four CPD-prone dipyrimidines (TT, TC, CC and CT), individually and combined. This file is based on read count without normalization by naked DNA CPD-Seq data.

`cpd_position.txt_ReadNormalizedCount.pdf`

This file contains the CPD damage positional aggregate figures for all four CPD-prone dipyrimidines, individually and combined. This file is based on read count with normalization by naked DNA CPD-Seq data.

`cpd_position.txt_SiteCount.pdf`

This file contains the CPD damage positional aggregate figures for all four CPD-prone dipyrimidines, individually and combined. This file is based on CPD site count without normalization by naked DNA CPD-Seq data.

`cpd_position.txt_SiteNormalizedCount.pdf`

This file contains the CPD damage positional aggregate figures for all four CPD-prone dipyrimidines, individually and combined. This file is based on CPD site count with normalization by naked DNA CPD-Seq data.

Other output files include a text file that contains the position count information used to generate the figure, a log file that contains log information for debugging purposes and an R script that is dynamically generated on the basis of the user input. An example of a nucleosome positional aggregate figure after normalization by naked CPD-Seq data can be seen in Fig. 5. A total of 113 BED files have been preprocessed.

Steps 13–17

There are three major comparison scenarios: (i) comparing one sample or more samples of one group with the reference genome, (ii) comparing two regions in one sample or more samples of one group and (iii) comparing two groups. Steps 13 and 14 concern scenario i, Step 15 concerns scenario ii and Steps 16 and 17 concern scenario iii. In each scenario, the comparisons are set forth in two different modalities: the one-versus-others modality comparing one specific dipyrimidine category against ‘others’, or dinucleotide types that should not be affected by CPD damage (those other than TT, TC, CC and CT); and the overall modality that accommodates all four dipyrimidine categories as opposed to the others. Scenarios i and iii allow analyses on two alternative quantification values: raw read count and CPM, which can be designated by supplying a null normalization file and a valid normalization file, respectively. Each of Steps 13–17 returns an HTML analysis report that contains analysis results and explanation.

Step 13

Step 13 is used when the user intends to compare the CPD damage of one or more UV light-treated samples against the reference genome in a genome-wide manner. The essential question being answered here is if the read count number of a dipyrimidine type is significantly greater than 'others', or dinucleotide types that should not be affected by CPD damage (those other than TT, TC, CC and CT), while taking into account the dinucleotide type proportion in the reference genome. The resultant table includes five rows and six columns. The top four rows show one-versus-others comparison results, each focusing on a particular dipyrimidine type. The fifth row shows the overall statistical test results, where the read counts are counted toward each of five categories (TT, TC, CC, CT and others). The six columns from left to right are expected and observed fractions, unadjusted and adjusted *P* values of a one-sided exact test and unadjusted and adjusted *P* values of a chi-squared test. A *P* value $< 1 \times 10^{-15}$ will be displayed as '1E-15'. In the one-versus-others rows, the expected/observed situations reflect the reference/observed frequency of the concerned dipyrimidine; in the overall row, they designate the ratios between all five dinucleotide types. In Steps 13–17, whenever actual read counts rather than probability-like frequency values are required by the statistical test, we use the count numbers preferentially in millions units or otherwise in thousands units. Hence, the expected/observed values may contain total numbers counted in millions or thousands, which are explicitly explained in companion notations. The major output for Step 13 is an HTML report. Other output files include a dynamically generated R script, an R markdown file, an option file and a log file.

Step 14

Step 14 is used when the user intends to compare CPD damage between one or more UV-treated samples against the reference genome in a type of genomic region (e.g., nucleosomes and introns). During this step, the specific dinucleotide background information will be computed on the basis of the [COORDINATE_FILE] parameter from the genome and compared to the CPD damage counted within the same specific regions in the UV light-treated sample. As in Step 13, a binomial test (exact test) and a chi-squared test are used for this comparison. The major output for Step 14 is an HTML report. Other output files include a dynamically generated R script, an R markdown file, an option file, a log file and a regional CPD Count file.

Step 15

Step 15 is used to compare CPD damage distribution between two genomic regions within one or more samples of the same group. The essential question being answered here is if there is a significant difference in the read count number of dipyrimidine type between two genomic regions of interest, while taking into account the inherent compositions of the dipyrimidine type of these two regions in the reference genome. The resultant table includes five rows and seven columns. As in Steps 13 and 14, a binomial test (exact test) and a chi-squared test are used for this comparison. The top four rows each focus on a particular dipyrimidine type. The fifth row reflects the summary over the four dipyrimidine types. The seven columns from left to right are expected and observed ratios, unadjusted and adjusted *P* values of a one-sided exact test, unadjusted and adjusted *P* values of a chi-squared test and

directionality of the comparison result (column '1stVS2nd'). In the one-versus-others rows, the expected/observed ratios are expressed as the ratios of two counts from the two regions of interest; in the overall row, they are shown as nonapplicable. When column 1stVS2nd shows '>', the first type of regions tend to show a higher UV damage than the second type of regions; when 1stVS2nd shows '<', the first type of regions tend to show a lower UV damage than the second type of regions. The statistical significance (*P* value) of this tendency is indicated in the column 'p.exact'. The major output for Step 15 is an HTML report. Other output files include a dynamically generated R script, an R markdown file, an option file, a log file and a regional CPD Count file.

Step 16

Step 16 is used to perform genome-wide CPD damage comparison between two groups of samples. The essential question being answered here is if there is a significant difference in the proportion of a dipyrimidine type between the two groups. A hypergeometric test (exact test) and a chi-squared test are used for this comparison. The resultant table includes five rows and seven columns. The top four rows each focus on a particular dipyrimidine type. The fifth row shows the overall statistical test results, where the read counts are counted toward each of five categories (TT, TC, CC, CT and others). The seven columns from left to right are expected and observed read counts, unadjusted and adjusted *P* values of a one-sided exact test, unadjusted and adjusted *P* values of a chi-squared test and directionality of the comparison result (column '1stVS2nd'). In the one-versus-others rows, the expected/observed situations show expected/observed counts of the concerned dipyrimidine between the two samples; in the overall row, they recapitulate the actual/expected two-by-five contingency tables entailed in the overall chi-squared test. When column 1stVS2nd shows '>', the first group of samples tend to show a higher UV damage than the second group; when 1stVS2nd shows '<', the first group of samples tends to show a lower UV damage than the second group. The statistical significance (*P* value) of this tendency is indicated in the column 'p.exact'. The major output for Step 16 is an HTML report. Other output files include a dynamically generated R script, an R markdown file, an option file, a log file and a regional CPD Count file.

Step 17

Step 17 is used to compare CPD damage between two groups of samples within a specific type of region. Unlike when comparing to the reference genome, comparing CPD damage between two groups within the same region does not require the computation of background dinucleotide information. The statistical methods used are the same as those used in Step 16. The major output for Step 17 is an HTML report. Other output files include a dynamically generated R script, an R markdown file, an option file, a log file and a regional CPD count file.

Code availability

This protocol, including all scripts (Shell, Python and R), is hosted at <https://github.com/shengqh/cpdseger>. A comprehensive test case involving all 17 steps entails an empirical CPD-Seq dataset and corresponding stepwise testing code scripts, which are available at <https://cqsweb.app.vumc.org/Data/cpdseger/>.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgements

This study was supported by a Cancer Center Support Grant (P30CA118100) and R01ES030993-01A1 from the National Cancer Institute, funding from the National Institutes of Health (R21ES029302), a pilot grant from the UNM Center for Metals in Biology and Medicine (P20GM130422), the Bioinformatics Shared Resources and the Biostatistics Shared Resources at The Comprehensive Cancer Center. None of the funding bodies were involved in the study design; data collection, analysis or interpretation; or writing of the manuscript.

Data availability

All datasets (GSE103487⁵, GSE79977³ and GSE119249⁶) used in the demonstration for this protocol are available through the NCBI Short Read Archive (<https://www.ncbi.nlm.nih.gov/sra>). All figures used in this article are original. All preprocessed resource files listed in Supplementary Data 1 are available at <https://cqsweb.app.vumc.org/Data/cpdseqqr/>.

References

1. Guy GP, Machlin SR, Ekwueme DU & Yabroff KR Prevalence and costs of skin cancer treatment in the US, 2002–2006 and 2007–2011. *Am. J. Prev. Med* 48, 183–187 (2015). [PubMed: 25442229]
2. Mouret S et al. Cyclobutane pyrimidine dimers are predominant DNA lesions in whole human skin exposed to UVA radiation. *Proc. Natl Acad. Sci. USA* 103, 13765–13770 (2006). [PubMed: 16954188]
3. Mao P, Smerdon MJ, Roberts SA & Wyrick JJ Chromosomal landscape of UV damage formation and repair at single-nucleotide resolution. *Proc. Natl Acad. Sci. USA* 113, 9057–9062 (2016). [PubMed: 27457959]
4. Mao P, Wyrick JJ, Roberts SA & Smerdon MJ UV-induced DNA damage and mutagenesis in chromatin. *Photochem. Photobiol* 93, 216–228 (2017). [PubMed: 27716995]
5. Mao P et al. ETS transcription factors induce a unique UV damage signature that drives recurrent mutagenesis in melanoma. *Nat. Commun* 9, 2626 (2018). [PubMed: 29980679]
6. Elliott K et al. Elevated pyrimidine dimer formation at distinct genomic bases underlies promoter mutation hotspots in UV-exposed cancers. *PLoS Genet.* 14, e1007849 (2018). [PubMed: 30586386]
7. Premi S et al. Genomic sites hypersensitive to ultraviolet radiation. *Proc. Natl Acad. Sci. USA* 116, 24196–24205 (2019). [PubMed: 31723047]
8. Lindberg M, Bostrom M, Elliott K & Larsson E Intragenomic variability and extended sequence patterns in the mutational signature of ultraviolet light. *Proc. Natl Acad. Sci. USA* 116, 20411–20417 (2019). [PubMed: 31548379]
9. Brown AJ, Mao P, Smerdon MJ, Wyrick JJ & Roberts SA Nucleosome positions establish an extended mutation signature in melanoma. *PLoS Genet.* 14, e1007823 (2018). [PubMed: 30485262]
10. Mao P, Smerdon MJ, Roberts SA & Wyrick JJ Asymmetric repair of UV damage in nucleosomes imposes a DNA strand polarity on somatic mutations in skin cancer. *Genome Res.* 30, 12–21 (2020). [PubMed: 31871068]
11. Duan M, Selvam K, Wyrick JJ & Mao P Genome-wide role of Rad26 in promoting transcription-coupled nucleotide excision repair in yeast chromatin. *Proc. Natl Acad. Sci. USA* 117, 18608–18616 (2020). [PubMed: 32690696]
12. Mao P et al. Genome-wide maps of alkylation damage, repair, and mutagenesis in yeast reveal mechanisms of mutational heterogeneity. *Genome Res.* 27, 1674–1684 (2017). [PubMed: 28912372]
13. Robinson MD & Oshlack A A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biol.* 11, R25 (2010). [PubMed: 20196867]

14. Langmead B & Salzberg SL Fast gapped-read alignment with Bowtie 2. *Nat. Methods* 9, 357–359 (2012). [PubMed: 22388286]
15. Ramirez F et al. deepTools2: a next generation web server for deep-sequencing data analysis. *Nucleic Acids Res.* 44, W160–W165 (2016). [PubMed: 27079975]
16. Li H et al. The Sequence Alignment/Map format and SAMtools. *Bioinformatics* 25, 2078–2079 (2009). [PubMed: 19505943]
17. Ward CM, To TH & Pederson SM ngsReports: a Bioconductor package for managing FastQC reports and other NGS related log files. *Bioinformatics* 36, 2587–2588 (2020). [PubMed: 31841127]
18. Guo Y, Ye F, Sheng QH, Clark T & Samuels DC Three-stage quality control strategies for DNA re-sequencing data. *Brief. Bioinform.* 15, 879–889 (2014). [PubMed: 24067931]
19. Patel RK & Jain M NGS QC Toolkit: a toolkit for quality control of next generation sequencing data. *PLoS One* 7, e30619 (2012). [PubMed: 22312429]
20. Girardot C, Scholtalbers J, Sauer S, Su SY & Furlong EE Je, a versatile suite to handle multiplexed NGS libraries with unique molecular identifiers. *BMC Bioinformatics* 17, 419 (2016). [PubMed: 27717304]
21. Andrews S A Quality Control Tool for High Throughput Sequence Data. Available at <http://www.bioinformatics.babraham.ac.uk/projects/fastqc/> (2010).
22. Guo Y et al. Multi-perspective quality control of Illumina exome sequencing data using QC3. *Genomics* 103, 323–328 (2014). [PubMed: 24703969]
23. Yu H et al. Non-canonical RNA-DNA differences and other human genomic features are enriched within very short tandem repeats. *PLoS Comput. Biol* 16, e1007968 (2020). [PubMed: 32511223]

Fig. 1 |. Graphic illustration of the CPD-Seq methodology.

For CPD-Seq library preparation, UV-damaged genomic DNA is sonicated to ~400-bp fragments and then treated with enzymes for end repair and dA-tailing, before ligation to the first adaptor DNA. All free 3'-OH groups are blocked with dideoxy-ATP (dd) to prevent ligation of the second adaptor to the free 3' end. The DNA is subsequently digested with T4 endonuclease V (T4 endoV) and AP endonuclease (APE1) to generate a new ligatable 3'-OH group at the CPD damage site. After denaturing to single-stranded DNA, the damaged strand is ligated to the second adaptor DNA. The ligation product is captured by streptavidin beads, and the damaged strand is used as a template to synthesize the opposite strand. The resultant CPD-Seq library is briefly PCR-amplified by using primers complementary to the first and second adaptors. Sequencing reads are aligned to the reference genome. The position of the two nucleotides immediately upstream of the 5' end of each read is identified, and the dinucleotide sequence (shown in dashed ovals) on the opposing strand is recognized as the CPD damage site, as described in Steps 3–9.

Fig. 2 |. The CPDSeqr protocol is composed of 17 steps.

Black rectangular boxes represent individual analysis steps, with a solid border for mandatory steps and dashed border for optional steps. The primary output files of several major steps in the protocol are highlighted with blue boxes. Thick gray arrows indicate primary products flowing out of steps, and black arrows indicate either inflows to steps or steps in succession. §The normalization factor file can be generated in Step 9 or compiled ab initio. *Step 11: generating a genome-wide UV damage distribution map. *Step 12: plotting a dinucleotide pileup figure in a specific genomic region type. *Step 13: comparing UV radiation damage of sample(s) against the reference genome background. *Step 14: comparing UV damage of sample(s) against the reference genome background within a specific region type. *Step 15: comparing UV damage between two regions for one or

multiple samples. *Step 16: comparing genome-wide UV damage between two groups of samples. *Step 17: comparing UV damage between two groups of samples within a specific region type. BAM, binary alignment map; BED, browser extensible data.

Fig. 3 |. CPD-Seq-specific QC diagnostic plots (demonstrated on Gene Expression Omnibus dataset GSE119249).

a, Efficiency and contrast values of one UV-treated sample (left) and one control sample (right). The horizontal lines denote reference ranges of efficiency and contrast based on currently available CPD-Seq datasets. The efficiency and contrast values of the UV-treated sample fall within the expected range, whereas those of the control sample fall below the expected range. **b**, Distribution of frequency of discrete read count statistics (top) and distribution of percentage of dipyrimidine sites satisfying the specified read count criteria (bottom). **c**, Symmetry of efficiency, contrast and read counts of four dipyrimidines between the forward strand and reverse strand. DINUC4 means all four dipyrimidines are combined. Each pair of original indices, for forward and reverse strands, are converted to sum up to one. **d**, Efficiency and contrast values demonstrated for four samples simultaneously. UV-treated samples consist of SRR7770310 and SRR7770311, and control samples consist of SRR7770313 and SRR7770314. **e**, Distribution of frequency of raw read count statistics

(top) and CPM (bottom) statistics. Here, all dipyrimidine types (TT, TC, CC and CT) are combined. **f**, Principal component analysis plot. **g**, Heatmap plot with clustering dendrograms. **h**, Cross-sample pairwise scatter plot with calculated Pearson correlation coefficient values. For generation of **f,g** and **h**, each sample contributes a vector of read count numbers summarized for all possible chromosome-dinucleotide combinations (for humans, there are 368 such combinations), and these vectors are collated into a matrix upon which various analytical graphics are rendered.

Fig. 4 |. Graphic output of Step 11 to show genome-wide CPD damage site distribution. One no-UV control sample (top) and one UV-treated sample (bottom) were used in the demonstration. In this case, site count was used, meaning that the grayness signifies the density of distinct CPD-occurring sites within chromosome bins (default bin size is 100,000 bp). The intensity (numerical representation of raw or normalized dinucleotide counts) of each bin was subjected to local GC content normalization. More CPD damage sites were found in the UV-treated sample than in the control sample.

Fig. 5 |. Example positional CPD damage aggregate figure for a nucleosome region (output of Step 12).

In this example, we supplied over one million nucleosome regions and used naked DNA CPD-Seq data for normalization. Site count and normalized data were used for this figure. The positional aggregation based on nucleosome regions shows a clear oscillation pattern of CPD damage peak in the UV light-treated sample (right), but not in the control (left).