Supplementary Information

Engineered pegRNAs that improve prime editing efficiency

James W. Nelson[1,2,3]*, Peyton B. Randolph[1,2,3]*, Simon P. Shen[1,2,3], Kelcee A. Everette[1,2,3], Peter J. Chen[1,2,3], Andrew V. Anzalone[1,2,3], Meirui An[1,2,3], Gregory A. Newby[1,2,3], Jon C. Chen[1,2,3], Alvin Hsu[1,2,3], David R. Liu[1,2,3,‡]

[1]*Merkin institute of Transformative Technologies in Healthcare, Broad Institute of Harvard and MIT, Cambridge, MA, USA*
[2]*Howard Hughes Medical Institute, Harvard University, Cambridge, MA, USA*
[3]*Department of Chemistry and Chemical Biology, Harvard University, Cambridge, MA, USA*

* These authors contributed equally.
‡ Correspondence should be addressed to David R. Liu: drliu@fas.harvard.edu, @davidrliu

Supplementary Discussion

Supplementary Figures

1. Sequence and secondary structure of RNA structural motifs examined in this study
2. PE3-mediated edit:indel ratio for pegRNAs and epegRNAs shown in Fig. 2.
3. Linker-length dependence of epegRNA activity.
4. Improvement in PE3-mediated editing efficiency at various genomic loci from to the addition of 3′ RNA structural motifs to pegRNAs
5. PE3-mediated edit:indel ratio for pegRNAs and epegRNAs shown in Supplementary Fig. 4
6. Engineered pegRNAs demonstrate no increase in detected off-target activity compared to canonical pegRNAs
7. Site-dependent expression differences of pegRNAs and epegRNAs
8. High-throughput sequencing analysis of PE2-mediated genomic reverse transcriptase products.
9. PE3-mediated editing efficiency of pegRNAs containing other RNA structural motifs
10. PE3-mediated editing efficiency of epegRNAs containing evopreQ$_1$ or mpknot variants
11. Effect of the (F+E) scaffold on PE2-editing efficiency with lentivirally transduced epegRNAs
12. Effect of (F+E) scaffold modifications on PE3-editing efficiency with epegRNAs
13. Computational prediction of effective linker sequences between the PBS and structural motif of epegRNAs
14. Improvements in editing efficiency upon electroporation of chemically synthesized epegRNAs
15. PE2-mediated efficiency of installation of FLAG tags at the indicated genomic sites
16. Uncropped agarose gel in Figure 3
17. Uncropped northern blots in Supplementary Figure 7

Supplementary Tables

1. Sequences of pegRNAs, epegRNAs and sgRNAs used in this study
2. Sequences of RNA structural motifs examined in this study
3. Sequences of primers used for genomic DNA amplification

4. Sequences of amplicons analyzed with high-throughput sequencing
5. Sequences of primers, amplicons, and probes used in RTqPCR and northern blot experiments
6. Reference SNP numbers of pathogenic mutations installed with pegRNAs or epegRNAs

Supplementary Notes

1. Guidelines for epegRNA cloning via Golden Gate DNA assembly
2. pegRNA Linker Identification Tool (pegLIT) code
3. Python script for quantifying prime editing intermediates

Supplementary Discussion

pegLIT strategy for identifying optimal linker sequences

pegLIT uses simulated annealing to sample the analyzed linker space efficiently[1]. Linkers that are adenosine- or cytosine-rich are preferred by pegLIT since these nucleotides have been reported to function better as flexible RNA linkers[2]. Additionally, pegLIT filters out linkers that contain runs of four or more uridines, since such sequences could cause premature transcriptional termination[3].

The pegLIT tool then analyzes linkers that pass these requirements using ViennaRNA[4] to predict potential interactions between the linker sequence and the pegRNA spacer, PBS, template, or scaffold. The base pair probabilities of these predicted interactions are used to generate subscores for each region of the pegRNA, each of which represents the degree to which the linker is predicted to avoid interaction with the associated region. For example, a subscore of 0.95 for the PBS essentially indicates that, on average, the predicted probability of a pegRNA folded state lacking base pairing between any linker nucleotide and the PBS is 95%.

We sought to validate the use of pegLIT for linker design and examine which interactions identified by pegLIT were most detrimental to editing efficiency. We generated 30 linker sequences (10 recommended by pegLIT, 10 interacting with the spacer, and 10 interacting with the PBS) to test with evopreQ$_1$ epegRNAs templating either a C•G-to-A•T transversion at *RNF2* or a 15-bp deletion at *DNMT1*. The average spacer and PBS subscores were 0.94 and 0.97 for the optimal sequences, 0.66 and .95 for the spacer sequences, and 0.86 and 0.21 for the PBS sequences. Relative to the recommended designs, use of the PBS-interacting linkers was associated with 1.3- and 1.1-fold lower editing efficiency at *RNF2* and *DNMT1* respectively (**Supplementary Fig. 13**), whereas the spacer-interacting linkers had a negligible effect on editing efficiency. This difference may be because the closer proximity of the linker to the PBS compared to the spacer may give linker:PBS interactions an entropic advantage compared to linker:spacer pairing.

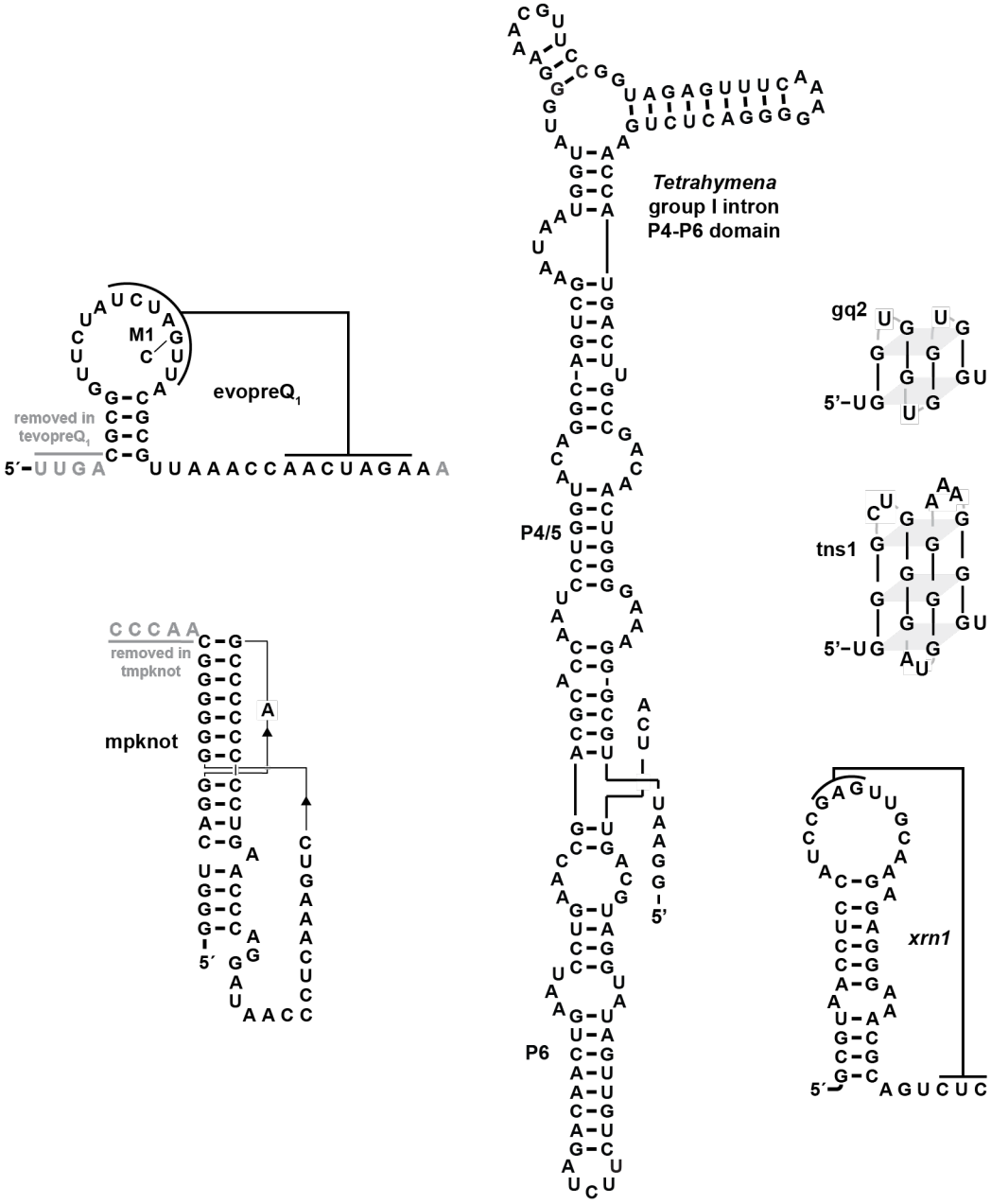
epegRNAs delivered via plasmid transfection with optimized guide RNA scaffolds in HEK293T cells

To mimic lower expression conditions, we transfected HEK293T cells with 20 ng of PE2 plasmid and 4 ng of pegRNA or epegRNA plasmid when assessing the applicability of “flip and extension” (F+E) sgRNA scaffold variants for PE. We compared the editing efficiency of epegRNAs targeted to *PRNP*, *HEK3*, *RUNX1*, and *EMX1* that contained the canonical sgRNA scaffold, an (F+E) scaffold[5], or one of six (F+E) scaffolds bearing mutations previously shown to increase Cas9-nuclease activity[6]. We found that these alternative scaffolds overall either maintained or improved PE efficiency relative to the standard scaffold, with cr772 exhibiting the best improvement (**Supplementary Fig. 12**). While efficiency improvements were less consistent under these conditions compared to lentiviral transduction (**Supplementary Fig. 11**), this may stem from differences in expression. EpegRNA

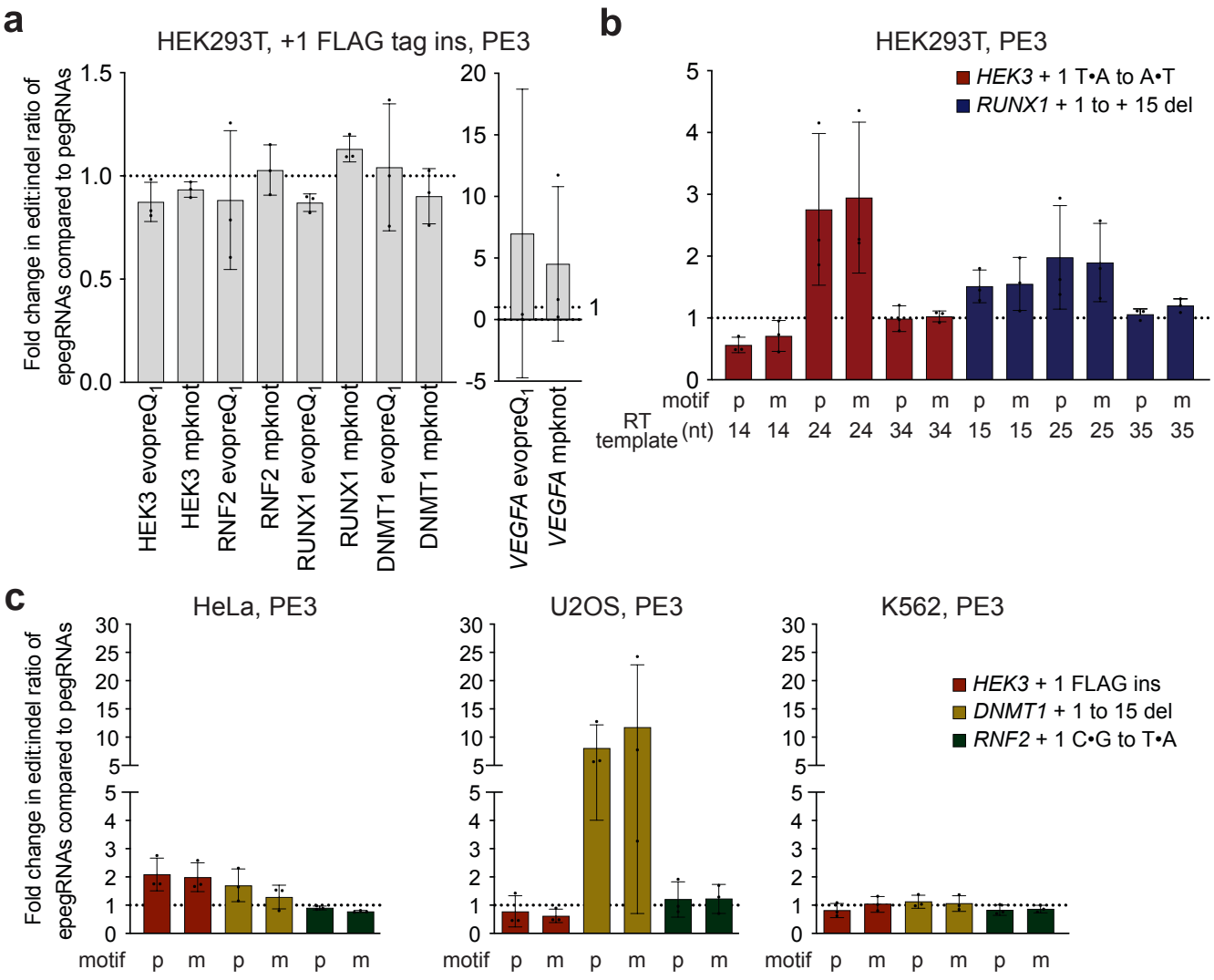
expression is likely several-fold higher following plasmid transfection than that following single-copy lentiviral transduction, which may partially obfuscate the benefits of more efficient transcription and Cas9 binding affinity. We recommend testing cr772 or the original (F+E) scaffold to further improve PE efficiency with epegRNAs, especially for applications with lower expression than plasmid transfection.

Installation of FLAG tags using unoptimized epegRNAs

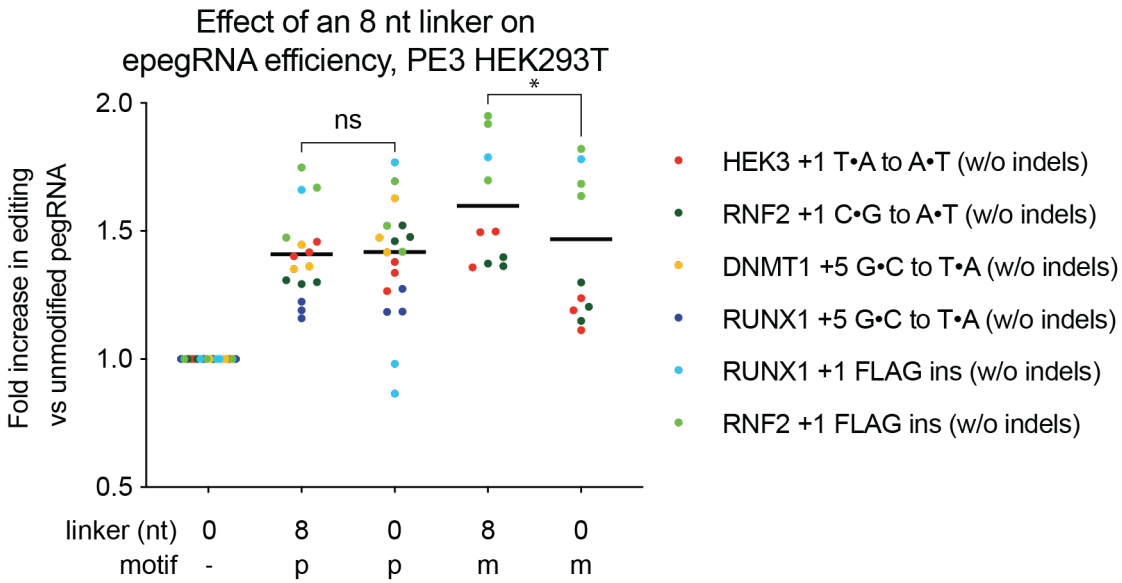
We compared epegRNAs and pegRNAs for the installation of more challenging edits, such as insertion of the 24-bp FLAG epitope tag (**Fig. 2a**). We assessed the ability of unoptimized pegRNAs and tevopreQ$_1$ epegRNAs containing one of two loci-specific pegLIT-designed 8-nt linkers to template the installation of a FLAG epitope tag at 15 loci in HEK293T cells using PE2 (**Supplementary Fig. 15**). The unoptimized epegRNAs and pegRNAs were designed with a 13-nt PBS and an RT template containing 25 nt of homology downstream of the inserted FLAG epitope tag, except when the 3′ extension would begin with cytosine[7], in which case it was extended to the nearest non-C nucleotide. The use of epegRNAs enabled FLAG tags to be installed with PE2 at ≥10% efficiency with no PBS and RT template optimization at 5 of the 15 sites, while ≥10% efficiency was not observed with any pegRNAs (**Supplementary Fig. 15**). These observations further demonstrate that epegRNAs can enhance prime editing performance for a variety of edits at many different endogenous human genomic loci.



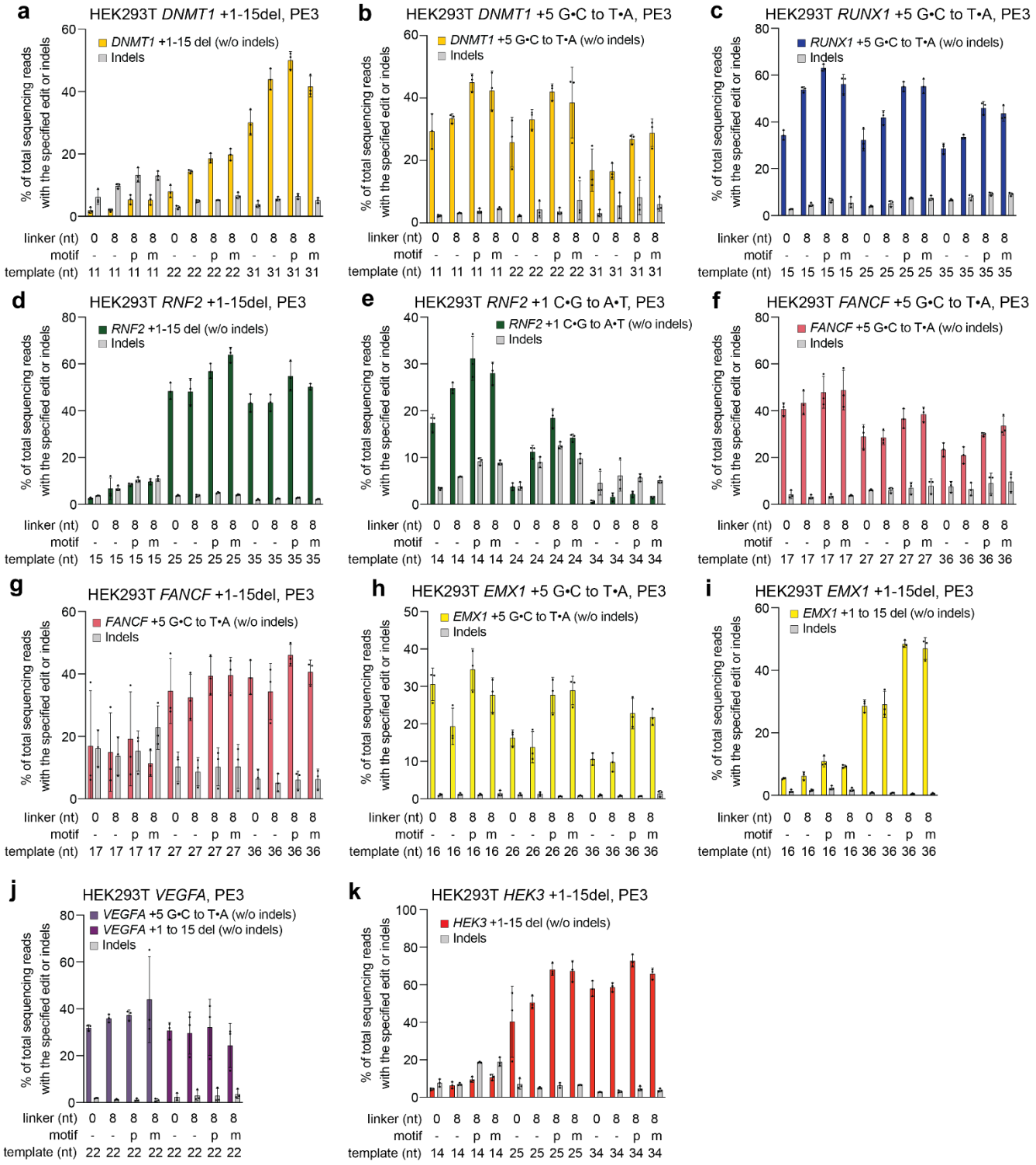
Supplementary Figure 1. Sequence and secondary structure of RNA structural motifs examined in this study. Structures are based on predictions from previously published structural or bioinformatic analyses[8-11]. Only two G-quadruplexes of the 11 tested are shown for brevity. Sequences of all motifs are provided in **Supplementary Table 2**.



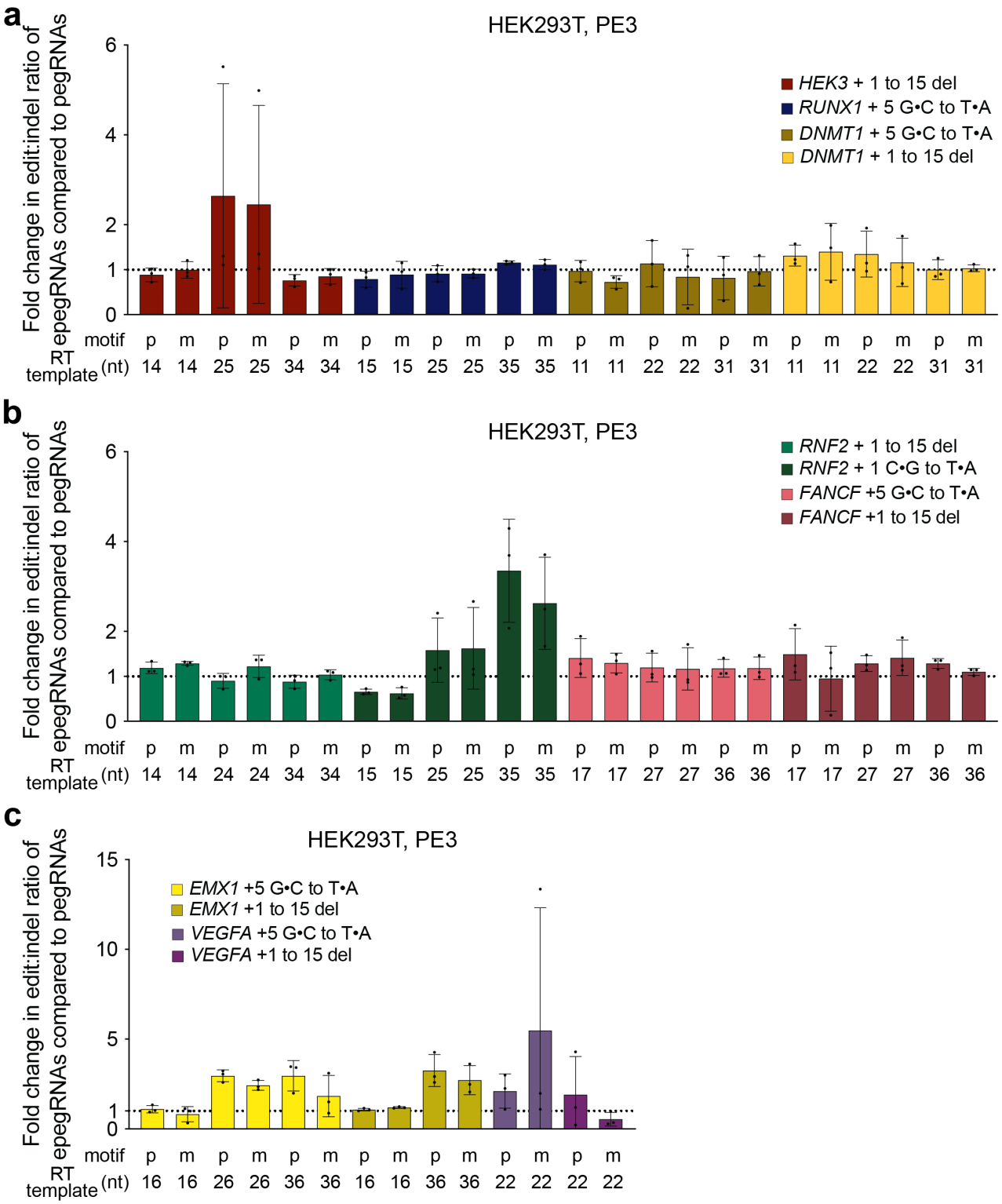
Supplementary Figure 2. PE3-mediated edit:indel ratio for pegRNAs and epegRNAs shown in Fig. 2. Fold-change in the observed prime editing edit:indel ratio for installation of a FLAG epitope tag (**a**) or the indicated transversion or deletion (**b**) in HEK293T cells, or the indicated edit in HeLa, U2OS, or K562 cells (**c**) of epegRNAs bearing either evopreQ$_1$ (p) or mpknot (m) compared to unmodified pegRNA (dashed line). Values were calculated from the data presented in Fig. 2a, 2c and 2d respectively. Data and error bars reflect the mean and standard deviation of three independent biological replicates.



Supplementary Figure 3. Linker-length dependence of epegRNA activity. Effect of removing the 8-nt linkers used in **Fig. 2** and **Supplementary Fig. 4 a-k** on PE3 editing efficiency. Either evopreQ$_1$ (p) or mpknot (m) was appended to the PBS via either no linker or an 8-nt linker. The distance from the Cas9 nick site to the installed mutation in nucleotides is as indicated in the legend. Dots indicate the average of three biological replicates. Bars indicate the grand median. Significance was calculated via a two-tailed paired Student's t test (p=0.022).



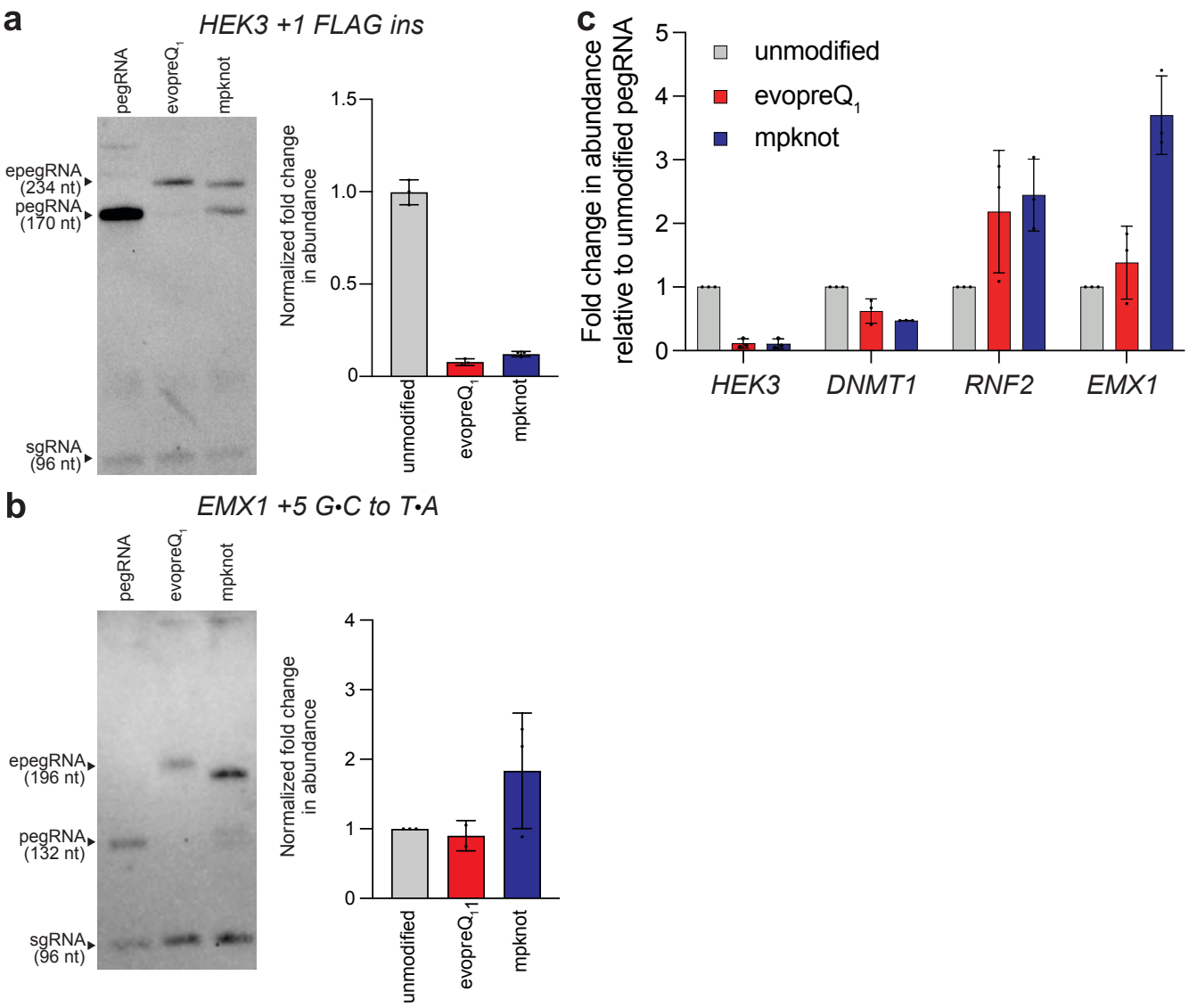
Supplementary Figure 4. Improvement in PE3-mediated editing efficiency at various genomic loci from to the addition of 3′ RNA structural motifs to pegRNAs. (**a-k**) PE3-mediated installation of the indicated edit at (**a, b**) *DNMT1*, (**c**) *RUNX1*, (**d, e**) *RNF2*, (**f, g**) *FANCF*, (**h, i**) *EMX1*, (**j**) *VEGFA,* or (**k**) *HEK3*. Either an 8-nt linker alone or the linker in conjunction with evopreQ$_1$ (p) or mpknot (m) was appended to pegRNAs of increasing template lengths and compared to canonical pegRNAs. The distance from the Cas9 nick site to the installed mutation in nucleotides is indicated. Data and error bars reflect the mean and standard deviation of three independent biological replicates.



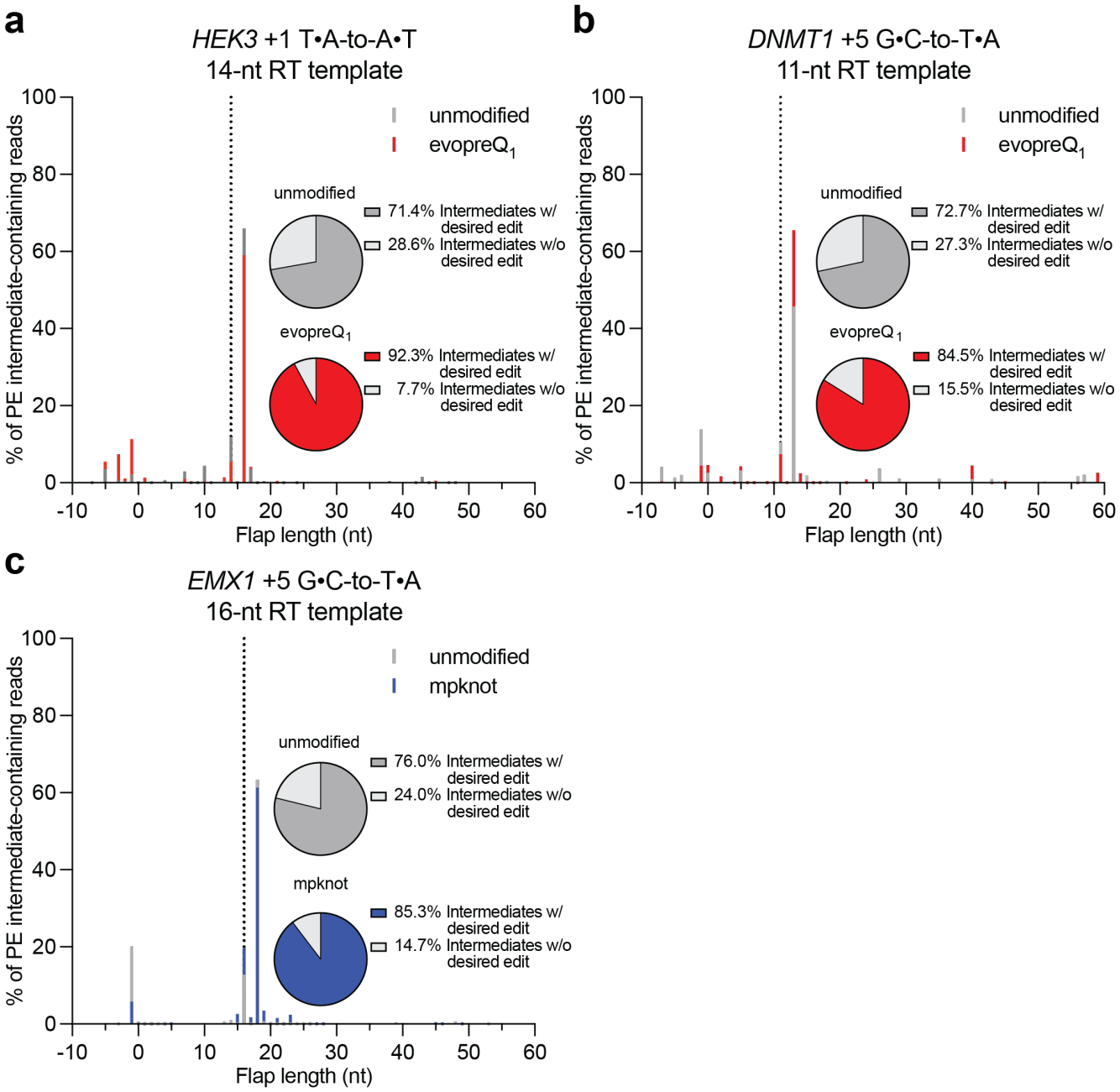
Supplementary Figure 5. PE3-mediated edit:indel ratio for pegRNAs and epegRNAs shown in Supplementary Fig. 4. Fold-change in the observed edit:indel ratio for the indicated transversion or deletion at *HEK3*, *RUNX1*, or *DNMT1* (**a**), *RNF2* or *FANCF* (**b**), or *EMX1* or *VEGFA* (**c**) of epegRNAs bearing either evopreQ$_1$ (p) or mpknot (m) compared to unmodified pegRNA (dashed line). Values were calculated from the data presented in **Supplementary Fig. 2a-k**. Data and error bars reflect the mean and standard deviation of three independent biological replicates.

	HEK3 (PE3)					*EMX1* (PE3)					*FANCF* (PE3)				
pegRNA / site	---	pt mtn, -	pt mtn, m	del, -	del, p	---	pt mtn, -	pt mtn, p	del, -	del, m	---	pt mtn, -	pt mtn, m	del, -	del, m
On-target		21.5 (4.0)	44.1 (11.7)	57.9 (2.8)	72.7 (4.7)		30.5 (1.1)	34.5 (1.1)	28.6 (0.8)	47 (0.5)		40.7 (4.2)	48.8 (3.7)	38.9 (6.4)	40.6 (6.3)
Off-target 1		<0.1 (<0.1)	<0.1 (<0.1)	<0.1 (<0.1)	<0.1 (<0.1)		<0.1 (<0.1)	<0.1 (<0.1)	<0.1 (<0.1)	<0.1 (<0.1)		<0.1 (<0.1)	<0.1 (<0.1)	<0.1 (<0.1)	<0.1 (<0.1)
Off-target 2		<0.1 (<0.1)	<0.1 (<0.1)	<0.1 (<0.1)	<0.1 (<0.1)		<0.1 (<0.1)	<0.1 (<0.1)	<0.1 (<0.1)	<0.1 (<0.1)		<0.1 (<0.1)	<0.1 (<0.1)	<0.1 (<0.1)	<0.1 (<0.1)
Off-target 3		<0.1 (<0.1)	<0.1 (<0.1)	<0.1 (<0.1)	<0.1 (<0.1)		<0.1 (<0.1)	<0.1 (<0.1)	<0.1 (<0.1)	<0.1 (<0.1)		<0.1 (<0.1)	<0.1 (<0.1)	<0.1 (<0.1)	<0.1 (<0.1)
Off-target 4		<0.1 (<0.1)	<0.1 (0.1)	<0.1 (0.1)	<0.1 (0.1)		<0.1 (0.1)	<0.1 (0.1)	<0.1 (<0.1)	<0.1 (0.1)		<0.1 (<0.1)	<0.1 (<0.1)	<0.1 (<0.1)	<0.1 (<0.1)

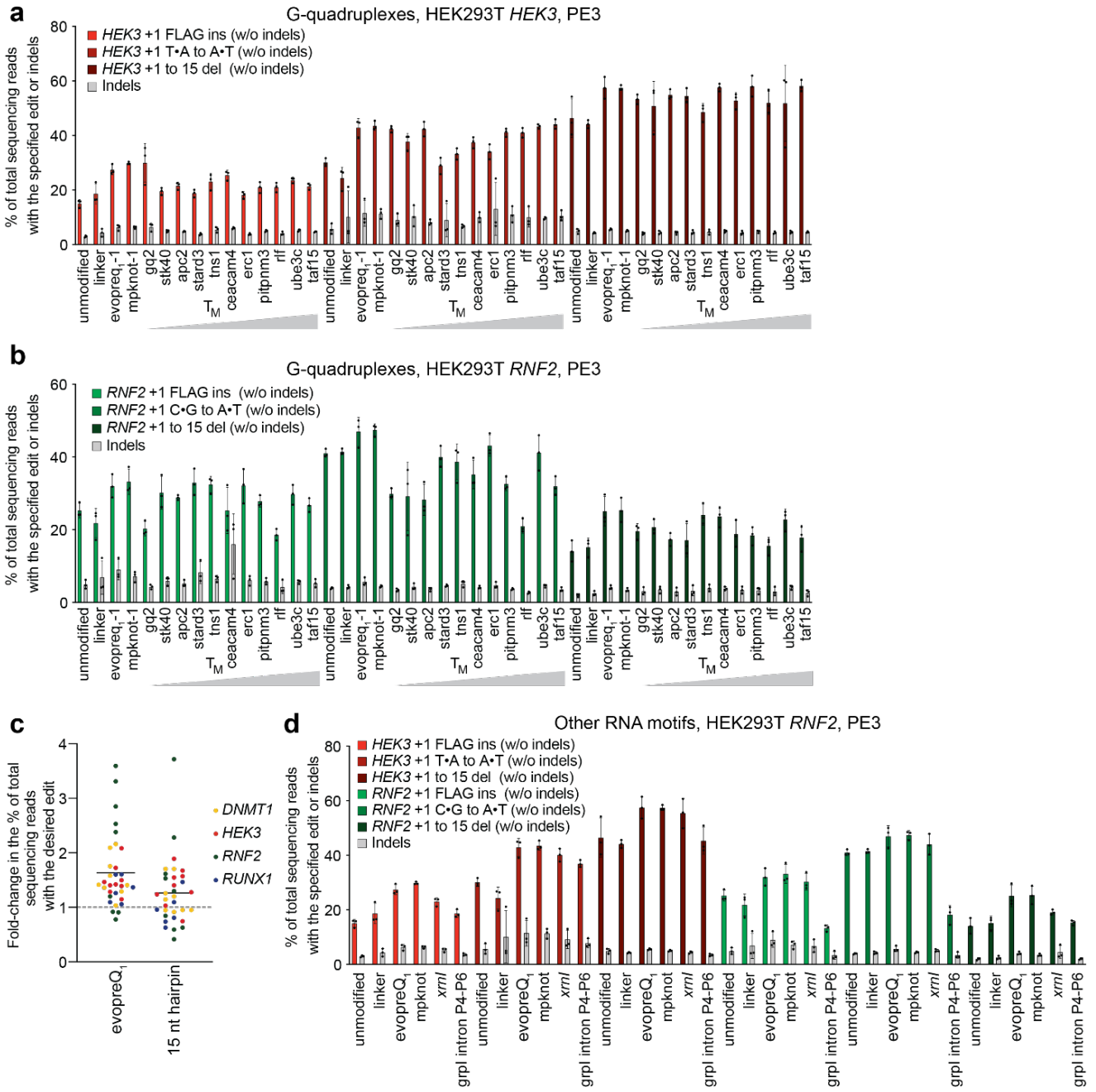
Supplementary Figure 6. Engineered pegRNAs demonstrate no increase in detected off-target activity compared to canonical pegRNAs. On- and off-target PE3 editing of pegRNAs and epegRNAs targeted to HEK3, EMX1, or FANCF and templating either a nucleotide transversion (T•A to A•T at *HEK3* or G•C to T•A at *EMX1* and *FANCF*; pt mtn) or a 15-nt deletion (del). –, canonical pegRNA; m, epegRNA containing mpknot; p, epegRNA containing evopreQ$_1$. Indel frequencies are shown in parentheses. For *EMX1* off-target 1, indels were obtained by subtracting the percentage of sequencing reads containing indels in cells transfected with a non-targeting pegRNA. Off-target loci are listed in **Supplementary Table 4**. Data are the average of three biological replicates.



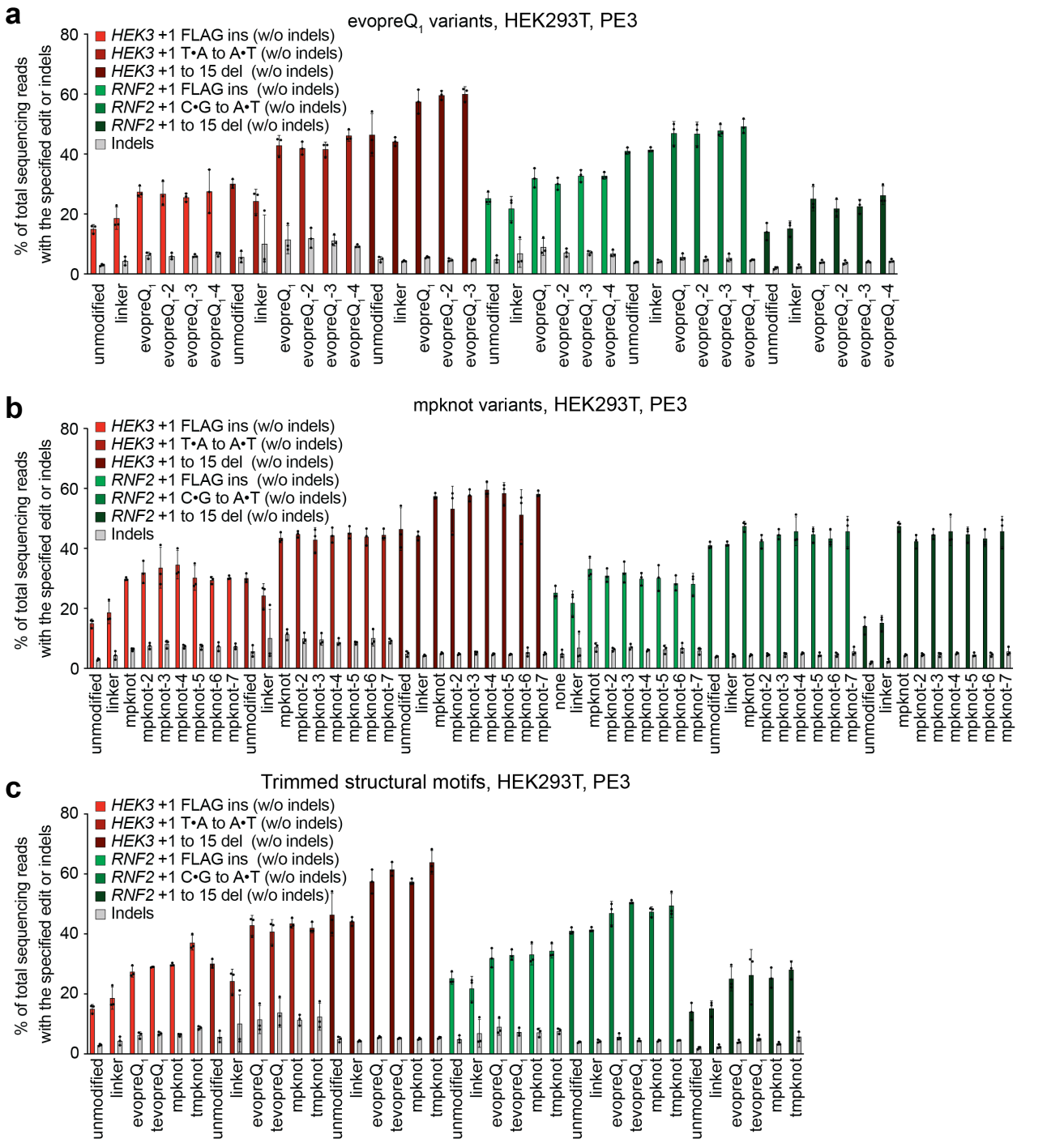
Supplementary Figure 7. Site-dependent expression differences of pegRNAs and epegRNAs. Northern blot of HEK293T lysates containing pegRNAs or epegRNAs targeted to (**a**) *HEK3* or (**b**) *EMX1* after hybridization with a DIG-labeled RNA probe complementary to the sgRNA scaffold. PAGE gels shown are representative of multiple independent biological replicates. The normalized fold change in abundance relative to unmodified pegRNA as determined by densitometry is shown (right). Abundance was calculated by including both full-length pegRNA and epegRNA for samples in which full length pegRNA is present. Band identity was confirmed using untreated *in vitro* transcribed pegRNAs and epegRNAs as standards, DIG-labeled ssRNA ladder, and purified RNA from HEK293T cells transfected with sgRNA as markers. (**c**) Abundance of epegRNA and canonical pegRNA targeted to *HEK3*, *DNMT1*, *RNF2* or *EMX1* in HEK293T cells by RT-qPCR amplification and quantification of the sgRNA scaffold. Primers for qPCR amplification can be found in **Supplementary Table 5**. Data and error bars reflect the mean and standard deviation of three independent biological replicates.



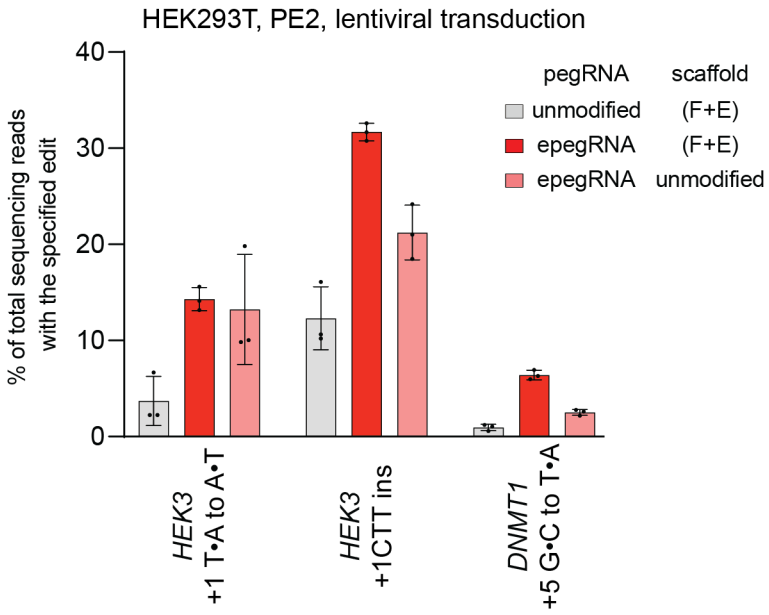
Supplemental Figure 8. High-throughput sequencing analysis of PE2-mediated genomic reverse transcriptase products. Comparison of prime-editing intermediates generated by PE2 with either pegRNAs or epegRNAs at (**a**) *HEK3*, (**b**) *DNMT1*, or (**c**) *EMX1* as indicated. Dotted lines indicate the full-length reverse transcriptase product templated by the pegRNA or epegRNA tested at the indicated locus. X axis is relative to the position of the PE2-induced nick with the first base 3′ downstream represented as position +1. Histograms and pie charts are generated from the average of three independent biological replicates.



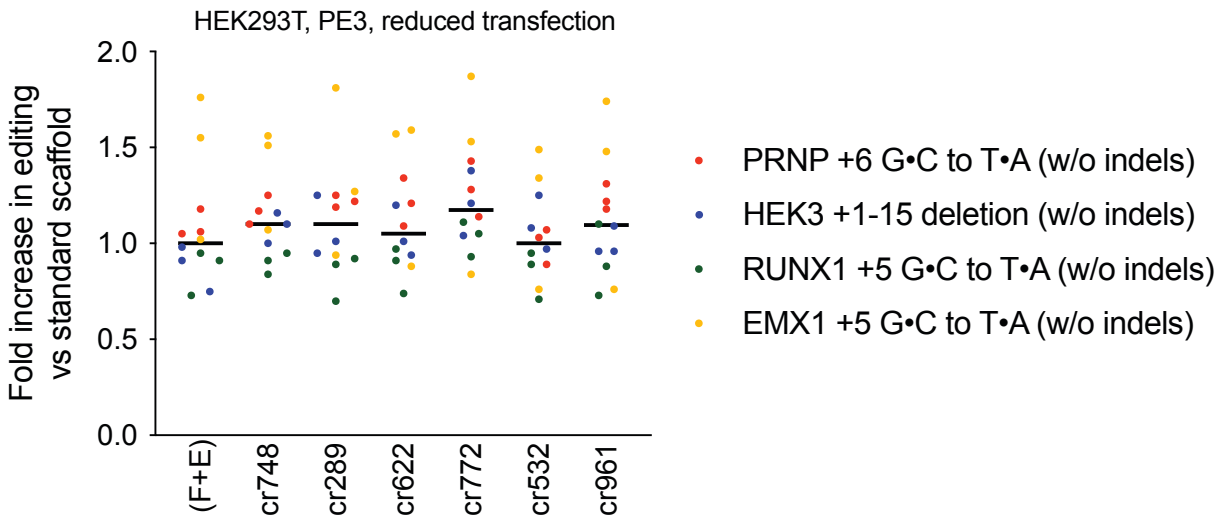
Supplementary Figure 9. PE3-mediated editing efficiency of pegRNAs containing other RNA structural motifs. Comparison of PE3-mediated editing efficiencies for the installation of the FLAG epitope tag, a 15-nt deletion, or a point mutation at *HEK3* (**a**) and *RNF2* (**b**) with epegRNAs to which various G-quadruplexes have been appended via an 8-nt linker. G-quadruplexes are ordered based on melting temperature, ranging from 60 to >90 °C, as previously determined[12]. (**c**) PE3-mediated efficiency of installation of point mutations at the indicated genomic loci using pegRNAs containing the evopreQ$_1$ motif or a 15-bp (34-nt) hairpin. (**d**) Addition of either a pseudoknot known to inhibit the 5′ exonuclease XrnI (*xrnI*)[10] or a large tertiary RNA structure (the P4-P6 domain of the group I intron from *Tetrahymena thermophila*[11]) to the 3′ terminus of the pegRNA via an 8-nt linker does not yield more efficient editing than addition of either evopreQ$_1$ or mpnkot by the same linker. The distance from the Cas9 nick site to the installed mutation is indicated. Data and error bars reflect the mean and standard deviation of three independent biological replicates.



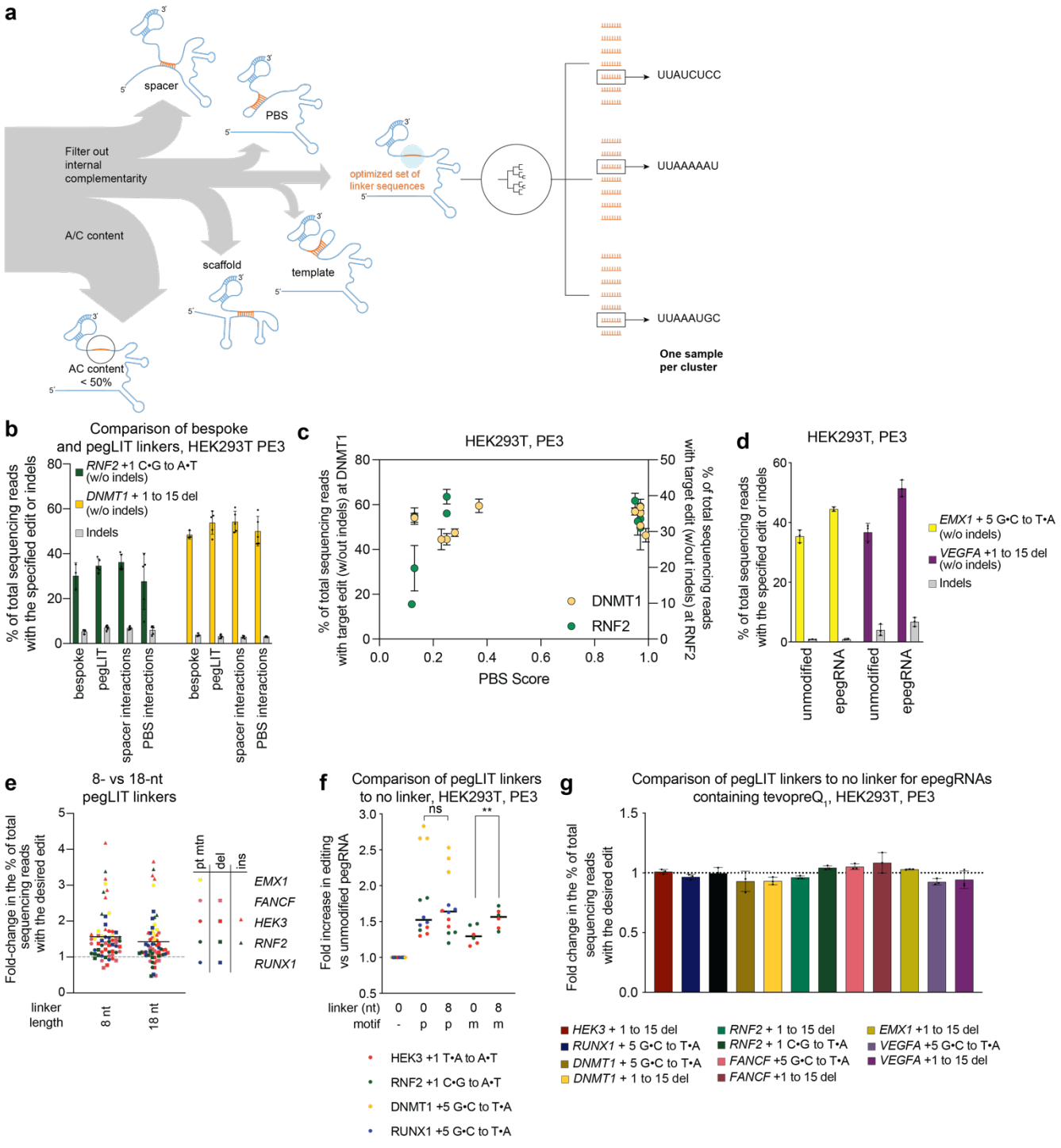
Supplementary Figure 10. PE3-mediated editing efficiency of epegRNAs containing evopreQ$_1$ or mpknot variants. Comparison of PE3-mediated editing efficiencies for the installation of the FLAG epitope tag, a 15-nt deletion, or a point mutation at *HEK3* and *RNF2* with epegRNAs containing various RNA motifs, where the distance between the Cas9 nick and the edit is indicated by +1. PE3 editing efficiencies of additional evolved prequeosine$_1$-1 riboswitch aptamer variants (**a**) or modifications to mpknot (**b**) compared to evopreQ$_1$ or mpknot. (**c**) PE3 editing efficiencies of epegRNAs trimmed to remove nucleotides 5′ and 3′ of evopreQ$_1$ (tevopreQ$_1$) and mpknot (tmpknot) compared to parent epegRNAs. Data and error bars reflect the mean and standard deviation of three independent biological replicates.



Supplementary Figure 11. Effect of the (F+E) scaffold on PE2-editing efficiency with lentivirally transduced epegRNAs. PE2-editing efficiency of lentivirally-transduced prime editor and pegRNA or epegRNA that contain tevopreQ$_1$ and either the canonical or (F+E) sgRNA scaffold and that template the indicated edit at *HEK3* or *DNMT1* in HEK293T cells. Data and error bars reflect the mean and standard deviation of three independent biological replicates.

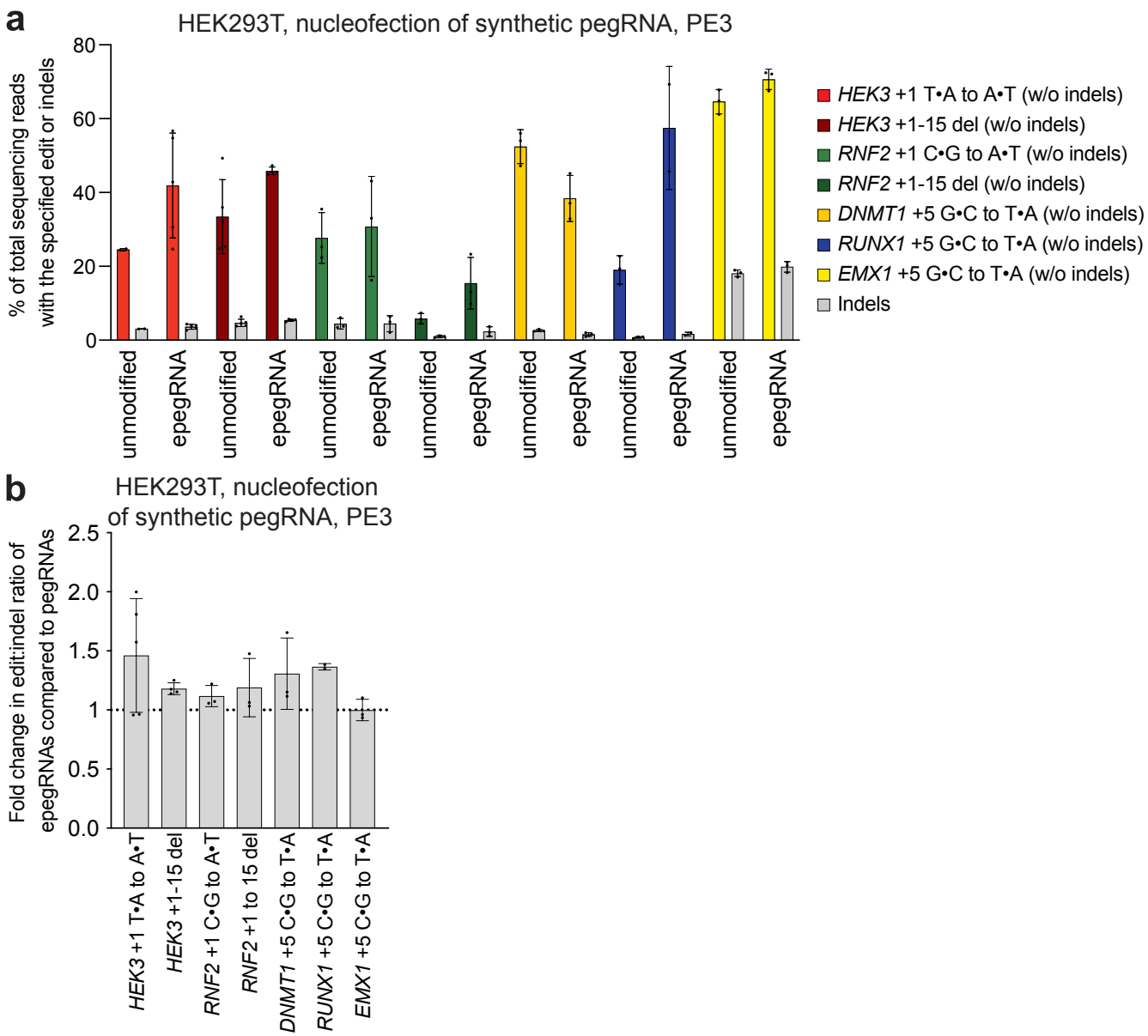


Supplementary Figure 12. Effect of (F+E) scaffold modifications on prime editing efficiency with epegRNAs. Comparison of PE3-mediated editing efficiencies of epegRNAs with the indicated scaffold to epegRNAs with the standard SpCas9 sgRNA scaffold. One-tenth the normal amount of plasmids encoding PE2 and pegRNA or epegRNA was transfected in HEK293T cells in these experiments. Edits templated were either a transversion at *PRNP, RUNX1, or EMX1* or a 15-nt deletion at *HEK3*. Modified scaffold sequences all contain the "flip and extension" (F+E) modification. Scaffolds designated cr also contain mutations to the (F+E) scaffold previously identified as potentially improving Cas9 nuclease activity at some sites[6]. Sequences of all scaffolds can be found in **Supplementary Table 1**. Lines indicate the grand medians.

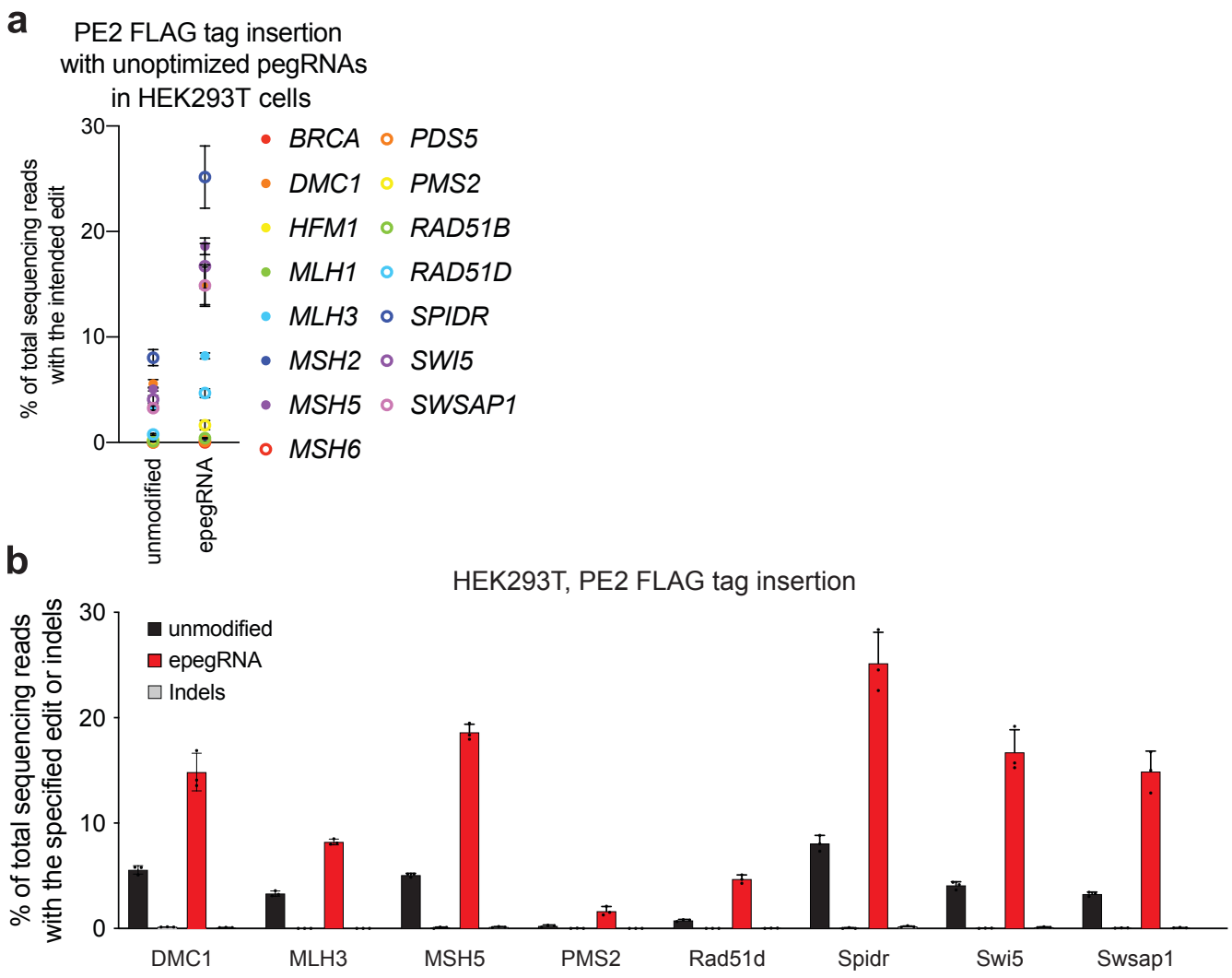


Supplementary Figure 13. Computational prediction of effective linker sequences between the PBS and structural motif of epegRNAs. (**a**) Schematic illustrating the workflow of pegLIT, a computational script to select appropriate linker sequences for epegRNAs. Potential linker sequences are filtered by sequence identity and propensity for base pairing to other regions of the epegRNA. Sequences passing the filter are then optionally clustered based on identity and individual sequences are selected from different clusters to promote diversity in the final output. (**b** and **c**) epegRNAs containing evopreQ$_1$ connected via linker sequences recommended by pegLIT lead to modestly

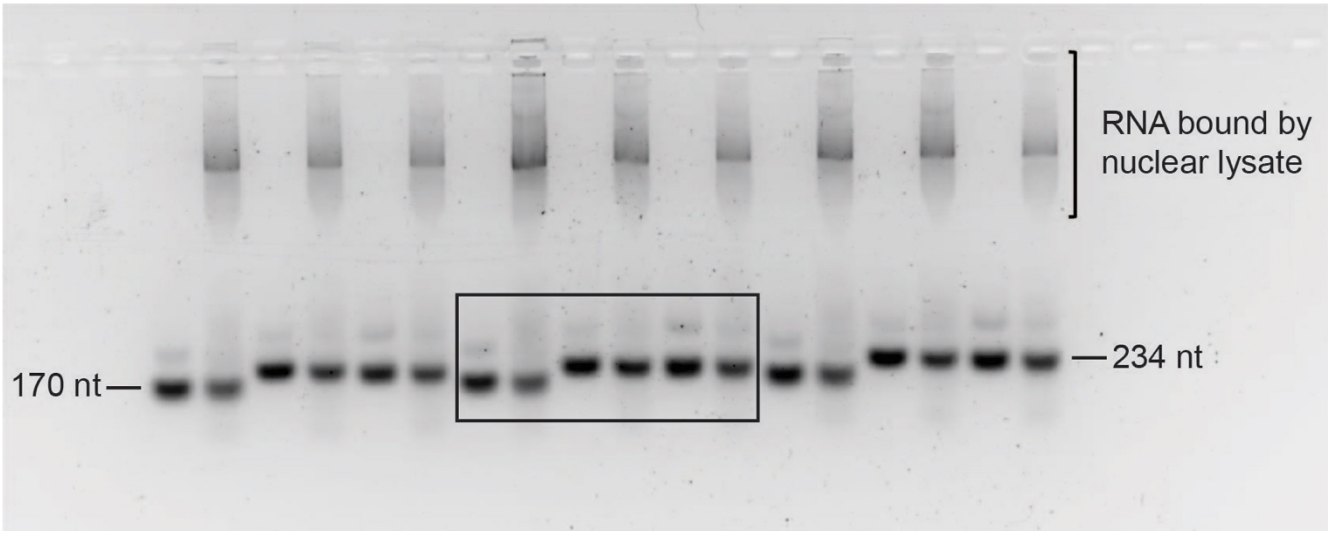
improved PE editing efficiency compared to epegRNAs containing evopreQ$_1$ connected via a human-designed linker or linkers that were predicted by pegLIT to interact with the PBS and (**d**) rescued activity at those sites at which epegRNAs did not initially yield improvements (**Supplementary Fig. 4**). (**e**) Comparison of PE3-mediated editing efficiencies of epegRNAs with evopreQ$_1$ and either 8- or 18-nt long linkers suggests no significant improvement is achieved by increasing linker length. (**f**) Comparison of PE3-mediated editing efficiencies of epegRNAs with either evopreQ$_1$ (p) or mpknot (m) and either an 8-nt pegLIT linker (8) or no linker (0). Significance was calculated using a two-tailed paired student's t test ($p=0.0061$). (**g**) Fold increase in PE3-mediated editing efficiencies of epegRNAs with tevopreQ$_1$ containing an 8-nt pegLIT linker compared to no linker. Data are presented as the mean with error bars indicating either (for b) the standard deviation of the mean for five pegLIT-designed linkers, each in triplicate, or the standard deviation of three replicates for manually designed linker sequences, (for c ,d, and g) the standard deviation of three biological replicates, or (for e and f) the grand median of the average fold-change in editing efficiency for each indicated site and edit.



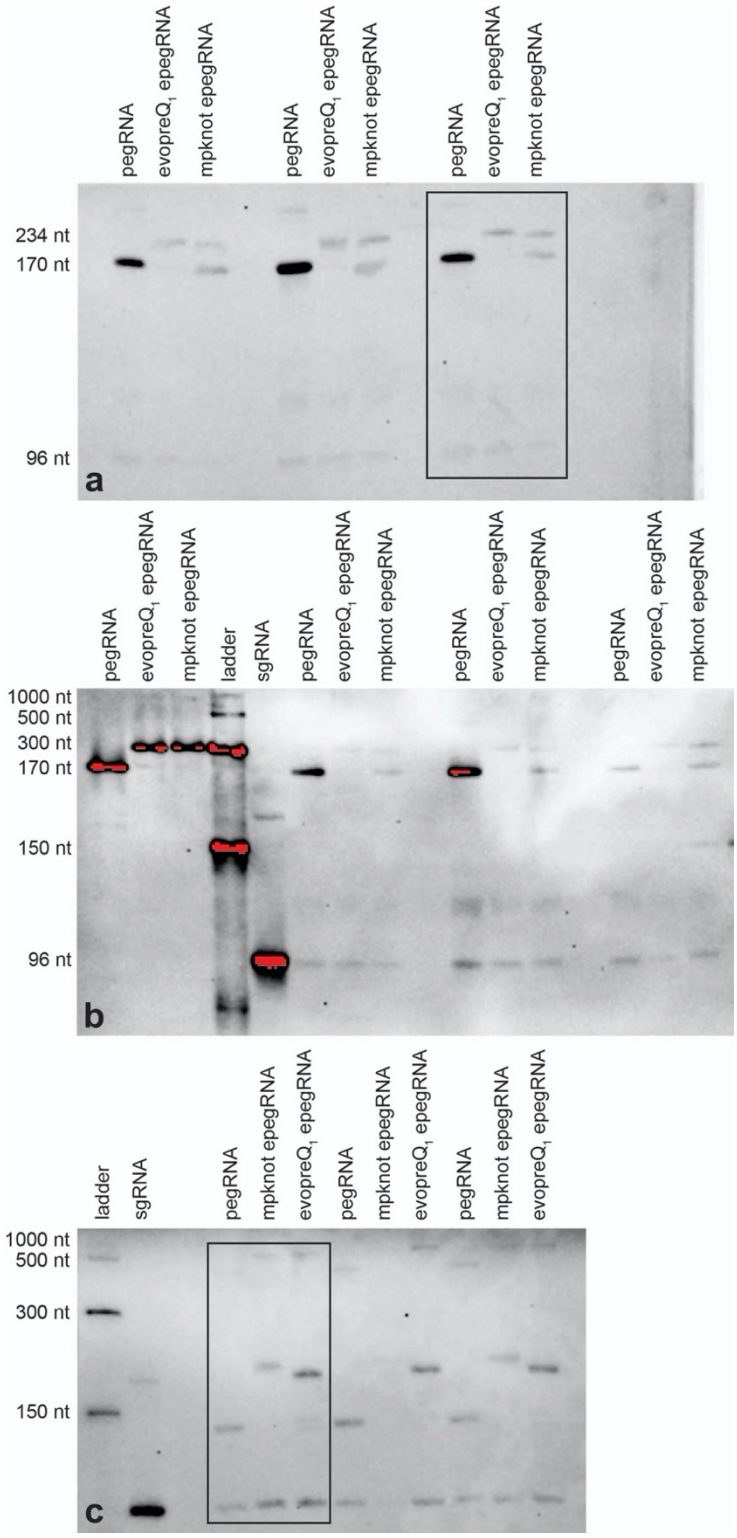
Supplementary Figure 14. Improvements in editing efficiency upon electroporation of chemically synthesized epegRNAs. (**a**) Efficiency of PE3-mediated installation of the indicated edit upon nucleofection of mRNA which encodes PE2, a chemically synthesized nicking sgRNA, and either chemically synthesized pegRNA or epegRNA containing evopreQ$_1$ via an 8-nt linker. (**b**) Observed fold-change in the edit:indel ratio for epegRNAs compared to pegRNAs for the indicated site and edit, based on data in (a). Data and error bars reflect the mean and standard deviation of two or more independent biological replicates.



Supplementary Figure 15. PE2-mediated efficiency of installation of FLAG tags at the indicated genomic sites. (**a**) PE2-mediated editing efficiency of FLAG epitope tag insertion at 15 genomic loci in HEK293T cells using unoptimized epegRNAs compared to unoptimized canonical pegRNAs. (**b**) Data from (a) shown in bar chart form. Sites with sub 1% editing efficiency with both pegRNAs and epegRNAs are not shown but are listed in **Supplemental Table 1**. Data and error bars reflect the mean and standard deviation of three independent biological replicates.



Supplementary Figure 16. Uncropped agarose gel in Figure 3. Uncropped image of the agarose gel used for Figure 3a with the excerpted region outlined in black. Untreated *in vitro* transcribed pegRNAs or epegRNAs were used as molecular weight standards.


pegRNA
evopreQ1 epegRNA
mpknot epegRNA
234 nt
170 nt
96 nt
a
ladder
sgRNA
1000 nt
500 nt
300 nt
150 nt
b
c

Supplementary Figure 17. Uncropped northern blots in Supplementary Figure 7. (**a**) Uncropped image of the northern blot used for Supplementary Figure 7a with the excerpted region outlined in black. Species lengths were confirmed using untreated *in vitro* transcribed pegRNA and epegRNA as molecular weight standards on a separate blot with a molecular weight ladder (shown in b). (**b**) Uncropped image of the northern blot used to confirm the band identities and molecular weights of standards in (a). (**c**) Uncropped image of the northern blot used for Supplementary Figure 7b with the excerpted region outlined in black.

Supplementary Tables are provided in a separate Microsoft Excel file.

Supplementary Table 1. Sequences of pegRNAs and sgRNAs used in this study. This table lists all (e)pegRNAs and sgRNA used in the study. For each RNA, the spacer, template, PBS, linker, and motif added, if any, are listed separately. RNAs are organized by figure.

Supplementary Table 2. Sequences of RNA structural motifs examined in this study. This table contains a separate list of RNA structural motifs which were appended to epegRNAs. We recommend epegRNAs that contain tevopreQ$_1$ (highlighted).

Supplementary Table 3. Sequences of primers used for genomic DNA amplification. This table lists all primers used for genomic DNA amplification prior to high-throughput sequencing. For most forward primers, offset forward primers with either 4 or 5 Ns were used, as indicated.

Supplementary Table 4. Sequences of amplicons analyzed with high-throughput sequencing. This table lists all genomic regions analyzed by high-throughput sequencing, including known Cas9 off-target sites for *HEK3, EMX1,* and *FANCF.*

Supplementary Table 5. Sequences of primers used in RTqPCR experiments. This table lists all primers used for RTqPCR analysis of pegRNA expression levels.

Supplementary Table 6. Reference SNP numbers of pathogenic mutations installed with pegRNAs or epegRNAs. This table lists NCBI reference SNP designations for mutations installed in **Fig. 4d**.

Supplementary Note 1. Guidelines for epegRNA cloning via Golden Gate DNA assembly[13].
When cloning epegRNAs using the Golden Gate method, the same protocol as previously described[7] is appropriate with the important note that the junction sequence between the 3' extension oligo and the plasmid backbone is different for epegRNAs using tevopreQ$_1$ and trimmed mpknot (tmpknot), as shown below. More details on pegRNA design and cloning are available at http://liugroup.us. Plasmid backbones used for Golden Gate cloning have been deposited with Addgene.

Forward oligo for 3' extension of pegRNAs and epegRNAs	5′–	GTGCNNNNNNNNNNNNNNNNNNNNNNNNNN –3′
Reverse oligo for 3' extension of pegRNAs	3′–	NNNNNNNNNNNNNNNNNNNNNNNNNNAAAA–5′
Reverse oligo for 3' extension of epegRNAs with tevopreq$_1$-1	3′–	NNNNNNNNNNNNNNNNNNNNNNNNNNGCGC–5′
Reverse oligo for 3' extension of epegRNAs with tmpknot	3′–	NNNNNNNNNNNNNNNNNNNNNNNNNNGGGAGTC–5′

Supplementary Note 2. **pegRNA Linker Identification Tool (pegLIT) code**

```
from math import prod
import random
import heapq
import numpy as np
from scipy.special import expit as sigmoid
from sklearn.cluster import AgglomerativeClustering as HAC
from Levenshtein import distance as levenshtein_distance
import RNA # ViennaRNA

BASE_SYMBOLS = {
    "A": ("A",), "C": ("C",), "G": ("G",), "T": ("T",), "U": ("T",),
    "W": ("A", "T"), "S": ("C", "G"), "M": ("A", "C"),
    "K": ("G", "T"), "R": ("A", "G"), "Y": ("C", "T"),
    "B": ("C", "G", "T"), "D": ("A", "G", "T"), "H": ("A", "C", "T"), "V": ("A", "C", "G"),
    "N": ("A", "C", "G", "T")}

def apply_filters(seq_pre, seq_linker, seq_post, ac_thresh, u_thresh, n_thresh):
    """
    Returns False if any filter is failed i.e. AC content < ac_thresh OR consecutive Us > u_thresh
    OR consecutive Ns > n_thresh. Otherwise, True if all filters are passed. All thresholds have
    units nt (i.e. ac_thresh is not a percent). Ts are treated as Us.
    """
    # AC content
    if seq_linker.count("A") + seq_linker.count("C") < ac_thresh:
        return False
    # Consecutive U
    seq_neighborhood = seq_pre[-(u_thresh):] + seq_linker + seq_post[:u_thresh]
    seq_neighborhood = seq_neighborhood.replace("T", "U")
    if "U" * (u_thresh + 1) in seq_neighborhood:
        return False
    # Consecutive N
    seq_neighborhood = seq_pre[-(n_thresh):] + seq_linker + seq_post[:n_thresh]
    seq_neighborhood = seq_neighborhood.replace("T", "U")
    if any(nt * (n_thresh + 1) in seq_neighborhood for nt in set(seq_linker)):
        return False
    return True

def calc_subscores(linker_pos, *sequence_components):
    """
    Calculate base-pairing probs marginalized for each nucleotide
    """
    # Calculate bpp from ViennaRNA
    pegrna = RNA.fold_compound("".join(sequence_components))
    _ = pegrna.pf() # need to first internally calculate partition function
    basepair_probs = np.array(pegrna.bpp())[1:, 1:]
    # Fill in lower-triangle and diagonal of ViennaRNA's upper-triangular bpp matrix
    unpaired_probs = 1. - (basepair_probs.sum(axis=0) + basepair_probs.sum(axis=1))
    # copy data to make symmetric
    i_lower = np.tril_indices(len(basepair_probs), -1)
    i_diag = np.eye(len(basepair_probs), dtype=bool)
    basepair_probs[i_lower] = basepair_probs.T[i_lower]
```

```
    basepair_probs[i_diag] = unpaired_probs
    # Track indices of subsequences
    idx_cur = 0
    seq_idx = []
    for subseq in sequence_components:
        idx_prev = idx_cur
        idx_cur += len(subseq)
        seq_idx.append(slice(idx_prev, idx_cur))
    # Extract subscores for subsequences
    bpp_subseq = np.ma.masked_all(len(sequence_components))
    for i, subseq in enumerate(sequence_components):
        bpp_within_subseq = basepair_probs[seq_idx[i], seq_idx[linker_pos]]
        bpp_subseq[i] = np.mean(np.sum(bpp_within_subseq, axis=0))
    return bpp_subseq

def apply_score(seq_spacer, seq_scaffold, seq_template, seq_pbs, seq_linker,
                score_to_beat=None, epsilon=0.01):
    """
    Calculates subscores then outputs hashed score. Terminates calculation early if score will
    be less than score_to_beat. Prioritize PBS, spacer, template, scaffold.
    """
    # Cas9 complex at R loop subscore
    bpp_subseq1 = calc_subscores(2, seq_template, seq_pbs, seq_linker)
    subscore_pbs = 1. - bpp_subseq1[1]
    subscore_template = 1. - bpp_subseq1[0]
    # Free pegRNA subscore
    if ((score_to_beat is not None)
            and (epsilon * int(subscore_pbs / epsilon) < score_to_beat[0])):
        subscore_spacer = 0.
        subscore_scaffold = 0.
    else:
        bpp_subseq2 = calc_subscores(4, seq_spacer, seq_scaffold,
                                     seq_template, seq_pbs, seq_linker)
        subscore_spacer = 1. - bpp_subseq2[0]
        subscore_scaffold = 1. - bpp_subseq2[1]
    # Turn subscores into a single score
    return tuple(
        epsilon * int(val / epsilon)
        if val is not None else 0
        for val in (subscore_pbs, subscore_spacer, subscore_template, subscore_scaffold)
        )

def optimize(seq_spacer, seq_scaffold, seq_template, seq_pbs, seq_motif,
             linker_pattern, ac_thresh, u_thresh, n_thresh, topn, epsilon,
             num_repeats, num_steps, temp_init, temp_decay, seed):
    """
    Simulated annealing optimization of linkers
    """
    ## Pre-process inputs
    random.seed(seed)
    seq_pre = seq_spacer + seq_scaffold + seq_template + seq_pbs
    seq_post = seq_motif
    linker_pattern = linker_pattern.upper()
```

```
    ac_thresh = ac_thresh * len(linker_pattern)
    ## Simulated annealing to optimize linker sequence
    # Initialize hashmap of sequences already considered
    linker_skip = {}
    len_sequence_space = prod(len(BASE_SYMBOLS[nt]) for nt in linker_pattern)
    # Initialize min heap of topn linkers
    linker_heap = []
    for _ in range(num_repeats):
        # Initialize simulated annealing
        seq_linker_prev = "".join([random.choice(BASE_SYMBOLS[nt]) for nt in linker_pattern])
        score_prev = None
        temp = temp_init
        for _ in range(num_steps):
            # Generate new sequence by substituting characters in sequence until pass filters
            seq_linker = seq_linker_prev
            keep_going = True
            while keep_going:
                char_pos = random.randint(0, len(linker_pattern) - 1)
                seq_linker = (
                    seq_linker[:char_pos]
                    + random.choice(BASE_SYMBOLS[linker_pattern[char_pos]])
                    + seq_linker[(char_pos + 1):])
                keep_going = (
                    (seq_linker in linker_skip
                    or not apply_filters(seq_pre, seq_linker, seq_post,
                                ac_thresh, u_thresh, n_thresh))
                    and len(linker_skip) < len_sequence_space) # already screened whole seq space
                linker_skip[seq_linker] = True
            # Calculate score for linker sequence
            score_to_beat = linker_heap[0][0] if len(linker_heap) >= topn else None
            score = apply_score(seq_spacer, seq_scaffold, seq_template, seq_pbs, seq_linker,
                            score_to_beat=score_to_beat, epsilon=epsilon)
            # Add to min heap i.e. maintains the top `topn` largest entries
            if score_to_beat is None: # heap is not yet full
                heapq.heappush(linker_heap, (score, seq_linker))
            elif score > score_to_beat:
                heapq.heapreplace(linker_heap, (score, seq_linker))
            # Decide if keep proposal
            if (score_prev is None                                  # initialize
                or score > score_prev                               # exploit improvement
                or random.random() < sigmoid(                               # explore
                    sum((s1 - s2) * (epsilon ** i)
                        for i, (s1, s2) in enumerate(zip(score, score_prev))) / temp
                    )):
                seq_linker_prev = seq_linker
                score_prev = score
            # Update simulated annealing param
            temp *= temp_decay
    linker_heap_scores, linker_heap = zip(*linker_heap)
    return linker_heap_scores, linker_heap

def apply_bottleneck(heap_scores, heap, bottleneck, seed):
    """
```

```
    Cluster sequences and output top-scoring sequence per cluster.
    """
    random.seed(seed)
    # Pick best, randomly tiebreak if needed
    def _pick_best(scores, choices):
        idx_maxed = np.where(scores == np.max(scores))[0]
        idx_chosen = random.choice(idx_maxed)
        return choices[idx_chosen]
    # Can just pick best output
    if bottleneck == 1:
        return [_pick_best(heap_scores, heap)]
    # Calculate features for each linker sequence i.e. edit distance to all other linker sequences
    features = np.zeros((len(heap), len(heap)), dtype=int)
    for i, seq_x in enumerate(heap):
        for j, seq_y in enumerate(heap):
            features[i, j] = levenshtein_distance(seq_x, seq_y)
    # Cluster linker sequences
    clusters = HAC(n_clusters=bottleneck, linkage="complete").fit_predict(features)
    # Output highest-scoring linker sequence from each cluster
    output = []
    heap = np.array(heap)
    heap_scores_mean = np.mean(heap_scores, axis=1)
    for cluster_num in range(bottleneck):
        idx_cluster = clusters == cluster_num
        heap_cluster = heap[idx_cluster]
        cluster_scores = heap_scores_mean[idx_cluster]
        output.append(_pick_best(cluster_scores, heap_cluster))
    return output

def pegLIT(seq_spacer, seq_scaffold, seq_template, seq_pbs, seq_motif,
           linker_pattern="NNNNNNNN", ac_thresh=0.5, u_thresh=3, n_thresh=3, topn=100,
           epsilon=1e-2, num_repeats=10, num_steps=250, temp_init=0.15, temp_decay=0.95,
           bottleneck=1, seed=2020):
    """
    Optimizes+bottlenecks linker for an inputted pegRNA. Outputs linker recommendation(s).
    """
    # Simulated annealing to optimize linker sequence
    linker_heap_scores, linker_heap = optimize(
        seq_spacer, seq_scaffold, seq_template, seq_pbs, seq_motif,
        linker_pattern=linker_pattern, ac_thresh=ac_thresh, u_thresh=u_thresh,
        n_thresh=n_thresh, topn=topn, epsilon=epsilon, num_repeats=num_repeats,
        num_steps=num_steps, temp_init=temp_init, temp_decay=temp_decay, seed=seed)
    # Sample diverse sequences
    linker_output = apply_bottleneck(linker_heap_scores, linker_heap,
                                     bottleneck=bottleneck, seed=seed)
    return linker_output

if __name__ == "__main__":
    # Example usage for HEK3 +1 FLAG ins
    print(pegLIT(
        seq_spacer="GGCCCAGACTGAGCACGTGA",
        seq_scaffold="GTTTTAGAGCTAGAAATAGCAAGTTAAAATAAGGCTAGTCCGTTAT"
                     "CAACTTGAAAAAGTGGCACCGAGTCGGTGC",
```

```
seq_template="TGGAGGAAGCAGGGCTTCCTTTCCTCTGCCATCACTTATCG"
          "TCGTCATCCTTGTAATC",
seq_pbs="CGTGCTCAGTCTG",
seq_motif="CGCGGTTCTATCTAGTTACGCGTTAAACCAACTAGAA"))
```

Supplementary Note 3. Python script for quantifying prime editing intermediates

```
import pandas as pd
import glob
import re
import os
import subprocess
from subprocess import Popen
from subprocess import PIPE

#generates list of fastq files to analyze
fastqs = glob.glob('*.fastq')

#collects sequences of prime editing intermediates as any sequence between first 10 #nucleotides of the targeted protospacer and a poly(G) sequence installed by TdT, writes #sequences to a new "trimmed" text file
first10nts = {
                'HEK3':'GGCCCAGACT',
                'DNMT1':'GATTCCTGGT',
                'RNF2':'GTCATCTTAG',
                'EMX1':'GAGTCCGAGC'
                }

for fname in fastqs:
    with open(f'{fname[:-6]}_trimmed.txt','w+') as f:
        for spacer in first10nts.keys():
            if spacer in fname:

                nt_readARGS = ['grep', '-o', f'{first10nts[spacer]}.*GGGGGGGG', fname]
                nt_readproc = Popen(nt_readARGS, stdout=subprocess.PIPE,
                                    universal_newlines=True)
                f.write(str(nt_readproc.stdout.read())+'\n')

trimmedfastqs = glob.glob('*trimmed.txt')

#determines length of intermediate that was tailed, whether it contains the desired edit, and the #degree to which it contains sequence belonging to the reverse complement of the pegRNA #scaffold and spacer
for fname in trimmedfastqs:
    sequences_b = open(fname, 'r')

    if 'HEK3' in fname:
        edit_pos = 1
        designed_flap = 'AGATGGCAGAGGAA'
        RT_temp_length = 14
        scaffold_revcomp = 'GCACCGACTCGGTGCCACTTTTTCAAGTTGATAACGGACTAGCC
                            TTATTTTAACTTGCTATTTCTAGCTCTAAAACTCACGTGCTCAGTCTGGG
                            CC'
    if 'RNF2' in fname:
        edit_pos = 1
        designed_flap = 'ATGAGGTGTTCGTT'
        RT_temp_length = 14
        scaffold_revcomp = 'GCACCGACTCGGTGCCACTTTTTCAAGTTGATAACGGACTAGCC
```

```
                        TTATTTTAACTTGCTATTTCTAGCTCTAAAACCAGGTAATGACTAAGATG
                        AC'
if 'DNMT1' in fname:
    edit_pos = 5
    designed_flap = 'ACAGTGGTGAC'
    RT_temp_length = 11
    scaffold_revcomp = 'GCACCGACTCGGTGCCACTTTTTCAAGTTGATAACGGACTAGCC
                        TTATTTTAACTTGCTATTTCTAGCTCTAAAACGCGCGAACAGCTCCAGCC
                        CGC'
if 'EMX1' in fname:
    edit_pos = 5
    designed_flap = 'GAAGTGCTCCCATCAC'
    RT_temp_length = 16
    scaffold_revcomp = 'GCACCGACTCGGTGCCACTTTTTCAAGTTGATAACGGACTAGCC
                        TTATTTTAACTTGCTATTTCTAGCTCTAAAACTTCTTCTTCTGCTCGGACT
                        C'

seq_list = [ ]

df = pd.DataFrame({'flap seq':[],'flap length':[],'contains edit?':[],'scaffold insertion length':[]})

for line in sequences_b:
    seq = str(line)
    seq_list.append(seq)

counter = 0
counter_b = 1
for read in seq_list:
    edit = 0
    scaff_RT = 0
    for k in range(len(read) - 5):
        window = read[k:k+5]
        if window == 'GGGGG':
            spacer_flap = read[0:k]
            flap_length = len(spacer_flap) - 17
            if flap_length > edit_pos:
                if flap_length < len(designed_flap):
                    three_prime = flap_length
                else:
                    three_prime = len(designed_flap)
                if spacer_flap[17:17+three_prime] == designed_flap[:three_prime]:
                    edit = 1
            if flap_length > RT_temp_length:
                scaff_ins_len = flap_length - RT_temp_length
                scaff_ins_seq = spacer_flap[RT_temp_length+17:]
                if scaffold_revcomp[0:scaff_ins_len] == scaff_ins_seq:
                    scaff_RT = scaff_ins_len
            new_row = pd.DataFrame({'flap seq' : [spacer_flap],'flap length' :
                            [flap_length],'contains edit?' : [edit],'scaffold insertion length' : [scaff_RT]})
            df = df.append(new_row)
            break
df = df[['flap seq','flap length','contains edit?','scaffold insertion length']]
df.to_csv(f'{fname[:-4]}_output.csv', index=False)
```

Supplementary References

1. Bertsimas, D. & Tsitsiklis, J. Simulated Annealing. *Stat Sci* **8**, 10-15 (1993).
2. Win, M.N. & Smolke, C.D. A modular and extensible RNA-based gene-regulatory platform for engineering cellular function. *Proc Natl Acad Sci USA* **104**, 14283-14288 (2007).
3. Nielsen, S., Yuzenkova, Y. & Zenkin, N. Mechanism of eukaryotic RNA polymerase III transcription termination. *Science* **340**, 1577-1580 (2013).
4. Lorenz, R. et al. ViennaRNA package 2.0. *Algorithms Mol Biol* **6**, 26 (2011).
5. Chen, B. et al. Dynamic imaging of genomic loci in living human cells by an optimized CRISPR/Cas system. *Cell* **155**, 1479-1491 (2014).
6. Jost, M. et al. Titrating gene expression using libraries of systematically attenuated CRISPR guide RNAs. *Nat Biotechnol* **38**, 355-364 (2020).
7. Anzalone, A.V. et al. Search-and-replace genome editing without double-strand breaks or donor DNA. *Nature* **576**, 149-157 (2019).
8. Roth, A. et al. A riboswitch selective for the queuosine precursor preQ1 contains an unusually small aptamer domain. *Nat Struct Mol Biol* **14**, 308-317 (2007).
9. Houck-Loomis, B. et al. An equilibrium-dependent retroviral mRNA switch regulates translational recoding. *Nature* **480**, 561-564 (2011).
10. Steckelberg, A.L. et al. A folded viral noncoding RNA blocks host cell exoribonucleases through a conformationally dynamic RNA structure. *Proc Natl Acad Sci USA* **115**, 6404-6409 (2018).
11. Cate, J.H. et al. Crystal structure of a group I ribozyme domain: principles of RNA packing. *Science* **273**, 1678-1685 (1996).
12. Pandey, S., Agarwala, P. & Maiti, S. Effect of loops and G-quartets on the stability of RNA G-quadruplexes. *J Phys Chem B* **117**, 6896-6905 (2013).
13. Engler, C., Gruetzner, R., Kandzia, R. & Marillonnet, S. Golden gate shuffling: a one-pot DNA shuffling method based on type IIs restriction enzymes. *PLoS One* **4**, e5553 (2009).