rev_site 17_HTS	TGGAGTTCAGACGTGTGCTCTTCCGATCTCCTGTCTCTGCTCCTTTGTCCCC
fwd_site 18/19_HTS	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNGCATTACCTGGGAGCCTGTT
rev_site 18/19_HTS	TGGAGTTCAGACGTGTGCTCTTCCGATCTAACTTCAGCGGGCATCAGAA
fwd_site HGB1/2_HTS	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNGTGGAGTTTAGCCAGGGACC
rev_site HGB1/2_HTS	TGGAGTTCAGACGTGTGCTCTTCCGATCTTACAGGCCTCACTGGAGCTA
fwd_site HFE_HTS	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNGGCTGGATAACCTTGGCTGT
rev_site HFE_HTS	TGGAGTTCAGACGTGTGCTCTTCCGATCTTCCTCAGGCACTCCTCTCAA
fwd_HEK2_HTS	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNCCAGCCCCATCTGTCAAACT
rev_HEK2_HTS	TGGAGTTCAGACGTGTGCTCTTCCGATCTTGAATGGATTCCTTGGAAACAATGA
fwd_HEK3_HTS	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNATGTGGGCTGCCTAGAAAGG
rev_HEK3_HTS	TGGAGTTCAGACGTGTGCTCTTCCGATCTCCCAGCCAAACTTGTCAACC
fwd_HEK4_HTS	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNGAACCCAGGTAGCCAGAGAC
rev_HEK4_HTS	TGGAGTTCAGACGTGTGCTCTTCCGATCTTCCTTTCAACCCGAACGGAG
fwd_HEK2_off1_HTS	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNGTGTGGAGAGTGAGTAAGCCA
rev_HEK2_off1_HTS	TGGAGTTCAGACGTGTGCTCTTCCGATCTACGGTAGGATGATTTCAGGCA
fwd_HEK2_off2_HTS	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNCACAAAGCAGTGTAGCTCAGG
rev_HEK2_off2_HTS	TGGAGTTCAGACGTGTGCTCTTCCGATCTTTTTTGGTACTCGAGTGTTATTCAG
fwd_HEK3 off1_HTS	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNTCCCCTGTTGACCTGGAGAA
rev_HEK3_off1_HTS	TGGAGTTCAGACGTGTGCTCTTCCGATCTCACTGTACTTGCCCTGACCA
fwd_HEK3_off2_HTS	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNTTGGTGTTGACAGGGAGCAA
rev_ HEK3_off2_HTS	TGGAGTTCAGACGTGTGCTCTTCCGATCTCTGAGATGTGGGCAGAAGGG
fwd_HEK3_off3_HTS	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNTGAGAGGGAACAGAAGGGCT
rev_ HEK3_off3_HTS	TGGAGTTCAGACGTGTGCTCTTCCGATCTGTCCAAAGGCCCAAGAACCT
fwd_HEK3_off4_HTS	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNTCCTAGCACTTTGGAAGGTCG
rev_HEK3_off4_HTS	TGGAGTTCAGACGTGTGCTCTTCCGATCTGCTCATCTTAATCTGCTCAGCC
fwd_HEK3_off5_HTS	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNAAAGGAGCAGCTCTTCCTGG
rev_HEK3_off5_HTS	TGGAGTTCAGACGTGTGCTCTTCCGATCTGTCTGCACCATCTCCCACAA
fwd_HEK4_off1_HTS	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNGGCATGGCTTCTGAGACTCA
rev_ HEK4_off1_HTS	TGGAGTTCAGACGTGTGCTCTTCCGATCTGTCTCCCTTGCACTCCCTGTCTTT
fwd_ HEK4_off2_HTS	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNTTTGGCAATGGAGGCATTGG
rev_ HEK4_off2_HTS	TGGAGTTCAGACGTGTGCTCTTCCGATCTGAAGAGGCTGCCCATGAGAG
fwd_ HEK4_off3_HTS	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNGGTCTGAGGCTCGAATCCTG
rev_ HEK4_off3_HTS	TGGAGTTCAGACGTGTGCTCTTCCGATCTCTGTGGCCTCCATATCCCTG
fwd_ HEK4_off4_HTS	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNTTTCCACCAGAACTCAGCCC
rev_ HEK4_off4_HTS	TGGAGTTCAGACGTGTGCTCTTCCGATCTCCTCGGTTCCTCCACAACAC

fwd_ HEK4_off5_HTS	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNCACGGGAAGGACAGGAGAAG
rev_HEK4_off5_HTS	TGGAGTTCAGACGTGTGCTCTTCCGATCTGCAGGGGAGGGATAAAGCAG
fwd_ site 1_HDR_HTS	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNCCAGCCCCATCTGTCAAACT
rev_ site 1_HDR_HTS	TGGAGTTCAGACGTGTGCTCTTCCGATCTTGAATGGATTCCTTGGAAACAATGA
fwd_ site 2_HDR_HTS	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNCCCTGAGATACAGTCACGAGGT
rev_ site 2_HDR_HTS	TGGAGTTCAGACGTGTGCTCTTCCGATCTCCTGAAATGCTGTGCGTGTCTA
fwd_ site 3_HDR_HTS	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNGCCCACATTACCTTGGTGCATA
rev_ site 3_HDR_HTS	TGGAGTTCAGACGTGTGCTCTTCCGATCTGGCAGGCAGATTATCATTCCCA
fwd_ site 4_HDR_HTS	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNAAAGTGCTGCGATTACAGGC
rev_ site 4_HDR_HTS	TGGAGTTCAGACGTGTGCTCTTCCGATCTGTGGCATCCAGAGACATGGT
fwd_ site 6_HDR_HTS	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNATGTGGGCTGCCTAGAAAGG
rev_ site 6_HDR_HTS	TGGAGTTCAGACGTGTGCTCTTCCGATCTCCCAGCCAAACTTGTCAACC
B_Catenin_mRNA _fwd	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNATTTGATGGAGTTGGACATGGCC
B_Catenin_mRNA _rev	TGGAGTTCAGACGTGTGCTCTCCAGCTACTTGTTCTTGAGTGAAGG
B_Actin_mRNA_fwd	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNGACAAAACCTAACTTGCGCAGAAAACAAGATG
B_Actin_mRNA_rev	TGGAGTTCAGACGTGTGCTCTGCTTTTAGGATGGCAAGGGACTTCCTG
GAPDH_mRNA_fwd	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNGGCTACAGCAACAGGGTGGTGGAC
GAPDH_mRNA_rev	TGGAGTTCAGACGTGTGCTCTCCATCAATAAAGTACCCTGTGCTCAACC
RB1_mRNA_fwd	ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNGGAAGGATTATGATAGGGACAAGG
RB1_mRNA_rev	TGGAGTTCAGACGTGTGCTCTCCACAATTTCCTTCATATGTTCAAAC

Supplementary Table 10. 100-mer single-stranded oligonucleotide donor templates (ssODNs) used in HDR experiments.

Target site	Sequence
1	5'-TTTTCCAGCCCGCTGGCCCTGTAAAGGAAACTGGAACGCAAAGCATAGACTGCGCGGCGG GCCAGCCTGAATAGCTGCAAACAAGTGCAGAATATCTGAT-3'
2	5'-CATGAAAAAGAGACTGATTGCGTGGAGTTCATGGAGTGTGAGGCATAGACTGCACGAGACA TAAACCATGACTTGCAGATGAAGAAGCATTTTAAAAGT-3'
3	5'-GACAGCCAGTGGTTAAGTCAGAACCCGACTCAGGTCAGGAAAGCAGAGACTGCCCGGGGT TGGGAAGGCGGTGAACTCAGAGATAGAAACAGGGTGGGTG-3'
4	5'-ATTTTAAGCTGTAGTATTATGAAGGGAAATCTGGAGCGAAGAGAATAGACTGTACGGAAACC AGTTAAGAAATAGGACATGGAGGCTAGGTGCAGTGGCT-3'
6	5'-CCTCTGCCATCACGTGCTCAGTCTGGGCCCCAAGGATTGGCCCAGGCCAGGGCTCGAGAA GCAGAAAAAAAGCATCAAGCCTACAAATGCATGCTTACTT-3'

Supplementary Sequences 1. DNA sequences of adenine deaminases used in this study.

Bacterial codon-optimized ecTadA (wild-type):
ATGTCTGAAGTCGAATTTAGCCACGAATACTGGATGCGTCACGCGCTGACGCTGGCGAAACGTGCCTGGGATGAGC
GGGAAGTGCCGGTCGGCGCGGTATTAGTGCATAACAATCGGGTAATCGGCGAAGGCTGGAACCGCCCGATTGGTC
GCCATGATCCCACCGCACATGCAGAAATCATGGCCCTGCGGCAGGGTGGTCTGGTGATGCAAAATTATCGTCTGATC
GACGCCACGTTGTATGTCACGCTTGAACCATGTGTAATGTGTGCCGGAGCGATGATCCACAGTCGCATTGGTCGCGT
GGTCTTTGGTGCGCGTGACGCGAAAACTGGCGCTGCGGGATCTTTAATGGATGTGCTGCATCATCCGGGTATGAATC
ACCGAGTGGAAATTACGGAAGGAATACTGGCGGATGAGTGCGCGGCGTTGCTCAGTGACTTCTTTCGCATGCGCCG
CCAGGAAATTAAAGCGCAGAAAAAAGCGCAATCCTCGACGGAT

Mammalian codon-optimized ecTadA (wild-type):
ATGTCCGAAGTCGAGTTTTCCCATGAGTACTGGATGAGACACGCATTGACTCTCGCAAAGAGGGCTTGGGATGAACG
CGAGGTGCCCGTGGGGGCAGTACTCGTGCATAACAATCGCGTAATCGGCGAAGGTTGGAATAGGCCGATCGGACG
CCACGACCCCACTGCACATGCGGAAATCATGGCCCTTCGACAGGGAGGGCTTGTGATGCAGAATTATCGACTTATCG
ATGCGACGCTGTACGTCACGCTTGAACCTTGCGTAATGTGCGCGGGAGCTATGATTCACTCCCGCATTGGACGAGTT
GTATTCGGTGCCCGCGACGCCAAGACGGGTGCCGCAGGTTCACTGATGGACGTGCTGCATCACCCAGGCATGAACC
ACCGGGTAGAAATCACAGAAGGCATATTGGCGGACGAATGTGCGGCGCTGTTGTCCGACTTTTTTCGCATGCGGAG
GCAGGAGATCAAGGCCCAGAAAAAAGCACAATCCTCTACTGAC

Mammalian codon-optimized mADA:
ATGGCCCAGACACCCGCATTCAACAAACCCAAAGTAGAGTTACACGTCCACCTGGATGGAGCCATCAAGCCAGAAAC
CATCTTATACTTTGGCAAGAAGAGAGGCATCGCCCTCCCGGCAGATACAGTGGAGGAGCTGCGCAACATTATCGGCA
TGGACAAGCCCCTCTCGCTCCCAGGCTTCCTGGCCAAGTTTGACTACTACATGCCTGTGATTGCGGGCTGCAGAGAG
GCCATCAAGAGGATCGCCTACGAGTTTGTGGAGATGAAGGCAAAGGAGGGCGTGGTCTATGTGGAAGTGCGCTATA
GCCCACACCTGCTGGCCAATTCCAAGGTGGACCCAATGCCCTGGAACCAGACTGAAGGGGACGTCACCCCTGATGA
CGTTGTGGATCTTGTGAACCAGGGCCTGCAGGAGGGAGAGCAAGCATTTGGCATCAAGGTCCGGTCCATTCTGTGC
TGCATGCGCCACCAGCCCAGCTGGTCCCTTGAGGTGTTGGAGCTGTGTAAGAAGTACAATCAGAAGACCGTGGTGG
CTATGGACTTGGCTGGGGATGAGACCATTGAAGGAAGTAGCCTCTTCCCAGGCCACGTGGAAGCCTATGAGGGCGC
AGTAAAGAATGGCATTCATCGGACCGTCCACGCTGGCGAGGTGGGCTCTCCTGAGGTTGTGCGTGAGGCTGTGGAC
ATCCTCAAGACAGAGAGGGTGGGACATGGTTATCACACCATCGAGGATGAAGCTCTCTACAACAGACTACTGAAAGA
AAACATGCACTTTGAGGTCTGCCCCTGGTCCAGCTACCTCACAGGCGCCTGGGATCCCAAAACGACGCATGCGGTT
GTTCGCTTCAAGAATGATAAGGCCAACTACTCACTCAACACAGACGACCCCCTCATCTTCAAGTCCACCCTAGACACT
GACTACCAGATGACCAAGAAAGACATGGGCTTCACTGAGGAGGAGTTCAAGCGACTGAACATCAACGCAGCGAAGT
CAAGCTTCCTCCCAGAGGAAGAGAAGAAGGAACTTCTGGAACGGCTCTACAGAGAATACCAA

Mammalian codon optimized hADAR2 (catalytic domain):
ATGCATCTCGATCAAACCCCGAGCCGCCAACCAATCCCGAGTGAAGGCCTGCAACTGCATCTGCCACAAGTTCTGGC
GGATGCCGTTAGCCGCCTGGTCTTGGGTAAGTTCGGTGATCTGACAGACAACTTTTCTAGTCCACATGCTCGCCGTA
AGGTGCTGGCTGGCGTTGTGATGACCACAGGTACAGACGTCAAAGATGCTAAAGTGATTTCTGTGTCTACTGGCACG
AAGTGCATTAACGGCGAATATATGTCTGACCGTGGCTTAGCGCTTAACGATTGTCATGCCGAAATCATCTCCCGTCGT
TCATTGCTTCGCTTCCTGTACACGCAGTTGGAACTGTATCTGAATAACAAAGACGATCAGAAGCGTTCTATTTTCCAG
AAGTCTGAGCGCGGCGGGTTCCGTCTTAAAGAGAATGTGCAGTTTCACCTTTATATTTCAACCTCTCCTTGTGGTGAT
GCCCGTATTTTTTCACCACACGAACCTATTTTAGAGGAACCGGCCGATCGTCATCCGAACCGCAAAGCCCGTGGGCA
GCTGCGTACGAAAATCGAATCAGGTGAAGGCACCATTCCCGTCCGCTCCAATGCGAGCATTCAAACGTGGGACGGT
GTGTTACAGGGCGAACGCCTGTTAACCATGAGCTGCTCAGACAAAATTGCACGTTGGAACGTGGTAGGCATCCAGG
GCTCGTTATTGAGCATTTTCGTGGAGCCGATTTATTTTAGTTCCATCATTTTGGGCTCACTCTACCACGGCGATCACCT
TAGCCGCGCGATGTACCAGCGCATTAGTAACATCGAAGATTTACCGCCCCTGTATACCCTGAACAAACCACTGTTAA
GCGGTATTTCTAACGCGGAGGCGCGTCAGCCTGGTAAAGCCCCGAACTTCAGTGTGAACTGGACTGTGGGTGATTC
TGCAATTGAGGTAATTAACGCGACGACGGGTAAAGATGAACTGGGCCGTGCCTCTCGTCTGTGTAAACACGCGCTGT
ACTGTCGTTGGATGCGCGTGCACGGTAAAGTTCCCAGTCATCTGTTACGTAGCAAGATCACCAAGCCAAATGTCTAC
CACGAATCGAAGCTGGCCGCGAAAGAATACCAAGCGGCTAAGGCGCGTCTGTTCACCGCCTTTATTAAGGCTGGCTT
AGGGGCCTGGGTGGAAAAACCAACCGAGCAAGATCAATTCAGTCTGACCCCG

Mammalian codon optimized hADAT2:
ATGGAGGCGAAGGCGGCACCCAAGCCAGCTGCAAGCGGCGCGTGCTCGGTGTCGGCAGAGGAGACCGAAAAGTG
GATGGAGGAGGCGATGCACATGGCCAAAGAAGCCCTCGAAAATACTGAAGTTCCTGTTGGCTGTCTTATGGTCTACA
ACAATGAAGTTGTAGGGAAGGGGAGAAATGAAGTTAACCAAACCAAAAATGCTACTCGACATGCAGAAATGGTGGCC
ATCGATCAGGTCCTCGATTGGTGTCGTCAAAGTGGCAAGAGTCCCTCTGAAGTATTTGAACACACTGTGTTGTATGTC
ACTGTGGAGCCGTGCATTATGTGTGCAGCTGCTCTCCGCCTGATGAAAATCCCGCTGGTTGTATATGGCTGTCAGAA
TGAACGATTTGGTGGTTGTGGCTCTGTTCTAAATATTGCCTCTGCTGACCTACCAAACACTGGGAGACCATTTCAGTG

TATCCCTGGATATCGGGCTGAGGAAGCAGTGGAAATGTTAAAGACCTTCTACAAACAAGAAAATCCAAATGCACCAAA
ATCGAAAGTTCGGAAAAAGGAATGTCAGAAATCT

Supplementary Sequences 2. DNA sequences of antibiotic resistance genes used in this study. Inactivating mutations are shown in red.

Chloramphenicol resistance gene (Cam^R) H193Y:

ATGGAGAAAAAAATCACTGGATATACCACCGTTGATATATCCCAATGGCATCGTAAAGAACATTTTGAGGCATTTCAGT
CAGTTGCTCAATGTACCTATAACCAGACCGTTCAGCTGGATATTACGGCCTTTTTAAAGACCGTAAAGAAAAATAAGC
ACAAGTTTTATCCGGCCTTTATTCACATTCTTGCCCGCCTGATGAATGCTCATCCGGAGTTCCGTATGGCAATGAAAG
ACGGTGAGCTGGTGATATGGGATAGTGTTCACCCTTGTTACACCGTTTTCCATGAGCAAACTGAAACGTTTTCATCGC
TCTGGAGTGAATACCACGACGATTTCCGGCAGTTTCTACACATATATTCGCAAGATGTGGCGTGTTACGGTGAAAACC
TGGCCTATTTCCCTAAAGGGTTTATTGAGAATATGTTTTTCGTCTCAGCCAATCCCTGGGTGAGTTTCACCAGTTTTGA
TTTAAACGTGGCCAATATGGACAACTTCTTCGCCCCCGTTTTCACTATGGGCAAATATTATACGCAAGGCGACAAGGT
GCTGATGCCGCTGGCCATCCAGGTGCAC**T**ACGCCGTATGCGACGGCTTCCATGTCGGCAGAATGCTTAATGAATTA
CAACAGTACTGCGATGAGTGGCAGGGCGGGGCGTAA

Kanamycin resistance gene (Kan^R) Q4STOP and W15STOP:

ATGATCGAA**T**AAGATGGATTGCACGCAGGTTCTCCGGCCGCTT**A**GGTGGAGCGCCTATTCGGCTATGACTGGGCAC
AACAGACAATCGGCTGCTCTGATGCCGCCGTGTTCCGGCTGTCAGCGCAGGGGCGCCCGGTTCTTTTTGTCAAGAC
CGACCTGTCCGGTGCCCTGAATGAACTGCAGGACGAGGCAGCGCGGCTATCGTGGCTGGCCACGACGGGCGTTCC
TTGCGCAGCTGTGCTCGACGTTGTCACTGAAGCGGGAAGGGACTGGCTGCTATTGGGCGAAGTGCCGGGGCAGGA
TCTCCTGTCATCTCACCTTGCTCCTGCCGAGAAAGTATCCATCATGGCTGATGCAATGCGGCGGCTGCATACGCTTG
ATCCGGCTACCTGCCCATTCGACCACCAAGCGAAACATCGCATCGAGCGAGCACGTACTCGGATGGAAGCCGGTCT
TGTCGATCAGGATGATCTGGACGAAGAGCATCAGGGGCTCGCGCCAGCCGAACTGTTCGCCAGGCTCAAGGCGCG
CATGCCCGACGGCGAGGATCTCGTCGTGACCCATGGCGATGCCTGCTTGCCGAATATCATGGTGGAAAATGGCCGC
TTTTCTGGATTCATCGACTGTGGCCGGCTGGGTGTGGCGGACCGCTATCAGGACATAGCGTTGGCTACCCGTGATAT
TGCTGAAGAGCTTGGCGGCGAATGGGCTGACCGCTTCCTCGTGCTTTACGGTATCGCCGCTCCCGATTCGCAGCGC
ATCGCCTTCTATCGCCTTCTTGACGAGTTCTTCTAA

Spectinomycin resistance gene ($Spect^R$) T89I:

ATGAGGGAAGCGGTGATCGCCGAAGTATCGACTCAACTATCAGAGGTAGTTGGCGTCATCGAGCGCCATCTCGAAC
CGACGTTGCTGGCCGTACATTTGTACGGCTCCGCAGTGGATGGCGGCCTGAAGCCACACAGTGATATTGATTTGCTG
GTTACGGTGACCGTAAGGCTTGATGAAACAACGCGGCGAGCTTTGATCAACGACCTTTTGGAAACTTCGGCTTCCCC
TGGAGAGAGCGAGATTCTCCGCGCTGTAGAAGTCA**T**CATTGTTGTGCACGACGACATCATTCCGTGGCGTTATCCAG
CTAAGCGCGAACTGCAATTTGGAGAATGGCAGCGCAATGACATTCTTGCAGGTATCTTCGAGCCAGCCACGATCGAC
ATTGATCTGGCTATCTTGCTGACAAAAGCAAGAGAACATAGCGTTGCCTTGGTAGGTCCAGCGGCGGAGGAACTCTT
TGATCCGGTTCCTGAACAGGATCTATTTGAGGCGCTAAATGAAACCTTAACGCTATGGAACTCGCCGCCCGACTGGG
CTGGCGATGAGCGAAATGTAGTGCTTACGTTGTCCCGCATTTGGTACAGCGCAGTAACCGGCAAAATCGCGCCGAA
GGATGTCGCTGCCGACTGGGCAATGGAGCGCCTGCCGGCCCAGTATCAGCCCGTCATACTTGAAGCTAGACAGGCT
TATCTTGGACAAGAAGAAGATCGCTTGGCCTCGCGCGCAGATCAGTTGGAAGAATTTGTCCACTACGTGAAAGGCGA
GATCACCAAGGTAGTCGGCAAATAA

Kanamycin resistance gene (Kan^R) Q4STOP and D208N:

ATGATCGAA**T**AAGATGGATTGCACGCAGGTTCTCCGGCCGCTTGGGTGGAGAGGCTATTCGGCTATGACTGGGCAC
AACAGACAATCGGCTGCTCTGATGCCGCCGTGTTCCGGCTGTCAGCGCAGGGGCGCCCGGTTCTTTTTGTCAAGAC
CGACCTGTCCGGTGCCCTGAATGAACTGCAGGACGAGGCAGCGCGGCTATCGTGGCTGGCCACGACGGGCGTTCC
TTGCGCAGCTGTGCTCGACGTTGTCACTGAAGCGGGAAGGGACTGGCTGCTAT AGCCGGCCACAGTTAATGAA
TGGGCGAAGTGCCGGGGCAGGATCTCCTGTCATCTCACCTTGCTCCTGCCGAGAAAGTATCCATCATGGCTGATGCA
ATGCGGCGGCTGCATACGCTTGATCCGGCTACCTGCCCATTCGACCACCAAGCGAAACATCGCATCGAGCGAGCAC
GTACTCGGATGGAAGCCGGTCTTGTCGATCAGGATGATCTGGACGAAGAGCATCAGGGGCTCGCGCCAGCCGAACT
GTTCGCCAGGCTCAAGGCGCGCATGCCCGACGGCGAGGATCTCGTCGTGACCCATGGCGATGCCTGCTTGCCGAA
TATCATGGTGGAAAATGGCCGCTTTTCTGGATTCATT**A**ACTGTGGCCGGCTGGGTGTGGCGGACCGCTATCAGGAC
ATAGCGTTGGCTACCCGTGATATTGCTGAAGAGCTTGGCGGCGAATGGGCTGACCGCTTCCTCGTGCTTTACGGTAT
CGCCGCTCCCGATTCGCAGCGCATCGCCTTCTATCGCCTTCTTGACGAGTTCTTCTAA

Supplementary Sequences 3. Amino acid sequences of late-stage ABEs developed in this study. Color coding is as follows:
green = ecTadA (wt), monomer 1 of 2
orange = linker
black + red = evolved ecTadA* internal monomer 2 of 2, with mutations highlighted in red
blue = Cas9 nickase (D10A mutation underlined)
purple = NLS

ABE6.3 (ecTadA(wt)–linker(32 aa)–ecTadA*(6.3)–linker(32 aa)–Cas9 nickase–NLS):
MSEVEFSHEYWMRHALTLAKRAWDEREVPVGAVLVHNNRVIGEGWNRPIGRHDPTAHAEIMALRQGGLVMQNYRLIDAT
LYVTLEPCVMCAGAMIHSRIGRVVFGARDAKTGAAGSLMDVLHHPGMNHRVEITEGILADECAALLSDFFRMRRQEIKAQK
KAQSSTDSGGSSGGSSGSETPGTSESATPESSGGSSGGSSEVEFSHEYWMRHALTLAKRAWDEREVPVGAVLVLNNRV
IGEGWNRSIGLHDPTAHAEIMALRQGGLVMQNYRLIDATLYVTFEPCVMCAGAMIHSRIGRVVFGVRNAKTGAAGSLMDV
LHYPGMNHRVEITEGILADECAALLCYFFRMRRQVFNAQKKAQSSTDSGGSSGGSSGSETPGTSESATPESSGGSSGGS
DKKYSIGLAIGTNSVGWAVITDEYKVPSKKFKVLGNTDRHSIKKNLIGALLFDSGETAEATRLKRTARRRYTRRKNRICYLQE
IFSNEMAKVDDSFFHRLEESFLVEEDKKHERHPIFGNIVDEVAYHEKYPTIYHLRKKLVDSTDKADLRLIYLALAHMIKFRGH
FLIEGDLNPDNSDVDKLFIQLVQTYNQLFEENPINASGVDAKAILSARLSKSRRLENLIAQLPGEKKNGLFGNLIALSLGLTPN
FKSNFDLAEDAKLQLSKDTYDDDLDNLLAQIGDQYADLFLAAKNLSDAILLSDILRVNTEITKAPLSASMIKRYDEHHQDLTLL
KALVRQQLPEKYKEIFFDQSKNGYAGYIDGGASQEEFYKFIKPILEKMDGTEELLVKLNREDLLRKQRTFDNGSIPHQIHLG
ELHAILRRQEDFYPFLKDNREKIEKILTFRIPYYVGPLARGNSRFAWMTRKSEETITPWNFEEVVDKGASAQSFIERMTNFD
KNLPNEKVLPKHSLLYEYFTVYNELTKVKYVTEGMRKPAFLSGEQKKAIVDLLFKTNRKVTVKQLKEDYFKKIECFDSVEISG
VEDRFNASLGTYHDLLKIIKDKDFLDNEENEDILEDIVLTLTLFEDREMIEERLKTYAHLFDDKVMKQLKRRRYTGWGRLSRK
LINGIRDKQSGKTILDFLKSDGFANRNFMQLIHDDSLTFKEDIQKAQVSGQGDSLHEHIANLAGSPAIKKGILQTVKVVDELV
KVMGRHKPENIVIEMARENQTTQKGQKNSRERMKRIEEGIKELGSQILKEHPVENTQLQNEKLYLYYLQNGRDMYVDQEL
DINRLSDYDVDHIVPQSFLKDDSIDNKVLTRSDKNRGKSDNVPSEEVVKKMKNYWRQLLNAKLITQRKFDNLTKAERGGLS
ELDKAGFIKRQLVETRQITKHVAQILDSRMNTKYDENDKLIREVKVITLKSKLVSDFRKDFQFYKVREINNYHHAHDAYLNAV
VGTALIKKYPKLESEFVYGDYKVYDVRKMIAKSEQEIGKATAKYFFYSNIMNFFKTEITLANGEIRKRPLIETNGETGEIVWDK
GRDFATVRKVLSMPQVNIVKKTEVQTGGFSKESILPKRNSDKLIARKKDWDPKKYGGFDSPTVAYSVLVVAKVEKGKSKKL
KSVKELLGITIMERSSFEKNPIDFLEAKGYKEVKKDLIIKLPKYSLFELENGRKRMLASAGELQKGNELALPSKYVNFLYLASH
YEKLKGSPEDNEQKQLFVEQHKHYLDEIIEQISEFSKRVILADANLDKVLSAYNKHRDKPIREQAENIIHLFTLTNLGAPAAFK
YFDTTIDRKRYTSTKEVLDATLIHQSITGLYETRIDLSQLGGDSGGSPKKKRKV*

ABE7.8 (ecTadA(wt)–linker(32 aa)–ecTadA*(7.8)–linker(32 aa)–Cas9 nickase–NLS):
MSEVEFSHEYWMRHALTLAKRAWDEREVPVGAVLVHNNRVIGEGWNRPIGRHDPTAHAEIMALRQGGLVMQNYRLIDAT
LYVTLEPCVMCAGAMIHSRIGRVVFGARDAKTGAAGSLMDVLHHPGMNHRVEITEGILADECAALLSDFFRMRRQEIKAQK
KAQSSTDSGGSSGGSSGSETPGTSESATPESSGGSSGGSSEVEFSHEYWMRHALTLAKRALDEREVPVGAVLVLNNRVI
GEGWNRAIGLHDPTAHAEIMALRQGGLVMQNYRLIDATLYVTFEPCVMCAGAMIHSRIGRVVFGVRNAKTGAAGSLMDVL
HYPGMNHRVEITEGILADECNALLCYFFRMRRQVFNAQKKAQSSTDSGGSSGGSSGSETPGTSESATPESSGGSSGGSD
KKYSIGLAIGTNSVGWAVITDEYKVPSKKFKVLGNTDRHSIKKNLIGALLFDSGETAEATRLKRTARRRYTRRKNRICYLQEI
FSNEMAKVDDSFFHRLEESFLVEEDKKHERHPIFGNIVDEVAYHEKYPTIYHLRKKLVDSTDKADLRLIYLALAHMIKFRGHF
LIEGDLNPDNSDVDKLFIQLVQTYNQLFEENPINASGVDAKAILSARLSKSRRLENLIAQLPGEKKNGLFGNLIALSLGLTPNF
KSNFDLAEDAKLQLSKDTYDDDLDNLLAQIGDQYADLFLAAKNLSDAILLSDILRVNTEITKAPLSASMIKRYDEHHQDLTLLK
ALVRQQLPEKYKEIFFDQSKNGYAGYIDGGASQEEFYKFIKPILEKMDGTEELLVKLNREDLLRKQRTFDNGSIPHQIHLGEL
HAILRRQEDFYPFLKDNREKIEKILTFRIPYYVGPLARGNSRFAWMTRKSEETITPWNFEEVVDKGASAQSFIERMTNFDKN
LPNEKVLPKHSLLYEYFTVYNELTKVKYVTEGMRKPAFLSGEQKKAIVDLLFKTNRKVTVKQLKEDYFKKIECFDSVEISGVE
DRFNASLGTYHDLLKIIKDKDFLDNEENEDILEDIVLTLTLFEDREMIEERLKTYAHLFDDKVMKQLKRRRYTGWGRLSRKLI
NGIRDKQSGKTILDFLKSDGFANRNFMQLIHDDSLTFKEDIQKAQVSGQGDSLHEHIANLAGSPAIKKGILQTVKVVDELVKV
MGRHKPENIVIEMARENQTTQKGQKNSRERMKRIEEGIKELGSQILKEHPVENTQLQNEKLYLYYLQNGRDMYVDQELDIN
RLSDYDVDHIVPQSFLKDDSIDNKVLTRSDKNRGKSDNVPSEEVVKKMKNYWRQLLNAKLITQRKFDNLTKAERGGLSELD
KAGFIKRQLVETRQITKHVAQILDSRMNTKYDENDKLIREVKVITLKSKLVSDFRKDFQFYKVREINNYHHAHDAYLNAVVGT
ALIKKYPKLESEFVYGDYKVYDVRKMIAKSEQEIGKATAKYFFYSNIMNFFKTEITLANGEIRKRPLIETNGETGEIVWDKGRD
FATVRKVLSMPQVNIVKKTEVQTGGFSKESILPKRNSDKLIARKKDWDPKKYGGFDSPTVAYSVLVVAKVEKGKSKKLKSV
KELLGITIMERSSFEKNPIDFLEAKGYKEVKKDLIIKLPKYSLFELENGRKRMLASAGELQKGNELALPSKYVNFLYLASHYEK
LKGSPEDNEQKQLFVEQHKHYLDEIIEQISEFSKRVILADANLDKVLSAYNKHRDKPIREQAENIIHLFTLTNLGAPAAFKYFD
TTIDRKRYTSTKEVLDATLIHQSITGLYETRIDLSQLGGDSGGSPKKKRKV*

ABE7.9 (ecTadA(wt)–linker(32 aa)–ecTadA*(7.9)–linker(32 aa)–Cas9 nickase–NLS):
MSEVEFSHEYWMRHALTLAKRAWDEREVPVGAVLVHNNRVIGEGWNRPIGRHDPTAHAEIMALRQGGLVMQNYRLIDAT
LYVTLEPCVMCAGAMIHSRIGRVVFGARDAKTGAAGSLMDVLHHPGMNHRVEITEGILADECAALLSDFFRMRRQEIKAQK
KAQSSTDSGGSSGGSSGSETPGTSESATPESSGGSSGGSSEVEFSHEYWMRHALTLAKRALDEREVPVGAVLVLNNRVI

GEGWNRAIGLHDPTAHAEIMALRQGGLVMQNYRLIDATLYVTFEPCVMCAGAMIHSRIGRVVFGVRNAKTGAAGSLMDVL
HYPGMNHRVEITEGILADECNALLCYFFRMPRQVFNAQKKAQSSTDSGGSSGGSSGSETPGTSESATPESSGGSSGGSD
KKYSIGLAIGTNSVGWAVITDEYKVPSKKFKVLGNTDRHSIKKNLIGALLFDSGETAEATRLKRTARRRYTRRKNRICYLQEI
FSNEMAKVDDSFFHRLEESFLVEEDKKHERHPIFGNIVDEVAYHEKYPTIYHLRKKLVDSTDKADLRLIYLALAHMIKFRGHF
LIEGDLNPDNSDVDKLFIQLVQTYNQLFEENPINASGVDAKAILSARLSKSRRLENLIAQLPGEKKNGLFGNLIALSLGLTPNF
KSNFDLAEDAKLQLSKDTYDDDLDNLLAQIGDQYADLFLAAKNLSDAILLSDILRVNTEITKAPLSASMIKRYDEHHQDLTLLK
ALVRQQLPEKYKEIFFDQSKNGYAGYIDGGASQEEFYKFIKPILEKMDGTEELLVKLNREDLLRKQRTFDNGSIPHQIHLGEL
HAILRRQEDFYPFLKDNREKIEKILTFRIPYYVGPLARGNSRFAWMTRKSEETITPWNFEEVVDKGASAQSFIERMTNFDKN
LPNEKVLPKHSLLYEYFTVYNELTKVKYVTEGMRKPAFLSGEQKKAIVDLLFKTNRKVTVKQLKEDYFKKIECFDSVEISGVE
DRFNASLGTYHDLLKIIKDKDFLDNEENEDILEDIVLTLTLFEDREMIEERLKTYAHLFDDKVMKQLKRRRYTGWGRLSRKLI
NGIRDKQSGKTILDFLKSDGFANRNFMQLIHDDSLTFKEDIQKAQVSGQGDSLHEHIANLAGSPAIKKGILQTVKVVDELVKV
MGRHKPENIVIEMARENQTTQKGQKNSRERMKRIEEGIKELGSQILKEHPVENTQLQNEKLYLYYLQNGRDMYVDQELDIN
RLSDYDVDHIVPQSFLKDDSIDNKVLTRSDKNRGKSDNVPSEEVVKKMKNYWRQLLNAKLITQRKFDNLTKAERGGLSELD
KAGFIKRQLVETRQITKHVAQILDSRMNTKYDENDKLIREVKVITLKSKLVSDFRKDFQFYKVREINNYHHAHDAYLNAVVGT
ALIKKYPKLESEFVYGDYKVYDVRKMIAKSEQEIGKATAKYFFYSNIMNFFKTEITLANGEIRKRPLIETNGETGEIVWDKGRD
FATVRKVLSMPQVNIVKKTEVQTGGFSKESILPKRNSDKLIARKKDWDPKKYGGFDSPTVAYSVLVVAKVEKGKSKKLKSV
KELLGITIMERSSFEKNPIDFLEAKGYKEVKKDLIIKLPKYSLFELENGRKRMLASAGELQKGNELALPSKYVNFLYLASHYEK
LKGSPEDNEQKQLFVEQHKHYLDEIIEQISEFSKRVILADANLDKVLSAYNKHRDKPIREQAENIIHLFTLTNLGAPAAFKYFD
TTIDRKRYTSTKEVLDATLIHQSITGLYETRIDLSQLGGDSGGSPKKKRKV*

ABE7.10 (ecTadA(wt)–linker(32 aa)–ecTadA*(7.10)–linker(32 aa)–Cas9 nickase–NLS):
MSEVEFSHEYWMRHALTLAKRAWDEREVPVGAVLVHNNRVIGEGWNRPIGRHDPTAHAEIMALRQGGLVMQNYRLIDAT
LYVTLEPCVMCAGAMIHSRIGRVVFGARDAKTGAAGSLMDVLHHPGMNHRVEITEGILADECAALLSDFFRMRRQEIKAQK
KAQSSTDSGGSSGGSSGSETPGTSESATPESSGGSSGGSSEVEFSHEYWMRHALTLAKRARDEREVPVGAVLVLNNRVI
GEGWNRAIGLHDPTAHAEIMALRQGGLVMQNYRLIDATLYVTFEPCVMCAGAMIHSRIGRVVFGVRNAKTGAAGSLMDVL
HYPGMNHRVEITEGILADECAALLCYFFRMPRQVFNAQKKAQSSTDSGGSSGGSSGSETPGTSESATPESSGGSSGGSD
KKYSIGLAIGTNSVGWAVITDEYKVPSKKFKVLGNTDRHSIKKNLIGALLFDSGETAEATRLKRTARRRYTRRKNRICYLQEI
FSNEMAKVDDSFFHRLEESFLVEEDKKHERHPIFGNIVDEVAYHEKYPTIYHLRKKLVDSTDKADLRLIYLALAHMIKFRGHF
LIEGDLNPDNSDVDKLFIQLVQTYNQLFEENPINASGVDAKAILSARLSKSRRLENLIAQLPGEKKNGLFGNLIALSLGLTPNF
KSNFDLAEDAKLQLSKDTYDDDLDNLLAQIGDQYADLFLAAKNLSDAILLSDILRVNTEITKAPLSASMIKRYDEHHQDLTLLK
ALVRQQLPEKYKEIFFDQSKNGYAGYIDGGASQEEFYKFIKPILEKMDGTEELLVKLNREDLLRKQRTFDNGSIPHQIHLGEL
HAILRRQEDFYPFLKDNREKIEKILTFRIPYYVGPLARGNSRFAWMTRKSEETITPWNFEEVVDKGASAQSFIERMTNFDKN
LPNEKVLPKHSLLYEYFTVYNELTKVKYVTEGMRKPAFLSGEQKKAIVDLLFKTNRKVTVKQLKEDYFKKIECFDSVEISGVE
DRFNASLGTYHDLLKIIKDKDFLDNEENEDILEDIVLTLTLFEDREMIEERLKTYAHLFDDKVMKQLKRRRYTGWGRLSRKLI
NGIRDKQSGKTILDFLKSDGFANRNFMQLIHDDSLTFKEDIQKAQVSGQGDSLHEHIANLAGSPAIKKGILQTVKVVDELVKV
MGRHKPENIVIEMARENQTTQKGQKNSRERMKRIEEGIKELGSQILKEHPVENTQLQNEKLYLYYLQNGRDMYVDQELDIN
RLSDYDVDHIVPQSFLKDDSIDNKVLTRSDKNRGKSDNVPSEEVVKKMKNYWRQLLNAKLITQRKFDNLTKAERGGLSELD
KAGFIKRQLVETRQITKHVAQILDSRMNTKYDENDKLIREVKVITLKSKLVSDFRKDFQFYKVREINNYHHAHDAYLNAVVGT
ALIKKYPKLESEFVYGDYKVYDVRKMIAKSEQEIGKATAKYFFYSNIMNFFKTEITLANGEIRKRPLIETNGETGEIVWDKGRD
FATVRKVLSMPQVNIVKKTEVQTGGFSKESILPKRNSDKLIARKKDWDPKKYGGFDSPTVAYSVLVVAKVEKGKSKKLKSV
KELLGITIMERSSFEKNPIDFLEAKGYKEVKKDLIIKLPKYSLFELENGRKRMLASAGELQKGNELALPSKYVNFLYLASHYEK
LKGSPEDNEQKQLFVEQHKHYLDEIIEQISEFSKRVILADANLDKVLSAYNKHRDKPIREQAENIIHLFTLTNLGAPAAFKYFD
TTIDRKRYTSTKEVLDATLIHQSITGLYETRIDLSQLGGDSGGSPKKKRKV*

Supplementary Note 1. Matlab script for base calling.

```
function basecall(WTnuc, directory)
%cycle through fastq files for different samples
cd directory
files=dir('*.fastq');
for d=1:2
    filename=files(d).name;
    %read fastq file
    [header,seqs,qscore] = fastqread(filename);
    seqsLength = length(seqs);          % number of sequences
    seqsFile = strrep(filename,'.fastq','');      % trims off .fastq
    %create a directory with the same name as fastq file
    if exist(seqsFile,'dir');
        error('Directory already exists. Please rename or move it before moving on.');
    end
    mkdir(seqsFile);                    % make directory
    wtLength = length(WTnuc);              % length of wildtype sequence
    %% aligning back to the wildtype nucleotide sequence
    %
    % AlN is a matrix of the nucleotide alignment
    window=1:wtLength;
    sBLength = length(seqs);         % number of sequences
    % counts number of skips
    nSkips = 0;
    ALN=repmat(' ',[sBLength wtLength]);
    % iterate through each sequencing read
    for i = 1:sBLength
        %If you only have forward read fastq files leave as is
        %If you have R1 foward and R2 is reverse fastq files uncomment the
        %next four lines of code and the subsequent end statement
%         if mod(d,2)==0;
%             reverse = seqrcomplement(seqs{i});
%             [score,alignment,start] = swalign(reverse,WTnuc,'Alphabet','NT');
%         else

        [score,alignment,start] = swalign(seqs{i},WTnuc,'Alphabet','NT');
%         end

        % length of the sequencing read
        len = length(alignment(3,:));
        % if there is a gap in the alignment , skip = 1 and we will
        % throw away the entire read
        skip = 0;
        for j = 1:len
            if (alignment(3,j) == '-' || alignment(1,j) == '-')
                skip = 1;
                break;
            end
            %in addition if the qscore for any given base in the read is
            %below 31 the nucleotide is turned into an N (fastq qscores that are not
letters)
            if isletter(qscore{i}(start(1)+j-1))
            else
                alignment(1,j) = 'N';
            end

        end
        if skip == 0 && len>10
            ALN(i, start(2):(start(2)+length(alignment)-1))=alignment(1,:);
        end
    end
    % with the alignment matrices we can simply tally up the occurrences of
```

```
    % each nucleotide at each column in the alignment these
    % tallies ignore bases annotated as N
    % due to low qscores
    TallyNTD=zeros(5,wtLength);
    FreqNTD=zeros(4,wtLength);
    SUM=zeros(1,wtLength);
    for i=1:wtLength

TallyNTD(:,i)=[sum(ALN(:,i)=='A'),sum(ALN(:,i)=='C'),sum(ALN(:,i)=='G'),sum(ALN(:,i)=='T'
),sum(ALN(:,i)=='N')];
    end

    for i=1:wtLength
        FreqNTD(:,i)=100*TallyNTD(1:4,i)/sum(TallyNTD(1:4,i));
    end
    for i=1:wtLength
        SUM(:,i)=sum(TallyNTD(1:4,i));
    end

    % we then save these tally matrices in the respective folder for
    % further processing

    save(strcat(seqsFile, '/TallyNTD'), 'TallyNTD');
    dlmwrite(strcat(seqsFile, '/TallyNTD.csv'), TallyNTD, 'precision', '%.3f', 'newline',
'pc');
    save(strcat(seqsFile, '/FreqNTD'), 'FreqNTD');
    dlmwrite(strcat(seqsFile, '/FreqNTD.csv'), FreqNTD, 'precision', '%.3f', 'newline',
'pc');
    fid = fopen('FrequencySummary.csv', 'a');
    fprintf(fid, '\n \n');
    fprintf(fid, filename);
    fprintf(fid, '\n \n');
    dlmwrite('FrequencySummary.csv', FreqNTD, 'precision', '%.3f', 'newline', 'pc', '-
append');
    dlmwrite('FrequencySummary.csv', SUM, 'precision', '%.3f', 'newline', 'pc', '-
append');
end


% set up queue of basecalling runs

% change directory to folder of fastq files for a given target site
cd('/Users/michaelpacker/Documents/MATLAB/BaseCallingWithSummary')
cd PUTFOLDERNAMEHERE
% call upon the basecall program
basecall(PUTWTSEQUENCEHERE)
% and repeat
cd('/Users/michaelpacker/Documents/MATLAB/BaseCallingWithSummary')
cd PUTFOLDERNAMEHERE
basecall(PUTWTSEQUENCEHERE)
% and repeat…
```

Supplementary Note 2. Matlab script for indel analysis.

```
%WTnuc='CGGTGGGAGGTCTATATAAGCAGAGCTGGTTTAGTGAACCGTCAGATCCGCTAGAGATCCGCGGCCGCTAATACGACTCAC
CCTAGGGAGAGCCGCCACCGTGGTGAGCAAGGGCGAGGAGCTGTTCACCGGGGTGGTGCCCATCCTGGTCGAGCTGGACGGCGACGTAA
ACGGCCACAAGTTCAGCGTGTCCGGCGAG';
%cycle through fastq files for different samples
files=dir('*.fastq');
indelstart=55;
width=30;
flank=10;

for d=1:2
    filename=files(d).name;
    %read fastq file
    [header,seqs,qscore] = fastqread(filename);
    seqsLength = length(seqs);          % number of sequences
    seqsFile = strcat(strrep(filename,'.fastq',''),'_INDELS');      % trims off .fastq
    %create a directory with the same name as fastq file+_INDELS
    if exist(seqsFile,'dir');
        error('Directory already exists. Please rename or move it before moving on.');
    end
    mkdir(seqsFile);                    % make directory
    wtLength = length(WTnuc);           % length of wildtype sequence
    sBLength = length(seqs);            % number of sequences

    % initialize counters and cell arrays
    nSkips = 0;
    notINDEL=0;
    ins={};
    dels={};
    NumIns=0;
    NumDels=0;
    % iterate through each sequencing read
    for i = 1:sBLength
    %search for 10BP sequences that should flank both sides of the "INDEL WINDOW"
        windowstart=strfind(seqs{i},WTnuc(indelstart-flank:indelstart));
        windowend=strfind(seqs{i},WTnuc(indelstart+width:indelstart+width+flank));
        %if these flanks are found and more than half of base calls
        %are above Q31 THEN proceed OTHERWISE save as a skip
        if length(windowstart)==1 && length(windowend)==1 &&
(sum(isletter(qscore{i}))/length(qscore{i}))>=0.5
            %if the sequence length matches the INDEL window length save as
            %not INDEL
            if windowend-windowstart==width+flank
                notINDEL=notINDEL+1;
            %if the sequence is ONE or more baseslonger than the INDEL
            %window length save as an Insertion
            elseif windowend-windowstart>=width+flank+1
                NumIns=NumIns+1;
                ins{NumIns}=seqs{i};
            %if the sequence is ONE or more bases shorter than the INDEL
            %window length save as a Deletion
            elseif windowend-windowstart<=width+flank-1
                NumDels=NumDels+1;
                dels{NumDels}=seqs{i};
            end
        %keep track of skipped sequences that do not posess matching flank
        %sequences and do not pass quality cutoff
        else
            nSkips=nSkips+1;

        end
```

```
    end
    INDELrate=(NumIns+NumDels)/(NumIns+NumDels+notINDEL)*100.;
    FID = fopen('INDELSummary.csv', 'a');
    fprintf(FID, '\n \n');
    fprintf(FID, filename);
    fprintf(FID, '\n');
    fprintf(FID, num2str(INDELrate));

    fid=fopen(strcat(seqsFile, '/summary.txt'), 'wt');
    fprintf(fid, 'Skipped reads %i\n not INDEL %i\n Insertions %i\n Deletions %i\n INDEL
percent %e\n', [nSkips, notINDEL, NumIns, NumDels,INDELrate]);
    fclose(fid);
    save(strcat(seqsFile, '/nSkips'), 'nSkips');
    save(strcat(seqsFile, '/notINDEL'), 'notINDEL');
    save(strcat(seqsFile, '/NumIns'), 'NumIns');
    save(strcat(seqsFile, '/NumDels'), 'NumDels');
    save(strcat(seqsFile, '/INDELrate'), 'INDELrate');
    save(strcat(seqsFile, '/dels'), 'dels');
    C = dels;
    fid = fopen(strcat(seqsFile, '/dels.txt'), 'wt');
    fprintf(fid, '"%s"\n', C{:});
    fclose(fid);
    save(strcat(seqsFile, '/ins'), 'ins');
    C = ins;
    fid = fopen(strcat(seqsFile, '/ins.txt'), 'wt');
    fprintf(fid, '"%s"\n', C{:});
    fclose(fid);
```

Supplementary Note 3. Python script for analysis of *HBG1* and *HBG2* base editing and indels.

```
%matplotlib inline
import numpy as np
import scipy as sp
import matplotlib as mpl
import matplotlib.cm as cm
import matplotlib.pyplot as plt
import pandas as pd
pd.set_option('display.width', 500)
pd.set_option('display.max_columns', 100)
pd.set_option('display.notebook_repr_html', True)
import seaborn as sns
sns.set_style("whitegrid")
sns.set_context("poster")
import requests
import time
from bs4 import BeautifulSoup
import regex
import re
import os
from Bio import SeqIO
import Bio
from Bio import motifs
from Bio import pairwise2
from Bio.pairwise2 import format_alignment
from Bio.Alphabet import IUPAC
from sklearn import preprocessing

basecall analysis with 50% Q31 cutoff on protospacer region (as defined by flanks)
#includes a check for match with two HBG1 SNPs
#inputs:
#directory, working directory folder containing all fastq files
#site, genomic site name as it appears in the fastq filenames
#orientation, 'FWD' if you want output in the same direction as the sequencing read or
'REV' if you want reverse complement output,
#flank1, sequence that is used to define the 5' end of protospacer in the sequencing read
direction,
#flank2, sequence that is used to define the 5' end of protospacer in the sequencing read
direction,
#width, expected bp length of basecalling window
#
#outputs:
#'_counts.csv', all base editing product sequences with corresponding number of
occurences
#'_rawsummary.csv', summarizes base call counts for all samples
#'_normalizedsummary.csv', summarizes base call percentages for all samples
def basecallhbg1(directory, site, orientation, flank1, flank2, width):
    indir=directory
    outdir=directory
    filenames=os.listdir(indir)
    for i in range(len(filenames)):
        seqs={}
        if (filenames[i][-5:]=='fastq') and (site in filenames[i]):
            for record in SeqIO.parse(indir+filenames[i], "fastq") :
                recordqual=[x>31 for x in record.letter_annotations['phred_quality']]
                #only process reads that have more than half of basecalls >Q31 and
contain two HBG1 specific SNPs at 3' end of read
                if (record.seq.find('GTTTTTCTCTAATTTATTCTTCCCTTTAGCTAGTTTC')>0) and
(float(sum(recordqual))/float(len(recordqual))>=.5):
                    recordseq="".join([y if x else 'N' for (x,y) in zip(recordqual,
record.seq)])
```

```
                    recordseq="".join([y if x else 'N' for (x,y) in zip(recordqual,
record.seq)])
                    #split prior to spacer window
                    split1=recordseq.split(flank1)
                    if len(split1)==2:
                        #take second item in first split
                        #split again at the sequence right after the protospacer and take
first item
                        split2=split1[1].split(flank2)[0]
                        #keep only entries with exact width
                        if (len(split2)==width):
                            if orientation=='FWD':
                                seqs[record.id]=split2
                            elif orientation=='REV':
                                seqs[record.id]=Bio.Seq.reverse_complement(split2)
            frame=pd.DataFrame({'Spacer':seqs.values()}, index=seqs.keys())
            Motif=motifs.create(frame.Spacer.values, alphabet=IUPAC.IUPACAmbiguousDNA())
            raw=pd.DataFrame(Motif.counts, index=[str(s+1) for s in
range(width)])[['A','C','G','T','N']].transpose()
            normalized=pd.DataFrame(Motif.counts, index=[str(s+1) for s in
range(width)])[['A','C','G','T']].transpose()
            normalized=normalized/normalized.sum(axis=0)*100.
            normalized=normalized.round(2)
            Counts=pd.DataFrame(seqs.items(), columns=['ID','Window'])
            Counts=Counts[['N' not in x for x in Counts.Window]]
            Counts=Counts.groupby('Window').count().sort_values('ID', ascending=False)
            Counts.to_csv(outdir+filenames[i].strip('.fastq')+'_hbg1.csv')
            fd=open(directory+site+'_normalizedsummary_hbg1.csv','a')
            fd.write('\n'+filenames[i]+'\n')
            normalized.to_csv(fd)
            fd.close()
            fd=open(directory+site+'_rawsummary_hbg1.csv','a')
            fd.write('\n'+filenames[i]+'\n')
            raw.to_csv(fd)
            fd.close()
    return


#basecall analysis with 50% Q31 cutoff on protospacer region (as defined by flanks)
#includes a check for match with two HBG2 SNPs
#inputs:
#directory, working directory folder containing all fastq files
#site, genomic site name as it appears in the fastq filenames
#orientation, 'FWD' if you want output in the same direction as the sequencing read or
'REV' if you want reverse complement output,
#flank1, sequence that is used to define the 5' end of protospacer in the sequencing read
direction,
#flank2, sequence that is used to define the 5' end of protospacer in the sequencing read
direction,
#width, expected bp length of basecalling window
#
#outputs:
#'_counts.csv', all base editing product sequences with corresponding number of
occurences
#'_rawsummary.csv', summarizes base call counts for all samples
#'_normalizedsummary.csv', summarizes base call percentages for all samples
def basecallhbg2(directory, site, orientation, flank1, flank2, width):
    indir=directory
    outdir=directory
    filenames=os.listdir(indir)
    for i in range(len(filenames)):
        seqs={}
        if (filenames[i][-5:]=='fastq') and (site in filenames[i]):
```

```
            for record in SeqIO.parse(indir+filenames[i], "fastq") :
                recordqual=[x>31 for x in record.letter_annotations['phred_quality']]
                #only process reads that have more than half of basecalls >Q31 and
contain two HBG2 specific SNPs at 3' end of read
                if (record.seq.find('ATTTTTCTCTAATTTATTCTTCCCTTTAGCTAGTTTT')>0) and
(float(sum(recordqual))/float(len(recordqual))>=.5):
                    recordseq="".join([y if x else 'N' for (x,y) in zip(recordqual,
record.seq)])
                    #split prior to spacer window
                    split1=recordseq.split(flank1)
                    if len(split1)==2:
                        #take second item in first split
                        #split again at the sequence right after the protospacer and take
first item
                        split2=split1[1].split(flank2)[0]
                        #keep only entries with exact width
                        if (len(split2)==width):
                            if orientation=='FWD':
                                seqs[record.id]=split2
                            elif orientation=='REV':
                                seqs[record.id]=Bio.Seq.reverse_complement(split2)
            frame=pd.DataFrame({'Spacer':seqs.values()}, index=seqs.keys())
            Motif=motifs.create(frame.Spacer.values, alphabet=IUPAC.IUPACAmbiguousDNA())
            raw=pd.DataFrame(Motif.counts, index=[str(s+1) for s in
range(width)])[['A','C','G','T','N']].transpose()
            normalized=pd.DataFrame(Motif.counts, index=[str(s+1) for s in
range(width)])[['A','C','G','T']].transpose()
            normalized=normalized/normalized.sum(axis=0)*100.
            normalized=normalized.round(2)
            Counts=pd.DataFrame(seqs.items(), columns=['ID','Window'])
            Counts=Counts[['N' not in x for x in Counts.Window]]
            Counts=Counts.groupby('Window').count().sort_values('ID', ascending=False)
            Counts.to_csv(outdir+filenames[i].strip('.fastq')+'_hbg2.csv')
            fd=open(directory+site+'_normalizedsummary_hbg2.csv','a')
            fd.write('\n'+filenames[i]+'\n')
            normalized.to_csv(fd)
            fd.close()
            fd=open(directory+site+'_rawsummary_hbg2.csv','a')
            fd.write('\n'+filenames[i]+'\n')
            raw.to_csv(fd)
            fd.close()
    return


#indel analysis
#includes a check for match with two HBG1 SNPs
#inputs:
#directory, working directory folder containing all fastq files
#site, genomic site name as it appears in the fastq filenames
#orientation, 'FWD' if you want output in the same direction as the sequencing read or
'REV' if you want reverse complement output,
#flank1, sequence that is used to define the 5' end of protospacer in the sequencing read
direction,
#flank2, sequence that is used to define the 5' end of protospacer in the sequencing read
direction,
#width, expected bp length of basecalling window
#ouputs:
#"_Insertions_hbg1.csv", sequences of all insertion reads
#"_deletions_hbg1.csv", sequences of all deletion reads
#'indelsummary_hbg1.csv', contains all indel stats for all fastq files
def indelshbg1(directory, site, flank1, flank2, width):
    indir=directory
    outdir=directory
```

```
    filenames=os.listdir(indir)
    for i in range(len(filenames)):
        seqs={}
        if (filenames[i][-5:]=='fastq') and (site in filenames[i]):
            skips=0
            ins=0
            insertions=[]
            dels=0
            deletions=[]
            notindel=0
            for record in SeqIO.parse(indir+filenames[i], "fastq") :
                recordqual=[x>31 for x in record.letter_annotations['phred_quality']]
                #only process reads that have more than half of basecalls >Q31 and
contain two HBG1 specific SNPs at 3' end of read
                if (record.seq.find('GTTTTTCTCTAATTTATTCTTCCCTTTAGCTAGTTTC')>0) and
(float(sum(recordqual))/float(len(recordqual))>=.5):
                    #split prior to indel window
                    split1=record.seq.split(flank1)
                    if len(split1)==2:
                        #take second item in first split
                        #split again at the sequence right after the indel window
                        if len(split1[1].split(flank2))==2:
                            split2=split1[1].split(flank2)[0]
                        #if INDEL window is +1 add to Insertions
                            if (len(split2)>=width+1):
                                ins=ins+1
                                insertions.append(split2)
                        #if INDEL window is -1 add to Deletions
                            if (len(split2)<=width-1):
                                dels=dels+1
                                deletions.append(split2)
                            if len(split2)==width:
                                notindel=notindel+1
                        else:
                            skips=skips+1
                    else:
                        skips=skips+1
                else:
                    skips=skips+1
            fd=open(directory+'indelsummary_hbg1.csv','a')
            fd.write('\n'+filenames[i]+'\n')
            fd.write('skipped reads: '+str(skips)+'\n')
            fd.write('insertions: '+str(ins)+'\n')
            fd.write('deletions: '+str(dels)+'\n')
            fd.write('notindels: '+str(notindel)+'\n')
            fd.write('indel rate:
'+str(float(ins+dels)/float(ins+dels+notindel)*100.)+'%'+'\n')
            fd.close()
            pd.DataFrame(insertions).to_csv(directory+filenames[i]+'Insertions_hbg1.csv')
            pd.DataFrame(deletions).to_csv(directory+filenames[i]+'Deletions_hbg1.csv')
    return



#indel analysis
#includes a check for match with two HBG2 SNPs
#inputs:
#directory, working directory folder containing all fastq files
#site, genomic site name as it appears in the fastq filenames
#orientation, 'FWD' if you want output in the same direction as the sequencing read or
'REV' if you want reverse complement output,
```

```
#flank1, sequence that is used to define the 5' end of protospacer in the sequencing read
direction,
#flank2, sequence that is used to define the 5' end of protospacer in the sequencing read
direction,
#width, expected bp length of basecalling window
#ouputs:
#"_Insertions_hbg2.csv", sequences of all insertion reads
#"_deletions_hbg2.csv", sequences of all deletion reads
#'indelsummary_hbg2.csv', contains all indel stats for all fastq files
def indelshbg2(directory, site, flank1, flank2, width):
    indir=directory
    outdir=directory
    filenames=os.listdir(indir)
    for i in range(len(filenames)):
        seqs={}
        if (filenames[i][-5:]=='fastq') and (site in filenames[i]):
            skips=0
            ins=0
            insertions=[]
            dels=0
            deletions=[]
            notindel=0
            for record in SeqIO.parse(indir+filenames[i], "fastq") :
                recordqual=[x>31 for x in record.letter_annotations['phred_quality']]
                #only process reads that have more than half of basecalls >Q31 and
contain two HBG2 specific SNPs at 3' end of read
                if (record.seq.find('ATTTTTCTCTAATTTATTCTTCCCTTTAGCTAGTTTT')>0) and
(float(sum(recordqual))/float(len(recordqual))>=.5):
                    #split prior to indel window
                    split1=record.seq.split(flank1)
                    if len(split1)==2:
                        #take second item in first split
                        #split again at the sequence right after the indel window
                        if len(split1[1].split(flank2))==2:
                            split2=split1[1].split(flank2)[0]
                        #if INDEL window is +1 add to Insertions
                            if (len(split2)>=width+1):
                                ins=ins+1
                                insertions.append(split2)
                        #if INDEL window is -1 add to Deletions
                            if (len(split2)<=width-1):
                                dels=dels+1
                                deletions.append(split2)
                            if len(split2)==width:
                                notindel=notindel+1
                        else:
                            skips=skips+1
                    else:
                        skips=skips+1
                else:
                    skips=skips+1
            fd=open(directory+'indelsummary_hbg2.csv','a')
            fd.write('\n'+filenames[i]+'\n')
            fd.write('skipped reads: '+str(skips)+'\n')
            fd.write('insertions: '+str(ins)+'\n')
            fd.write('deletions: '+str(dels)+'\n')
            fd.write('notindels: '+str(notindel)+'\n')
            fd.write('indel rate:
'+str(float(ins+dels)/float(ins+dels+notindel)*100.)+'%'+'\n')
            fd.close()
            pd.DataFrame(insertions).to_csv(directory+filenames[i]+'Insertions_hbg2.csv')
            pd.DataFrame(deletions).to_csv(directory+filenames[i]+'Deletions_hbg2.csv')
    return
```

```
directory1='/Users/michaelpacker/Desktop/Liu_Lab/MiSeqData/y-globin_632/'
basecallhbg1(directory1, '632', 'FWD','ATTTGCA','TTAATTTTTT', 43)
basecallhbg2(directory1, '632', 'FWD','ATTTGCA','TTAATTTTTT', 43)
indelshbg1(directory1, '632','ATTTGCA','TTAATTTTTT',43)
indelshbg2(directory1, '632','ATTTGCA','TTAATTTTTT',43)
```

Supplementary Note 4. Python script for analysis of base editing linkage disequilibrium.

```
%matplotlib inline
import numpy as np
import scipy as sp
import matplotlib as mpl
import matplotlib.cm as cm
import matplotlib.pyplot as plt
import pandas as pd
pd.set_option('display.width', 500)
pd.set_option('display.max_columns', 100)
pd.set_option('display.notebook_repr_html', True)
import seaborn as sns
sns.set_style("whitegrid")
sns.set_context("poster")
import requests
import time
from bs4 import BeautifulSoup
import regex
import re
import os
from Bio import SeqIO
import Bio
from Bio import motifs

#ABE processivity analysis
#inputs:
#directory, working directory folder containing all fastq files for a single ABE
#site, genomic site name as it appears in the fastq filenames
#orientation, 'FWD' if you want output in the same direction as the sequencing read or
'REV' if you want reverse complement output,
#flank1, sequence that is used to define the 5' end of protospacer in the sequencing read
direction,
#flank2, sequence that is used to define the 5' end of protospacer in the sequencing read
direction,
#primaryposition, site of primary target A in protospacer with the position furthest from
the PAM as 0
#secondaryposition, site of secondary target A in protospacer with the position furthest
from the PAM as 0
#outputs:
#'_counts.csv', all base editing product sequences with corresponding number of
occurrences
#'_RawMotifs.csv', unnormalized nucleotide counts at all 20 positions of protospacer as
well as counts conditional on the identity of the primary target position
#'_NormalizedMotifs.csv' normalized nucleotide frequencies at all 20 positions of
protospacer as well as frequencies conditional on the identity of the primary target
position
#'_probability.csv', summary containing editing probabilities at both positions as well
as observed probability of double editing
def processivity(directory, site, orientation, flank1, flank2, primaryposition,
secondaryposition):
    indir=directory
    outdir=directory
    filenames=os.listdir(indir)
    probabilities=pd.DataFrame({'P1':[],'P2':[],'P21':[], 'P2P1':[]})
    for i in range(len(filenames)):
        seqs={}
        if (filenames[i][-5:]=='fastq') and (site in filenames[i]):
            for record in SeqIO.parse(indir+filenames[i], "fastq") :
                #split prior to spacer window
                split1=record.seq.tostring().split(flank1)
                if len(split1)==2:
                    #take second item in first split
```

```
                    #split again at the sequence right after the protospacer and take
first item
                    split2=split1[1].split(flank2)[0]
                    #keep only 20 basepair long protospacers
                    if (len(split2)==20) & (split2.find('N')==-1):
                        if orientation=='FWD':
                            seqs[record.id]=split2
                        elif orientation=='REV':
                            seqs[record.id]=Bio.Seq.reverse_complement(split2)
            frame=pd.DataFrame({'Spacer':seqs.values(),
'Primary_Position':[x[primaryposition] for x in seqs.values()]}, index=seqs.keys())
            MotifAll=motifs.create(frame.Spacer.values)
            #in the event that no reads have a given base call at the primary position we
will save a dummy motif for a polyA sequence
            if len(frame[frame.Primary_Position=='A'])>0:
                MotifA=motifs.create(frame[frame.Primary_Position=='A'].Spacer.values)
            else:
                MotifA=motifs.create(['A'*20])
            if len(frame[frame.Primary_Position=='C'])>0:
                MotifC=motifs.create(frame[frame.Primary_Position=='C'].Spacer.values)
            else:
                MotifC=motifs.create(['A'*20])
            if len(frame[frame.Primary_Position=='G'])>0:
                MotifG=motifs.create(frame[frame.Primary_Position=='G'].Spacer.values)
            else:
                MotifG=motifs.create(['A'*20])
            if len(frame[frame.Primary_Position=='T'])>0:
                MotifT=motifs.create(frame[frame.Primary_Position=='T'].Spacer.values)
            else:
                MotifT=motifs.create(['A'*20])
            #save motifs both raw and normalized conditional on the primary position
being each of the four bases
            a=pd.DataFrame(MotifA.counts, index=['A'+str(s) for s in range(20)])
            A=pd.DataFrame(MotifA.counts.normalize(),index=['A'+str(s) for s in
range(20)])
            c=pd.DataFrame(MotifC.counts, index=['C'+str(s) for s in range(20)])
            C=pd.DataFrame(MotifC.counts.normalize(), index=['C'+str(s) for s in
range(20)])
            g=pd.DataFrame(MotifG.counts, index=['G'+str(s) for s in range(20)])
            G=pd.DataFrame(MotifG.counts.normalize(), index=['G'+str(s) for s in
range(20)])
            t=pd.DataFrame(MotifT.counts, index=['T'+str(s) for s in range(20)])
            T=pd.DataFrame(MotifT.counts.normalize(), index=['T'+str(s) for s in
range(20)])
            #save motifs both raw and normalized for all base editing products
            All=pd.DataFrame(MotifAll.counts, index=['All'+str(s) for s in range(20)])
            ALL=pd.DataFrame(MotifAll.counts.normalize(),index=['All'+str(s) for s in
range(20)])
            #append all motifs and export, indices contain protospacer position as well
as an identifier for the primary position

All.append(a).append(c).append(g).append(t).to_csv(outdir+filenames[i].strip('.fastq')+'R
awMotifs.csv')

ALL.append(A).append(C).append(G).append(T).to_csv(outdir+filenames[i].strip('.fastq')+'N
ormalizedMotifs.csv')
            #save the base editing product sequences and corresponding number of
occurrences
            Counts=pd.DataFrame(seqs.items(),
columns=['ID','Window']).groupby('Window').count().sort_values('ID', ascending=False)
            Counts.to_csv(outdir+filenames[i].strip('.fastq')+'.csv')
            #evaluate editing probability at both primary and secondary positions
            P1=ALL['G'].iloc[primaryposition]
```

```
            P2=ALL['G'].iloc[secondaryposition]
            #evaluate observed probability of joint editing as P(2|1)*P(1)
            P21=G['G'].iloc[secondaryposition]*P1
            #evaluate expected probability of joint editing given statistical
independence as P(1)*P(2)
            P2P1=P1*P2
            #export probabilities

probabilities=probabilities.append(pd.DataFrame({'P1':[P1],'P2':[P2],'P21':[P21],
'P2P1':[P2P1]}, index=[site]))
            probabilities.to_csv(outdir+site+'_probabilities.csv')
    return


#ABE processivity analysis, for when flank1 needs to be short, we instead split on flank2
first and then find flank1
#program is otherwise identical to processivity
def processivity2(directory, site, orientation, flank1, flank2, primaryposition,
secondaryposition):
    indir=directory
    outdir=directory
    filenames=os.listdir(indir)
    probabilities=pd.DataFrame({'P1':[],'P2':[],'P21':[], 'P2P1':[]})
    for i in range(len(filenames)):
        seqs={}
        if (filenames[i][-5:]=='fastq') and (site in filenames[i]):
            for record in SeqIO.parse(indir+filenames[i], "fastq") :
                #split prior to spacer window
                split1=record.seq.tostring().split(flank2)
                if len(split1)==2:
                    #take second item in first split
                    #split again at the sequence right after the protospacer and take
first item
                    if len(split1[0].split(flank1))==2:
                        split2=split1[0].split(flank1)[1]
                    #keep only 20 basepair long protospacers
                        if (len(split2)==20) & (split2.find('N')==-1):
                            if orientation=='FWD':
                                seqs[record.id]=split2
                            elif orientation=='REV':
                                seqs[record.id]=Bio.Seq.reverse_complement(split2)
            frame=pd.DataFrame({'Spacer':seqs.values(),
'Primary_Position':[x[primaryposition] for x in seqs.values()]}, index=seqs.keys())
            MotifAll=motifs.create(frame.Spacer.values)
            if len(frame[frame.Primary_Position=='A'])>0:
                MotifA=motifs.create(frame[frame.Primary_Position=='A'].Spacer.values)
            else:
                MotifA=motifs.create(['A'*20])
            if len(frame[frame.Primary_Position=='C'])>0:
                MotifC=motifs.create(frame[frame.Primary_Position=='C'].Spacer.values)
            else:
                MotifC=motifs.create(['A'*20])
            if len(frame[frame.Primary_Position=='G'])>0:
                MotifG=motifs.create(frame[frame.Primary_Position=='G'].Spacer.values)
            else:
                MotifG=motifs.create(['A'*20])
            if len(frame[frame.Primary_Position=='T'])>0:
                MotifT=motifs.create(frame[frame.Primary_Position=='T'].Spacer.values)
            else:
                MotifT=motifs.create(['A'*20])
            a=pd.DataFrame(MotifA.counts, index=['A'+str(s) for s in range(20)])
            A=pd.DataFrame(MotifA.counts.normalize(),index=['A'+str(s) for s in
range(20)])
```

```
            c=pd.DataFrame(MotifC.counts, index=['C'+str(s) for s in range(20)])
            C=pd.DataFrame(MotifC.counts.normalize(), index=['C'+str(s) for s in
range(20)])
            g=pd.DataFrame(MotifG.counts, index=['G'+str(s) for s in range(20)])
            G=pd.DataFrame(MotifG.counts.normalize(), index=['G'+str(s) for s in
range(20)])
            t=pd.DataFrame(MotifT.counts, index=['T'+str(s) for s in range(20)])
            T=pd.DataFrame(MotifT.counts.normalize(), index=['T'+str(s) for s in
range(20)])
            All=pd.DataFrame(MotifAll.counts, index=['All'+str(s) for s in range(20)])
            ALL=pd.DataFrame(MotifAll.counts.normalize(),index=['All'+str(s) for s in
range(20)])

All.append(a).append(c).append(g).append(t).to_csv(outdir+filenames[i].strip('.fastq')+'R
awMotifs.csv')

ALL.append(A).append(C).append(G).append(T).to_csv(outdir+filenames[i].strip('.fastq')+'N
ormalizedMotifs.csv')
            Counts=pd.DataFrame(seqs.items(),
columns=['ID','Window']).groupby('Window').count().sort_values('ID', ascending=False)
            Counts.to_csv(outdir+filenames[i].strip('.fastq')+'.csv')
            P1=ALL['G'].iloc[primaryposition]
            P2=ALL['G'].iloc[secondaryposition]
            P21=G['G'].iloc[secondaryposition]*P1
            P2P1=P1*P2

probabilities=probabilities.append(pd.DataFrame({'P1':[P1],'P2':[P2],'P21':[P21],
'P2P1':[P2P1]}, index=[site]))
            probabilities.to_csv(outdir+site+'_probabilities.csv')
    return


directory1='/Users/michaelpacker/Desktop/Liu_Lab/MiSeqData/2017_0824_MSP/144/'
processivity(directory1, '299', 'REV', 'CCGCCCC','CAGTTTC',5-1, 7-1)
processivity(directory1, '310',  'FWD', 'ATCGAAA','AGGATAA', 5-1,8-1)
processivity(directory1, '311',  'FWD', 'ACTCAGA','GGGGTAC', 5-1,8-1)
processivity2(directory1, '314', 'FWD',  'AAGT','TGGGCTTG', 5-1, 8-1)
processivity(directory1, '318', 'REV','GTAACCA','ATGAGTTCA',5-1,7-1)
processivity(directory1, '463',  'FWD', 'GATACAA','GGGT', 5-1, 3-1)
processivity(directory1, '464',  'FWD', 'ACCAGGA','AGGCAAA', 5-1, 6-1)
processivity(directory1, '466',  'FWD', 'ATCTCAT','TGGTTAC', 5-1, 7-1)
processivity(directory1, '467',  'FWD', 'GAGACTG','GGGAATG', 5-1, 6-1)
processivity(directory1, '468',  'FWD', 'AACGACT','TGGTATC', 5-1, 8-1)
processivity(directory1, '469',  'FWD', 'TCATG','AGGAGAC', 5-1, 8-1)
processivity(directory1, '470',  'FWD', 'GACTCAG','CGGGGGT', 5-1, 7-1)
processivity(directory1, '471',  'FWD', 'GCCTCAG','TGGACAA', 5-1, 7-1)
processivity(directory1, '472', 'REV', 'TGGTTTCCCT','CAGATTT', 5-1, 6-1)
processivity(directory1, '501',  'FWD', 'CTGAGAG','GGGAGA', 5-1, 6-1)
processivity2(directory1, '505',  'FWD', 'AGT','GGGTCGCTGAAAA', 5-1, 8-1)
processivity(directory1, '508',  'FWD', 'GGTGAGG','GGGCTTC', 5-1, 7-1)
processivity(directory1, '536', 'REV', 'TTCTCCA','TTGGGGC', 7-1, 3-1)
processivity(directory1, '601',  'FWD', 'CACAGAC','TGGGAGT',5-1, 7-1)
processivity(directory1, '602', 'FWD', 'ACAGACA','GGGAGTG',6-1, 8-1)

#script to combine all sites into one summary file for each ABE
indir=directory1
filenames=os.listdir(indir)
summary=pd.DataFrame({'P1':[],'P2':[],'P21':[], 'P2P1':[]})
for i in range(len(filenames)):
    if 'probabilities' in filenames[i]:
        summary=summary.append(pd.read_csv(indir+filenames[i], index_col=0))
summary.to_csv(indir+'summary.csv')
```

Supplementary References

1 Tsai, S. Q. *et al.* GUIDE-seq enables genome-wide profiling of off-target cleavage by CRISPR-Cas nucleases. *Nature Biotechnology* **33**, 187-197, doi:10.1038/nbt.3117 (2015).

2 Reddy, R., Henning, D., Das, G., Harless, M. & Wright, D. The capped U6 small nuclear RNA is transcribed by RNA polymerase III. *The Journal of biological chemistry* **262**, 75-81 (1987).

3 Kunkel, G. R., Maser, R. L., Calvet, J. P. & Pederson, T. U6 small nuclear RNA is transcribed by RNA polymerase III. *Proceedings of the National Academy of Sciences of the United States of America* **83**, 8575-8579 (1986).

4 Mussolino, C. & Cathomen, T. in *Nature Biotechnology* Vol. 31 208-209 (2013).

5 Komor, A. C., Kim, Y. B., Packer, M. S., Zuris, J. A. & Liu, D. R. Programmable editing of a target base in genomic DNA without double-stranded DNA cleavage. *Nature* **533**, 420-424, doi:10.1038/nature17946 (2016).