Supporting Online Material for

A Simple Cipher Governs DNA Recognition by TAL Effectors

Matthew J. Moscou and Adam J. Bogdanove*

*To whom correspondence should be addressed. E-mail: ajbog@iastate.edu

Published 29 October 2009 on Science Express
DOI: 10.1126/science.1178817

This PDF file includes:

SOM Text
Figs. S1 and S2
Tables S1 and S2
References

Correction: The frequencies in the script are now the ones corrected for the misannotations in the previous version.
A simple cipher governs DNA recognition by TAL effectors

Matthew J. Moscou and Adam J. Bogdanove*

Department of Plant Pathology and Bioinformatics and Computational Biology Program, Iowa State University, Ames, IA 50011 USA

*To whom correspondence should be addressed. E-mail: ajbog@iastate.edu

Supporting Online Material

**Fig. S1.** Evidence that OsHen1 is activated by Tal1c of *Xanthomonas oryzae* pv. oryzaicola strain BLS256. (A) Semi quantitative RT-PCR results showing relative transcript abundance of OsHen1, and an actin gene for reference, in rice leaves 24 hours after inoculation with BLS256 marker exchange mutant M51, M51 carrying the empty cosmid vector (ev), M51 carrying cosmid pIJF92, which contains tal1a, tal1b, and tal1c, and the wild type (WT) strain. (B) Mapping of the single marker exchange mutation in M51 by rescue and end sequencing of a marker-containing *XmaI* fragment. A schematic of the genome region, the coordinates of the rescued fragment, and the coordinates of the BLS256 genomic fragment contained in cosmid pIJF92 are shown.
**Fig. S2.** Single best alignments in all gene promoters in rice (*Oryza sativa* spp. japonica cv. Nipponbare, Osa1, Release 6.0, http://rice.plantbiology.msu.edu/) for five *Xanthomonas oryzae* TAL effectors, scored (y axis) using a weight matrix derived from the RVD-nucleotide association frequencies in Fig. 1B and represented by the nucleotide at position -1. The x-axis shows the distance from the start codon. Arrows point to experimentally identified targets. Promoters were taken as the 1,000 bp upstream of the start codon for all annotated genes. For the AvrXa27 scan, we included the sequence upstream of *Xa27* (GenBank accession AY986492), which is not present in Nipponbare. In the weight matrix, observed RVD-nucleotide association frequencies were weighted at 90% with the remaining 10% distributed equally across all possible associations.
Table S1. Predicted target site features for ten experimentally identified TAL effector-target pairs. RVDs, repeat-variable diresidues; TcS, annotated transcriptional start site; TlS, translational start site. Locations are relative to the 5’ end of the target site.

<table>
<thead>
<tr>
<th>TAL effector</th>
<th>Source</th>
<th>RVDs</th>
<th>Target gene</th>
<th>TcS</th>
<th>TATA-box</th>
<th>TlS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AvrXa27 (1)</td>
<td>Xanthomonas oryzae pv. oryzae PXO99&lt;sup&gt;A&lt;/sup&gt;</td>
<td>17</td>
<td>Xa27 (rice)</td>
<td>27</td>
<td>-7</td>
<td>87</td>
</tr>
<tr>
<td>AvrBs3 (2)</td>
<td>X. campestris pv. vesicatoria</td>
<td>18</td>
<td>Bs3 (pepper)</td>
<td>59</td>
<td>1</td>
<td>123</td>
</tr>
<tr>
<td>AvrBs3 (3)</td>
<td>X. campestris pv. vesicatoria</td>
<td>18</td>
<td>UPA20 (pepper)</td>
<td>72</td>
<td>1</td>
<td>150</td>
</tr>
<tr>
<td>AvrBs3Δrep16 (4, 5)</td>
<td>Modified AvrBs3</td>
<td>14</td>
<td>Bs3-E (pepper)</td>
<td>85</td>
<td>1</td>
<td>136</td>
</tr>
<tr>
<td>AvrBs3Δrep109 (4)</td>
<td>Modified AvrBs3</td>
<td>15</td>
<td>Bs3 (pepper)</td>
<td>59</td>
<td>1</td>
<td>123</td>
</tr>
<tr>
<td>AvrHah1 (6)</td>
<td>X. gardneri</td>
<td>14</td>
<td>Bs3 (pepper)</td>
<td>59</td>
<td>1</td>
<td>121</td>
</tr>
<tr>
<td>PthXo1 (7)</td>
<td>X. oryzae pv. oryzae PXO99&lt;sup&gt;A&lt;/sup&gt;</td>
<td>24</td>
<td>Os8N3 (rice)</td>
<td>79</td>
<td>46</td>
<td>251</td>
</tr>
<tr>
<td>PthXo6 (8)</td>
<td>X. oryzae pv. oryzae PXO99&lt;sup&gt;A&lt;/sup&gt;</td>
<td>23</td>
<td>OsTFX1 (rice)</td>
<td>31</td>
<td>-780</td>
<td>136</td>
</tr>
<tr>
<td>PthXo7(8)</td>
<td>X. oryzae pv. oryzae PXO99&lt;sup&gt;A&lt;/sup&gt;</td>
<td>22</td>
<td>OsTFIIAγ1 (rice)</td>
<td>333</td>
<td>44</td>
<td>469</td>
</tr>
<tr>
<td>Tal1c</td>
<td>X. oryzae pv. oryzicola BLS256</td>
<td>16</td>
<td>OsHEN1 (rice)</td>
<td>10</td>
<td>-265</td>
<td>217</td>
</tr>
</tbody>
</table>

**Table S2.** *Xanthomonas oryzae* TAL effector candidate targets in rice activated during infection. RVDs, repeat-variable diresidues; r, rank out of 58,918 gene models scanned, based on the RVD weight matrix score; TcS, annotated transcriptional start site; n.p., not present; TlS, translational start site. Locations are relative to the 5’ end of the target site. *q* values are for a comparison to mock across five time points up to 96 hrs after inoculation, replicated four times; fold change given is at 96 hours (PLEXdb, accession OS3).

<table>
<thead>
<tr>
<th>Effector</th>
<th>Strain</th>
<th>RVDs</th>
<th>Rice locus</th>
<th>r</th>
<th>TcS</th>
<th>TATA box</th>
<th>TlS</th>
<th><em>q</em></th>
<th>Fold change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tal1c</td>
<td>BLS256</td>
<td>16</td>
<td>OsHen1</td>
<td>1</td>
<td>10</td>
<td>-265</td>
<td>217</td>
<td>0.01</td>
<td>3.3</td>
</tr>
<tr>
<td>Tal2c</td>
<td>BLS256</td>
<td>27</td>
<td>Os03g03034</td>
<td>15</td>
<td>-16</td>
<td>-145</td>
<td>143</td>
<td>0.01</td>
<td>5.2</td>
</tr>
<tr>
<td>Tal2d</td>
<td>BLS256</td>
<td>16</td>
<td>Os04g49194</td>
<td>9</td>
<td>27</td>
<td>n.p.</td>
<td>102</td>
<td>3.9E-07</td>
<td>29.7</td>
</tr>
<tr>
<td>Tal3b</td>
<td>BLS256</td>
<td>18</td>
<td>Os05g27590</td>
<td>42</td>
<td>34</td>
<td>-1</td>
<td>104</td>
<td>3.4E-08</td>
<td>8.5</td>
</tr>
<tr>
<td>Tal4a</td>
<td>BLS256</td>
<td>26</td>
<td>Os03g37840</td>
<td>1</td>
<td>152</td>
<td>221</td>
<td>363</td>
<td>2.2E-04</td>
<td>2.6</td>
</tr>
<tr>
<td>Tal4b</td>
<td>BLS256</td>
<td>14</td>
<td>Os09g32100</td>
<td>72</td>
<td>68</td>
<td>n.p.</td>
<td>271</td>
<td>8.0E-03</td>
<td>3.6</td>
</tr>
<tr>
<td>Tal4c</td>
<td>BLS256</td>
<td>23</td>
<td>Os06g37080</td>
<td>18</td>
<td>31</td>
<td>n.p.</td>
<td>151</td>
<td>2.7E-10</td>
<td>17.1</td>
</tr>
<tr>
<td>Tal6</td>
<td>BLS256</td>
<td>20</td>
<td>Os07g47790</td>
<td>16</td>
<td>-15</td>
<td>-70</td>
<td>93</td>
<td>3.6E-02</td>
<td>21.6</td>
</tr>
<tr>
<td>PthXo1</td>
<td>PXO99A</td>
<td>24</td>
<td>Os8N3</td>
<td>1</td>
<td>79</td>
<td>46</td>
<td>251</td>
<td>1.0E-08</td>
<td>84.2</td>
</tr>
<tr>
<td>PthXo6</td>
<td>PXO99A</td>
<td>23</td>
<td>OsTFX1</td>
<td>2</td>
<td>31</td>
<td>-780</td>
<td>136</td>
<td>3.5E-03</td>
<td>2.8</td>
</tr>
<tr>
<td>PthXo7</td>
<td>PXO99A</td>
<td>22</td>
<td>OsTFIIAγ1</td>
<td>7</td>
<td>333</td>
<td>44</td>
<td>469</td>
<td>1.6E-06</td>
<td>4.5</td>
</tr>
<tr>
<td>Tal9a</td>
<td>PXO99A</td>
<td>20</td>
<td>OsHen1</td>
<td>1</td>
<td>44</td>
<td>-3</td>
<td>93</td>
<td>0.13</td>
<td>8.2</td>
</tr>
<tr>
<td>Tal7a/8a</td>
<td>PXO99A</td>
<td>18</td>
<td>Os01g68740</td>
<td>2</td>
<td>32</td>
<td>-197</td>
<td>102</td>
<td>1.8E-01</td>
<td>1.7</td>
</tr>
<tr>
<td>Tal7b/8b</td>
<td>PXO99A</td>
<td>20</td>
<td>Os01g40290</td>
<td>57</td>
<td>-2</td>
<td>-276</td>
<td>206</td>
<td>1.8E-01</td>
<td>1.7</td>
</tr>
</tbody>
</table>
Script S1. Script for searching a DNA sequence for TAL effector target sites, written in Python v2.5.

#! /usr/bin/python

# Usage: Script_S1.py [options]

# Options:
#   -h, --help
#       show this help message and exit
#   -M MODEL, --model=MODEL
#       Frequency distributions to use in the analysis (2 or 3).
#   -w WEIGHT, --weight=WEIGHT
#       Weight factor for observed versus unobserved frequencies (choose between 0.01-0.99).
#   -f FASTA, --fasta=FASTA
#       FASTA file containing promoter sequence(s).
#   -T TAL, --TAL=TAL
#       File containing TAL(s) diresidue information (format: TAL1 HD NG .. NN).
#   -o OUTPUT, --output=OUTPUT
#       Output file containing scores and additional information.

# import modules
import datetime
import math
from math import *
import optparse
from optparse import OptionParser
import pickle
import sets
import string

# import arguments and options
usage = 'usage: %prog [options]'
parser = OptionParser(usage=usage)
parser.add_option('-M', '--model', dest='model', type='int', default=3,
 help='Frequency distributions to use in the analysis (2 or 3).')
parser.add_option('-w', '--weight', dest='weight', type='float', default=0.9,
 help='Weight factor for observed versus unobserved frequencies (choose between 0.01-0.99).')
parser.add_option('-f', '--fasta', dest='fasta', type='string', default='NA',
 help='FASTA file containing promoter sequence(s).')
parser.add_option('-T', '--TAL', dest='TAL', type='string', default='NA',
 help='File containing TAL(s) diresidue information (format: TAL1 HD NG .. NN).')
parser.add_option('-o', '--output', dest='output', type='string',
default='output.txt', help='Output file containing scores and additional
information.')
(options, args) = parser.parse_args()

if options.model not in [2, 3]:
    parser.error('Only Models 2 and 3 may be selected.')

if options.weight > 0.999 or options.weight < 0.001:
    parser.error('Select weight between 0.01-0.99.')

if options.fasta == 'NA':
    parser.error('FASTA file required.

diresidue_counts = {}

# Diresidue counts are determined by user-selected model
if options.model == 2:
    # known
diresidue_counts['HD'] = [ 6, 45, 0, 1]
diresidue_counts['NI'] = [35, 4, 0, 0]
diresidue_counts['NG'] = [3, 4, 0, 32]
diresidue_counts['NN'] = [7, 3, 8, 1]
diresidue_counts['NS'] = [11, 3, 2, 0]
diresidue_counts['N*'] = [1, 7, 0, 1]
diresidue_counts['HG'] = [0, 2, 0, 3]
diresidue_counts['IG'] = [0, 0, 0, 1]
diresidue_counts['H*'] = [0, 0, 0, 1]

    # unknown
diresidue_counts['ND'] = [1, 1, 1, 1]
diresidue_counts['HA'] = [1, 1, 1, 1]
diresidue_counts['HI'] = [1, 1, 1, 1]
diresidue_counts['HN'] = [1, 1, 1, 1]
diresidue_counts['NA'] = [1, 1, 1, 1]
diresidue_counts['NK'] = [1, 1, 1, 1]
diresidue_counts['S*'] = [1, 1, 1, 1]
diresidue_counts['NH'] = [1, 1, 1, 1]
diresidue_counts['NC'] = [1, 1, 1, 1]
diresidue_counts['YG'] = [1, 1, 1, 1]
diresidue_counts['HH'] = [1, 1, 1, 1]
diresidue_counts['SN'] = [1, 1, 1, 1]
diresidue_counts['SS'] = [1, 1, 1, 1]

elif options.model == 3:
    # known
diresidue_counts['HD'] = [7, 99, 0, 1]
diresidue_counts['NI'] = [58, 6, 0, 0]
diresidue_counts['NG'] = [6, 6, 1, 57]
diresidue_counts['NN'] = [21, 8, 26, 2]
diresidue_counts['NS'] = [20, 6, 4, 0]
diresidue_counts['N*'] = [ 1, 1, 1, 7]
diresidue_counts['HG'] = [ 1, 2, 0, 15]
diresidue_counts['HA'] = [ 1, 4, 1, 0]
diresidue_counts['ND'] = [ 0, 4, 0, 0]
diresidue_counts['NK'] = [ 0, 0, 2, 0]
diresidue_counts['HI'] = [ 0, 1, 0, 0]
diresidue_counts['HN'] = [ 0, 0, 1, 0]
diresidue_counts['NA'] = [ 0, 0, 0, 0]
diresidue_counts['IG'] = [ 0, 0, 0, 1]
diresidue_counts['H*'] = [ 0, 0, 0, 1]

# unknown
diresidue_counts['S*'] = [1, 1, 1, 1]
diresidue_counts['NH'] = [1, 1, 1, 1]
diresidue_counts['YG'] = [1, 1, 1, 1]
diresidue_counts['SN'] = [1, 1, 1, 1]
diresidue_counts['SS'] = [1, 1, 1, 1]
diresidue_counts['NC'] = [1, 1, 1, 1]
diresidue_counts['HH'] = [1, 1, 1, 1]

# Process diresidue counts based on user-defined weight (default=0.9)
diresidue_probability = {}

for diresidue in diresidue_counts.keys():
    diresidue_probability[diresidue] = [float(diresidue_counts[diresidue][0]) / sum(diresidue_counts[diresidue]) * options.weight * 10 + (1 - options.weight) / 4 * 10, float(diresidue_counts[diresidue][1]) / sum(diresidue_counts[diresidue]) * options.weight * 10 + (1 - options.weight) / 4 * 10, float(diresidue_counts[diresidue][2]) / sum(diresidue_counts[diresidue]) * options.weight * 10 + (1 - options.weight) / 4 * 10, float(diresidue_counts[diresidue][3]) / sum(diresidue_counts[diresidue]) * options.weight * 10 + (1 - options.weight) / 4 * 10]

# DNA: List and dictionary
DNA = ['A', 'C', 'G', 'T']
DNA_dict = {'A':0, 'C':1, 'G':2, 'T':3}

# import promoter sequences
promoter_sequences = open(options.fasta, 'r')
gene_promoter = {}
seq_name = ''

for line in promoter_sequences.readlines():
    sline = string.split(line)
    if line[0] == '>':
        seq_name = sline[0][1:]
    elif seq_name not in gene_promoter.keys():
    gene_promoter[seq_name] = string.upper(sline[0])
else:
    gene_promoter[seq_name] += string.upper(sline[0])

promoter_sequences.close()

# import TAL diresidues
diresidues_file = open(options.TAL, 'r')

TAL_diresidues = {}
observed_diresidues = []

for line in diresidues_file.readlines():
    sline = string.split(line)
    TAL_diresidues[sline[0]] = sline[1:]
    observed_diresidues = list(sets.Set(observed_diresidues) |
    sets.Set(sline[1:])))

diresidues_file.close()

# generate null distributions for all TALs
for diresidue in observed_diresidues:
    if diresidue not in diresidue_probability.keys():
        diresidue_probability[diresidue] = [0.25, 0.25, 0.25, 0.25]

# generate sum of all probabilities (to normalize to 1.0)
for diresidue in diresidue_probability.keys():
    diresidue_probability[diresidue].append(float(sum(diresidue_probability[diresidue])))

# processing of gene promoters with position weight matrix (PWM)
# save best hit for each individual promoter
# TAL -> gene -> [minimum score, start position of target site]
TAL_gene_best_hit = {}

output_file = open(options.output, 'w')

output_file.write('TAL' + '	' + 'gene' + '	' + 'score' + '	' + 'position' + '	' + 'preceding-DNA' + '	' + 'target site' + '	' + 'succeeding-DNA' + '
')

for TAL in TAL_diresidues.keys():
    print 'Processing -->', TAL, '@', datetime.datetime.today()

    # initialize data structure for given TAL
    TAL_gene_best_hit[TAL] = {}

    # for all genes in the user-defined FASTA file
    for gene in gene_promoter.keys():
        positional_probabilities = []

        for TAL in TAL_diresidues.keys():
            print 'Processing -->', TAL, '@', datetime.datetime.today()
# scan the promoter for each gene
for sindex in range(len(gene_promoter[gene]) -
    len(TAL_diresidues[TAL])): 
    probs = []
    if len(sets.Set(DNA) |
        sets.Set(gene_promoter[gene][sindex:sindex+len(TAL_diresidues[TAL]))]])) == 4: 
        for tindex in range(len(TAL_diresidues[TAL])): 
            probs.append(-1 * 
                log(diresidue_probability[TAL_diresidues[TAL][tindex]][DNA_dict[gene_promoter[gene][sindex + tindex]]] /
                    diresidue_probability[TAL_diresidues[TAL][tindex]][4]))
    positional_probabilities.append(sum(probs))
    else:
        positional_probabilities.append(100)

# storing the minimum score for a given promoter
TAL_gene_best_hit[TAL][gene] = [min(positional_probabilities),
    positional_probabilities.index(min(positional_probabilities))]

if TAL_gene_best_hit[TAL][gene][1] > 10: 
    pre_region =
        gene_promoter[gene][(TAL_gene_best_hit[TAL][gene][1] -
        11):TAL_gene_best_hit[TAL][gene][1]]
    else:
        pre_region =
            gene_promoter[:TAL_gene_best_hit[TAL][gene][1]]

target_region =
    gene_promoter[gene][TAL_gene_best_hit[TAL][gene][1]:
        (TAL_gene_best_hit[TAL][gene][1] +
            len(TAL_diresidues[TAL]))]

post_region =
    gene_promoter[gene][(TAL_gene_best_hit[TAL][gene][1] +
        len(TAL_diresidues[TAL])):
            TAL_gene_best_hit[TAL][gene][1] +
                len(TAL_diresidues[TAL]) + 10]

    output_file.write(TAL + '\t' + gene + '\t' +
        str(TAL_gene_best_hit[TAL][gene][0]) + '\t' +
        str(TAL_gene_best_hit[TAL][gene][1]) + '\t' + pre_region + '\t' + target_region + '\t' + post_region + '\n')

output_file.close()