Although this code has been processed successfully on a computer system at the U.S. Geological Survey (USGS), no warranty expressed or implied is made regarding the display or utility of the data for other purposes, nor on all computer systems, nor shall the act of distribution constitute any such warranty. The USGS or the U.S. Government shall not be held liable for improper or incorrect use of the data described and/or contained herein.

Supplemental Material

This is a complete description of the methods used to assemble the puma (*Puma concolor*) genome presented in Ochoa *et al.* 2019. The following methods also include supplemental tables, figures, and code embedded where appropriate. Please contact the corresponding author ochoa.43@osu.edu for additional information.

Sample Collection and Sequencing

We obtained 1 ml of whole blood from five female Texas pumas (TX101, TX105, TX106, TX107, and TX108), four male Florida panthers (FP16, FP45, FP60, and FP79), and one female Florida panther (FP73). A pedigree of the relationship among the samples is shown below in Figure S1. Briefly, Texas pumas TX101, TX105, TX106, and TX108 and Florida panthers FP16, FP45, FP60, and CM7 (unsampled male) constitute the known founder individuals to viable F_1 panthers that resulted from the mid-1990s genetic restoration program. TX101 and CM7 produced two F_1 offspring: FP73 and FP79. The latter, FP79, would then breed with TX107. Specifically, FP45, FP60, and CM7 were 'pure' or canonical Florida panthers from Big Cypress National Preserve; FP16 was a non-canonical Florida panther with Costa Rican and Panamanian ancestry from the Everglades National Park.

We isolated genomic DNA from these samples using a phenol-chloroform extraction protocol, and prepared 500-bp paired-end (PE) sequencing libraries for each individual. We generated an additional 5-kb insert size mate-pair (MP) sequencing library for sample FP16. We sequenced each DNA library on independent lanes on an Illumina HiSeq 2000/2500 system (Illumina Inc., San Diego, California) at the University of Arizona Genetics Core (UAGC, Tucson, Arizona).

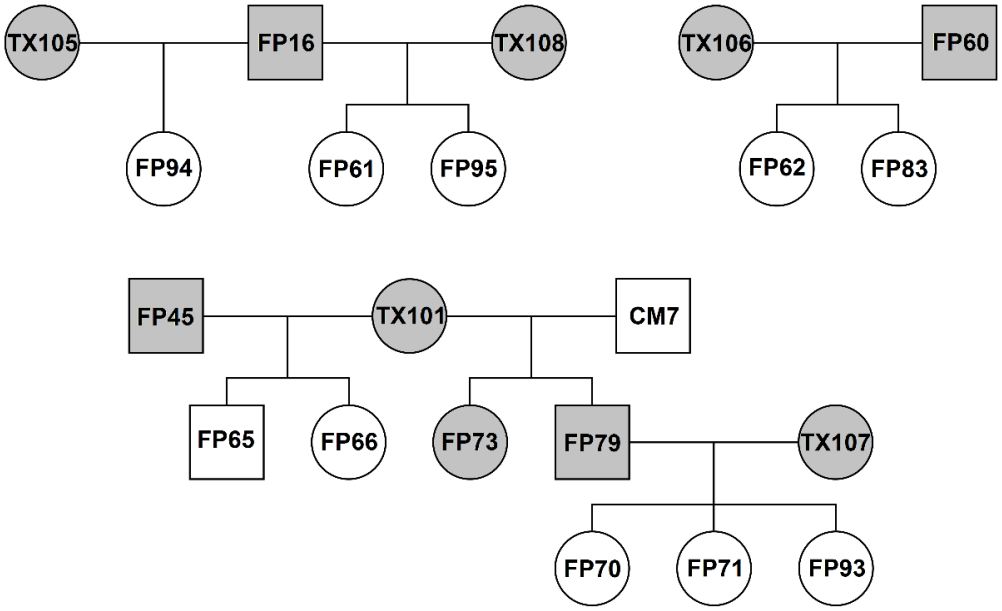

Figure S1. Pedigree of the relationships among the 10 puma samples used in this study (modified from Johnson *et al.* 2010). Individuals sequenced in this study are shown in gray. Squares = males; circles = females; FP = Florida panther; CM7 = unsampled Florida panther sire; TX = Texas puma.

Read Trimming and Error Correction

We cleaned the sequencing reads using Trimmomatic v.0.35 (Bolger *et al.* 2014). The trimming process consisted of removing library and sequencing adapters, eliminating leading and trailing bases with a Phred-scaled quality score of < 20, removing leading and trailing 'N' bases, eliminating the 3' end of the reads with a 4-bp sliding window if the average Phred-scaled quality per base was < 20, and omitting reads with an average Phred-scaled quality score of < 30 or with a length of < 50 bp.

Code for trimming raw reads:

```
# Run Trimmomatic
  # the file 'all.fa' contains all Illumina adapter sequences
  # in fasta format
trimmomatic \
  PE \
  -threads 8 \
  ${fwd}.fastq \
  ${rev}.fastq \
  ${fwd}.trimmed.fastq \
  ${fwd}.SE.trimmed.fastq \
  ${rev}.trimmed.fastq \
  ${rev}.SE.trimmed.fastq \
  ILLUMINACLIP:all.fa:2:30:7 \
  LEADING:20 \
  TRAILING:20 \
  SLIDINGWINDOW:4:20 \
  AVGQUAL:30 \
  MINLEN:50
```

Next, we error-corrected the trimmed reads using 21-mer and 23-mer abundance profiles with Musket v.1.1 (Liu *et al.* 2013; Table S1). The correction of sequencing errors, especially when using assembly tools without an inherent error-correction step, greatly improves the *de novo* assembly of genomes (Salzberg *et al.* 2012). Musket generally outperforms other error-correction algorithms with respect to speed and sensitivity (Akogwu *et al.* 2016; Heydari *et al.* 2017).

Code for performing error correction for both PE and MP reads:

```
# Perform error correction of paired-reads
musket \
  -k 21 536870912 -k 23 268435456 \
  -multik 1 \
  -p 16 \
  ${fwd}.trimmed.fastq \
  ${rev}.trimmed.fastq \
  ${fwd}.SE.trimmed.fastq \
  ${rev}.SE.trimmed.fastq \
  -zlib 1 \
  -maxtrim 70 \
  -lowercase \
  -inorder \
  -omulti $out
```

Table S1. Summary statistics of the raw and processed (trimmed and error-corrected) reads used for this study.

Sample	DNA library	Raw reads with partner	Processed reads with partner	Proportion of retained reads (%)	Mean read length (bp)
FP16	500-bp PE	146,956,816	132,302,486	90.0	98.71
FP45	500-bp PE	150,553,746	133,405,502	88.6	98.43
FP60	500-bp PE	174,846,193	161,077,621	92.1	98.82
FP73	500-bp PE	129,892,080	115,631,502	89.0	98.38
FP79	500-bp PE	98,874,217	91,975,512	93.0	98.34
TX101	500-bp PE	136,391,343	123,755,311	90.7	98.57
TX105	500-bp PE	113,743,817	101,625,014	89.3	98.32
TX106	500-bp PE	94,070,557	87,853,654	93.4	98.45
TX107	500-bp PE	111,974,533	103,416,878	92.4	98.60
TX108	500-bp PE	131,940,171	121,326,422	92.0	98.73
Total	500-bp PE	1,289,243,473	1,172,369,902	90.9	98.56
FP16	5-kb MP	136,126,902	60,844,542	44.7	89.47

De novo Assembly

We performed a *de novo* assembly using ABySS v.1.3.6 (Simpson *et al.* 2009). ABySS is designed for Illumina short-read sequencing, including PE, MP, and single-end (SE) reads. To determine the optimal *k*-mer length to use in ABySS, we repeated the *de novo* assembly for *k*-mer lengths of 45, 51, 55, 59, 61, 63, 65, 69, and 75 bp. We chose the best assembly by comparing the number of scaffolds, scaffold N50, and proportion of retained core eukaryotic genes (CEGs) identified by CEGMA (see below).

Code for an example assembly at *k* = 61 and for removing resulting scaffolds < 500 bp:

```
# Run the de novo assembly
abyss-pe \
  v=-v \
  np=16 \
  k=61 \
  n=5 \
  s=200 \
  name=Pco \
  lib='FP16 FP45 FP60 FP73 FP79 TX101 TX105 TX106 TX107 TX108' \
  mp='mp1' \
  FP16='16.PE.R1.cor.fastq.gz 16.PE.R2.cor.fastq.gz' \
  FP45='45.PE.R1.cor.fastq.gz 45.PE.R2.cor.fastq.gz' \
  FP60='60.PE.R1.cor.fastq.gz 60.PE.R2.cor.fastq.gz' \
  FP73='73.PE.R1.cor.fastq.gz 73.PE.R2.cor.fastq.gz' \
  FP79='79.PE.R1.cor.fastq.gz 79.PE.R2.cor.fastq.gz' \
  TX101='101.PE.R1.cor.fastq.gz 101.PE.R2.cor.fastq.gz' \
  TX105='105.PE.R1.cor.fastq.gz 105.PE.R2.cor.fastq.gz' \
  TX106='106.PE.R1.cor.fastq.gz 106.PE.R2.cor.fastq.gz' \
  TX107='107.PE.R1.cor.fastq.gz 107.PE.R2.cor.fastq.gz' \
  TX108='108.PE.R1.cor.fastq.gz 108.PE.R2.cor.fastq.gz' \
  se='16.SE.R1.cor.fastq.gz 16.SE.R2.cor.fastq.gz \
  FP16.MP-SE.R1.cor.fastq.gz FP16.MP-SE.R2.cor.fastq.gz \
  45.SE.R1.cor.fastq.gz 45.SE.R2.cor.fastq.gz 60.SE.R1.cor.fastq.gz \
  60.SE.R2.cor.fastq.gz 73.SE.R1.cor.fastq.gz 73.SE.R2.cor.fastq.gz \
  79.SE.R1.cor.fastq.gz 79.SE.R2.cor.fastq.gz 101.SE.R1.cor.fastq.gz \
  101.SE.R2.cor.fastq.gz 105.SE.R1.cor.fastq.gz 105.SE.R2.cor.fastq.gz \
  106.SE.R1.cor.fastq.gz 106.SE.R2.cor.fastq.gz 107.SE.R1.cor.fastq.gz \
  107.SE.R2.cor.fastq.gz 108.SE.R1.cor.fastq.gz 108.SE.R2.cor.fastq.gz' \
  mp1='FP16.MP-PE.R1.cor.fastq.gz FP16.MP-PE.R2.cor.fastq.gz'

# The 'lib' includes all trimmed and corrected
  # PE libraries for each individual
  # The 'mp' is the single trimmed and corrected MP library from FP16
```

```
  # the 'se' are the remaining unpaired trimmed and corrected reads
  # from each individual and the MP library.
  # Remove all scaffolds less than 500-bp using 'awk' script
cat Pco-scaffolds.fa | \
  awk '!/^>/ {next}; {getline s}; length(s) >= 500 { print $0 "\n" s }' \
  > Pco-scaffolds-500.fa
```

Afterwards, we assessed the completeness of each assembly using CEGMA v.2.4 (Parra *et al.* 2007). This program annotates highly conserved CEGs that should be present in the genome.

We used the '--mam' parameter to model mammalian intron structure:

```
# Load dependencies
module load blast/2.2.29
module load hmmer/3.0
export CEGMA=/path/to/cegma_v2.4.010312
export PERL5LIB=/path/to/cegma_v2.4.010312/lib/
export WISECONFIGDIR=/path/to/wise2.2.3-rc7/wisecfg/
export PATH=$PATH:/path/to/geneid/bin/
export PATH=$PATH:/path/to/cegma_v2.4.010312/bin
export PATH=$PATH:/path/to/wise2.2.3-rc7/src/bin/

# Run CEGMA
cegma \
  --mam \
  -o Pco \
  -v \
  -T 16 \
  -g Pco-scaffolds-500.fa
```

Across the various assemblies using different *k*-mer lengths, we selected *k* = 61 for the final assembly. This value for *k* gave the best balance of proportion of CEGs, number of scaffolds, and scaffold N50 (Figure S2A-C). Furthermore, although we found a large number of scaffolds when using *k* = 61, 95% of the total assembly was contained within the longest ~37,000 scaffolds (Figure S2D).

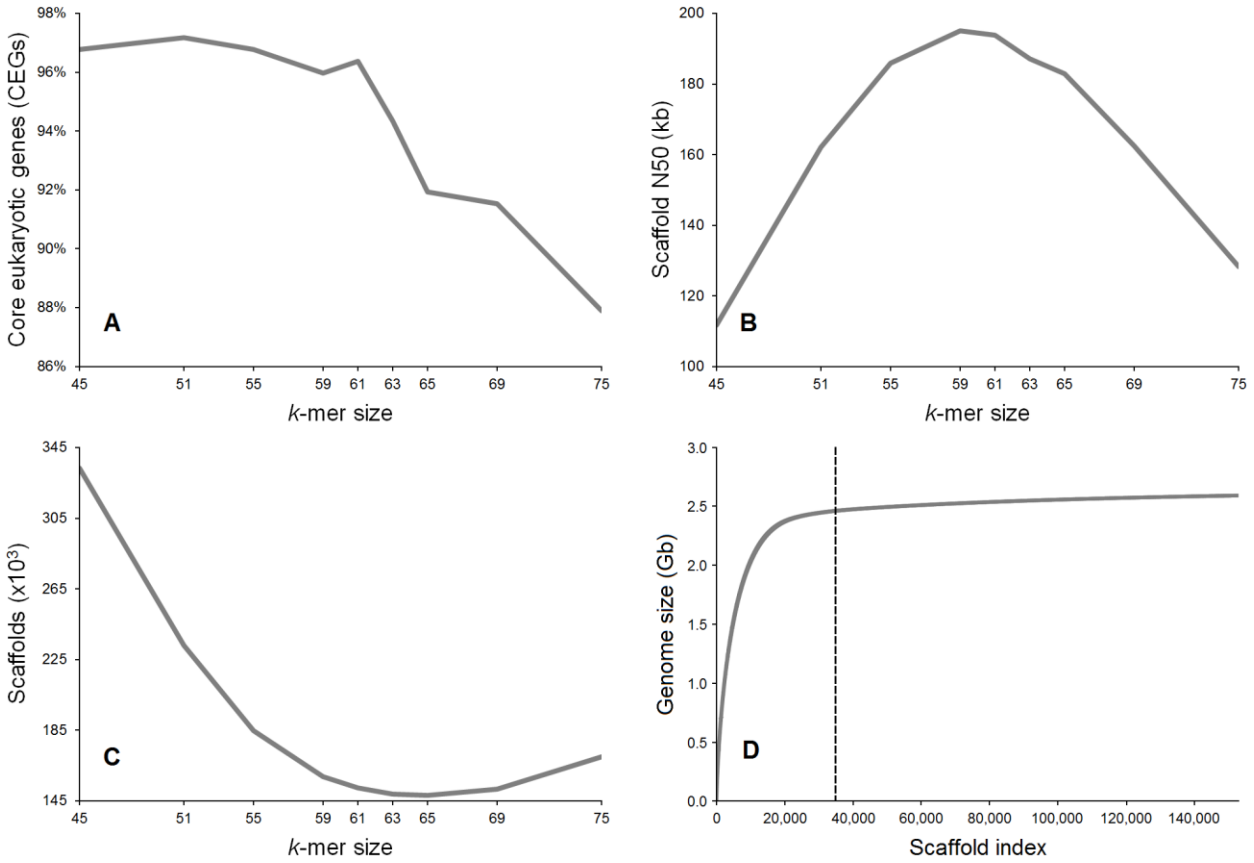

Figure S2. Quality metrics for *de novo* genome assemblies. (A) Proportion of core eukaryotic genes (CEGs), (B) scaffold N50, and (C) number of scaffolds for assemblies based on a different *k*-mer size (45, 51, 55, 59, 61, 63, 65, 69, and 75 bp). (D) Cumulative length of the assembly based on a *k*-mer size of 61 bp. Scaffolds are sorted from longest to smallest along the horizontal axis. The vertical dotted line indicates the number of scaffolds containing 95% of the total assembly.

Genome Annotation

Repetitive elements and non-coding RNAs

We annotated and masked repetitive elements in the *de novo* assembly using RepeatMasker v.4.0.7 (Smit *et al.* 1996-2010) and the Repbase v.19.07 library (Jurka *et al.* 2005). Additionally, we identified and annotated non-coding RNA loci with Infernal v.1.1.2 (Nawrocki *et al.* 2009). Infernal predicts non-coding RNA molecules based on homology searches and comparisons with the Rfam database (Release 12.0; Griffiths-Jones *et al.* 2003). When predicting RNAs, we employed a gathering score threshold of 85% for the covariance models along with a confidence threshold, or e-value, of 10^{-9}. Both the repetitive elements and non-coding RNAs predicted are presented below in Tables S2 and S3, respectively.

Code for annotating repetitive elements:

```
# Run RepeatMasker v.4.0.7
RepeatMasker \
  -pa 16 \
  -gff \
  -xsmall \
  All_k61.final.fa
```

Table S2. Repetitive elements characterized in the puma genome assembly.

Class	Elements	Length occupied (bp)	Genome %
SINEs	467,738	68,542,447	2.6
ALUs	4	259	0.0
MIRs	460,056	67,642,334	2.6
LINEs	886,693	416,873,394	16.1
LINE1	517,553	321,962,875	12.4
LINE2	315,106	82,908,505	3.2
L3/CR1	40,169	8,681,856	0.3
LTR elements	305,154	113,444,496	4.4
ERVL	91,477	39,896,259	1.5
ERVL-MaLRs	151,043	51,741,991	2.0
ERVL class I	38,973	15,877,812	0.6
ERVL class II	290	120,563	0.0
DNA elements	354,824	71,100,928	2.7
hAT-Charlie	196,881	36,710,523	1.4
TcMar-Tigger	63,897	16,660,285	0.6
Small RNA	106,881	7,840,011	0.3
Satellites	237	46,117	0.0
Simple repeats	1,246,144	56,888,686	2.2
Low complexity	403,994	21,190,431	0.8
Other	5,873	1,082,071	0.0
Total	3,777,538	756,773,358	29.2

Code for annotating non-coding RNAs:

```
# Run Infernal v.1.1.2
# Set the gathering score cutoff to 0.85 for each RNA model.
# Only one example model is shown, repeat for all models.
GA=$(grep "^GA" RF00001.cm | \
  sed 's/^GA[ ]*//g' | \
  perl -ne '$_=0.85 * $_; print "$_"')

# Run prediction for the example model.
cmsearch \
  -Z 5400 \
  -T $GA \
  --tblout RF00001.cm.tbl \
  RF00001.cm \
  All_k61.final.fa \
  > RF00001.cm.cmsearch
```

Table S3. RNA loci and families identified in the puma genome assembly.

Class	Loci	Families
cis-regulatory RNA	63	31
lncRNA	160	155
miRNA	1,224	201
Ribozyme	12	5
rRNA	248	9
snoRNA	570	209
snRNA	1,041	10
tRNA	258	2
Other	355	14
Total	3,931	636

Structural annotation of protein-coding genes

We used the MAKER2 v.2.31.6 pipeline (Cantarel *et al.* 2008; Holt and Yandell 2011) to annotate protein-coding genes in the final puma assembly. This pipeline assembles evidence from a variety of *ab initio* gene modelers, ESTs, and/or proteins (Figure S3). Briefly, the first iteration consisted of the annotation of genes with SNAP (Korf 2004) using hidden Markov models (hmms) for genes trained from CEGs using CEGMA; *ab initio* predictions of genes from GeneMark-ES (Lomsadze *et al.* 2005); transcriptome sequences from the puma (Fitak *et al.* 2016a); and protein sequences from the cheetah, cat, tiger, and dog downloaded from Ensembl (www.ensembl.org). The second iteration was similar to the first one, but included *ab initio* predictions of genes with AUGUSTUS v.2.5.5 (Stanke *et al.* 2006) in addition to predictions from SNAP using an hmm built from the first round of predictions produced by MAKER2. We then removed the gene predictions with an exon annotation edit distance (eAED) ≥ 0.75 (Holt and Yandell 2011; Figure S4).

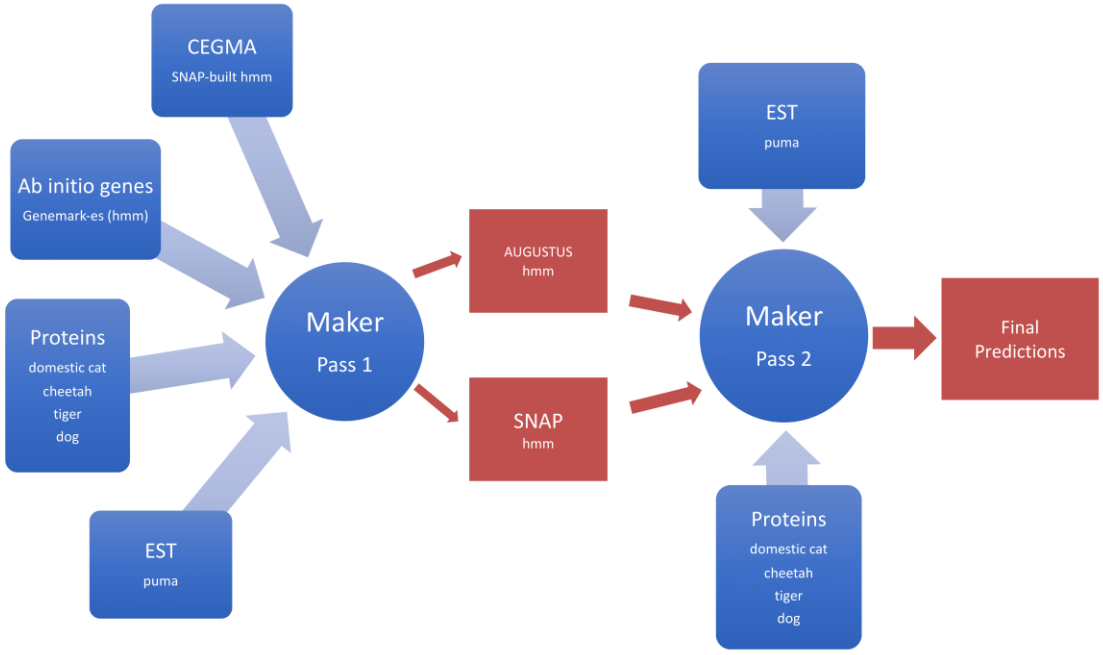

Figure S3. Visual representation of the evidence used in the two-step gene prediction process implemented in MAKER2.

Code for downloading species-specific databases:

```
# Downloaded the Puma ESTs from Fitak et al. 2016a
# the dataset is available at Pangaea
  # https://doi.org/10.1594/PANGAEA.835154
# the file name is Puma_SNP_Data.zip
# unpack and reformat into a fasta file
unzip Puma_SNP_Data.zip
tr \
  "\t" \
  "\n" \
  < Puma_SNP_Data/Puma_cDNA_transcriptome.tab \
  > puma-transcriptome.fasta

# 2,109 assembled EST contigs in total
# Download and concatenate cat, tiger, cheetah and dog protein sequences
# NOTE: Pay attention to the specific genome versions of each species
```

```
genome=ftp://ftp.ncbi.nlm.nih.gov/genomes

# Cat - 33,220 protein sequences
wget ${genome}/Felis_catus/ARCHIVE/ANNOTATION_RELEASE.102/protein/protein.fa.gz
gunzip protein.fa.gz; mv protein.fa Fcatus.protein.fa

# Tiger - 29,473 protein sequences
wget ${genome}/Panthera_tigris_altaica/protein/protein.fa.gz
gunzip protein.fa.gz; mv protein.fa Ptigris.protein.fa

# Cheetah - 27,284 protein sequences
wget ${genome}/Acinonyx_jubatus/ARCHIVE/ANNOTATION_RELEASE.100/protein/protein.fa.gz
gunzip protein.fa.gz; mv protein.fa Ajubatus.protein.fa

# Dog - 47,087 protein sequences
wget ${genome}/Canis_lupus_familiaris/ARCHIVE/ANNOTATION_RELEASE.104/protein/protein.fa.gz
gunzip protein.fa.gz; mv protein.fa Cfamiliaris.protein.fa

# Merge all together
cat \
Fcatus.protein.fa \
Ptigris.protein.fa \
Ajubatus.protein.fa \
Cfamiliaris.protein.fa > homologs.fa
```

Code for running the first pass of maker MAKER2:

```
# Build an hmm from the CEMGA output using MAKER2 and SNAP
cegma2zff Pco.cegma.gff Pco-scaffolds-500.fa
  # Output = genome.ann and genome.dna
fathom genome.ann genome.dna -categorize 1000
  # Output = uni.dna, uni.ann, wrn.dna, wrn.ann, olp.ann,
  # olp.dna, err.ann, err.dna, alt.dna, alt.ann
fathom -export 1000 -plus uni.ann uni.dna
  # Output = export.aa, export.ann, export.dna, export.tx
forge export.ann export.dna
  # Output = many files
hmm-assembler.pl Pco . > Pco.cegmasnap.hmm
  # Output = Pco.cegmasnap.hmm

# Run GENEMARK-ES
gmes_petap.pl --cores 4 --v --ES --sequence Pco-scaffolds-500.fa

# Load dependencies for MAKER2
module load gcc/4.7.2
module load intelmpi/4.1.0
module load bioperl/1.6.9
module load exonerate/2.2.0
export AUGUSTUS_CONFIG_PATH=/path/to/augustus.2.5.5/config/
export ZOE=/path/to/snap
export PATH=$PATH:/path/to/snap/
export PATH=$PATH:/path/to/gm_et_linux_64/
export PATH=$PATH:/path/to/augustus.2.5.5/bin/
export PATH=$PATH:/path/to/ncbi-blast-2.2.25+/bin/
export PERL5LIB=/path/to/maker/perl/lib/

srun maker -base MAKER1 maker1_opts.ctl maker1_bopts.ctl maker1_exe.ctl
```

The contents of the configuration files were:

maker1_opts.ctl

```
#-----Genome (these are always required)
genome=Pco-scaffolds-500.fa #genome sequence
organism_type=eukaryotic #eukaryotic or prokaryotic.

#-----Re-annotation Using MAKER Derived GFF3
maker_gff= #MAKER derived GFF3 file
est_pass=0 #use ESTs in maker_gff: 1 = yes, 0 = no
altest_pass=0 #use alternate organism ESTs in maker_gff: 1 = yes, 0 = no
protein_pass=0 #use protein alignments in maker_gff: 1 = yes, 0 = no
rm_pass=0 #use repeats in maker_gff: 1 = yes, 0 = no
model_pass=0 #use gene models in maker_gff: 1 = yes, 0 = no
pred_pass=0 #use ab-initio predictions in maker_gff: 1 = yes, 0 = no
other_pass=0 #passthrough anything else in maker_gff: 1 = yes, 0 = no

#-----EST Evidence (for best results provide a file for at least one)
est=puma-transcriptome.fasta #set of ESTs from Fitak et al. 2016a
altest= #EST/cDNA sequence file in fasta format from an alternate organism
est_gff= #aligned ESTs or mRNA-seq from an external GFF3 file
altest_gff= #aligned ESTs from a closely relate species in GFF3 format

#-----Protein Homology Evidence (for best results provide a file for at least one)
protein=homologs.fa # protein sequence fasta file concatenated from cat, tiger, cheetah, dog
protein_gff= #aligned protein homology evidence from an external GFF3 file
#-----Repeat Masking (leave values blank to skip repeat masking)
model_org=all #select a model organism for RepBase masking in RepeatMasker
rmlib= #provide an organism specific repeat library in fasta format for RepeatMasker
repeat_protein=te_proteins.fasta #fasta file of transposable elements for RepeatRunner
rm_gff= #pre-identified repeat elements from an external GFF3 file
prok_rm=0 #forces MAKER to repeatmask prokaryotes, 1 = yes, 0 = no
softmask=1 #use soft-masking rather than hard-masking in BLAST
     # (i.e. seg and dust filtering)

#-----Gene Prediction
snaphmm=Pco.cegmasnap.hmm #SNAP HMM file
gmhmm=gmhmm.mod #GeneMark HMM file
augustus_species= #Augustus gene prediction species model
fgenesh_par_file= #FGENESH parameter file
pred_gff= #ab-initio predictions from an external GFF3 file
model_gff= #annotated gene models from an external GFF3 file (annotation pass-through)
est2genome=1 #infer gene predictions directly from ESTs, 1 = yes, 0 = no
protein2genome=1 #infer predictions from protein homology, 1 = yes, 0 = no
trna=1 #find tRNAs with tRNAscan, 1 = yes, 0 = no
snoscan_rrna= #rRNA file to have Snoscan find snoRNAs
unmask=0 #also run ab-initio prediction programs on unmasked sequence, 1 = yes, 0 = no

#-----Other Annotation Feature Types (features MAKER doesn't recognize)
other_gff= #extra features to pass-through to final MAKER generated GFF3 file

#-----External Application Behavior Options
alt_peptide=C #amino acid used to replace non-standard amino acids in BLAST databases
cpus=1 #max number of cpus to use in BLAST and RepeatMasker (leave 1 when using MPI)

#-----MAKER Behavior Options
max_dna_len=100000 #length for dividing up contigs (increases/decreases memory usage)
min_contig=1 #skip genome contigs below this length (under 10kb are often useless)
pred_flank=200 #flank for extending evidence clusters sent to gene predictors
pred_stats=0 #report AED and QI statistics for all predictions as well as models
```

```
AED_threshold=1 #Maximum Annotation Edit Distance allowed (bound by 0 and 1)
min_protein=0 #require at least this many amino acids in predicted proteins
alt_splice=0 #Take extra steps to try and find alternative splicing, 1 = yes, 0 = no
always_complete=0 #extra steps to force start and stop codons, 1 = yes, 0 = no
map_forward=0 #map names and attributes forward from old GFF3 genes, 1 = yes, 0 = no
keep_preds=1 #Concordance threshold to add unsupported gene prediction (0 to 1)
split_hit=10000 #length for the splitting of hits
single_exon=1 #consider single exon EST evidence when generating annotations
single_length=250 #min length for single exon ESTs if 'single_exon is enabled'
correct_est_fusion=0 #limits use of ESTs in annotation to avoid fusion genes
tries=2 #number of times to try a contig if there is a failure for some reason
clean_try=0 #remove all data from previous run before retrying, 1 = yes, 0 = no
clean_up=0 #removes the Void directory with individual analysis files, 1 = yes, 0 = no
TMP= #specify a temporary directory for temporary files
```

maker1_bopts.ctl

```
#-----BLAST and Exonerate Statistics Thresholds
blast_type=ncbi+ #set to 'ncbi+', 'ncbi' or 'wublast'

pcov_blastn=0.8 #Blastn Percent Coverage Threshold EST-Genome Alignments
pid_blastn=0.85 #Blastn Percent Identity Threshold EST-Genome Alignments
eval_blastn=1e-10 #Blastn eval cutoff
bit_blastn=40 #Blastn bit cutoff
depth_blastn=0 #Blastn depth cutoff (0 to disable cutoff)

pcov_blastx=0.5 #Blastx Percent Coverage Threshold Protein-Genome Alignments
pid_blastx=0.4 #Blastx Percent Identity Threshold Protein-Genome Alignments
eval_blastx=1e-06 #Blastx eval cutoff
bit_blastx=30 #Blastx bit cutoff
depth_blastx=0 #Blastx depth cutoff (0 to disable cutoff)

pcov_tblastx=0.8 #tBlastx Percent Coverage Threshold alt-EST-Genome Alignments
pid_tblastx=0.85 #tBlastx Percent Identity Threshold alt-EST-Genome Alignments
eval_tblastx=1e-10 #tBlastx eval cutoff
bit_tblastx=40 #tBlastx bit cutoff
depth_tblastx=0 #tBlastx depth cutoff (0 to disable cutoff)

pcov_rm_blastx=0.5 #Blastx Percent Coverage Threshold For Transposable Element Masking
pid_rm_blastx=0.4 #Blastx Percent Identity Threshold For Transposable Element Masking
eval_rm_blastx=1e-06 #Blastx eval cutoff for transposable element masking
bit_rm_blastx=30 #Blastx bit cutoff for transposable element masking

ep_score_limit=20 #Exonerate protein percent of maximal score threshold
en_score_limit=20 #Exonerate nucleotide percent of maximal score threshold
```

maker1_exe.ctl

```
# Please set the paths to the various executables to match your computer system.
```

Code for preparing the output from the first iteration for AUGUSTUS and the second pass of MAKER2:

```
# Merge MAKER output
gff3_merge \
   -d MAKER1.maker.output/MAKER1_master_datastore_index.log \
   -o maker1.gff

# Build new SNAP hmm
mkdir MAKER1_HMM
cd MAKER1_HMM
maker2zff ../maker1.gff
fathom genome.ann genome.dna -categorize 1000
fathom -export 1000 -plus uni.ann uni.dna
forge export.ann export.dna
hmm-assembler.pl maker1 . > ../maker_snap1.hmm
zff2gff3.pl genome.ann | \
   perl -plne 's/\t(\S+)$/\t\.\t$1/' > genome.gff3

# Build AUGUSTUS hmm
# This had to be run as several different parts,
# only the basic outline is shown below
cd MAKER1_HMM
module purge
module load blat
module load blast
export AUGUSTUS_CONFIG_PATH=/path/to/augustus.2.5.5/config/
export PATH=$PATH:/path/to/augustus.2.5.5/bin

autoAug.pl \
   --genome=Pco-scaffolds-500.fa \
   --species=pconcolor \
   --cdna=puma-transcriptome.fasta \
   --trainingset=genome.gff3 \
   -v -v -v \
   --useexisting
```

Code for running the second pass of MAKER2:

```
# Load dependencies for MAKER2
module load gcc/4.7.2
module load intelmpi/4.1.0
module load bioperl/1.6.9
module load exonerate/2.2.0
export AUGUSTUS_CONFIG_PATH=/path/to/augustus.2.5.5/config/
export ZOE=/path/to/snap
export PATH=$PATH:/path/to/snap/
export PATH=$PATH:/path/to/gm_et_linux_64/
export PATH=$PATH:/path/to/augustus.2.5.5/bin/
export PATH=$PATH:/path/to/ncbi-blast-2.2.25+/bin/
export PERL5LIB=/path/to/maker/perl/lib/

srun maker -base MAKER2 maker2_opts.ctl maker2_bopts.ctl maker2_exe.ctl
```

The contents of the configuration files were:

maker2_opts.ctl

```
#-----Genome (these are always required)
genome=Pco-scaffolds-500.fa #genome sequence
organism_type=eukaryotic #eukaryotic or prokaryotic.

#-----Re-annotation Using MAKER Derived GFF3
maker_gff= #MAKER derived GFF3 file
est_pass=0 #use ESTs in maker_gff: 1 = yes, 0 = no
altest_pass=0 #use alternate organism ESTs in maker_gff: 1 = yes, 0 = no
protein_pass=0 #use protein alignments in maker_gff: 1 = yes, 0 = no
rm_pass=0 #use repeats in maker_gff: 1 = yes, 0 = no
model_pass=0 #use gene models in maker_gff: 1 = yes, 0 = no
pred_pass=0 #use ab-initio predictions in maker_gff: 1 = yes, 0 = no
other_pass=0 #passthrough anything else in maker_gff: 1 = yes, 0 = no

#-----EST Evidence (for best results provide a file for at least one)
est=puma-transcriptome.fasta #set of ESTs in fasta format from Fitak et al. 2016a
altest= #EST/cDNA sequence file in fasta format from an alternate organism
est_gff= #aligned ESTs or mRNA-seq from an external GFF3 file
altest_gff= #aligned ESTs from a closely relate species in GFF3 format

#-----Protein Homology Evidence (for best results provide a file for at least one)
protein=homologs.fa # protein sequence fasta file
   # concatenated from cat, cheetah, tiger, and dog
protein_gff= #aligned protein homology evidence from an external GFF3 file

#-----Repeat Masking (leave values blank to skip repeat masking)
model_org=all #select a model organism for RepBase masking in RepeatMasker
rmlib= #provide an organism specific repeat library in fasta format for RepeatMasker
repeat_protein=te_proteins.fasta #RepeatRunner fasta file of transposable elements
rm_gff= #pre-identified repeat elements from an external GFF3 file
prok_rm=0 #forces MAKER to repeatmask prokaryotes, 1 = yes, 0 = no
softmask=1 #use soft-masking rather than hard-masking in BLAST

#-----Gene Prediction
snaphmm=maker1_snap.hmm #SNAP HMM file
gmhmm=gmhmm.mod #GeneMark HMM file
augustus_species=pconcolor #Augustus gene prediction species model
fgenesh_par_file= #FGENESH parameter file
pred_gff= #ab-initio predictions from an external GFF3 file
model_gff= #annotated gene models from an external GFF3 file (annotation pass-through)
est2genome=0 #infer gene predictions directly from ESTs, 1 = yes, 0 = no
protein2genome=0 #infer predictions from protein homology, 1 = yes, 0 = no
trna=1 #find tRNAs with tRNAscan, 1 = yes, 0 = no
snoscan_rrna= #rRNA file to have Snoscan find snoRNAs
unmask=0 #also run ab-initio prediction programs on unmasked sequence, 1=yes, 0=no

#-----Other Annotation Feature Types (features MAKER doesn't recognize)
other_gff= #extra features to pass-through to final MAKER generated GFF3 file

#-----External Application Behavior Options
alt_peptide=C #amino acid used to replace non-standard amino acids in BLAST databases
cpus=1 #max number of cpus to use in BLAST and RepeatMasker (leave 1 when using MPI)

#-----MAKER Behavior Options
max_dna_len=100000 #length for dividing up contigs (increases/decreases memory usage)
min_contig=1 #skip genome contigs below this length (under 10kb are often useless)
```

```
pred_flank=200 #flank for extending evidence clusters sent to gene predictors
pred_stats=1 #report AED and QI statistics for all predictions as well as models
AED_threshold=1 #Maximum Annotation Edit Distance allowed (bound by 0 and 1)
min_protein=30 #require at least this many amino acids in predicted protein
alt_splice=1 #Take extra steps to try and find alternative splicing, 1 = yes, 0 = no
always_complete=0 #extra steps to force start and stop codons, 1 = yes, 0 = no
map_forward=0 #map names and attributes forward from old GFF3 genes, 1 = yes, 0 = no
keep_preds=1 #Concordance threshold to add unsupported gene prediction (0 to 1)

split_hit=10000 #length for the splitting of hits
single_exon=1 #consider single exon EST evidence when generating annotations
single_length=250 #min length for single exon ESTs if 'single_exon is enabled'
correct_est_fusion=0 #limits use of ESTs in annotation to avoid fusion genes

tries=2 #number of times to try a contig if there is a failure for some reason
clean_try=0 #remove all data from previous run before retrying, 1 = yes, 0 = no
clean_up=0 #removes the Void directory with individual analysis files, 1 = yes, 0 = no
TMP= #specify a temporary directory for temporary files
```

maker2_bopts.ctl

```
#-----BLAST and Exonerate Statistics Thresholds
blast_type=ncbi+ #set to 'ncbi+', 'ncbi' or 'wublast'

pcov_blastn=0.8 #Blastn Percent Coverage Threshold EST-Genome Alignments
pid_blastn=0.85 #Blastn Percent Identity Threshold EST-Genome Alignments
eval_blastn=1e-10 #Blastn eval cutoff
bit_blastn=40 #Blastn bit cutoff
depth_blastn=0 #Blastn depth cutoff (0 to disable cutoff)

pcov_blastx=0.5 #Blastx Percent Coverage Threshold Protein-Genome Alignments
pid_blastx=0.4 #Blastx Percent Identity Threshold Protein-Genome Alignments
eval_blastx=1e-06 #Blastx eval cutoff
bit_blastx=30 #Blastx bit cutoff
depth_blastx=0 #Blastx depth cutoff (0 to disable cutoff)

pcov_tblastx=0.8 #tBlastx Percent Coverage Threshold alt-EST-Genome Alignments
pid_tblastx=0.85 #tBlastx Percent Identity Threshold alt-EST-Genome Alignments
eval_tblastx=1e-10 #tBlastx eval cutoff
bit_tblastx=40 #tBlastx bit cutoff
depth_tblastx=0 #tBlastx depth cutoff (0 to disable cutoff)

pcov_rm_blastx=0.5 #Blastx Percent Coverage Threshold For Transposable Element Masking
pid_rm_blastx=0.4 #Blastx Percent Identity Threshold For Transposable Element Masking
eval_rm_blastx=1e-06 #Blastx eval cutoff for transposable element masking
bit_rm_blastx=30 #Blastx bit cutoff for transposable element masking

ep_score_limit=20 #Exonerate protein percent of maximal score threshold
en_score_limit=20 #Exonerate nucleotide percent of maximal score threshold
```

maker2_exe.ctl

```
# Please set the paths to the various executables to match your computer system.
```

Code for retaining predictions with an eAED score < 0.75 (Holt and Yandell 2011; Figure S4):

```
# Merge gff output files
gff3_merge \
  -d MAKER2.maker.output/MAKER2_master_datastore_index.log \
  -o maker2.gff

# Merge fasta output files
fasta_merge \
  -d MAKER2.maker.output/MAKER2_master_datastore_index.log \
  -o maker2.fasta

# Filter MAKER proteins for AED scores < 0.75 and then for eAED scores < 0.75
perl -ne \
  'chomp $_; if ($_ =~ /^>/){print "\n$_\n";}else { print "$_";}' \
  maker2.fasta.all.maker.proteins.fasta | \
  sed '1d' | \
  tr "\n" "\t" | \
  sed 's/\t>/\n>/g' | \
  perl -ne \
  '@a=split(/\s/, $_); $a[3]=~s/eAED://; if($a[3]<0.75){print "$_";}' | \
  tr "\t" "\n" > maker.proteins.faa

# Make of GFF file for the gene predictions
grep "^>" maker.proteins.faa | \
  sed 's/^>//g' | \
  cut -d' ' -f1 > maker.proteins.list
count=1
while read line
  do
  echo "Processing protein $count"
  grep "$line" maker2.All.gff >> maker.proteins.final.gff
  count=$((count + 1))
done < maker.proteins.list
```

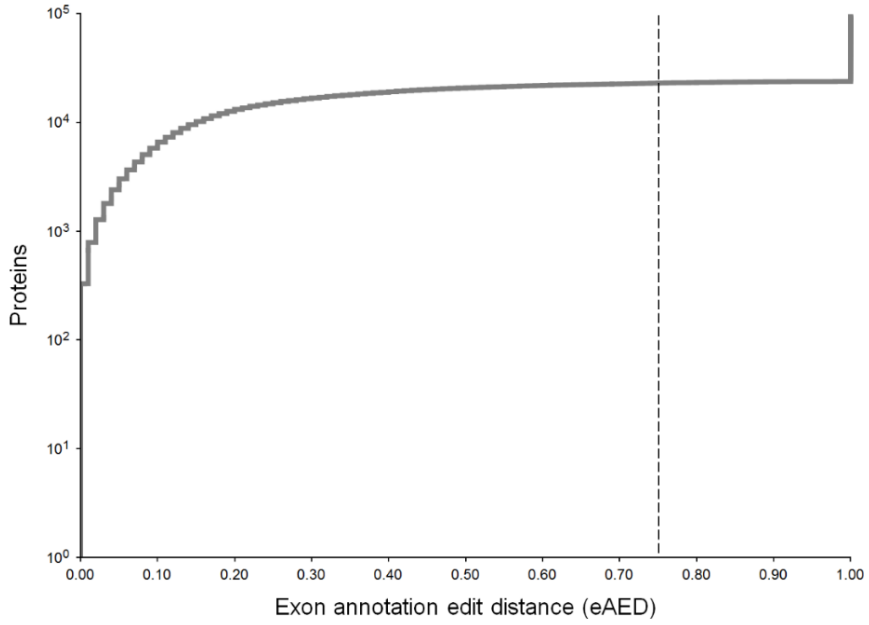

Figure S4. The exon annotation edit distance (eAED) of proteins annotated in the puma genome assembly using the two-step procedure in MAKER2. The vertical dotted line indicates an eAED of 0.75, the cutoff for retaining final gene predictions.

Table S4. Summary statistics of the proteins and genes found across different mammalian species compared with the puma.

Species	Proteins	Genes	Genes with isoforms	Mean gene length (aa)[§]
Puma	22,926*	22,745	177	392.9
Cat	20,259	19,493	727	525.6
Dog	25,157	19,856	4,554	547.1
Panda	21,136	19,343	1,605	531.8
Cow	22,118	19,994	1,844	535.7
Human	102,915	22,964	17,234	553.2
Mouse	59,203	22,770	12,782	526.5

*Exon annotation edit distance (eAED) < 0.75.
[§]After keeping only the longest isoform from each gene.
Mammalian genomes other than the puma were downloaded from Ensembl (www.ensembl.org).

Functional annotation of protein-coding genes

We used the longest isoform from each predicted gene to functionally annotate them with Blast2GO v.4.0.7 (Conesa *et al.* 2005). As input into Blast2GO, we obtained the top 20 BLAST matches for each protein sequence to the metazoan 'nr' database using an e-value threshold of 10^{-3} with BLASTP v.2.2.30. We also combined the BLAST results with assignments of each predicted protein sequence to various domains and motifs using InterProScan v.5.7.48 (Jones *et al.* 2014). The latter searches a variety of databases (i.e., TIGRFAM, ProDom, SMART, HAMAP, PROSITE, SUPERFAMILY, PRINTS, PANTHER, Gene3D, PIRSF, Pfam, COILS) to make these functional assignments in addition to linking gene ontology (GO) terms to each protein. We loaded the BLASTP and InterProScan matches into Blast2GO, where we performed additional GO 'Mapping' and 'Annotation' steps using the default parameters. A summary of the functional annotations is shown in Figures S5−S8.

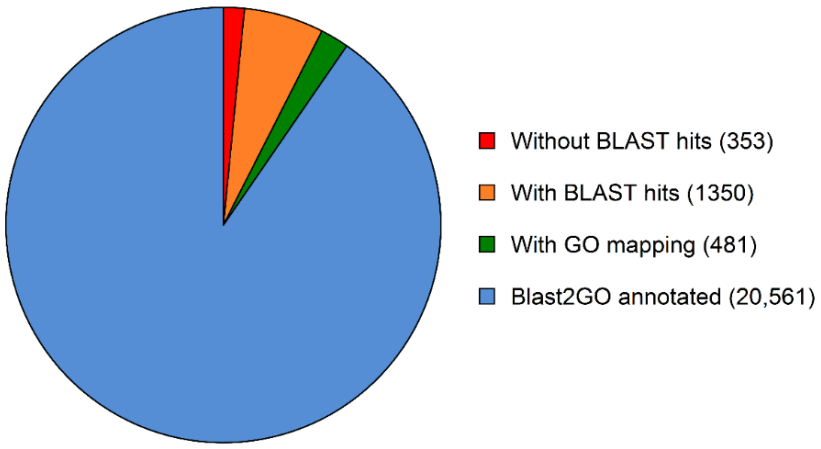

Figure S5. Summary statistics of the genes functionally annotated with Blast2GO.

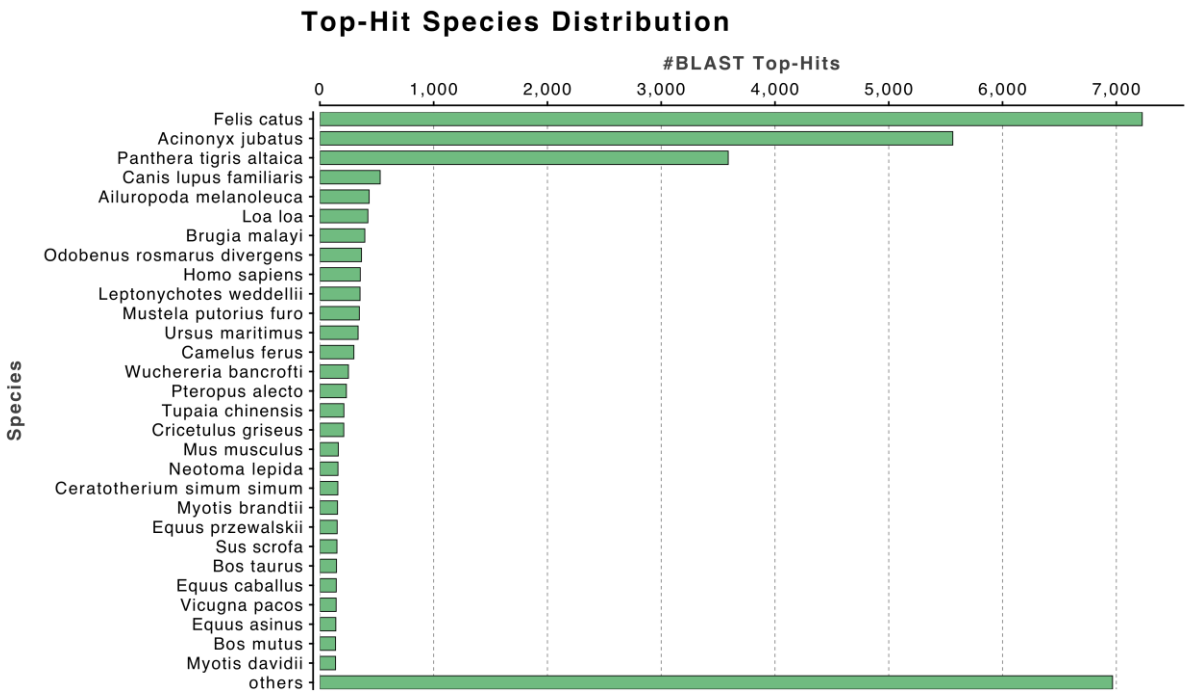

Figure S6. Species distribution of the top hit for each protein sequence with Blast2GO.

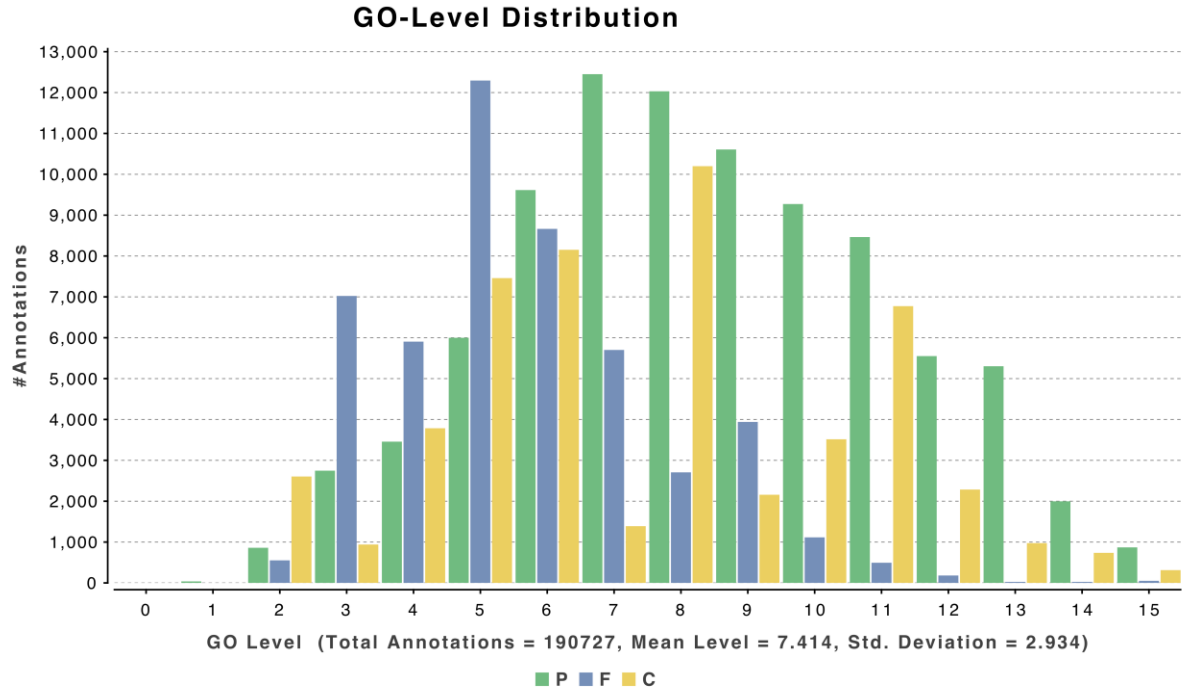

Figure S7. Summary of the number of gene ontology (GO) terms annotated to the protein sequences using Blast2GO. The three types of terms are for biological processes (P; green), molecular functions (F; blue), and cellular components (C; yellow).

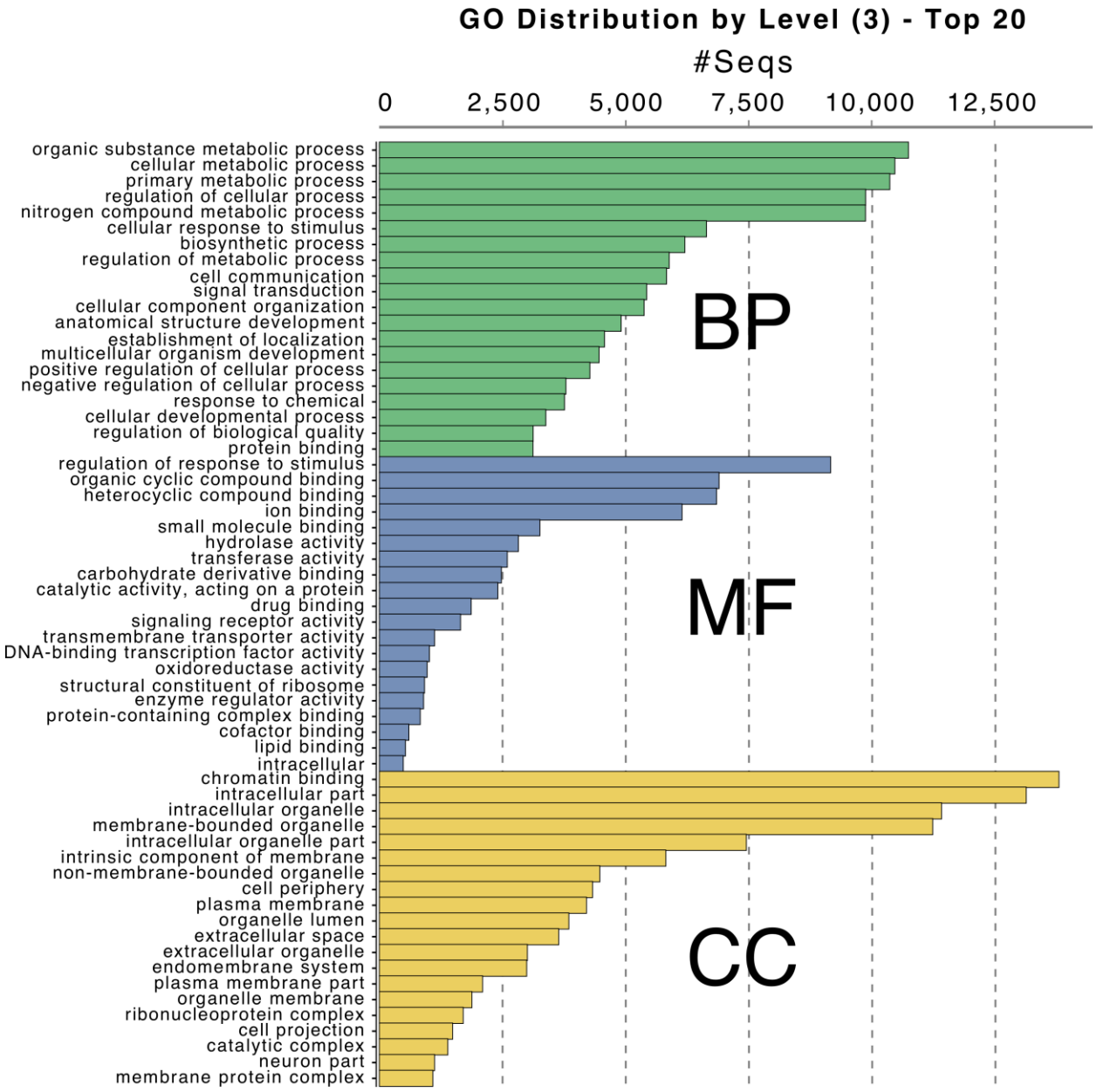

Figure S8. The 20 most abundant gene ontology (GO) terms annotated with Blast2GO in each of the types biological processes (BP; green), molecular functions (MF; blue), and cellular components (CC; yellow). Only GO terms at Level 3 are shown (GO terms from the top two levels of the GO hierarchy are omitted for lack of specificity).

Code for BLAST and InterProScan searches:

```
# Separate fasta file into a separate file for each sequence
# fasta-splitter.pl is a script available upon request
fasta-splitter.pl \
  --part-size 1 \
  --measure count \
  --line-length 0 \
  maker.proteins.longestORFs.faa
```

```
# Get a list of metazoan taxa IDs from NCBI
  # txid33208.gids

# Blast each sequence independently, a cluster job would be faster
for i in *.fa
  do
  echo "Blasting sequence $i"
  blastp \
    -query $i \
    -db nr \
    -outfmt 5 \
    -out $i.xml \
    -evalue 0.001 \
    -max_target_seqs 20 \
    -gilist txid33208.gids
done

# Run interproscan
cluster_interproscan.sh \
  -i maker.proteins.longestORFs.faa \
  -o PCO_IPRSCAN \
  -f XML \
  -t p \
  -nsplit 30 \
  -blast2go
```

Identification of Orthologs and Interspecific Analyses

Proteome clustering

We clustered the protein-coding genes identified in the puma genome along with proteomes from the domestic cat, dog, panda, cow, mouse, and human genomes into orthologous gene families using OrthoMCL v.1.0 (Li *et al.* 2003). We removed any sequences < 10 amino acids in length, and containing more than 20% early stop codons. In total, 85 sequences were removed, leaving 147,080 protein sequences. We used a minimum identity of 50% and an e-value cut-off of 10^{-5} to call orthologs.

Code for clustering proteomes:

```
# Setup new folders
mkdir ORTHOMCL
cd ORTHOMCL

# Download and prep all proteomes
# cat, dog, panda, cow, human, and mouse protein sequences
# NOTE: Pay attention to the specific genome versions of each species
genome=ftp://ftp.ensembl.org/pub
# NOTE: see below for the code for the short script "longest.sh",
# which grabs the longest isoform for each gene ID

# Cat - v6.2 - 20259 protein sequences (19493 genes)
wget \
${genome}/release-90/fasta/felis_catus//pep/Felis_catus.Felis_catus_6.2.pep.all.fa.gz
gunzip Felis_catus.Felis_catus_6.2.pep.all.fa.gz
longest.sh Felis_catus.Felis_catus_6.2.pep.all.fa Fcatus.protein.fa
```

```
# Dog - v3.1 - 25157 protein sequences (19856 genes)
wget \
${genome}/release-90/fasta/canis_familiaris//pep/Canis_familiaris.CanFam3.1.pep.all.fa.gz
gunzip Canis_familiaris.CanFam3.1.pep.all.fa.gz
longest.sh Canis_familiaris.CanFam3.1.pep.all.fa Cfamiliarus.protein.fa

# Panda - v1.0 - 21136 protein sequences (19343 genes)
wget \
${genome}/release-90/fasta/ailuropoda_melanoleuca//pep/Ailuropoda_melanoleuca.ailMel1.pep.all.fa.gz
gunzip Ailuropoda_melanoleuca.ailMel1.pep.all.fa.gz
longest.sh Ailuropoda_melanoleuca.ailMel1.pep.all.fa Amelanoleuca.protein.fa

# Cow - vUMD3.1 - 22118 protein sequences (19994 genes)
wget \
${genome}/release-90/fasta/bos_taurus//pep/Bos_taurus.UMD3.1.pep.all.fa.gz
gunzip Bos_taurus.UMD3.1.pep.all.fa.gz
longest.sh Bos_taurus.UMD3.1.pep.all.fa Btaurus.protein.fa

# Human - vGRCh38 - 102915 protein sequences (22964 genes)
wget \
${genome}/release-86/fasta/homo_sapiens//pep/Homo_sapiens.GRCh38.pep.all.fa.gz
gunzip Homo_sapiens.GRCh38.pep.all.fa.gz
longest.sh Homo_sapiens.GRCh38.pep.all.fa Hsapiens.protein.fa

# Mouse - vGRCm38 - 59203 protein sequences (22770 genes)
wget \
${genome}/release-86/fasta/mus_musculus//pep/Mus_musculus.GRCm38.pep.all.fa.gz
gunzip Mus_musculus.GRCm38.pep.all.fa.gz
longest.sh Mus_musculus.GRCm38.pep.all.fa Mmusculus.protein.fa

# Prepare fasta files using OrthoMCL 2.0
orthomclAdjustFasta Pcon maker.proteins.longestORFs.faa 1
orthomclAdjustFasta Fcat Fcatus.protein.fa 1
orthomclAdjustFasta Cfam Cfamiliaris.protein.fa 1
orthomclAdjustFasta Amel Amelanoleuca.protein.fa.faa 1
orthomclAdjustFasta Btau Btaurus.protein.fa 1
orthomclAdjustFasta Hsap Hsapiens.protein.fa 1
orthomclAdjustFasta Mmus Mmusculus.protein.fa 1
mkdir ALL_SEQS
mv *.fasta ALL_SEQS
...

# Filter Fasta files:
# minumum length=10, max percent stop codons=20; these are defaults
orthomclFilterFasta ALL_SEQS 10 20
  # Resulting files are:
  # 'goodProteins.fasta' with 147,080 sequences
  # 'poorProteins.fasta' with 85 sequences

# Perform All vs All BLAST
makeblastdb \
  -in goodProteins.fasta \
  -dbtype prot \
  -parse_seqids \
  -out goodProteins.fasta

# Ran on a 16 core node, convert to a cluster script if necessary
blastp \
  -db goodProteins.fasta \
```

```
  -query goodProteins.fasta \
  -outfmt 6 \
  -num_threads 16 \
  -out blastresults.tsv

# Parse BLAST results
orthomclBlastParser blastresults.tsv ALL_SEQS >> parsedSequences.txt

# Configure SQL database; remove any old databases and make a new one
mysql -u username -p
  # enter password: password
DROP DATABASE orthomcl;
create database orthomcl;
exit

# Install database schema
orthomclInstallSchema mysql.config mysql.log

# Contents of mysql.config
  # dbVendor=mysql
  # dbConnectString=dbi:mysql:orthomcl:mysql_local_infile=1:localhost
  # dbLogin=username
  # dbPassword=password
  # similarSequencesTable=SimilarSequences
  # orthologTable=Ortholog
  # inParalogTable=InParalog
  # coOrthologTable=CoOrtholog
  # interTaxonMatchView=InterTaxonMatch
  # percentMatchCutoff=50
  # evalueExponentCutoff=-5
  # oracleIndexTblSpc=NONE

# Load BLAST results into database
orthomclLoadBlast mysql.config parsedSequences.txt

# Call pairs (potential orthologs, co-orthologs, and in-paralogs)
orthomclPairs mysql.config pairs.log cleanup=no
orthomclDumpPairsFiles mysql.config

# Run mcl program
mcl mclInput --abc -I 1.5 -o groups_1.5.txt

# Name the groups
orthomclMclToGroups OG1.5_ 1000 < groups_1.5.txt > named_groups_1.5.txt

# Summarize Results as Tables (see bash scripts below this code block)
CopyNumberGen.sh named_groups_1.5.txt > named_groups_1.5_frequency.txt
  # This table lists the count of each ortholog group in each genome
ExtractSCOs.sh named_groups_1.5_frequency.txt > scos_list.txt
  # This table lists the orthology groups that have a single copy in each species

# Get the IDs of the single-copy ortholog groups then extract the protein IDs
cut -f1 scos_list.txt | sed '1d' > ids.txt
c=1
while read line
  do
  echo "Processing id $c"
  grep -w "$line" named_groups_1.5.txt >> named_sco_groups_1.5.txt
  c=$(( $c + 1 ))
done < ids.txt

# Extract the sequences for the these single-copy orthologs
```

```
mkdir OG_SEQS
ExtractSeq.sh \
  -o OG_SEQS/ \
  named_sco_groups_1.5.txt \
  goodProteins.fasta

# Download all CDS sequences
...

# Make concatenated set of CDS sequences
cat *.cds.fa | seqtk seq -l0 -A > all.cds.fa
cat *.pep.orthomcl.fa | seqtk seq -l0 -A > all.prot.fa

# Get the CDS sequences for each alignment
a=$(grep "ENSP000002174 ../all.prot.fa | perl -ne '$_ =~ m/ transcript:([^ ]*) /;
  print "$1\n"')
grep "$a" ../all.cds.fa
```

Contents of the script *longest.sh*

```
#!/bin/bash

### Notes:
# Requires seqtk
# First argument is the input fasta file from ENSEMBL
# second argument is the name of an output fasta file
# Convert file to better format

seqtk seq -l0 $1 > tmp.seq

# Get list of uniq gene names
grep "^>" tmp.seq | \
  perl -ne '/ gene:([^ ]*) /; print "$1\n"' | \
  sort | \
  uniq > all.ids
c=$(grep -c "^" all.ids)
echo "Found $c unique gene names. Grabbing longest isoform of each"

# Grab longest isoform
n=1
while read l
do
grep "$l" tmp.seq | \
cut -f1 -d" " | \
sed "s/>//g" | \
seqtk subseq tmp.seq - | \
seqtk comp | \
cut -f1-2 | \
sort -nr -k2,2 | \
cut -f1 | \
head -1 | \
seqtk subseq tmp.seq - >> $2
echo "Finished $n sequences"
n=$(( $n + 1 ))
done < all.ids
rm tmp.seq all.ids
```

Contents of the script *CopyNumGen.sh*

```
#!/bin/bash
# This is a bash script that generates the table for number of orthologs
# present in each species.
# It takes the output generated by "orthomclMclToGroups" and
# converts ids to numbers. by default it prints to stdout
# Arun Seetharam aseetharam@purdue.edu

scriptName="${0##*/}"

function printUsage() {
  cat <<EOF

Synopsis

  $scriptName [-h | --help] input_file

Description

  Generates the count table from the orthologous group ids file
  generated by "orthomclMclToGroups"
  The count table gives gene copy number in all species for
  each othologous group.

  input_file
    Input file should contain orthologous group and IDs
    This file has to be generated by "orthomclMclToGroups" command

  -h, --help
    Brings up this help page

Author

  Arun Seetharam, Bioinformatics Core, Purdue University.
  aseetharam@purdue.edu


EOF
}

if [ $# -lt 1 ] ; then
  printUsage
  exit 1
fi

while getopts ':h:' option; do
  case "$option" in
  h) printUsage
    exit
    ;;
  help) printUsage
    exit
    ;;
  esac
done

file=${1};
sed \
  -e 's/ /" /g' -e 's/|/ "/g' -e 's/$/"/g' -e 's/:"/ /g' \
  ${file} > ${file}.temp # separate gene ids from the species identifier
```

```
names=`head -n 500 ${file}.temp | \
  tr -s " " "\n" |sed '/^".*/d'|sed '/^OG.*/d'| \
  sort |uniq |tr -s "\n" " "; echo ""` #array of all species names
echo -en "OG_name\t" #print the header line
for name in ${names[@]}; do
echo -en "$name\t";
done
echo "";
while read line; do #count frequency
ogroup=$(echo $line|cut -d " " -f 1 );
echo -en "$ogroup\t";
for name2 in ${names[@]}; do
freq=`echo $line | awk -F "$name2" '{print NF-1}'`;
echo -ne "$freq\t";
done
echo "";
done<${file}.temp;
rm ${file}.temp; #delete temp file
```

Contents of the script *ExtractSCOs.sh*

```
#!/bin/bash
# This is a bash script that extracts all ortholog groups
# having single copy gene in different species.
# It takes the output generated by "CopyNumberGen.sh" and
# prints single copy orthologs.
# by default it prints to stdout

# Arun Seetharam aseetharam@purdue.edu

scriptName="${0##*/}"
declare -i DEFAULT_COPY=1
declare -i copynum=DEFAULT_COPY
function printUsage() {
  cat <<EOF

Synopsis

  $scriptName [-h | --help] [-c number] input_file

Description

  Reads the count table and prints only lines having
  one copy gene in every genome.
  Output is printed to STDOUT

  input_file
    Input file should contain count table and ortholog group IDs
    This file has to be generated by "CopyNumberGen.sh" script.

  -c number
    By default, all orthologs groups that have single copy gene
    are printed, but you can specify the desired number of genes
    per ortholog group. Using 2 will print all ortholog groups having
    two copy genes per genome.

  -h, --help
    Brings up this help page
```

```
Author

   Arun Seetharam, Bioinformatics Core, Purdue University.
   aseetharam@purdue.edu


EOF
}

if [ $# -lt 1 ] ; then
   printUsage
   exit 1
fi

while getopts ':c:' option; do
   case "$option" in
   c) copynum=$OPTARG
      shift
      ;;
   h) printUsage
      exit
      ;;
```

Contents of the script *ExtractSeq.sh*

```
#!/bin/bash
# This is a bash script that extracts the sequences for all orthologous groups (OG).
# It takes the a OG ids list as input and saves all sequences belonging to that group
# from all organism in a file named with OG group in fasta format.
# Note that after the script is executed, there will be 'n' number of files (where
# n=total number of OG's in the input list

# Arun Seetharam aseetharam@purdue.edu

scriptName="${0##*/}"
outdir=$(pwd)
function printUsage() {
   cat <<EOF

Synopsis

   $scriptName [-h | --help] [-o dir_name] input_ids_list database

Description

   Extracts sequences for all ortholog groups supplied as list. For each
   ID in the list a file containing FASTA sequences will be generated,
   which belong to that OG.
   Note: this script requires standalone blast+ software.

   input_ids_list
      Input list should contain orthologous group IDs one per line
      These IDs should be generated by "orthomclMclToGroups" command

   database
      Absolute path for the database should be specified. The database is
      generally named as 'goodProteins.fasta'.

   -o directory_name
```

```
    directory name to save the output files. By default all files will
    be saved in the current directory.

  -h, --help
    Brings up this help page

Author

  Arun Seetharam, Bioinformatics Core, Purdue University.
  aseetharam@purdue.edu


EOF
}

if [ $# -lt 1 ] ; then
  printUsage
  exit 1
fi

while getopts ':o:' option; do
  case "$option" in
    o) outdir=$OPTARG
    shift
    ;;
    h) printUsage
    exit
    ;;
    help) printUsage
    exit
    ;;
  esac
done
#module use /apps/group/bioinformatics/modules # uncomment these 2 lines
#if running this on
#module load blast
mkdir -p $outdir
shift $(( $# - 2 ))
file=${1}
pathdbname=${2}
sed -i 's/://g' ${file}
while IFS=$' ' read -r -a myArray
do
for i in "${myArray[@]:0}";
do
blastdbcmd -entry "$i" -db ${pathdbname} >> ${outdir}/${myArray[0]}.fa;
done
done <${file}
```

Gene family expansion and contraction

We used the DupliPHY v.1.0 (Ames *et al.* 2012; Ames and Lovell 2015) to identify gene family expansions and contractions across the branches of a phylogenetic tree of the following species: puma, cat, dog, panda, cow, mouse, and human. Divergence times (i.e., branch lengths) were obtained from www.timetree.org and Ochoa *et al.* (2017). The input tree used was formatted as:

```
((((Pcon:5.6,Fcat:5.6):50.6,(Cfam:46.6,Amel:46.6):9.6):24.9,Btau:81.1):13.9,(Hsap:90.1,Mmus:90.1):4.9);
```

We excluded gene families absent in either the puma, cat, dog, panda, and cow clade or the mouse and human clade. Therefore, we performed this analysis with 15,666 of the 17,131 gene families defined above using OrthoMCL.

Positive selection

We examined 8210 single-copy orthologs (see the folder OG_SEQS in the code above) for positive selection within the puma lineage. First, we aligned the amino acid sequences of each single-copy ortholog across species with PRANK v.170427 (Löytynoja and Goldman 2005) with the guide tree as described above. Next, we used TrimAl v.1.4 (Capella-Gutiérrez *et al.* 2009) with the '-strictplus' algorithm to automatically determine the optimal gap and similarity scores to use when removing poorly aligned regions, and to convert the resulting cleaned alignment to the corresponding codon (nucleic acid) sequence alignment. The use of the '-strictplus' parameter is known to improve subsequent phylogenetic accuracy and generally outperforms other methods such as Gblocks v.0.91b (Castresana 2000) under default conditions (Capella-Gutiérrez *et al.* 2009).

We performed a 'branch test' on the aligned codon sequences using the evolutionary models implemented in the codeml program of PAML v.4.9 (Yang 1997, 2007). In this regard, we first calculated the likelihood of a null model where a single non-synonymous to synonymous ratio (d_N/d_S, or ω) was estimated across the entire tree (see tree above used for DupliPHY). Next, we calculated the likelihood of an alternative model where ω could vary on the puma branch (foreground lineage) as opposed to remaining fixed across all remaining branches (background lineages) of the tree. We compared the models with a likelihood ratio test to determine if the alternative model had an improved fit over the null model. We corrected the *P*-values associated with the multiple tests using the false discovery rate (FDR) method (Benjamini and Hochberg 1995). We only retained single-copy orthologs with an FDR < 0.05 and where the ω estimated by the alternative model was both greater in the puma lineage relative to the background ω and ω > 1.

Code for estimating models of selection on each single-copy ortholog:

```
# The following code was repeated for each ortholog ${OG}, 8210 total

# Set input fasta file of proteins sequences in an orthologous group (OG)
# See the files in the folder OG_SEQS created in the OrthoMCL analysis
OG=OG1.5_10000
mkdir ${OG}
cd ${OG}
cp ../OG_SEQS/${OG}.fa .
cp ../tree.dat .
# tree.dat
# ((((Pcon:5.6, Fcat:5.6):50.6, (Cfam:46.6, Amel:46.6):9.6):24.9,
# Btau:81.1):13.9, (Hsap:90.1, Mmus:90.1):4.9);

# Set location of fasta list of all protein sequences
prot=../goodProteins.fa # See OrthoMCL analysis above

# Set location of fasta list of all cds sequences
cds=../all.cds.fa

# Make prank input file
cp ${OG}.fa input.fa
sed -i "s/>.*|\(.*\)|.*$/>\1/" input.fa

# Align protein sequences using PRANK
module load GCC/5.4.0
prank \
```

```
   -d=input.fa \
   -t=tree.dat \
   -o=${OG} \
   -f=fasta \
   -F \
   -protein \
   -iterate=1

# Get protein ID, then transcript ID from the protein ID
peps=$(grep -f <(grep "^>" ${OG}.fa | \
   cut -d"|" -f3 | \
   cut -d" " -f1) ${prot} | \
   perl -ne \
   '$_ =~ m/ transcript:([^ ]*) |>(ENSAPCO[^ ]*) .*eAED:/;print "$1$2\n"')

# Grab the matching cds sequences and write to file
grep -A 1 -f <(echo "$peps") $cds | \
   grep -v '\-\-' | \
   seqtk seq -l0 > ${OG}.cds

# Change name formatting to match other files
sed -i \
   -e "s/ENSAMET.*$/Amel/" \
   -e "s/ENSBTAT.*$/Btau/" \
   -e "s/ENSCAFT.*$/Cfam/" \
   -e "s/ENSFCAT.*$/Fcat/" \
   -e "s/ENST.*$/Hsap/" \
   -e "s/ENSMUST.*$/Mmus/" \
   -e "s/ENSAPCO.*$/Pcon/" ${OG}.cds

# Clean the alignment with TRIMAL v1.4
trimal \
   -in ${OG}.best.fas \
   -out ${OG}.trimmed.paml \
   -strictplus \
   -backtrans ${OG}.cds \
   -ignorestopcodon \
   -phylip_paml

# Run codeml model M0 = np=14; model=0; NSSites=0; fix_omega=0; omega=1
# The 'xxxx' was just a dummy variable in the .ctl files to be
# replaced by the ortholog group ID
sed "s/xxxx/$OG/g" ../M0.ctl > ${OG}_M0.ctl
codeml ${OG}_M0.ctl

# Run codeml model M1 = np=13; model=2; NSSites=0; fix_omega=1; omega=1
sed "s/xxxx/$OG/g" ../M1.ctl > ${OG}_M1.ctl
codeml ${OG}_M1.ctl

# Run codeml model M2 = np=15; model=2; NSSites=0; fix_omega=0; omega=1
sed -i "s/Pcon/Pcon #1/" tree.dat
sed "s/xxxx/$OG/g" ../M2.ctl > ${OG}_M2.ctl
codeml ${OG}_M2.ctl

# Run codeml model M3 = np=16; model=2; NSSites=2; fix_omega=1; omega=1
sed "s/xxxx/$OG/g" ../M3.ctl > ${OG}_M3.ctl
codeml ${OG}_M3.ctl

# Run codeml model M4 = np=17; model=2; NSSites=2; fix_omega=0; omega=1
sed "s/xxxx/$OG/g" ../M4.ctl > ${OG}_M4.ctl
```

```
codeml ${OG}_M4.ctl

  # Use this command if you want to clean up the extra files
  rm -rf rst rst1 rub 2* lnf 4fold.nuc

# Get Likelihoods and Omega
M0lik=$(grep "lnL" ${OG}_M0.mlc | tr -s " " | cut -d" " -f5)
M1lik=$(grep "lnL" ${OG}_M1.mlc | tr -s " " | cut -d" " -f5)
M2lik=$(grep "lnL" ${OG}_M2.mlc | tr -s " " | cut -d" " -f5)
M3lik=$(grep "lnL" ${OG}_M3.mlc | tr -s " " | cut -d" " -f5)
M4lik=$(grep "lnL" ${OG}_M4.mlc | tr -s " " | cut -d" " -f5)
omegas=$(grep "w (dN/dS) for branches:" ${OG}_M2.mlc | cut -d" " -f6-7)
echo "${OG} $M0lik $M1lik $M2lik $M3lik $M4lik $omegas" >> out.liks

# Repeat for all single-copy orthologs
# 14 sequences failed, so replace with NA
sed -i "s/ / NA NA NA NA NA NA NA/g" out.liks

# In R
# Make function to calculate likelihood ratio test
LRT = function(L1, L0, df1, df0) 1 - pchisq(2 * (L1 - L0), df = (df1 - df0))

# Get P values for likelihood ratio test
# Adjust for multiple comparisons
data = read.table("out.liks", sep = " ", header = F)
pval = LRT(data$V4, data$V2, 15, 14)
fdr = p.adjust(pval, method = "fdr")
data = cbind(data, pval, fdr)
colnames(data) = c("Ortholog", "M0_likelihood",
  "M1_likelihood", "M2_likelihood", "M3_likelihood",
  "M4_likelihood", "Omega_BG", "Omega_Puma", "P_LRT", "FDR_LRT")

# Write full output to table
write.table(data, file = "Full.table.tsv", sep = "\t",
  col.names = T, row.names = F, quote = F)

# Get list of positively selected genes
b = subset(data, FDR_LRT < 0.05 & Omega_Puma > Omega_BG & Omega_Puma > 1)
write.table(b, file = "Selected.table.tsv", sep = "\t",
  col.names = T, row.names = F, quote = F)
```

Finally, make a list of the genes under positive selection by converting the ortholog ID to the puma genome protein ID:

```
while read i
do
OG=$(echo "$i" | cut -f1)
cd "$OG"
grep "Pcon" ${OG}.fa | \
perl -ne '$_ =~ m/(ENSAPCO[^ ]*) /; print "$1\n"' >> ../genes.selected.txt.list
cd ..
done < <(sed '1d' Selected.table.tsv)
```

GO enrichment tests

We performed tests for the enrichment of GO terms (biological process only) in expanded gene families (Table S5), contracted (Table S6) gene families, and genes under positive selection (Table S7) using Blast2GO. We calculated *P*-values for each GO term using Fisher's exact test and corrected for multiple comparisons using the FDR method. We omitted GO terms assigned to ≤ 10 genes in the complete, reference gene set. Additionally, we identified 17 positively selected genes (FDR < 0.05) associated with the refinement of sensory perceptions (Table S8).

Table S5. Biological processes enriched in puma expanded gene families.

GO ID	GO name	FDR
GO:0002119	nematode larval development	4.85E-14
GO:0086010	membrane depolarization during action potential	5.25E-09
GO:0040035	hermaphrodite genitalia development	2.93E-08
GO:1902287	semaphorin-plexin signaling pathway involved in axon guidance	4.70E-08
GO:0019228	neuronal action potential	1.76E-07
GO:0021785	branchiomotor neuron axon guidance	3.55E-06
GO:0060078	regulation of postsynaptic membrane potential	4.68E-05
GO:0008340	determination of adult lifespan	1.80E-04
GO:1904764	chaperone-mediated autophagy translocation complex disassembly	0.001146
GO:0061741	chaperone-mediated protein transport involved in chaperone-mediated autophagy	0.001146
GO:0009792	embryo development ending in birth or egg hatching	0.005005
GO:0072318	clathrin coat disassembly	0.005035
GO:1904592	positive regulation of protein refolding	0.007998
GO:0007339	binding of sperm to zona pellucida	0.007998
GO:0097214	positive regulation of lysosomal membrane permeability	0.007998
GO:0009258	10-formyltetrahydrofolate catabolic process	0.007998
GO:0061740	protein targeting to lysosome involved in chaperone-mediated autophagy	0.007998
GO:0006085	acetyl-CoA biosynthetic process	0.008822
GO:1902667	regulation of axon guidance	0.009135
GO:0030240	skeletal muscle thin filament assembly	0.010867
GO:0048532	anatomical structure arrangement	0.010867
GO:0061738	late endosomal microautophagy	0.010867
GO:0048026	positive regulation of mRNA splicing, via spliceosome	0.013389
GO:0010501	RNA secondary structure unwinding	0.017941
GO:0090084	negative regulation of inclusion body assembly	0.022012
GO:0061635	regulation of protein complex stability	0.022012
GO:0045899	positive regulation of RNA polymerase II transcriptional preinitiation complex assembly	0.023092
GO:1900245	positive regulation of MDA-5 signaling pathway	0.025474
GO:0048677	axon extension involved in regeneration	0.025474
GO:0034765	regulation of ion transmembrane transport	0.028914
GO:0051084	*de novo* posttranslational protein folding	0.036777

Table S6. Biological processes enriched in puma contracted gene families.

GO ID	GO name	FDR
GO:0050911	detection of chemical stimulus involved in sensory perception of smell	8.32E-25
GO:0007186	G protein-coupled receptor signaling pathway	8.40E-16
GO:0043547	positive regulation of GTPase activity	2.24E-08
GO:0045332	phospholipid translocation	3.92E-08
GO:0007229	integrin-mediated signaling pathway	8.30E-08
GO:0038083	peptidyl-tyrosine autophosphorylation	2.20E-06
GO:0051453	regulation of intracellular pH	5.05E-06
GO:0007223	Wnt signaling pathway, calcium modulating pathway	1.88E-05
GO:0048843	negative regulation of axon extension involved in axon guidance	3.35E-05
GO:0045653	negative regulation of megakaryocyte differentiation	8.59E-05
GO:0015701	bicarbonate transport	9.49E-05
GO:0050919	negative chemotaxis	1.39E-04
GO:0000160	phosphorelay signal transduction system	1.50E-04
GO:0070588	calcium ion transmembrane transport	2.10E-04
GO:1902476	chloride transmembrane transport	4.12E-04
GO:0018105	peptidyl-serine phosphorylation	4.64E-04
GO:0006355	regulation of transcription, DNA-templated	5.48E-04
GO:0060012	synaptic transmission, glycinergic	7.18E-04
GO:0060341	regulation of cellular localization	8.68E-04
GO:0030335	positive regulation of cell migration	0.001053
GO:0048013	ephrin receptor signaling pathway	0.001071
GO:0045995	regulation of embryonic development	0.001763
GO:0008544	epidermis development	0.001829
GO:1904322	cellular response to forskolin	0.002031
GO:0060384	Innervation	0.002184
GO:2000811	negative regulation of anoikis	0.002559
GO:0071526	semaphorin-plexin signaling pathway	0.003020
GO:0010976	positive regulation of neuron projection development	0.003715
GO:0051241	negative regulation of multicellular organismal process	0.003753
GO:0007044	cell-substrate junction assembly	0.004039
GO:0001676	long-chain fatty acid metabolic process	0.004801
GO:0009582	detection of abiotic stimulus	0.006017
GO:0030155	regulation of cell adhesion	0.006112
GO:0000165	MAPK cascade	0.006218
GO:0009581	detection of external stimulus	0.006841
GO:0090330	regulation of platelet aggregation	0.006856
GO:0006335	DNA replication-dependent nucleosome assembly	0.006856
GO:0007420	brain development	0.006867
GO:0034080	CENP-A containing nucleosome assembly	0.008177
GO:0046942	carboxylic acid transport	0.009531
GO:0043535	regulation of blood vessel endothelial cell migration	0.010506
GO:0001707	mesoderm formation	0.011833
GO:0043113	receptor clustering	0.013351
GO:0048015	phosphatidylinositol-mediated signaling	0.013443
GO:0051924	regulation of calcium ion transport	0.013925
GO:0090625	mRNA cleavage involved in gene silencing by siRNA	0.013984
GO:0000088	mitotic prophase	0.013984
GO:1903804	glycine import across plasma membrane	0.013984
GO:0022007	convergent extension involved in neural plate elongation	0.013984
GO:0042391	regulation of membrane potential	0.014115
GO:0051290	protein heterotetramerization	0.014966
GO:0060429	epithelium development	0.015268
GO:0050808	synapse organization	0.015897
GO:0046578	regulation of Ras protein signal transduction	0.017246
GO:0071805	potassium ion transmembrane transport	0.018116
GO:0022604	regulation of cell morphogenesis	0.019422
GO:0010646	regulation of cell communication	0.020267
GO:0007163	establishment or maintenance of cell polarity	0.021449
GO:2000641	regulation of early endosome to late endosome transport	0.021687
GO:0008206	bile acid metabolic process	0.021718

Table S6 (continued). Biological processes enriched in puma contracted gene families.

GO ID	GO name	FDR
GO:0023051	regulation of signaling	0.021938
GO:0018107	peptidyl-threonine phosphorylation	0.021938
GO:0001505	regulation of neurotransmitter levels	0.021969
GO:0001952	regulation of cell-matrix adhesion	0.023182
GO:0007635	chemosensory behavior	0.025870
GO:0015824	proline transport	0.025870
GO:0006171	cAMP biosynthetic process	0.025870
GO:0045636	positive regulation of melanocyte differentiation	0.025870
GO:0051592	response to calcium ion	0.026521
GO:0007626	locomotory behavior	0.026524
GO:0060402	calcium ion transport into cytosol	0.026552
GO:0045214	sarcomere organization	0.026552
GO:0002040	sprouting angiogenesis	0.026949
GO:0030216	keratinocyte differentiation	0.030520
GO:0048583	regulation of response to stimulus	0.031763
GO:0060359	response to ammonium ion	0.031943
GO:0043666	regulation of phosphoprotein phosphatase activity	0.033179
GO:0046854	phosphatidylinositol phosphorylation	0.033179
GO:0007156	homophilic cell adhesion via plasma membrane adhesion molecules	0.033215
GO:0021700	developmental maturation	0.036510
GO:0043551	regulation of phosphatidylinositol 3-kinase activity	0.036798
GO:0060038	cardiac muscle cell proliferation	0.036798
GO:0031047	gene silencing by RNA	0.036859
GO:1904714	regulation of chaperone-mediated autophagy	0.041129
GO:0035279	mRNA cleavage involved in gene silencing by miRNA	0.041129
GO:0048319	axial mesoderm morphogenesis	0.041129
GO:0097264	self proteolysis	0.041129
GO:0070837	dehydroascorbic acid transport	0.041129
GO:0008286	insulin receptor signaling pathway	0.042338
GO:0044539	long-chain fatty acid import into cell	0.043039
GO:0035278	miRNA mediated inhibition of translation	0.043039
GO:0048339	paraxial mesoderm development	0.043039
GO:0033700	phospholipid efflux	0.043039
GO:0050848	regulation of calcium-mediated signaling	0.043484
GO:0040029	regulation of gene expression, epigenetic	0.044040
GO:0000183	chromatin silencing at rDNA	0.044113
GO:0035235	ionotropic glutamate receptor signaling pathway	0.044113
GO:1904837	beta-catenin-TCF complex assembly	0.044113
GO:0071801	regulation of podosome assembly	0.044113
GO:2000573	positive regulation of DNA biosynthetic process	0.045062
GO:0050891	multicellular organismal water homeostasis	0.046455
GO:0048016	inositol phosphate-mediated signaling	0.046455
GO:0035196	production of miRNAs involved in gene silencing by miRNA	0.047825
GO:0042327	positive regulation of phosphorylation	0.048253
GO:1903076	regulation of protein localization to plasma membrane	0.048253
GO:0014031	mesenchymal cell development	0.049212

Table S7. Biological processes enriched in puma positively selected genes.

GO ID	GO name	FDR
GO:0006614	SRP-dependent cotranslational protein targeting to membrane	0.014949
GO:0000184	nuclear-transcribed mRNA catabolic process, nonsense-mediated decay	0.025693
GO:0019083	viral transcription	0.033769

Table S8. Positively selected genes with biological processes related to sensory perceptions in pumas.

Gene ID	GO ID	GO name
Pcon00085	GO:0048592	eye morphogenesis
Pcon01782	GO:0010842	retina layer formation
Pcon02269	GO:0007601	visual perception
	GO:0060219	camera-type eye photoreceptor cell differentiation
	GO:0061298	retina vasculature development in camera-type eye
Pcon07031	GO:0007601	visual perception
	GO:0007602	Phototransduction
Pcon08773	GO:0002088	lens development in camera-type eye
Pcon10399	GO:0050911	detection of chemical stimulus involved in sensory perception of smell
Pcon11092	GO:0007601	visual perception
	GO:0045494	photoreceptor cell maintenance
Pcon11179	GO:0007605	sensory perception of sound
	GO:0060117	auditory receptor cell development
Pcon11335	GO:0042462	eye photoreceptor cell development
	GO:0050908	detection of light stimulus involved in visual perception
	GO:0060041	retina development in camera-type eye
Pcon12456	GO:0061303	cornea development in camera-type eye
Pcon12566	GO:0007605	sensory perception of sound
Pcon13543	GO:0046619	optic placode formation involved in camera-type eye formation
	GO:0048839	inner ear development
	GO:0060059	embryonic retina morphogenesis in camera-type eye
	GO:0070309	lens fiber cell morphogenesis
Pcon19057	GO:0002088	lens development in camera-type eye
	GO:0003406	retinal pigment epithelium development
Pcon19084	GO:0007186	G protein-coupled receptor signaling pathway
	GO:0007608	sensory perception of smell
Pcon19454	GO:0031290	retinal ganglion cell axon guidance
Pcon20022	GO:0045494	photoreceptor cell maintenance
Pcon20260	GO:0007186	G protein-coupled receptor signaling pathway
	GO:0050911	detection of chemical stimulus involved in sensory perception of smell

Identification of SNPs and Intraspecific Analyses

Read mapping to reference

In order to identify SNPs and further assess the quality of the de novo assembly, we mapped the trimmed and error-corrected PE reads to the assembly using BWA v.0.7.9 (Li and Durbin 2009). To this extent, we first indexed the *de novo* reference genome using:

```
bwa index -a bwtsw Pco-scaffolds-500.fa
```

We then mapped the processed reads, removed duplicates, and retained only properly paired reads with high mapping qualities (Phred-scaled mapping quality [MQ] ≥ 20). We used SAMtools v.1.1 (Li *et al.* 2009) to create corresponding BAM files.

Code for mapping reads (only individual FP16 is shown as an example):

```
# Set up some variables
puma=FP16
rg='@RG\tID:1\tPL:illumina\tPU:ALEX_OCHOA\tLB:n/a\tSM:FP16\tCN:UAGC'
ref="Pco-scaffolds-500.fa"
```

```
# Run BWA (PE reads); -M is for picard/gatk compatibility
# result is a sorted BAM file
bwa mem \
  -M \
  -t 16 \
  -R $rg \
  -v 3 \
  $ref \
  $puma.PE.R1.cor.fastq.gz \
  $puma.PE.R2.cor.fastq.gz | \
  samtools view -Shu - | \
  samtools sort - $puma.PE.sorted

# Now to Remove duplicates, and filter for only mapped reads with a MQ>20
samtools rmdup $puma.PE.sorted.bam - | \
  samtools view \
  -b \
  -q 20 \
  -f 0x0002 \
  -F 0x0004 \
  -F 0x0008 - > $puma.sorted.PE.rmdup.mq20.bam

# Index the final bam file
samtools index $puma.sorted.PE.rmdup.mq20.bam

# Produce simple statistics after cleaning
samtools stats $puma.sorted.PE.rmdup.mq20.bam > $puma.bamstats
```

Table S9. Summary of processed reads mapped to the puma genome assembly.

Sample	Mapped reads	Mean length of the aligned segment of each read (bp)	Sequence coverage (x)
FP16	227,742,373	98.35	8.6
FP45	207,626,232	97.91	7.8
FP60	246,974,466	98.56	9.4
FP73	195,225,872	98.04	7.4
FP79	157,850,474	98.06	6.0
TX101	196,532,789	97.95	7.4
TX105	178,103,944	98.09	6.7
TX106	156,018,643	98.21	5.9
TX107	182,207,581	98.34	6.9
TX108	203,055,912	98.30	7.7
Total	1,951,338,286	98.19	73.9

SNP calling

We identified variants using 'HaplotypeCaller' within the GATK v.3.8.0 (McKenna *et al.* 2010) suite of genomic tools. Prior to SNP discovery, we also included an InDel realignment step within GATK ('IndelRealigner'). Although computationally more intensive, the inclusion of local realignment algorithms does improve overall variant identification sensitivity and quality whilst reducing false-positive SNPs (Baes *et al.* 2014).

Code for calling SNPs:

```
# Repeat for each of the 10 puma BAM files
puma=FP16
ref="Pco-scaffolds-500.fa"
java -jar GenomeAnalysisTK.jar \
  -T RealignerTargetCreator \
  -R ${ref} \
  -I ${puma}.sorted.PE.rmdup.mq20.bam \
  -o ${puma}.intervals

# Perform indel realignment
java -jar GenomeAnalysisTK.jar \
  -T IndelRealigner \
  -R ${ref} \
  -I ${puma}.sorted.PE.rmdup.mq20.bam \
  -targetIntervals ${puma}.intervals \
  -o ${puma}.sorted.rmdup.mq20.realigned.bam

# Identifiy variants using the GATK HaplotypeCaller
java -jar GenomeAnalysisTK.jar \
  -T HaplotypeCaller \
  -R ${ref} \
  -I 16.sorted.rmdup.mq20.realigned.bam \
  -I 45.sorted.rmdup.mq20.realigned.bam \
  -I 60.sorted.rmdup.mq20.realigned.bam \
  -I 73.sorted.rmdup.mq20.realigned.bam \
  -I 79.sorted.rmdup.mq20.realigned.bam \
  -I 101.sorted.rmdup.mq20.realigned.bam \
  -I 105.sorted.rmdup.mq20.realigned.bam \
  -I 106.sorted.rmdup.mq20.realigned.bam \
  -I 107.sorted.rmdup.mq20.realigned.bam \
  -I 108.sorted.rmdup.mq20.realigned.bam \
  -o SNPs_raw.vcf
```

Next, we filtered the raw set of variants with GATK and VCFtools v.0.1.12 (Danecek *et al.* 2011) to obtain a set of high quality variants that included the following criteria:

- They must be biallelic SNPs (no insertions/deletions).
- FS ≤ 60; Phred-scaled *P*-value using Fisher's exact test to detect strand bias.
- MQ ≥ 40; root mean square of the mapping quality.
- MQRankSum ≥ −12.5; *Z*-score from Wilcoxon rank sum test of alternate *vs.* reference read mapping qualities.
- QD ≥ 2; variant confidence/quality by depth.
- ReadPosRankSum ≥ −8.0; allele specific *Z*-score from Wilcoxon rank sum test of each alternate *vs.* reference read position bias.
- SNPs cannot be in a cluster of ≥ 3 (--clusterSize) within a 20 bp window (--clusterWindowSize).
- Genotypes are only called within individuals as long as the site depth is ≥ 4 and ≤ 25.

```
# GATK code for selecting only biallelic SNPs:
java -jar GenomeAnalysisTK.jar \
   -T SelectVariants \
   -R ${ref} \
   -V SNPs_raw.vcf \
   -o SNPs.biallelic.vcf \
   -selectType SNP \
   --restrictAllelesTo BIALLELIC

# GATK code for filtering the variants:
java -jar GenomeAnalysisTK.jar \
   -T VariantFiltration \
   -R ${ref} \
   -V SNPs_biallelic.vcf \
   -o SNPs_biallelic.filtered.vcf \
   --filterExpression "FS > 60.0 || \
      MQ < 40.0 || \
      MQRankSum < -12.5 || \
      QD < 2.0 || \
      ReadPosRankSum < -8.0" \
   --filterName "LOW_QUALITY" \
   --clusterSize 3 \
   --clusterWindowSize 20

# VCFtools code for removing "LOW_QUALITY" SNPs and filter for depth:
vcftools \
   --vcf SNPs_biallelic.filtered.vcf \
   --remove-filtered-all \
   --minDP 4 \
   --maxDP 25 \
   --out SNPs_biallelic.filtered.PASS.vcf
```

We retained SNPs with genotypes present across each of the following groups: FL (samples FP45 and FP60), TX (samples TX101, TX105, TX106, TX107, and TX108), F_1 (samples FP73 and FP79), and sample FP16. We analyzed the final subset of SNPs for the ratio of transitions (pyrimidine ↔ pyrimidine or purine ↔ purine) to transversions (purine ↔ pyrimidine), or Ti/Tv ratio, using VCFtools. The Ti/Tv ratio is often employed as a measure of overall SNP quality, as ratios similar to that observed in humans (~2.1) are indicative of a generally high prediction accuracy (DePristo *et al.* 2011; Liu *et al.* 2012; Baes *et al.* 2014). We calculated the observed heterozygosity for each individual as the number of heterozygous genotypes scaled to the total number of validated genotypes for that sample within the final subset of retained SNPs (Table S10).

Table S10. Observed heterozygosity in Florida panthers and Texas pumas.

Sample	Number of heterozygous genotypes	Total number of genotypes examined	Observed heterozygosity
FP16	1,777,252	6,210,080	0.286
FP45	411,586	5,891,667	0.070
FP60	521,319	5,994,401	0.087
FP73	1,501,071	5,792,823	0.259
FP79	1,238,332	5,516,466	0.224
TX101	1,535,069	5,826,006	0.263
TX105	1,352,753	5,732,476	0.236
TX106	1,090,312	5,468,882	0.199
TX107	1,281,824	5,811,401	0.221
TX108	1,508,511	5,841,552	0.258

Demographic history

We examined the demographic history of each panther sample individually using PSMC v.0.6.4 (Li and Durbin 2011). In essence, the demographic history, inferred by the effective population size, is reconstructed based on the pattern and distribution of coalescent events (heterozygous sites) across a single diploid genome. We used the final set of filtered variants in a conservative, or strict, approach as employed by Fitak *et al.* 2016b. This approach, which included the omission of annotated repetitive elements (see *Repetitive elements and non-coding RNAs* above) is shown to produce nearly identical results to the standard variant-identification pipeline implemented in SAMtools (see Li and Durbin 2011). We ran PSMC below for each puma separately and ran 100 bootstrap replicates for each. We ran 25 iterations, a -t15, an initial theta/rho ratio (-r) of 5, and time interval parameter -p "4+25*2+4+6" to infer ~10 recombination events in each interval (Li and Durbin 2011).

Code for inferring demographic history:

```
# First compress and index the vcf file
bgzip \
   -c SNPs_biallelic.filtered.PASS.vcf > SNPs_biallelic.filtered.PASS.vcf.gz
tabix \
   -p vcf \
   SNPs_biallelic.filtered.PASS.vcf.gz

# Set name of individual to be analyzed
puma=FP16

# Make a vcf file for each individual
zcat SNPs_biallelic.filtered.PASS.vcf.gz | \
   vcf-subset -e -c ${puma} | \
   grep -e "^#" -e "0/1" | \
   bgzip > ${puma}.vcf.gz
tabix -p vcf ${puma}.vcf.gz

# Make a fasta file for each individual, masking repetitive elements
cat ${ref} | \
   seqtk seq -M repeats.bed -A - | \
   vcf-consensus -i ${puma}.vcf.gz > ${puma}.fa

# Convert fasta to psmc fasta format
fq2psmcfa ${puma}.fa > ${puma}.psmcfa

# Split for bootstrapping
splitfa ${puma}.psmcfa > ${puma}.split.psmcfa

# Run PSMC on the genome
psmc \
   -N25 \
   -t15 \
   -r5 \
   -p "4+25*2+4+6" \
   -o ${puma}.psmc \
   ${puma}.psmcfa

# Run bootstrap replicates
for i in {1..100}
   do
   psmc \
      -N25 \
      -t15 \
      -r5 \
```

```
    -b \
    -p "4+25*2+4+6" \
    -o ${puma}.$i.psmc \
    ${puma}.split.psmcfa
done

# Concatenate full and bootstrapped files for each individual
cat \
  ${puma}.psmc \
  ${puma}.{1..100}.psmc > ${puma}.boostrapped.psmc
```

Next, we made a plot across all individuals (Figure S9). We scaled the final effective population size estimates to a generation time of three years and a mutation rate of 6.6×10^{-9} substitutions site^{-1} generation^{-1} (2.2×10^{-9} substitutions site^{-1} year^{-1}; Kumar and Subramanian 2002).

```
# Plot PSMC results for sample FP16 only
puma=FP16
psmc_plot.pl \
  -P top \
  -X2000000 \
  -u6.6e-09 \
  -p \
  -g3 \
  -T ${puma} \
  -x1000 \
  ${puma}.plot \
  ${puma}.boostrapped.psmc

# Plot PSMC Results for all individuals
psmc_plot.pl \
  -P top \
  -X2000000 \
  -u6.6e-09 \
  -p \
  -R \
  -g3 \
  -x1000 \
  -M FP16,FP45,FP60,FP73,FP79,TX101,TX105,TX106,TX107,TX108 \
  All.pumas.plot \
  FP16.psmc \
  FP45.psmc \
  FP60.psmc \
  FP73.psmc \
  FP79.psmc \
  TX101.psmc \
  TX105.psmc \
  TX106.psmc \
  TX107.psmc \
  TX108.psmc
```

Alternatively, plot in R using the following code:

```
# Load list of individual IDs
id = scan("IDs.txt", what = "character")

# Setup the plot parameters
pdf("FigS9.psmc.pdf", width = 7, height = 5)
plot.new()
plot.window(xlim = c(1000, 2000000), ylim = c(0, 8), log = "x")
axis(1, tck = -0.01, at = c(1000, 10000, 100000,1000000),
  labels = expression(10^3, 10^4, 10^5, 10^6))
axis(1, tck = -0.01,
  at = c(1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000,
  10000, 20000, 30000, 40000, 50000, 60000, 70000, 80000,
  90000, 100000, 200000, 300000, 400000, 500000, 600000, 700000,
  800000, 900000, 1000000), labels = F)
y.axis = seq(0,8, by = 0.1)
axis(2, las = 1)
axis(2, las = 1, tck = -0.01, at = y.axis, labels = F)
box()
mtext(side = 2,
  expression(paste("Effective Population Size (x", "10"^"4",")")),
  line = 2.5, cex = 1.25)
mtext(side = 1, "Time (Years ago)", line = 2.5, cex = 1.25)
col = c("#67001f", "#b2182b", "#d6604d", "#f4a582", "#fddbc7",
  "#d1e5f0", "#92c5de", "#4393c3", "#2166ac", "#053061")
line.type = rep(c(1,3), each = 5)

# Plot each individual
for (i in 0:(length(id) - 1)){
  file = paste("All.pumas.plot.", i, ".txt", sep = "")
  b = read.table(file)
  points(b$V1, b$V2, type = "l",lwd = 3,lty=line.type[i+1], col=col[i+1])
}

# Plot the legend
legend("topleft", legend = id, col = col, lty = line.type,
  bty = "n", lwd = 3, cex = 1.2)
dev.off()
```

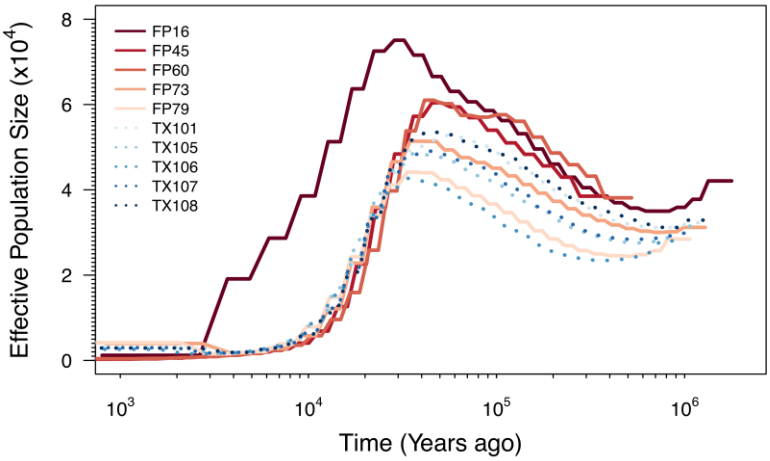

Figure S9. Historical changes in effective population sizes of the puma as examined for each puma sample used in this study.

References

Akogwu, I., N. Wang, C. Zhang, P. Gong, 2016 A comparative study of *k*-spectrum-based error correction methods for next-generation sequencing data analysis. Hum. Genomics 10: 20.

Ames, R.M., and S.C. Lovell, 2015 DupliPHY-Web: a web server for DupliPHY and DupliPHY-ML. Bioinformatics 31: 416–417.

Ames, R.M., D. Money, V.P. Ghatge, S. Whelan, and S.C. Lovell, 2012 Determining the evolutionary history of gene families. Bioinformatics 28: 48–55.

Baes, C.F., M.A. Dolezal, J.E. Koltes, B. Bapst, E. Fritz-Waters *et al*., 2014 Evaluation of variant identification methods for whole genome sequencing data in dairy cattle. BMC Genomics 15: 948.

Benjamini, Y., and Hochberg Y, 1995 Controlling the false discovery rate: a practical and powerful approach to multiple testing. J. Royal Stat. Soc., Series B (Methodological) 57: 289–300.

Bolger, A.M., M. Lohse, B. Usadel, 2014 Trimmomatic: a flexible trimmer for Illumina sequence data. Bioinformatics 30: 2114–2120.

Cantarel, B.L., I. Korf, S.M.C. Robb, G. Parra, E. Ross *et al.*, 2008 MAKER: an easy-to-use annotation pipeline designed for emerging model organism genomes. Genome Res. 18: 188–196.

Capella-Gutiérrez, S., J.M. Silla-Martínez, T. Gabaldón, 2009 TrimAl: a tool for automated alignment trimming in large-scale phylogenetic analyses. Bioinformatics 25: 1972–1973.

Castresana, J., 2000 Selection of conserved blocks from multiple alignments for their use in phylogenetic analysis. Mol. Biol. Evol. 17: 540–552.

Conesa, A., S. Götz, J.M. García-Gómez, J. Terol, M. Talón *et al.*, 2005 Blast2GO: a universal tool for annotation, visualization and analysis in functional genomics research. Bioinformatics 21: 3674–3676.

Danecek, P., A. Auton, G. Abecasis, C.A. Albers, E. Banks *et al.*, 2011 The variant call format and VCFtools. Bioinformatics 27: 2156–2158.

DePristo, M.A., E. Banks, R. Poplin, K.V. Garimella, J.R. Maguire *et al.*, 2011 A framework for variation discovery and genotyping using nextgeneration DNA sequencing data. Nature Genet. 43: 491–498.

Fitak, R.R., A. Naidu, R.W. Thompson, and M. Culver, 2016a A new panel of SNP markers for the individual identification of North American pumas. J. Fish. Wildl. Manag. 7: 13–27.

Fitak, R.R., E. Mohandesan, J. Corander, and P.A. Burger, 2016b The *de novo* genome assembly and annotation of a female domestic dromedary of North African origin. Mol. Ecol. Res. 16: 314–324.

Griffiths-Jones, S., A. Bateman, M. Marshall, A. Khanna, S.R. Eddy, 2003 Rfam: an RNA family database. Nucleic Acids Res. 31: 439–441.

Heydari, M., G. Miclotte, P. Demeester, Y. Van de Peer, J. Fostier, 2017 Evaluation of the impact of Illumina error correction tools on de novo genome assembly. BMC Bioinformatics 18: 374.

Holt, C., and M. Yandell, 2011 MAKER2: an annotation pipeline and genome-database management tool for second generation genome projects. BMC Bioinformatics 12: 491.

Johnson, W.E., D.P. Onorato, M.E. Roelke, E.D. Land, M. Cunningham *et al.*, 2010 Genetic restoration of the Florida panther. Science 329: 1641–1645.

Jones, P., D. Binns, H.-Y. Chang, M. Fraser, W. Li *et al.*, 2014 InterProScan 5: genome-scale protein function classification. Bioinformatics 30: 1236–1240.

Jurka, J., V.V. Kapitonov, A. Pavlicek, P. Klonowski, O. Kohany *et al.*, 2005 Repbase Update, a database of eukaryotic repetitive elements. Cytogenet. Genome Res. 110: 462–467.

Korf, I., 2004 Gene finding in novel genomes. BMC Bioinformatics 5: 59.

Kumar, S., and S. Subramanian, 2002 Mutation rates in mammalian genomes. Proc. Natl. Acad. Sci. USA. 99: 803–808.

Li, H., and R. Durbin, 2009 Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics 25: 1754–1760.

Li, H., and R. Durbin, 2011 Inference of human population history from whole genome sequence of a single individual. Nature 475: 493–496.

Li, H., B. Handsaker, A. Wysoker, T. Fennell, J. Ruan *et al.*, 2009 The Sequence alignment/map (SAM) format and SAMtools. Bioinformatics 25: 2078–2079.

Li, L., C.J. Stoeckert, and D.S. Roos, 2003 OrthoMCL: identification of ortholog groups for eukaryotic genomes. Genome Res. 13: 2178–2189.

Liu, Q., Y. Guo, J. Li, J. Long, B. Zhang, *et al.*, 2012 Steps to ensure accuracy in genotype and SNP calling from Illumina sequencing data. BMC Genomics 13: S8.
Liu, Y., J. Schröder, and B. Schmidt, 2013 Musket: a multistage k-mer spectrum-based error corrector for Illumina sequence data. Bioinformatics 29: 308–315.
Lomsadze, A., V. Ter-Hovhannisyan, Y. Chernoff, M. Borodovsky, 2005 Gene identification in novel eukaryotic genomes by self-training algorithm. Nucleic Acids Res. 33: 6494–6506.
Löytynoja, A., and N. Goldman, 2005 An algorithm for progressive multiple alignment of sequences with insertions. Proc. Natl. Acad. Sci. USA. 102: 10557–10562.
McKenna, A., M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis *et al.*, 2010 The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome Res. 20: 1297–1303.
Nawrocki, E.P., and S.R. Eddy, 2013 Infernal 1.1: 100-fold faster RNA homology searches. Bioinformatics 29: 2933–2935.
Ochoa, A., D.P. Onorato, R. Fitak, M.E. Roelke-Parker, and M. Culver, 2017 Evolutionary and functional mitogenomics associated with the genetic restoration of the Florida panther. J. Hered. 108: 449–455.
Parra, G., K. Bradnam, and I. Korf, 2007 CEGMA: a pipeline to accurately annotate core genes in eukaryotic genomes. Bioinformatics 23: 1061–1067.
Salzberg, S.L., A.M. Phillippy, A. Zimin, D. Puiu, T. Magoc *et al.*, 2012 GAGE: A critical evaluation of genome assemblies and assembly algorithms. Genome Res. 22: 557–567.
Simpson, J.T., K. Wong, S.D. Jackman, J.E. Schein, S.J.M. Jones *et al.*, 2009 ABySS: A parallel assembler for short read sequence data. Genome Res. 19: 1117–1123.
Smit, A.F.A., R. Hubley, and P. Green, 1996–2010 RepeatMasker Open-3.0. http://www.repeatmasker.org.
Stanke, M., A. Tzvetkova, and B. Morgenstern, 2006 AUGUSTUS at EGASP: using EST, protein and genomic alignments for improved gene prediction in the human genome. Genome Biol. 7: S11.
Yang, Z., 1997 PAML: a program package for phylogenetic analysis by maximum likelihood. Bioinformatics 13: 555–556.
Yang, Z., 2007 PAML 4: phylogenetic analysis by maximum likelihood. Mol. Biol. Evol. 24: 1586–1591.