VARIATION DISCOVERY AND EVOLUTIONARY PATTERNS OF PROTEIN DOMAIN FAMILIES IN 15 SPECIES OF BIRDS

Estudiante: Claudia Buhigas Novoa

MÁSTER EN BIOINFORMÁTICA Y BIOLOGÍA COMPUTACIONAL
ESCUELA NACIONAL DE SALUD- INSTITUTO DE SALUD CARLOS III

2014-2015

CENTRO/EMPRESA DONDE SE DESALLORARON LAS PRACTICAS
European Molecular Biology Laboratory, European Bioinformatics Institute (EMBL-EBI)
Wellcome Trust Genome Campus, Hinxton, UK
DIRECTOR DE LA TESIS: Dr. David Martín Gálvez, Dr. Paul Flicek
FECHA DE PRESENTACIÓN: Enero 2015
<table>
<thead>
<tr>
<th>INDEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT .................................................................</td>
</tr>
<tr>
<td>RESUMEN .................................................................</td>
</tr>
<tr>
<td>OBJECTIVES ...............................................................</td>
</tr>
<tr>
<td>INTRODUCTION ............................................................</td>
</tr>
<tr>
<td>Copy number variation ..................................................</td>
</tr>
<tr>
<td>Protein domain expansions .............................................</td>
</tr>
<tr>
<td>The use of CORTEX for heterozygous variation detection ..........</td>
</tr>
<tr>
<td>De Bruijn graphs ........................................................</td>
</tr>
<tr>
<td>MATERIALS AND METHODS ................................................</td>
</tr>
<tr>
<td>NGS DATA COLLECTION ..................................................</td>
</tr>
<tr>
<td>REFERENCE GENOME COLLECTION ........................................</td>
</tr>
<tr>
<td>QUALITY CONTROL ..........................................................</td>
</tr>
<tr>
<td>CORTEX ANALYSES .........................................................</td>
</tr>
<tr>
<td>• Binary dumping ........................................................</td>
</tr>
<tr>
<td>• Bubble calling ..........................................................</td>
</tr>
<tr>
<td>• Obtaining vcf files ....................................................</td>
</tr>
<tr>
<td>• Selecting and filtering vcf files ...................................</td>
</tr>
<tr>
<td>THE NUMBER OF VARIANTS ................................................</td>
</tr>
<tr>
<td>HETEROZYGOUS VARIATION IN RELATION TO POPULATION SIZE ......</td>
</tr>
<tr>
<td>ANALYSIS OF THE PROTEIN DOMAINS ....................................</td>
</tr>
<tr>
<td>• Protein domain detection and analysis ..............................</td>
</tr>
<tr>
<td>• Analysing the number of copies of the protein domains ..........</td>
</tr>
<tr>
<td>• Finding variants for the protein domains in all species .......</td>
</tr>
<tr>
<td>RESULTS ........................................................................</td>
</tr>
<tr>
<td>VARIANT CALLING .........................................................</td>
</tr>
<tr>
<td>NUMBER OF VARIANTS AND POPULATION SIZE ..........................</td>
</tr>
<tr>
<td>PROTEIN DOMAIN ANALYSIS .............................................</td>
</tr>
<tr>
<td>SEARCHING FOT SNVs IN PROTEIN DOMAINS ............................</td>
</tr>
<tr>
<td>DISCUSSION ....................................................................</td>
</tr>
<tr>
<td>TOTAL SNVs FOUND WITH CORTEX ......................................</td>
</tr>
<tr>
<td>PROTEIN DOMAIN ANALYSES ............................................</td>
</tr>
</tbody>
</table>
ABSTRACT

CORTEX software (an assembly-based variation discovery algorithm) was used to find heterozygous variation in 15 species of birds that were sequenced from a single outbreed individual. No variation was available in Ensembl for any of these species. The variation information found with CORTEX was then located in protein domain families for all the birds, and measured for each family. For this, first the protein domain families were characterized using Hmmscan software. The distribution for the variation across different protein domains and across the different species varied significantly, the Cadherin domain being the one with more heterozygous variation and also the most variable among species. We also analyzed the differences in copy number regarding the protein domain families, and found interesting patterns in protein domain families related to the process of retrovirus replication and infection that could suggest a domain expansion in the passerine birds.

RESUMEN

CORTEX, un software basado en un algoritmo de ensamblaje de novo, fue utilizado para la detección de variación heterocigota en un grupo de 15 especies de aves que fueron secuenciadas a partir de un solo individuo. Cuando se realizó el estudio no había variación disponible para ninguna de estas especies de aves. La información de variantes obtenida con CORTEX fue localizada en diversas familias de dominios proteicos en cada una de las especies. Para ello, los dominios proteicos fueron caracterizados con el programa Hmmscan. La distribución de la variación heterocigota a lo largo de las diferentes familias de proteínas fue bastante variable entre especies. La familia del dominio proteico que más variación heterocigota, y variación entre especies presentó, fue la de la Cadherina. También se analizaron las diferencias en número de copia de las diferentes familias de dominios, y se encontraron patrones interesantes en un grupo de familias de dominios relacionados con retrovirus que podrían sugerir una expansión de estas en el grupo de los paseriformes.
OBJECTIVES

The main objective of this project is the understanding of the implications heterozygous variation can have in the evolutionary patterns of a group of 15 species of birds.

The original idea for this project was to use CORTEX software (Iqbal et al. 2012) to obtain the heterozygous variation and then locate this variation in each base of a selected number of protein domains with a high number of copies. This would allow us to have a robust estimate of heterozygosity for each residue of the domain. Finally, we would compare the heterozygosity obtained for each residue with the evolutionary conservation (obtained from HMM information from Pfam). The goal was to give some insight regarding the conservation in protein domains and investigate the relationship between these findings and the function of the protein domain.

Nonetheless, we could not follow this initial idea, as we found that there were not protein domains with the sufficient amount of copies in order to have a robust estimate of the heterozygosity. Also, the variation found for each domain was lower than we had expected in the beginning. For this reason, we had to reconsider the original objectives and follow in a different direction.

Given we already had the heterozygous variation for all species, we decided to follow a slightly different approach:

We would analyse the differences in copy number of all the detected protein domains families for all the species, as variability in copy number such as protein domain expansions have been proven to affect diverse biological processes. Afterwards, we would locate the heterozygous variation found with CORTEX in each protein domain family and observe the different patterns regarding heterozygous variation in these protein domains families.
INTRODUCTION

The process of evolution is driven by genetic change in a population over many generations. For this reason, this genetic variation is an essential phenomenon for evolution to take place, as it allows the possibility of natural selection. Random mutations in the DNA sequence are key contributors to genetic variation. Because the number of mutations can be affected by the environment, they constitute a very important instrument to explain the course of evolution. Gene flow (movement of genes between populations) and sex also play an important role in providing for genetic variation.

In the human genome, the most common source of genetic variation corresponds to single nucleotide polymorphisms (SNPs), whose detection has given valuable insights in the development of common diseases, and have given rise to the development of genome-wide association studies (GWAS). Although SNPs are not always disease-causing mutations, these studies have allowed researchers to observe the relationship between the distribution of SNPs and the development of disease (Suh & Vijg, 2005). SNPs fall in different categories depending on where they are located in the genome sequence. They may be located in coding or non-coding regions of genes. SNPs located in a gene region can alter the production or function of a protein.

If it is located in a coding region, it can alter the amino acid sequence (non-synonymous SNP) or not (Synonymous SNP). On the other hand, if its location is in the non-coding region only the amount of protein produced may or may not be altered, as it can affect regulation of gene expression, transcription factor binding and degradation of mRNA.

Figure 1: Type of SNPs regarding location relative to a gene. Image borrowed from http://learn.genetics.utah.edu/content/pharma/snips/
**Copy number variation**

Another way we can measure the variation in the genome is by studying the changes in gene or protein domains copy number. Copy number variation (CNV) is a phenomenon that has become apparent after the completion of the Human Genome Project. It comprises structural variations (deletions and duplications) of the genomes that lead to an abnormal number of copies in different regions of DNA. It is been observed that copy number variation contribute to a variety of human diseases (Ionita-laza, Rogers, Lange, & Raby, 2010). This variation can be due to chromosome deletions or insertions, but also can happen on a lesser level, and affect the number of genes or protein domains.

**Protein domain expansions**

Protein domains are highly conserved protein regions with independent structural and functional properties that have evolved distinctly. The combination of different protein domains gives rise to varied protein architectures, each one with different functions determined by the domain composition of the protein. In this way, the gaining (domain insertion), losing (domain deletion) and rearrangement of domains in a given protein act as a driving force in protein evolution.

Domain repeats (the same family domain is repeated in tandem within a single protein) are big contributors to the increase of copies in many domain families (e.g LRR and C2H2 zinc fingers), but the other case is also a possibility: domains with a high number of copies found as a single unit in different proteins. Commonly, domain repeats intervene in interactions with proteins or ligands like DNA or RNA. Although the structure of these domains tends to be conserved, the sequence conservation is usually very low, as only a few residues of the sequence are needed for correct folding. This variation in sequence allows for the domain repeats to bind to diverse proteins and ligands. Thus, the proteins with a high number of repeats tend to have very varied functions, such as cell-cycle regulation, transcriptional regulation, protein transport and protein folding, among others. Because of this, it would be reasonable to think that protein domain repeats could have more variation within the domain.

It has also been observed that domain repeats expansion commonly occur by duplications of many domains at a time, while the single duplication of one domain is less likely (Björklund, Ekman, & Elofsson, 2006).
The mechanisms that explain domain expansion are diverse. In Figure 2 we can see multiple mechanisms that may explain this phenomenon: gene fusion (a), exon extension (b), exon recombination (c), intron recombination (d) and retroposition (e) are some known mechanisms (Marsh & Teichmann, 2010).

In this project the aim is the study of copy number regarding protein domains and its comparison between species. For example, it is been observed that the expansion of DUF1220 protein domain show a direct correlation with brain size in human populations and different forms of brain pathologies (Dumas et al., 2012; Popesco et al., 2006), as well as cognitive abilities (Davis et al., 2014). This domain would have undergone copy-number expansion during recent primate evolution and especially in the human lineage.

Figure 2. Possible mechanisms for the gain of protein domains. Colored blocks represent exons, with red, blue and green indicating exons coding for different domains. Solid black lines represent introns and red lines indicate intergenic regions.
(a) Gene fusion. The noncoding region between two genes is modified so that the exons of the first gene become spliced with the second. (b) Exon extension. The noncoding region following an exon becomes part of the exon and codes for a new domain. (c) Exon recombination. The exons of two genes become directly joined. (d) Intron recombination. An exon from one gene is inserted into the intron of another. (e) Retroposition. A retrotransposon sequence (RT, purple) mediates the copying of itself and a neighboring gene region via an mRNA intermediate, followed by insertion into another gene.

Figure retrieved from Marsh & Teichmann, 2010
The use of CORTEX software for heterozygous variation detection

Cortex software introduces the use of coloured *de Bruijn* graphs for the finding and genotyping of different kinds of genetic variants. The common approach of variant calling consists of the alignment of reads to a reference genome, and has proven to be powerful for the detection of SNPs and short insertion-deletion polymorphisms. Nevertheless it posed some problems when the genetic variants are extremely different or absent from the reference genome, as the alignment is not as good (Iqbal, Caccamo, Turner, Flicek, & McVean, 2012). Furthermore, the reference genome may not exist, or may be incomplete, in which case the absent reads may map to paralogous regions, constituting false variants. Cortex software solves many of these shortcomings, as it does not need the use of a reference genome. As a “novo assembler”, it uses classical de bruijn graphs, and colours the nodes and edges of each graph depending on the sample in which they are being observed.

*De bruijn graphs*

The strategy in modern genome assemblies and in Cortex software follows the use of *de Bruijn* graphs. A *de Bruijn* graph is a directed graph that connects a set of nodes (strings of a particular length *k* or *k*-mers) that overlap *n*-1 nucleotides. Directed edges connect *k*-mers (nodes) that are consecutively found in the genome (Figure 2).

Their importance in genome assembly of short sequence reads have been widely acknowledged, with many common algorithms based on them such as AllPaths-LG (Lindblad-Toh et al., 2011), SOAPdenovo (R. Li et al., 2010)Abyss (Simpson, Wong, Jackman, Schein, & Jones, 2009) and Velvet (Daniel R Zerbino & Birney, 2008; Daniel Robert Zerbino, 2009).

In this manner, a graph is created for the reference genome (or the novo assembly, as explained in Figure 3), and another one

---

**Figure 3:** strategy for genome assembly. Each read is the nodes (*k*-mers), the red edges join each *k*-mer, reconstructing the genome by forming an alignment. Image retrieved and modified from (Compeau, Pevzner, & Tesler, 2011).
will be constructed with the sample reads; when variation is found with Cortex, the variants appear as new nodes and edges that diverge from the other graph, constituting a “bubble” (Figures 3 and 4).

Figure 4. Discovery of variants in a single outbred diploid individual (blue), with a reference sequence (red). True polymorphisms generate bubbles that diverge from the reference, whereas repeat structures lead to bubbles that are also observed in the reference. Homozygous variants occur when the reference and the sample genome are completely separated in the bubble.

Figure 5: Bubble that shows a SNV. (Retrieved from Iqbal et al., 2012).
MATERIALS AND METHODS

NGS DATA COLLECTION

- **Birds**

All the data for the birds used in this project was brought together by the Avian Phylogenomics Consortium. They sequenced 44 birds, all of them sequenced by the Beijing Genome Institute (BGI, 2014). The files could be downloaded from the European Nucleotide Archive (ENA, http://www.ebi.ac.uk/ena) in fastq format via ftp. The NCBI accession numbers for finding the sequences were available on one of the papers of this project (Zhang et al., 2014).

For this project, I selected those genomes sequenced with the highest coverage. The species chosen were: Downy Woodpecker (*Picoides pubescens*), White-throated Tinamou (*Tinamus guttatus*), African Ostrich (*Struthio camelus*), Adelie Penguin (*Pygoscelis adeliae*), American Crow (*Corvus brachyrhynchos*), Golden-collared Manakin (*Manacus vitellinus*), Chimney Swift (*Chaetura pelagica*), Anna’s hummingbird (*Calypte anna*), Bald Eagle (*Haliaeetus leucocephalus*), Common Cuckoo (*Cuculus canorus*), Hoatzin (*Opisthocomus hoazin*), Killdeer (*Charadrius vociferus*), Little Egret (*Egretta garzetta*), Medium-ground finch (*Geospiza fortis*) and the Peregrine Falcon (*Falco peregrinus*).

All the avian genomes used, except the Bald Eagle were sequenced at BGI using the Illumina HiSeq 2000 platforms. The sequencing of the bald eagle was done at Washington University in St. Louis (WUSTL) using Illumina HiSeq technology.

REFERENCE GENOMES COLLECTION

- **Birds**

The references for each species were downloaded in FASTA format from the website Avianbase, a mirror based on code from Ensembl, a joint project between the Wellcome Trust Sanger Institute and the European Bioinformatic Institute (http://avianbase.narf.ac.uk).

To download the raw data and the reference, I wrote a script in bash called `get_"species".sub.`
QUALITY CONTROL

CORTEX already has a way to filter by the quality of the raw data, so we decided not to do any further analysis regarding the quality of the reads. Also, it is been discovered that the quality control of Illumina reads does not necessarily pose any advantage to the process of variant calling when using other known variant callers (Liu et al., 2012).

CORTEX ANALYSIS

The CORTEX_release_v1.0.5.17 was used for variant calling. The software can be downloaded from http://cortexassembler.sourceforge.net/index_cortex_var.html. After installing the software, it is essential to compile the Cortex binaries and calculate the memory and hash table requirements, the steps could are described in the CORTEX manual (Iqbal, 2012).

To obtain the variants we followed the same pipeline for all species:
1) Binary dumping.
2) Bubble calling.
3) Obtaining vcf files.
4) Selecting and filtering variants

1. Binary dumping

In this first step a binary for each species and its reference genome was created, meaning a de Bruijn graph was constructed in each case.

Nevertheless, the memory requirements were too high to dump a binary per genome, so the solution was to create one coloured binary for each fastq file and merge them later. Binaries were dumped using the script binaries_"species".sub (see Figure 6). The options chosen were as follows:

-pe_list LIST1,LIST2: it refers to the paired end fastq files to use. Reads 1 and reads 2 are both stored in lists with the files listed at the same positions in the two files.

-quality_score_threshold: to filter by quality scores of the input file. In this case we decided to filter the reads by 5, as this is the value recommended in the manual for Illumina reads. Any base with a phred-scale base quality less than or equal to 5 was removed.

-remove duplicates: removes PCR duplicate reads by ignoring read pairs if both reads start at the same k-mer as a previous read.
Error cleaning and binary merge

Binaries were merged and cleaned in one step, using the script "merge_binaries _"species".sub" (see Figure 6).
Is it advisable to clean the binaries to remove sequencing errors, especially when the sequences are Illumina sequences, as this method tends to produce more errors than Sanger sequencing.

The binaries were merged into one colour using the option -colour list FILE1. FILE 1 is a filelist that had to be generated (that was called list_merged_all) and contains FILE2 (a list of all the uncleaned binaries for each species). We created the file like this:

"Readlink -f .ctx > list_merged_all".

Then, the sequencing errors were removed using the option remove_low_coverage_supernodes, assigning here the value of 2 (the recommended value).

2. Bubble calling

In this step two coloured graphs were created, one representing the reference genome, and the other the individual’s genome.

The script used for bubble calling was bubbles_{species}.sub (see Figure 6). To introduce the input in the script we had to generate a list of filelists, that is a list (e.g colour_cuckoo_list) that contained two files: one list holding the binary for the reference genome and the other list holding the cleaned binary for each species.

For the reference genome (file1):

"Readlink -f Cuculus_canorus.ctx > list_acuckoo_ref_ctx"

For the genome (file2):

"Readlink -f binary_cuckoo_clean.ctx > list_bcuckoo_ind_ctx"

To generate colour_cuckoo_list:

"For x in `ls *_ctx`; do readlink -f $x ; done > colour_cuckoo_list"

For CORTEX to interpret the input correctly, it is necessary that the list containing the reference binary is in the first item in the list.

The options used in this script were the following:

--colour_list this is a list of filelists, one per colour, each containing a list of binaries to go into that colour.

--detect_bubbles1 0,1/0,1 to find heterozygous variants in colour 1, our species of interest.
- **--exclude_ref_bubbles** indicates that cortex to discard bubbles present in reference (i.e. branches that correspond to colour 0) which are surely due to repetitions or paralogs regions (see introduction).

- **--print_colour_coverages** prints coverages in all colours for supernodes and variants. This option is mandatory to dump VCF files.

- **--experiment_type EachColourADiploidSampleExceptTheRefColour** indicates the statistical models for determining likelihoods of genotypes.

The output files that contained the bubbles were called `{species}_bubbles`.

**Figure 6**: steps followed for variant discovery: binary dumping and bubble calling.

### 3) Obtaining vcf files.

- **Bubbles conversion to vcf format**

The conversion to vcf format is done with the cortex script `process_calls.pl`. Nevertheless, the conversion could not be done with the bubble files straightaway. Each bubble file was too big for the program to give a result in a reasonable time. The solution to this was to split the bubble file into many, smaller subfiles. This problem had already been solved by David Martin-Galvez during his internship. I used the script in
perl he wrote to split the bubbles (split_bubbles.pl).
This program splits the original bubble file into many bubble subfiles (each file containing 5000 bubbles), and creates as directory (“species”_bubbles_d) with as many subdirectories (“specie”_bubbles_subd_”number of file”) as bubbles files. Once bubbles have been splitted in many subfiles, we performed the conversion to VCF format by using the cortex script process_calls.pl.

- **Process calls functioning**

To generate the vcf file, the program needs to execute vcftools (Danecek et al., 2011). In order for process_calls.pl to map the variants into a reference we needed to use Stampy software (Lunter & Goodson, 2011), and create a stampy hash and index before running the program. This was done with the script I wrote, called stampy_”species”_sub.

I also had to generate some files for some of the options of the program, such as:

-- callfile_log: file containing the text Cortex printed to screen (“stdout” output). For this I had to modify the stdout, as I ran the bubble caller in the cluster. I removed the information concerning the cluster job.

--sample list: file containing one line per colour, and on each line, a sample identifier - these end up in the header line of the VCF.

--stampy hash: path to where is the stampy hash created earlier.

As we had many bubble subfiles in different subdirectories, the program was run as a job array in the cluster, to parallelize the analyses (see Annex 1). Thus, the number of jobs were the same as the number of bubble subfiles.

The resulting files are different types of vcf files: the raw vcf, that contains the actual calls, and the decomposed file (.decomp.vcf) file, where composite variants have been broken down into sub-SNP's and indels where possible. Here we only use the decomp.vcf files.

- **Merging of vcf subfiles**

To achieve this, I created a new directory in which I kept all the decomp.vcf files and proceeded to sort, compress and index the vcf files, with the following commands as
shown in this example code:

To sort the file: `for i in `ls *.vcf`;do
vcftools_0.1.6/perl/vcf-sort $i > sorted_$i; done

To compress (has to be bgzip): `for i in `ls sorted_*`;do
  tabix-0.2.6/bgzip -c $i > $i.gz; done

To index: `for i in `ls *.gz`;do
  tabix-0.2.6/tabix -p vcf $i; done

Finally, we concatenate all the vcf files for each species:

vcftools_0.1.6/perl/vcf-concat -f list_vcf > “species”_decomp_all.vcf

The final vcf is a file like this:

<table>
<thead>
<tr>
<th>CHROM</th>
<th>POS</th>
<th>ID</th>
<th>REF</th>
<th>ALT</th>
<th>QUAL</th>
<th>FILTER</th>
<th>INFO</th>
<th>FORMAT</th>
<th>CSQ</th>
<th>QC</th>
<th>PL</th>
<th>Strange</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C21929895</td>
<td>17</td>
<td>cortex_cuckoo_var_4388127</td>
<td>T</td>
<td>A</td>
<td>.</td>
<td>MISMAPPED_UNPLACEABLE</td>
<td>SLEN=8;SVTYPE=SNP;GT:COV:GT:CONF</td>
<td>1/0:11:35:31:38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C21929898</td>
<td>41</td>
<td>cortex_cuckoo_var_129097</td>
<td>A</td>
<td>G</td>
<td>.</td>
<td>MISMAPPED_UNPLACEABLE</td>
<td>SLEN=8;SVTYPE=SNP;GT:COV:GT:CONF</td>
<td>1/0:11:35:31:38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C21929899</td>
<td>41</td>
<td>cortex_cuckoo_var_3486419</td>
<td>T</td>
<td>G</td>
<td>.</td>
<td>MISMAPPED_UNPLACEABLE</td>
<td>SLEN=8;SVTYPE=SNP;GT:COV:GT:CONF</td>
<td>1/0:11:35:31:38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C21929900</td>
<td>51</td>
<td>cortex_cuckoo_var_2908029</td>
<td>T</td>
<td>A</td>
<td>.</td>
<td>MISMAPPED_UNPLACEABLE</td>
<td>SLEN=8;SVTYPE=SNP;GT:COV:GT:CONF</td>
<td>1/0:11:35:31:38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C21929901</td>
<td>64</td>
<td>cortex_cuckoo_var_3633408</td>
<td>A</td>
<td>C</td>
<td>.</td>
<td>MISMAPPED_UNPLACEABLE</td>
<td>SLEN=8;SVTYPE=SNP;GT:COV:GT:CONF</td>
<td>1/0:11:35:31:38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 7: example of a vcf file produced by CORTEX.

- Cleaning errors from vcf files

The variants that interest us are the ones that passed all the Cortex filters, those showing PASS under the filter column. Nevertheless, there was a bug in one of the Cortex filters. Some of the variants had reference alleles that did not match the allele in the reference genome. Thus, the allele that appears in the vcf file is not the same than the allele that is on the reference genome FASTA file. The warning MISMAPPED_UNPLACEABLE appears in the filter column in this cases.

This error was already reported when David Martin-Gálvez used Cortex in 2013, but it had not been fixed since he reported the mistake.

To solve the problem, I used a script that David wrote in perl (`filter_vcf_bad_ref.pl`). The script reads the vcf file and compares those places where there is a MISMAPPED_UNPLACEABLE tag with the reference genome.

This script parses the VCF file, compares reference alleles with information in the reference genome extracted with the command samtools faidx of samtools. It makes use of the module IPC::System::Simple.
function qw (capture) to retrieve alleles from the reference genome. Resulting vcf files were compressed and indexed again.

**4. Selecting and filtering variants**

We are only interested in those variants that show a PASS mark in the filter column of the vcf file (Figure 7), which indicates that they have passed all the CORTEX filters. Moreover, I also removed those variants that were homozygous (0,0) and those that had a coverage with a value of 1 (0,1 or 1,0). The variants that instead of PASS had MAPQ were also removed. I did this directly from the terminal:

```
egrep -v ":1,0::0,1::0,0::0/0:"
grep -v "MAPQ"
```

Second, I proceeded to remove the indels, as we are only going to use single nucleotide variants:

```
grep -v ";SVTYPE=INS"
grep -v ";SVTYPE=DEL"
grep -v ";SVTYPE=INDEL_FROM_COMPLEX"
```

After this, we had the file we called “species” _all_filtered _we were going to use for the rest of the project.

**THE NUMBER OF VARIANTS OBTAINED FOR ALL THE SPECIES**

To test if the differences in number of variants between species were significant, we did a chi-square test (see the script `final_analysis.R`)

**HETEROZYGOUS VARIATION IN RELATION TO POPULATION SIZE**

Estimations for the population size of some species were obtained from different sources. We could only find estimations for 10 species. Of those, 6 (American Crow, Downy Woodpecker, Peregrine Falcon, Anna’s Hummingbird, Bald Eagle and Common Swift) were obtained from the webpage Partners in Flight Science Committee 2013. Population Estimates Database, version 2013. Available at http://rmbo.org/pifpopestimates. The process that was followed to do the estimations is explained by Panjabi et al., 2013.
The rest (Common Cuckoo, Adelie Penguin, Killdeer and Little Egret) were collected from the webpage BirdLife International (http://www.birdlife.org/datazone/species/). With this information a plot was made using an R script I wrote (Final_analysis.R). All the code for subsequent analysis is gathered at that script.

**ANALYSIS OF THE PROTEIN DOMAINS**

The analysis of the protein domains consists in three parts:

1) Protein domain detection.
2) Analysing the copy number for each protein domain family in all species.
3) Finding the variants obtained with CORTEX in the protein domain family in all species.

**1) Protein domain detection and analysis**

The first step was to identify the domains of each protein from the fasta protein file (named “scientific_name.pep”). These files are available in the web page avianbase (http://avianbase.narf.ac.uk/index.html). Then, hmmscan software (http://hmmer.org/) was used to identify the protein domains. This software searches a sequence against a profile HMM database. It will annotate all the different known/detectable domains in a given sequence. It takes a single query sequence and an HMM database as input. The HMM database chosen in this case was Pfam, the profile HMM flatfile used was Pfam-A.hmm. Before running the software the database has to be prepared for the hmmscan to work. With the option –hmmpress, the flatfile is binary compressed and indexed. The script hmmscan.sub does this step, and the runs the hmmscan program, doing a loop to run the command in all species at once (all the .pep files).

After running the program, a tabular format file was obtained, that held the protein domains identified. All the fields of this file are described in the HMMER Users’s Guide (Eddy, 2010). The output file was called hmmscan_results.

For each protein_id (that represented each protein sequence in the .pep file) there could be one or more domains. The problem was that in many cases there was an overlapping of domains for the same protein. This is due to false identifications from Hmmscan, as the program provides with a list of those most probable domains for the sequence. For this cases in which domains overlapped, we decided to choose the non-overlapping domains with the lowest E-value. This task was accomplished with a script I wrote in Pyhton called “filtering_hmmscan.py”. First, it parses the Hmmscan output file and
discards those domains with an indE-value higher than 1e-05; second it selects the non-overlapping domains for each protein (that is, domains with the same ID). The program has to be run multiple times, until all the overlapping domains are removed.

The end result file was called “final hmmscan_0”, were we have the following fields:

<table>
<thead>
<tr>
<th>Name</th>
<th>Accession number</th>
<th>protein_id</th>
<th>Coordinates</th>
<th>ind.E-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>zf-H2C2_2</td>
<td>PF13465.1</td>
<td>Ppu_R006940</td>
<td>116 139</td>
<td>2 1e-07</td>
</tr>
<tr>
<td>zf-H2C2_2</td>
<td>PF13465.1</td>
<td>Ppu_R006940</td>
<td>143 165</td>
<td>3 2.7e-07</td>
</tr>
<tr>
<td>zf-H2C2_2</td>
<td>PF13465.1</td>
<td>Ppu_R006940</td>
<td>172 195</td>
<td>4 1.1e-07</td>
</tr>
<tr>
<td>Rdx</td>
<td>PF10262.4</td>
<td>Ppu_R006942</td>
<td>2 53 1</td>
<td>8.3e-13</td>
</tr>
<tr>
<td>RVT_1</td>
<td>PF00078.22</td>
<td>Ppu_R011055</td>
<td>15 115 1</td>
<td>3.1e-07</td>
</tr>
<tr>
<td>7tm_4</td>
<td>PF13853.1</td>
<td>Ppu_R011057</td>
<td>6 149 1</td>
<td>1.5e-53</td>
</tr>
</tbody>
</table>

**Figure 8:** the filtered hmmscan file that holds non-overlapping protein domains with indE-value smaller than 1e-05.

To test if the differences in number of protein domain families between species were significant, we did a chi-square test (see the script final_analysis.R)

### 2. Calculating the number of copies for each domain

For this step I wrote a script in Python called “copy_number.py”. It parses the filtered Hmmscan file (final_hmmscan_0) and counts the number of domains for each protein in each of the species. It has to be done with each one at a time. It creates a file that I called copy_number_“species” with two fields: the protein name and the number of domains for that protein.

- **Plotting the results**

The information holding the number of copies was transferred to LibreOffice Calc. I constructed a pivotal table that contained all the information regarding the protein domain copy number, as well as the average and standard deviation of the number of domains for each protein (table_copies.csv).

Afterwards, we proceeded to do a linear regression of the average of copies against the standard deviation, and plotted the results. We are interested in those domains that have the greatest change among species, as well as those that present the less change. The criteria followed to select these domains was to plot the residues of the linear regression model and choose the above a certain threshold (i.e, those that started to diverge from
the majority). Nevertheless, the protein domains selected were too many, and as preliminary analysis, we only selected between 10 to 15 domains families of those that were more divergent from the expected values. All these analyses were done with the script I wrote in R “final_analysis.R”.

• **Integrating this information with the phylogenetic history**

We created a tree and made a graph using a web page called ITOL (http://itol.embl.de/). The idea is to show the number of copies per protein domain for each species next to the phylogenetic tree. To construct the tree, I used as a model the phylogenetic tree from (Zhang G, 2014), and kept only the species I was selected for the project. We only used the information for those domains that presented the most variability in copy number among species. The tree had to be introduced in the following format, the parenthesis indicating the phylogenetic relationships:

```
((tinamou,ostrich),((cuckoo,(anna,swift)),((killdeer,hoatzin),((egret,penguin),((((crow,finch),manakin),falcon),(eagle,woodpecker)))))))
```

The information regarding the copy number of domains per species had to be uploaded in a file (“copies_set.txt”) with a specific format. To test whether there was a phylogenetic signal (i.e. if the phylogeny is affecting the protein domain copy number) I used the package phytools in R (Annex 2). The script is called `phylogenetic_signal.R`. To do this analysis I needed to generate the tree again in newick format (tree.phy), and introduce the information of the copy number in a specific format (see supplementary material, copies.txt)

3) **Finding variants in the protein domains for all species.**
We already have the variants obtained with CORTEX, and for each, the coordinate that indicate where they are in the reference genome. Now we need to see which of this variation is found in the protein domains. For that, the exact coordinates for each domain in the reference genome are needed. That can be achieved by doing alignment of each domain against the reference genome using BLAST. This way we will have all the coordinates for each protein domain, taking into account the existing introns. It is important to mention that the protein domains used in this step were all present in
the annotation file of each of the species (a file with the annotation for the all the characterized proteins in that species). This file is a .gff format, with one line per feature, each containing 9 columns of data. Among other things, it gives information regarding the location of the protein (scaffold and its coordinates) and the name of the protein (protein id). We are only interested in those lines with the mRNA feature, so with a shell command I erased the other lines:

\texttt{grep “mRNA” Picoides_publescens.gff > woodpecker_annotation.}

The first column corresponds to the scaffold in the genome, the 5 and 6 columns are the coordinates for each scaffold.

In this way will not use any protein domain of any protein that was not characterized in this file.

To explain the steps followed:

2.1) Use Hmmscan software to identify the protein domains. The input file is a fasta file (.pep) that holds the proteins identified for each species (This step is already done for the copy number analysis of the protein domains).

2.2) Using the coordinates of the protein domains, obtain the aminoacidic sequence for those protein domain that were characterized in the annotation file (.gff).

2.3) Running the Blast (protein sequence against the reference DNA sequence).

2.4) Finding variation in the protein domains.

\textbf{2.2) Using the coordinates of the protein domains, obtain the aminoacidic sequence for those protein domain that were characterized in the annotation file (.gff).}

To achieve this, I had to combine the information of the fasta file, the annotation file and the hmmscan_0 file. This process in explained in Figure 9. Using the protein domain coordinates of the hmmscan_0 file, we extracted only those protein domains that were present in the annotation file. To combine all this information and accomplish the task, I decided to create a database in POSTGRES and make queries to extract the information. I wrote the script “species”.sql, to create the tables for the database the tables are the following:

\textbf{Table 1} {Species}: the output of the hmmscan

\textbf{Table 2} {FASTA_species}: fasta file.

\textbf{Table 3} {REFERENCE_species}: annotation file.

\textbf{Table 4} {DOMAINS_species}: here the sequence for each domain will be introduced.
To introduce and relate all this information I wrote a script in Python, called `{specie}_into_database.py`. The script connects to the database and does the following:

1. Parses the hmmscan file (`final_hmmscan_0`), and introduces the corresponding fields in the first table.
2. Parses the fasta file ("{species}.pep"), retrieve the protein id and sequence to introduce it in the second table.
3. Parses the annotation file ("{species}.gff"), and introduces the corresponding fields (scaffold, protein id, coordinates of the scaffolds) and introduces it in the third table.
4. Selects the name, the id, sequence and coordinates (from the different tables) from the different tables for the same protein (indicated by the protein_id), and using the coordinates extracts the sequence domain from the sequence of the fasta file.

Afterwards, the domains are introduced to the fourth table.

```python
cur.execute("select target_name, id_proteina, secuencia, coord_inicio, coord_final from fasta_woodpecker f, woodpecker a where f.id_fasta = a.id_proteina;")
```

5. With another selection, it retrieves all the domains from the fourth table, and the coordinates of the scaffolds from the second table.

```python
cur.execute("select target_name, id_proteina, seq_dominio, scaffold, coord_ref_ini, coord_ref_end from dominios_woodpecker d, reference_woodpecker r where d.id_proteina = r.id_protein;")
```

The program outputs two files:

- **Summary_{specie}_all.txt**: this file has all the information, the name, id, scaffold and coordinates in the reference genome and the sequence for each domain.
- **domains_{specie}_all.fasta**: a fasta file with the name, id and sequence.

The summary file was created to serve as a control to check the future steps.
Figure 9: steps followed to characterize the domains and find the heterozygous variation. Hmmscan_filtering.py, “species”_into_database.py and blast_parser.py are the programs in Python used to fulfill each step, the arrows in different colours indicate the step this program intervenes.

2.3) Running the Blast

Once we have the fasta files (domains"species"_all.fasta) containing the domains sequences, we can run blast against a local database we create with the reference genome. I wrote the script database.sub in bash that created the database with the reference genome for all species. The option makedb was used.

Then, with another script I created (run_blast.sub) I run Blast (blast version) for all species. The option -outfmt ’6 std sseq means’ was chosen to have the output in a tabular format. Once I got the blast result, I had to filter the file to keep only the results with a 100% of identity.
There were also some hits that did not match the scaffold that the domain is supposed to be in, as indicated by the annotation file (in birds). That means that some domains have aligned in different parts of the genome. We are only going to consider those that are determined in the annotation file. To do this filtering I wrote a script in python (blast_parser.py).

This script searches for those sequences that aligned with a 100% of identity and removes the alignments that occurred in another scaffold. For this, it compares the scaffold in the blast hit with the summary file we created before (described in a previous step). The resulting file has the name final_blast_{specie}_all.

2.4) Finding variants in our protein domains

With the variation file obtained with CORTEX we can now look for variants that were found in our domains. To do this, I wrote a script in python, called get_ht.py. This script searches for variants that fall inside a protein domain. It checks if the position of the variation (in the vcf file) is between the coordinates of the protein domain. In the end, it outputs two files:

- **Final_summary_all**: here we have the following: target name of the protein domain, protein_id, the scaffold, the position it has in the genome, the reference allele, the alternate allele (the heterozygous variation) and the coordinates of the protein domain.
- **Variants_all**: here we have, in this order: the target name, the protein_id, the scaffold, the position the heterozygous variation has in the genome, the position the allele has in the protein domain, the reference allele, the coordinates of the protein domain and the sequence of the domain, in which the allele has been replaced with an N to indicate were the heterozygous variation is.

For what we are going to do, we do not need all this information. Knowing how many different variants are present in the protein domains is enough. I created these files with the idea of estimating the heterozygosity, for which I needed the exact position of the heterozygous variation inside the protein domain. In the end, that was not necessary, but the files are a good summary of the work done and also hold valuable information that could be checked for many different analyses.

Finally, we counted how many variants each protein domain had. It has to be noted that the counts correspond to the total heterozygous variation that is present in all the domains of the same family (and not the number of variants present in each sequence). Thus, families with more copies are expected to have more variants because of this fact.

---

21
For this purpose I created a script called variation_all.py. It parses the file variants_all and gives an output file with the name of the domain, the number of variants and the species. We decided to correct the variation values by the average length of the domain, as this feature can influence the number of variation a domain can have.

- **Heterozygous variation analyses and plotting the results**

As the number of copies varies for each protein domain family, we have to take this fact into account when analyzing the amount of heterozygous variation for each family. I created a pivotal table in LibreOffice Calc, and calculated the average and the standard deviation of copies per protein domain among species, and the average and standard deviation for the heterozygous variation found among species. It has to be noted that heterozygous variation was not found in all the protein domain families. To have only those domains where at least heterozygous variation was found to one of the species, I wrote a script in python called graph.py that selected only those protein domains that had heterozygous variation. It generates four files in order to create a table in LibreOffice Calc. The first has the names of the domains, the second the number of copies, the third the heterozygous variation and the fourth, the length of the protein domain sequence. All the data was in the same order in the file, so I could create the pivotal table (table_variation.csv).

Finally, I did two linear regressions in R (average number of copies~average number of variants and average number of copies~standard deviation of variants). Afterwards we proceeded to select those domains with more or less variation than expected (the same way we did when analysed the copy number variability among species), and plotted the results. All the analyses done with R are in the script final_analysis.R.

- **Integrating this information with the phylogenetic history**

This step is the same we followed in the case of the copy number analysis. We also tested for the phylogenetic signal in the same way we did previously.
RESULTS

VARIANT CALLING

In Table 1 are the total number of SNPs detected for all the bird species, a. There is not available variation in Ensembl (www.ensembl.org) to compare the variants obtained in this project. There seems to be differences in the total amount of variation between species. Species such as the Bald Eagle, the Peregrine Falcon and Anna’s Hummingbird have a very small number of variants. On the other side, the Common Cuckoo has much higher variation than the rest of species (Figure 10).

Table 1: Birds variation (SNVs)

<table>
<thead>
<tr>
<th>Specie</th>
<th>Variation</th>
<th>Specie</th>
<th>Variation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anna's hummingbird</td>
<td>1,218,411</td>
<td>Little egret</td>
<td>2,588,678</td>
</tr>
<tr>
<td>American crow</td>
<td>1,492,666</td>
<td>Golden-collared manakin</td>
<td>2,110,096</td>
</tr>
<tr>
<td>Medium-ground finch</td>
<td>3,493,100</td>
<td>Adelie penguin</td>
<td>3,014,441</td>
</tr>
<tr>
<td>Peregrine falcon</td>
<td>1,181,004</td>
<td>African ostrich</td>
<td>1,863,178</td>
</tr>
<tr>
<td>Common cuckoo</td>
<td>4,905,093</td>
<td>Common swift</td>
<td>1,633,711</td>
</tr>
<tr>
<td>Hoatzin</td>
<td>2,050,062</td>
<td>White-throated tinamou</td>
<td>2,538,247</td>
</tr>
<tr>
<td>Killdeer</td>
<td>2,767,103</td>
<td>Bald eagle</td>
<td>252,840</td>
</tr>
<tr>
<td>Downy woodpecker</td>
<td>2,584,160</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Chi-squared test for the variants obtained with CORTEX.

<table>
<thead>
<tr>
<th>Chi-squared test</th>
<th>Chi-squared</th>
<th>df</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7619000</td>
<td>14</td>
<td>2.2e-16</td>
</tr>
</tbody>
</table>

Figure 10: Total number of SNPs found with CORTEX.
NUMBER OF VARIANTS AND POPULATION SIZE

In **Figure 11** we can see there is a slight relationship between population size and the total number of SNPs found with CORTEX for each species. The species with the lowest variation, such as the Bald Eagle and the Peregrine Falcon are among those with the smaller population size, whereas the Common cuckoo is among those with the highest population size.

![Diagram](image)

**Figure 11:** Relationship between population size and total number of SNPs for each species.
PROTEIN DOMAIN ANALYSES

The number of domains detected with Hmmscan software (Figure 12) is very similar in all the species.

![Protein domain families](image)

**Figure 12:** total number of protein domains detected with Hmmscan.

<table>
<thead>
<tr>
<th>Chi-squared test</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-squared</td>
<td>df</td>
<td>p-value</td>
</tr>
<tr>
<td>22859</td>
<td>14</td>
<td>0.0626</td>
</tr>
</tbody>
</table>

*Table 3:* Chi-squared test for the total number of domains of all species.

Regarding the copy number of protein domains we can observe that there is variability among species, as well as between protein domains. As we can see in **Figure 13** we have two interesting scenarios. In the one hand, there is a group of domains that have a lower number of copies (ACFn3, RNase_H, RVT_thumb, Integrase_zn, rve, Gag_p30) but have a high variability among species. On the other hand, there is another group of domains whose number of copies is higher (Pkinase, WD40, I-set), but the variability among species is very low.
Per specie protein domain copy number variation (for selected domains) in a phylogenetic context. Comparative analyses

In Figure 14 we can observe there is a clear pattern for 4 protein domains, regarding the variability in protein domain copy number among species. All of them (RVT_thumb, rve, Integrase_zn and Rnase_H) have a very similar number of copies in all species.

No such effect is observed in the remaining protein domains. Moreover, all this protein domains intervene in the same biological process: the retrovirus infection cycle.

The analysis done with phytools suggest that there is phylogenetic signal for this 4 protein domains, as $\lambda$ is close to 1. Nevertheless, the P-value of the likelihood ratio test is greater than 0.05 in all cases (Table 4).
SEARCHING FOR SNVs IN PROTEIN DOMAINS

Heterozygous variation in protein domains families

The heterozygous variation found in each protein domain family is not equal. Figure 15 shows that those domains with a higher number of copies have more heterozygous variation, (a finding that was expected), whereas those protein domains with a lower number of copies present less amount of heterozygous variation. There are, however, a number of cases in which this does not occur (Pkinase, Collagen, RVT_1), as in this cases the number of copies is quite high, but the variation is very low.

![Figure 15: phylogenetic tree of the 15 bird species and per species protein domain copy number.](image)

**Table 4:** \( \lambda \) and \( P \)- values of the likelihood ratio test.

<table>
<thead>
<tr>
<th>Protein domains</th>
<th>( \lambda )</th>
<th>( P )</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVT_thumb</td>
<td>1</td>
<td>0.08</td>
</tr>
<tr>
<td>Rve</td>
<td>1</td>
<td>0.14</td>
</tr>
<tr>
<td>Rnase_H</td>
<td>0.98</td>
<td>0.19</td>
</tr>
<tr>
<td>Integrase_zn</td>
<td>1</td>
<td>0.13</td>
</tr>
<tr>
<td>RVT_1</td>
<td>0.38</td>
<td>0.40</td>
</tr>
<tr>
<td>Zf-C2H2_2</td>
<td>0.56</td>
<td>0.08</td>
</tr>
<tr>
<td>Gag_p30</td>
<td>0.23</td>
<td>0.48</td>
</tr>
<tr>
<td>Fn3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>ACfn3</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Variability of heterozygous variation of protein domains families across all species

The distribution of heterozygous variation among species varies depending on protein domains (Figure 16). Logically, those protein domain families with a higher number of repeats tend to have more variability on heterozygous variation (as we should expect by chance). However, there are interesting cases in which a protein domain family presents very different levels of heterozygous variation among species in spite of having a low number of copies (RNase_H, RVT_thumb). In other cases, the differences in heterozygous variation among species are higher than the expected values (RVT_1, 7tm_1, cadherin).

Finally, some domain families seem to have very little variability across species regarding heterozygous variation, independently of the number of copies (WD40, Collagen, zf-H2HC2_2).
Heterozygous variation (for selected domains) in a phylogenetic context.

Comparative analyses

There is no clear pattern when comparing the most variable protein domain families in relation with the distribution of heterozygous variation among species (Figure 17). In spite of having corrected by the total number of variants found in each of the species and the length of the protein domain sequence, we can observe that there are still some problems to interpret the data in the case of the eagle. For this bird, it seems that the variation found was too low in comparison with the rest. The analysis done with phytools suggest that there is phylogenetic signal for the protein domains RVT_thumb and VWA, as $\lambda$ is equal to 1 and the P-value of the likelihood ratio test is < 0.05 (Table 5).

Figure 16: linear regression analysis, depicting the variability of heterozygous variation of all the protein domains across species in relation to the protein domain copy number. In the Y axis it is represented the standard deviation of the heterozygous variation among species for each protein domain; in the X axis the average of copies of the protein domains among species. The coloured dots are those protein domains with values that differ more from the expected ones.
**Table 5**: $\lambda$ and P-values of the likelihood ratio test.

<table>
<thead>
<tr>
<th>Protein domains</th>
<th>$\lambda$</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNase_H</td>
<td>0.48</td>
<td>1</td>
</tr>
<tr>
<td>RVT_thumb</td>
<td>1</td>
<td>0.04</td>
</tr>
<tr>
<td>RVT_1</td>
<td>0.78</td>
<td>0.2</td>
</tr>
<tr>
<td>7TM_GPCR_Srsx</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>VWA</td>
<td>1</td>
<td>0.05</td>
</tr>
<tr>
<td>7tm_1</td>
<td>0.1</td>
<td>0.77</td>
</tr>
<tr>
<td>Cadherin</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
DISCUSSION

Total SNVs found with CORTEX

The results obtained with CORTEX show that the total number of SNVs varies widely between the fifteen species analyzed (Table 2, p-value < 0.05). As there is no variation data available in ensembl we cannot compare the current results with previous ones. Moreover, and given that all the data is been sequenced and processed for the same project and in the same conditions, it seems unlikely that our finding is due to differences between the data.

On the other hand, there are known factors that can affect genetic diversity (i.e. variation), such as population size. Many studies support the hypothesis that the size of the population, as well as conditions such as spatial isolation, influences negatively the genetic diversity of that population (Ortego, Aparicio, Cordero, & Calabuig, 2008).

Our results in Figure 11 suggest that although a direct correlation between population size and total variants does not exist in all species, population size could still be affecting the total amount of variation to some extent. It is notorious that the species with the lowest effective population size (specially the Bald eagle) are those with the lowest amount of SNVs, whereas the one with the most variation has the biggest effective population size. However, this results have to be taken with caution, has the effective population used for this analyses is an estimation. Moreover, in the case of the cuckoo it is very probable that the population is overestimated, considering the characteristic mating habits of this species (e.g see Gibbs et al., 2000).

It needs to be mentioned that these data are only from a single outbred individual, so logically we are missing data from the whole population and we could not make any assumptions with these analyses. However, the estimation of the effective population has already been done in many studies (H. Li & Durbin, 2012; Nadachowska-Brzyska, Li, Smeds, Zhang, & Ellegren, 2015). For this reason, these results indicate that population size could be an important factor affecting total amount of genetic variation found in a single outbred individual, among many others that need to be studied.
**Protein domain analyses**

- **Copy number (repeats), searching for domain expansions**

After estimating the total number of protein domains for all species, we can observe the number of total protein domains is very similar among all the birds. Concerning the copy number of protein domains (Figure 13), we can observe a wide difference among species and within domains. Protein domain families differ extensively, especially in vertebrates. Some protein domain families are known to have a high number of repeats, such as WD40, zinc finger and collagen (Björklund et al., 2006), a finding that also occurs in birds, as we corroborate with our results. It is been proposed that domain expansions could allow repetitive proteins to evolve more quickly than other proteins with less or without repeats. Interestingly, we can see that the domain repeats are not equally distributed in all species. This way, the number of repeats of WD40 and collagen domains would be more conserved than the rest. Looking at the phylogenetic tree (Figure 14) gives more insight into the distribution of domains among species.

Some protein domains families seem to follow the same pattern regarding number of domain repeats in all species (rve, Integrase_zn, RNase_H and RVT_thumb). Interestingly, all this protein domain families are related to the retroviral infection process. Rve and Integrase_zn protein domains are present in the enzyme retroviral integrase, produced by a retrovirus. The function of this enzyme is the integration of the viral DNA into the host chromosomal DNA. The other domains are included in the reverse transcriptase enzyme, also produced by retroviruses, and essential for the generation of a complementary cDNA from the RNA of the virus. This way, the produced cDNA could be integrated in the host genome.

The presence of endogenous viral elements is common in vertebrate genomes, especially in mammals, perhaps due to the lack of studies with avian genomes. Nonetheless, the increased presence of retroviral elements has been previously reported in passerine birds (Cui et al., 2014), suggesting an expansion in the oscine clade. Looking at our results, it is not clear, however, if there exist a strong phylogenetic signal. According to the analysis done with phytools there seems to be a signal ($\lambda$ is close to 1), but the P-values > 0.05 (Table 4). Nevertheless, it should be noted that the P-values are quite small, (some of them are near 0.05) and that there could be some factors affecting this results, such as the size of the tree (Blomberg, Garland, & Ives,
In this way, trees with less than 20 branches would not be very reliable. Thus, repeating the analysis with more species could be a good approach to see if the P-values show any change.

- **Heterozygous variation in protein domains families and its variability across species**

Important differences exist regarding the presence of heterozygous variation in protein domain families for avian species. Of all the families, the Cadherin family has shown no be the one that possess more heterozygous variation within the family in all birds, and also the one that suffers more fluctuations between species. Cadherins are type-1 transmembrane proteins that have important roles in cell adhesion and depend on calcium ions to function. Especially, they play an important role in different stages of development, as they intervene in processes like the separation of different tissue layers and cell migration (ref). The repeats we are analyzing here are present in all cadherins, and constitute the domains involved in extracellular calcium binding. The role of Cadherins in bird development have been investigated by Matsunaga *et al* (Matsunaga *et al.*, 2011). These authors studied the implications of Cadherins in the vocal learning of songbirds, and found that the expression of different types of Cadherins during development have an important effect in the learning ability to sing. It is known that singing birds have what is often called the “song system”, a neural network specialized for vocal learning.

Specifically, they concluded that the expression of Cad6B and Cad7 would activate different signaling pathways that intervene in the formation of the axonal branch in motor neurons. The first type of Cadherins would promote the formation of the axonal branch, while the other would inhibit it.

It is an interesting finding that we should take into account, particularly because this domain was the one with the higher heterozygous variation in all aspects (*Figure 15, Figure 16*).

In contrast, the Cadherin family does not show any interesting pattern in the phylogenetic tree (*Figure 17*), and the phytools analyses indicate that the phylogeny is not affecting the studied trait (heterozygous variation in the domain). They show, however, that the family VWA and the RVT_thumb to be influenced by the phylogenetic relationships (*Table 5, \( \lambda = 1, p\)-value < 0.05). In the case of RVT_thumb, this finding could be related to the fact that this family shows almost similar patterns regarding the
copy number of the domain, and also support the idea that a protein domain expansion actually occurred.

In spite of the Cadherin domain apparently not showing a clear pattern among species, more research should be done in this direction, to establish if the different patterns of variation observed among species are also related with differential vocal or other cognitive abilities. In this sense, more analysis could be done taking into account other aspects, such as ecological, anatomical (brain size, body size) or behavioral factors.

**CONCLUSION**

This project has given some insight into important aspects regarding heterozygous variation in 15 species birds and its distribution across different protein domain families. The heterozygous variation discovery, as well as the protein domain characterization was something that had never been done in this group of species. For this reason, we are considering the introduction of the heterozygous variation data into Ensembl, as it would facilitate further research.

Both the copy number analyses and the heterozygous variation found in them have rendered interesting results that could encourage other studies to further investigate in this direction. In the comparative analyses, four protein domain families related to retroviruses showed the same distribution regarding the copy number among all the species. The results suggest that an expansion of these families may have occurred in the Passerine clade.

Concerning the heterozygous variation in the protein domains, the Cadherin family was the one showing the most heterozygous variation and the most variability among species. In previous studies, this family has been linked to the learning ability in birds, so further studies could be done to investigate if the variation found is in some way related to differences among species in the role Cadherins play in the learning process. More comparative analyses should be done to examine possible relationships between the Cadherin domain family and phylogeny, considering other factors we have not explored in this project.
ANNEX 1:

SUBMITTING LSF JOBS TO THE CLUSTER OF THE EBI

The majority of the analysis I described for this project were done in the EBI cluster. To submit a job the cluster the following command and arguments were used:

Command: bsub

Arguments:
- J (job name)
- q (queue), it was always research –rh6
- M (the memory requirement in Mb)
- R (memory usage)
- o (lsf file that summarizes the output of the job done in the cluster, and indicates if the process was successful).

An example would be:

Bsub -J binary_penguin -q research-rh6 -M 42000 -R “rusage[mem=42000]” -o binary_penguin.lsfout

There was the possibility of submitting a script with the . An example of this would be:

Bsub < script.sh (command to submit the script)

#!/bin/bash
#BSUB -J download_penguin_fastq
#BSUB -R “rusage[mem=3000]”
#BSUB -M 3000
#BSUB -q research-rh6
#BSUB -o downloading_penguin_fastq.lsfout

[ordered job]

To execute many jobs at the same time we could submit it as job array. The number of jobs is indicated in the first argument (here 414). The %I means that a a lsf file will be created per job. An example:

#!/bin/bash
#BSUB -J splitted_to_vcf[1-414]
#BSUB -R “rusage[mem=8000]”
#BSUB -M 8000
#BSUB -q research-rh6
#BSUB -o splitted_to_vcf.lsfout%I
ANNEX 2

PHYTOOLS PACKAGE
The package provides with some function to do different types of phylogenetic analysis. It focuses, especially, in the phylogenetic analysis of comparative data from species. For this project we were interested to know if there was phylogenetic signal. This phenomenon occurs when related species have more similar characteristics than could be expected if we drawn species at random. There are many methods to quantify this pattern, but consensus has not been reached yet regarding which one is the best. Here we used Pagel method for the analysis.
The function used was `phylosig()`. The method chosen was the “lambda” method of Pagel, and P-values were calculated for a likelihood ratio test against the null hypothesis that lambda=0.

\[ \Lambda = 0 \quad \text{No phylogenetic signal} \]
\[ \Lambda = 1 \quad \text{Evidence of phylogenetic signal} \]

Pagel \( \lambda \) method (1999)
This method measures the level of phylogenetic dependence for an observed trait (Münkemüller et al., 2012), assuming a Brownian model of evolution. The relationships in the phylogeny are supposed to define the covariance matrix of the traits studied, nevertheless, there can be no phylogeny-related aspects that can be affecting the evolution of traits among species. This possibility is defined by the coefficient \( \lambda \), and searches for the phylogeny that confirms the best fit of the data to a Brownian model of evolution. The \( \lambda \) coefficient are not larger than 1, this value indicating that the related species share more similarities than expected under a Brownian model.

Brownian motion model of evolution
This model defines the evolution of a continuous trait as a random walk. If an especiation events occurs, the evolution of the trait continues independently. Under a Brownian model, the traits values are supposed to follow a normal distribution among the phylogeny.
BIBLIOGRAPHY


http://doi:10.1038/35025058