Bioinformatics, 2016, 0–0
doi: 10.1093/bioinformatics/xxxxx
Advance Access Publication Date: DD Month 2016
Genetics and Population Analysis

Genetics and Population Analysis

Human MicroRNAs and Copy Number Variation

Reardon M[1,*], Ronshaugen M[1] and Griffiths-Jones S [1]

[1]Faculty of Life Sciences, The University of Manchester, Carys Bannister Building, Dover Street, Manchester, M13 9PL, UK

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: MicroRNAs are ~22 nucleotide non-coding RNAs that play an important role in regulating expression levels of protein-coding genes. Here we investigate the distributions of genes with respect to copy number variations and miRNA targeting levels to better understand the buffering of protein dosage in aneuploidy and diseases such as cancer.

Results: We show that human miRNAs are significantly more likely than protein-coding genes to be gained in both healthy and pathogenic copy number variations. We also show that protein-coding genes that are gained or lost in pathogenic copy number variations are targeted by miRNAs significantly less than those that are not gained or lost, reinforcing the view that miRNAs buffer protein-coding gene dosage changes.

Availability: Source code is available in the appendix.

Contact: mark.reardon@postgrad.manchester.ac.uk

Supplementary information: Source code is available in the appendix.

Declaration

This dissertation is the student's original work unless referenced clearly to the contrary and no portion of the work referred to in the dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Intellectual Property Statement

i. The author of this dissertation (including any appendices and/or schedules to this dissertation) owns certain copyright or related rights in it (the "Copyright") and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this dissertation, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has entered into. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the "Intellectual Property") and any reproductions of copyright works in the dissertation, for example graphs and tables ("Reproductions"), which may be described in this dissertation, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this dissertation, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy, in any relevant Dissertation restriction declarations deposited in the University Library, and The University Library's regulations.

Word count: 5624 (calculated without appendix and therefore different to TurnItIn calculation)

1 Introduction

MicroRNAs (miRNAs) are short, non-coding RNAs with a median length of 22 nucleotides that were first discovered in 1993 (Lee, et al., 1993) and are found in animals, plants, algae and some DNA-based viruses. MiRNAs act as post-transcriptional regulators of gene expression by binding to complementary target regions in the 3'UTR of protein-coding gene mRNA transcripts and suppressing the translation of the mRNA into protein in the ribosome (Ambros, 2004).

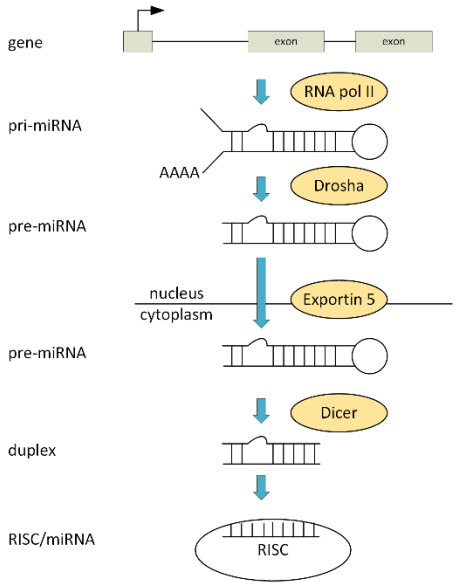

Figure 1. Biogenesis of an intergenic miRNA. Pri-miRNA are transcribed from the gene by RNA polymerase II and then cleaved into a pre-miRNA by Drosha. The pre-miRNA is exported to the cytoplasm by Exportin 5, cleaved by Dicer into an RNA duplex and then incorporated into the RISC. All enzymes are shown in yellow.

The human genome contains at least 1700 miRNAs which are transcribed either from non-coding miRNA genes or from the introns of other protein-coding genes (Bartel, 2004). Most animal miRNAs are transcribed by RNA polymerase II into transcripts called primary miRNAs (pri-miRNAs) containing one or more ~78 nucleotide RNA stem-loop hairpins called precursor miRNAs (pre-miRNAs) (Lee, et al., 2004). These pre-miRNAs are recognized by the DGCR8 protein which, together with the enzyme Drosha, causes the hairpin to be cleaved from the primary transcript (Lee, et al., 2003) before being exported from the nucleus to the cytoplasm by the Exportin-5 protein. In the cytoplasm the pre-miRNA is cleaved by the enzyme Dicer into a ~22 nucleotide RNA duplex, one strand of which is incorporated into the RNA-induced silencing complex (RISC) as the mature miRNA (Murchison & Hannon, 2004). These RISC with incorporated miRNAs then down-regulate the expression of protein-coding genes by degrading the mRNA or preventing its translation into protein (Bartel, 2009).

The target sites of human mature miRNAs are computationally predicted (and only in some cases experimentally confirmed), with most miRNAs predicted to target hundreds of different protein-coding gene transcripts which are in turn the targets of hundreds of different mature miRNAs (Friedman, et al., 2009); actual targeting depends on both miRNA and mRNA being co-expressed and so will be lower than these values. Different approaches have been taken to miRNA target prediction but most incorporate identification of sites in the 3'UTR with complementary Watson-Crick base pairing to a six, seven or eight nucleotide 'seed' region at the 5' end of the miRNA (Zheng, et al., 2013). Features of the miRNA near the seed region have also been shown to be important in target prediction (Grimson, et al., 2007). While a target prediction algorithm based solely on seed region complementarity will not find all experimentally confirmed target sites it is a useful simplification for fast target prediction.

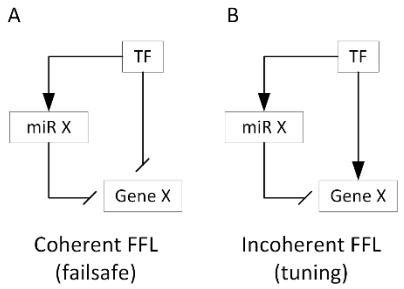

Figure 2. Examples of miRNA regulation of protein-coding genes. Pane A shows a coherent feedforward loop (FFL) with a transcription factor (TF) inhibiting a gene directly as well as by activating an inhibitory miRNA and thus acting as a failsafe FFL. Pane B shows an incoherent FFL with a transcription factor activating the transcription of a protein-coding gene while activating a miRNA to inhibit the translation of the protein-coding gene and so tuning the levels of the expressed protein over time.

Protein-coding genes and miRNAs together with transcription and signalling factors form complex regulatory networks characterized by various kinds of feedback and feedforward loops (Musilova & Mraz, 2015). In a coherent feedforward loop for example, a transcription factor inhibits the expression of a gene both directly and via miRNA inhibition (Musilova & Mraz, 2015). The parallel inhibitory paths have the effect of reinforcing cellular decisions against the effects of noise. Transcription factors can also activate gene transcription in an incoherent

feedforward loop while suppressing translation into protein by activating the corresponding miRNA, having the effect of dampening transcriptional noise and ensuring steadier protein levels across time (Musilova & Mraz, 2015).

Given the prevalence of these regulatory networks and their wide-ranging roles it is unsurprising that aberrant miRNA expression or repression can cause serious developmental problems as well as diseases such as cancer (Musilova & Mraz, 2015). In particular, it has been shown by knockout studies that losing a particular miRNA usually has little effect due to targeting redundancy but knock-in of a miRNA in either a different tissue or at a different developmental stage usually has serious effects by suppressing the production of essential proteins (Chen, et al., 2014).

Due to the role of miRNAs in buffering gene expression levels it seems likely that miRNAs are essential for the regulation of gene duplication and loss, whether in whole genome duplications or in smaller scale copy number variations (CNVs) known as aneuploidy. CNVs occur naturally in healthy individuals accounting for approximately 13% of human genetic variation but are also a common feature of cancerous cells and developmental problems such as trisomy 21 in patients with Down syndrome (Stankiewicz & Lupski, 2010).

The Database of Genomic Variation (DGV) (MacDonald, et al., 2014) is an online repository of all known CNVs in healthy humans whereas the Decipher project at the Sanger Institute publishes anonymized data from patients with serious pathologies arising from CNVs (Firth, et al., 2009). These datasets provide us with an amazing opportunity to not only examine the processes of miRNA and protein-coding gene gain and loss in aneuploidy but also to dissect the data with regard to gain or loss and CNV type using statistical analyses at the whole genome level.

Here we show that while there is no difference in the proportions of miRNAs and protein-coding genes that are lost in either healthy or pathogenic CNVs, a significantly higher proportion of miRNAs than protein-coding genes are gained in both healthy and pathogenic CNVs. In addition, we find that protein-coding genes that are gained or lost in pathogenic CNVs are targeted by miRNAs significantly less than genes unaffected by these CNVs whereas in healthy CNVs an increase in miRNA targeting of protein-coding genes is seen for losses with no difference for gains.

2 Materials and Methods

See Appendix for complete source code.

Gene location randomization simulations

To investigate the distribution of genes that overlap CNVs and to see whether the genomic locations of the genes could explain any variations in whether genes overlap with CNVs, a series of experiments were performed in R (version 3.2.3) (R Core Team, 2016) where the locations of the genes were randomized on their chromosomes 1000 times, the resulting simulated CNV overlap frequencies were obtained using R package GenomicRanges (version 1.22.4) (Lawrence, et al., 2013) and recorded. Due to the CPU-intensive nature of the simulations the experiment was run on multiple cores using R package snow (version 0.4-1) (Tierney, et al., 2015).

Precursor miRNA coordinates (n = 1,761) for chromosomes 1 to 22 were manually downloaded from ftp://mirbase.org/pub/mirbase/CURRENT/genomes/hsa.gff3 (miRBase version 21, genome assembly GRCh38) (Kozomara & Griffiths-Jones, 2014).

Protein-coding gene coordinates (n = 18,709) for chromosomes 1 to 22 were obtained from Ensembl (version 84, GRCh38) (Flicek, et al., 2014) using R package biomaRt (version 2.26.1) (Durinck, et al., 2009).

Chromosome sizes were manually downloaded from http://hgdownload.cse.ucsc.edu/goldenPath/hg38/bigZips/hg38.chrom.sizes (download date 17/4/16, GRCh38) (Rosenbloom, et al., 2015).

Decipher CNV coordinates (n = 228,437) were manually downloaded from https://decipher.sanger.ac.uk/files/downloads/population_cnv.txt.gz (version 9.6, GRCh37) (Firth, et al., 2009) and converted to GRCh38 using http://hgdownload.soe.ucsc.edu/goldenPath/hg19/liftOver/hg19ToHg38.over.chain and R packages rtracklayer (version 1.30.2) (Lawrence, et al., 2009) and liftOver (version 0.99.108006) (R Core Team, 2016).

DGV CNV coordinates (n = 325,560) were manually downloaded from http://dgv.tcag.ca/dgv/docs/GRCh38_hg38_variants_2015-07-23.txt (release date 23/7/15, GRCh38) (MacDonald, et al., 2014).

The means of the simulated frequencies were then compared to the observed frequencies with a p-value for each frequency calculated based on the difference and the standard deviation.

Gene and CNV overlap comparisons

To investigate whether the proportions of miRNAs and protein-coding genes that overlap pathogenic or healthy CNVs had significant differences and to see if the gene length was a possible confounding factor two experiments were performed, one using the full coordinates of the genes and one using only the midpoints of the genes. In both cases an overlap between a gene and a CNV was considered to be present if there was at least one nucleotide overlap (on the same strand where that information was present).

The proportions of each gene type (miRNA and protein-coding genes) overlapping each CNV type (gain, loss or none) were compared using R base function prop.test() with two-sided alternative hypothesis and Yate's continuity correction resulting in Pearson's chi-squared test statistics, estimates and p-values.

Further comparisons were made using the same statistical test of the proportions of each type of gene that were gained or lost in either pathogenic or healthy CNVs.

3'UTR targeting densities

To investigate whether the number of miRNAs targeting protein-coding genes influences the protein-coding genes' propensities to overlap CNVs a set of all possible targets was generated. All human 3'UTR sequences of protein-coding gene transcripts longer than 50bp (to avoid sequencing artefacts) (n = 52,093; downloaded with R package biomaRt) were converted into a FASTA file using R package Biostrings (version 2.38.4) (Pages, et al., 2005). All human mature miRNA sequences (n = 2,588; manually downloaded from ftp://mirbase.org/pub/mirbase/CURRENT/genomes/mature.fa.gz) were then analysed in conjunction with the 3'UTRs using an unpublished Perl script (Marco & Griffiths-Jones, 2009, personal communications) running on ActivePerl (version 5.20.2) that exhaustively determined the canonical six, seven and eight nucleotide seed possibilities as outlined in (Bartel, 2009).

The numbers of predicted miRNA targets on each protein-coding gene transcript were recorded and, to avoid the confounding factor of the widely varying 3'UTR lengths, normalized by the 3'UTR length to obtain a 3'UTR targeting 'density' for each transcript. The targeting density for each protein-coding gene was then calculated as the mean of its transcripts' targeting densities.

The genes were then divided into subsets overlapping gains and losses in each CNV dataset and their mean targeting densities were compared to that of the genes that overlapped no CNVs using analysis of variance.

3 Results

Gene location randomization simulations

In order to investigate whether there are any biological signals or processes acting on the retention or loss of miRNAs or protein-coding genes in copy number variations we wanted to see if the distributions of gained or lost miRNAs and protein-coding genes were different to random expectation. To investigate this a series of simulation experiments (n = 1000) were performed that assigned random genomic locations to the genes on each chromosome. In each simulation the same CNV overlap calculations with each gene's midpoint were performed as for the real observations and the mean count was calculated for each overlap frequency. These mean simulated overlap frequency counts were compared to the observed frequency counts and significance p-values calculated from the difference between the observed and simulated counts and the standard deviation of the simulated counts for both miRNAs and protein-coding genes.

miRNAs are both gained and lost more often than expected in healthy CNVs when the observed count of each overlap frequency is compared to the mean of the overlap frequencies arising from randomized gene locations and the difference is statistically significant (Figure 3). For pathogenic CNVs on the other hand there is a noticeably different pattern where gains are more likely than expected but losses are less likely.

The same differences between healthy and pathogenic CNVs are observed for protein-coding genes (Figure 4) with pathogenic gains observed more often than expected but losses less often; gains and losses are again both observed more often than expected in healthy CNVs with the exception that for some of the very rare high overlap counts the difference is reversed.

Gene and CNV overlap comparisons

It has been shown that miRNAs play an important regulatory role in buffering the effects of gene dosage (Musilova & Mraz, 2015) and have an effect on many genes if co-expressed. It could therefore be imagined that their gain or loss would be less acceptable to viable organisms than gain or loss of protein-coding genes. To determine whether this is seen in actual organisms the proportions of miRNAs and protein-coding genes gained or lost in both pathogenic and healthy CNVs were compared.

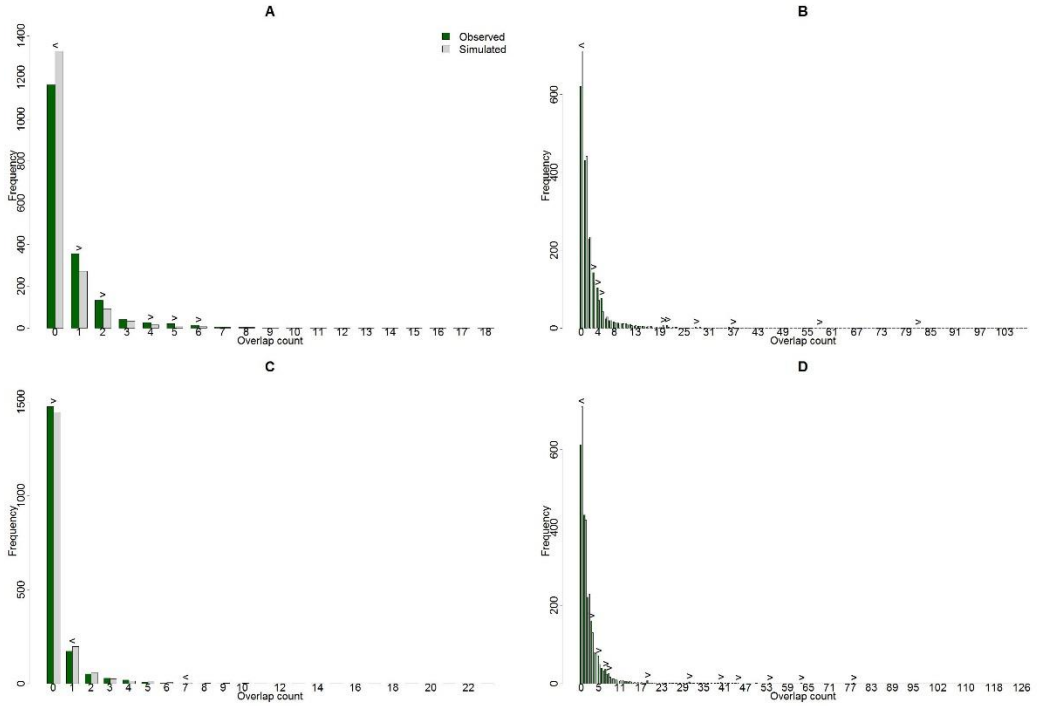

Figure 3. Observed and simulated miRNA gains and losses. The graphs show the observed frequencies in green and the mean of the simulated frequencies in grey. The arrows show where the observed is significantly more (>) or less (<) than the simulated. Pane A shows miRNAs gained in pathogenic CNVs, pane B shows miRNAs gained in healthy CNVs, pane C shows miRNAs lost in pathogenic CNVs and pane D shows miRNAs lost in healthy CNVs.

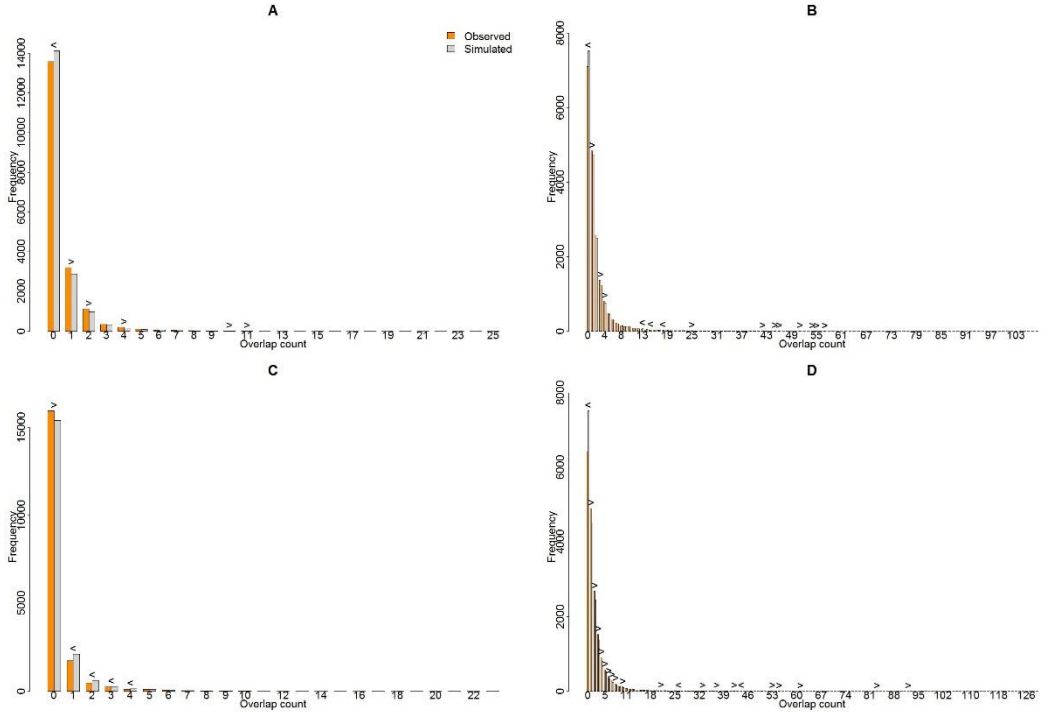

Figure 4. Observed and simulated protein-coding gene gains and losses. The graphs show the observed frequencies in orange and the mean of the simulated frequencies in grey. The arrows show where the observed is significantly more (>) or less (<) than the simulated. Pane A shows protein-coding genes gained in pathogenic CNVs, pane B shows protein-coding genes gained in healthy CNVs, pane C shows protein-coding genes lost in pathogenic CNVs and pane D shows miRNAs lost in protein-coding genes CNVs.

Unlike for protein-coding genes, gene models for miRNA genes are generally poor or have not yet been created and so only the precursor and mature miRNA coordinates are documented. The majority of intergenic miRNAs are transcribed and spliced from genes that are significantly longer than the pre-miRNA; for example, mir-9-1 has a promotor sequence approximately 20kb upstream as determined by spliced Expressed Sequence Tags (ESTs) and yet this transcript, with a total length ~30kb, only contains the pre-miRNA as far as is known.

Because of this lack of fully documented miRNA transcripts it is difficult to compare the overlap of miRNAs and protein-coding genes (which generally *do* have well-defined gene models) with CNVs in a manner unbiased by their lengths and, in the case of the miRNAs, the lack of the majority of their transcripts. To counter this two similar methods were investigated that respond differently to this bias (Figure 5).

In the first method investigated, the full known width of the gene in question is overlapped with the CNV and so correctly identifies protein-coding genes that overlap the CNV as the gene's correct width is known. In the case of miRNAs however, because the full transcript width is not known it is just the width of the precursor that can be considered with the current data and so CNVs that overlap parts of the transcript outside this region will be missed even though, in the case of a loss overlapping the promotor, the CNV would prevent the miRNA from being transcribed. In the second method only the midpoint of each gene is considered and so the genes are treated equally regardless of transcript length.

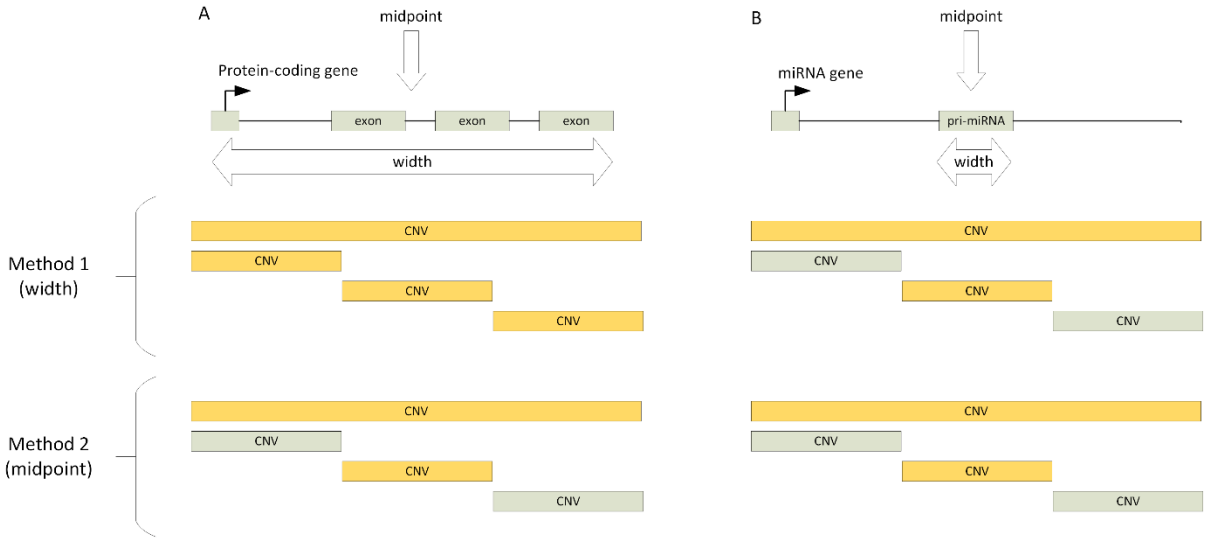

Figure 5. Two methods for considering gene/CNV overlaps. Pane A shows a protein-coding gene and a range of CNVs and pane B shows a miRNA gene and a range of CNVs. In method 1, the overlap between any part of the known width of the gene and the CNV is considered and it can be seen that because the miRNA's true transcript width is not known, method 1 will significantly under-report CNVs that overlap miRNAs (yellow highlight in figure). In method 2, only the midpoint of the gene is considered and so this bias is avoided.

The proportion of miRNAs and protein-coding genes that overlap both pathogenic and healthy CNVs were considered with both analysis methods (Figure 6) and it can be seen that the results are significantly different.

When analysed with the width-based method (Figure 6, panes A and B) miRNAs are significantly less likely than protein-coding genes to overlap CNVs of any kind. This result is biased however by the large difference in gene widths between miRNA genes and protein-coding genes.

To remove this bias the analysis was repeated with the midpoint-based method (Figure 6, panes C and D) and a strikingly different picture now emerges: miRNAs are now significantly *more* likely than protein-coding genes to overlap pathogenic gains ($X^2 = 32.8$, df = 1, p-value = 1×10^{-8}), significantly more likely to overlap healthy gains ($X^2 = 4.89$, df = 1, p-value = 0.027) and significantly less likely to overlap no pathogenic CNV at all ($X^2 = 25.9$, df = 1, p-value = 3.59×10^{-7}).

The midpoint-based method was used for all subsequent overlap analyses to avoid the bias introduced by the use in this study of only the precursor miRNAs which are much shorter than the protein-coding genes.

In addition to the differences between miRNAs and protein-coding genes discussed above, miRNAs were

significantly more likely to be gained than lost in pathogenic CNVs ($X^2 = 145$, df = 1, p-value < 2×10^{-16}) but there was no significant difference in healthy CNVs ($X^2 = 0.0799$, df = 1, p-value = 0.778). Protein-coding gene gains were significantly more likely to be gained than lost in pathogenic CNVs ($X^2 = 886$, df = 1, p-value < 2×10^{-16}) but significantly less likely to be gained than lost in healthy CNVs ($X^2 = 55$, df = 1, p-value = 1×10^{-13}).

Similarly, the differences between the CNV datasets for gained and lost miRNAs and protein-coding genes were analysed. miRNAs were significantly less likely to be gained in pathogenic CNVs than in healthy CNVs ($X^2 = 335$, df = 1, p-value < 2×10^{-16}) with the same pattern exhibited for miRNA losses ($X^2 = 876$, df = 1, p-value < 2×10^{-16}), protein-coding gene gains ($X^2 = 4536$, df = 1, p-value < 2×10^{-16}) and protein-coding gene losses ($X^2 = 10067$, df = 1, p-value < 2×10^{-16}).

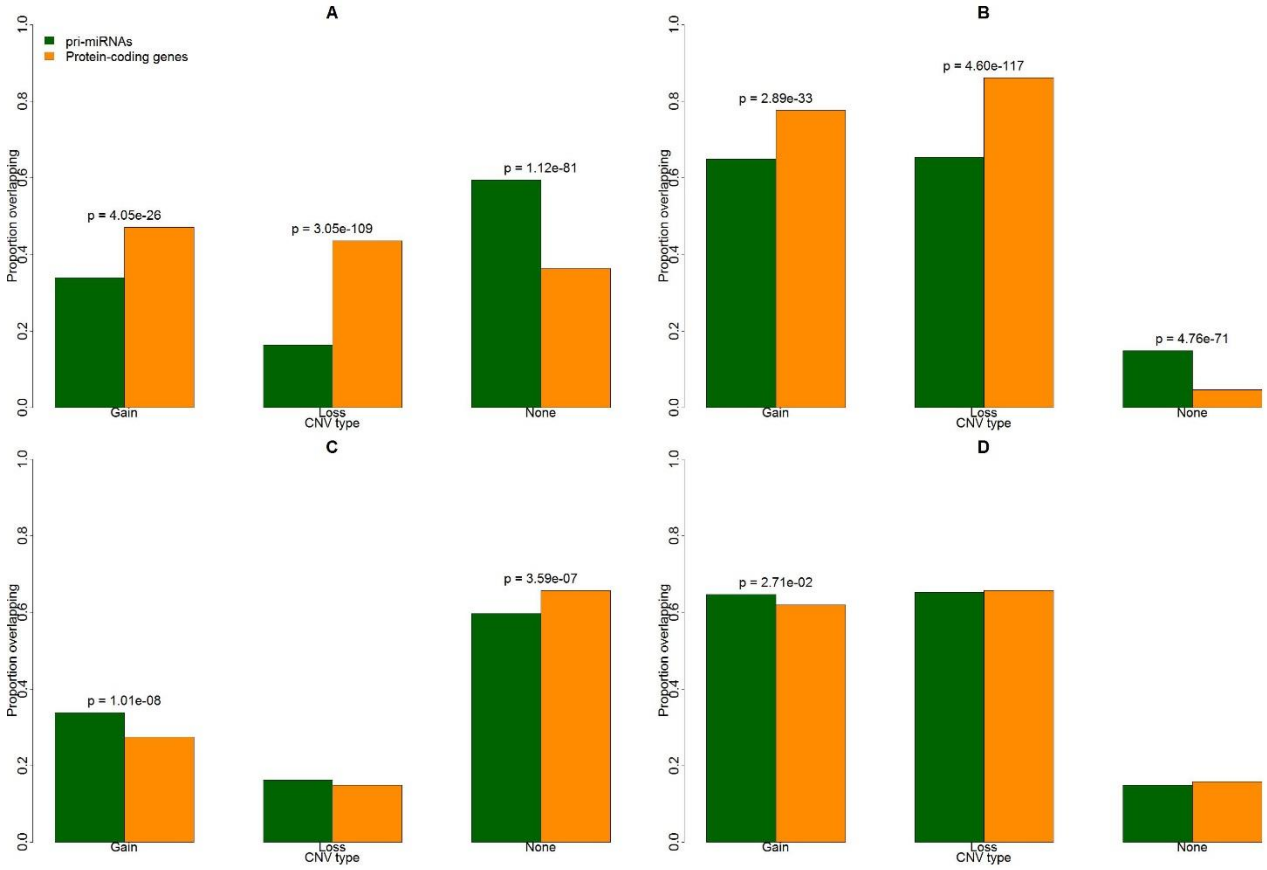

Figure 6. miRNAs and protein-coding genes overlapping CNVs compared. Pane A shows the proportions of miRNAs and protein-coding genes that are gained, lost or not in pathogenic CNVs, as determined by the width-based method. Pane B shows the proportions of miRNAs and protein-coding genes that are gained, lost or not in healthy CNVs, as determined by the width-based method. Pane C shows the proportions of miRNAs and protein-coding genes that are gained, lost or not in pathogenic CNVs, as determined by the midpoint-based method. Pane D shows the proportions of miRNAs and protein-coding genes that are gained, lost or not in healthy CNVs, as determined by the midpoint-based method. miRNAs are shown in green and protein-coding genes in orange. P-values are shown above the proportions when the difference is significant.

3'UTR targeting densities

As gene dosage is thought to be regulated by miRNAs we wanted to see if the protein-coding genes that are gained or lost in CNVs are differently targeted by miRNAs. The targeting 'density' for each protein-coding gene was calculated by counting the number of predicted miRNA targets of each transcript's 3'UTR, normalizing to the 3'UTR length and then averaging the transcript targeting densities for each gene (transcripts shorter than 50 nucleotides were excluded to avoid bias from sequencing artefacts).

After dividing the protein-coding genes into subsets of those that are gained (n = 5,127), lost (n = 2,776) or not in pathogenic CNVs (n = 6,789) the mean targeting densities of each subset were compared (Figure 7, pane A). When the null hypothesis that the means of the subsets are the same was considered using one way ANOVA it was found that there are highly significant differences between the means of the targeting densities of genes gained, lost and not in pathogenic CNVs (F-statistic: 74.9 on 2 and 14689 DF, p-value: $< 2 \times 10^{-16}$). Post hoc comparisons using the Tukey HSD test confirmed that the mean targeting densities for gains (mean = 0.363, SD = 0.135) and losses (mean = 0.342, SD = 0.148) were significantly different to those for genes in no CNVs (mean = 0.378, SD = 0.127), with an adjusted p-value = 0 in each case.

The same analysis was repeated for the genes that are gained (n = 11,606), lost (n = 12,296) or not in healthy CNVs (n = 869) (Figure 7, pane B). When the null hypothesis that the means of the subsets are the same was considered using one way ANOVA it was found that there is a highly significant difference between the means of the targeting densities of genes lost and not in pathogenic CNVs but no significant difference between gains and genes in no CNV (F-statistic: 15.6 on 2 and 24768 DF, p-value: 5.9×10^{-9}). Post hoc comparisons using the Tukey HSD test confirmed that the mean targeting density for gains (mean = 0.366, SD = 0.125) was not significantly different to that for genes in no CNV (mean = 0.365, SD = 0.133) but the mean targeting density for losses (mean = 0.375, SD = 0.129) was significantly different to that for genes in no CNVs (adjusted p-value = 0.049).

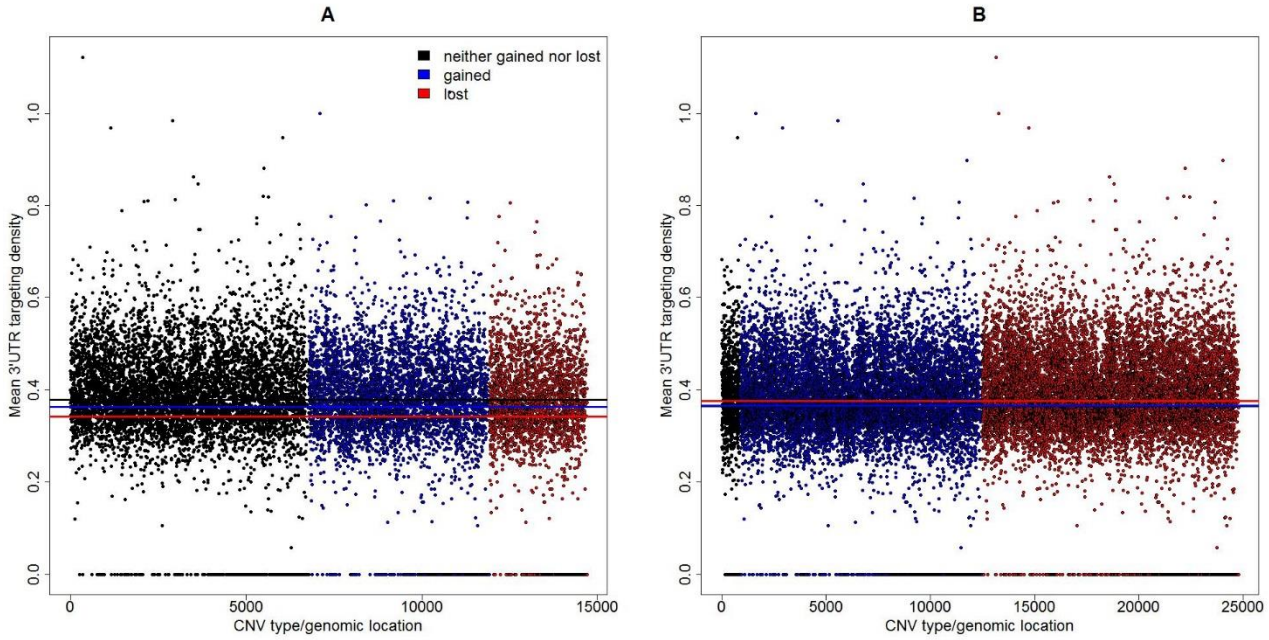

Figure 7. Protein-coding gene targeting densities. Pane A shows the mean miRNA targeting densities of all protein-coding genes grouped by whether they are in no, gained or lost pathogenic CNVs and represented as black, blue and red dots respectively. Pane B shows the mean miRNA targeting densities of all protein-coding genes grouped by whether they are in no, gained or lost healthy CNVs and again represented as black, blue and red dots respectively. Within each group the protein-coding genes are ordered by genomic location. The means of each group are superimposed as horizontal lines with the same colour as the protein-coding gene dots. 3'UTRs shorter than 50 nucleotides were excluded to avoid bias from sequencing artefacts.

4 Discussion

Where there are copy number variations in the genome there is the potential for change in the dosage of protein-coding genes to occur. MicroRNAs have been suggested to buffer or modulate these dosage changes as part of complex regulatory networks. We wanted to ask questions about the distribution of CNVs with respect to miRNAs and protein-coding genes as well as investigate the potential interactions between miRNAs and protein-coding genes located in CNV regions.

A recent study of miRNA conservation in rainbow trout showed that following a relatively recent whole genome duplication event between 25 and 100 million years ago around half of the duplicated protein-coding genes have been lost but almost all the duplicated miRNAs genes have been retained (Berthalot, et al., 2014). The same study also showed that this differential conservation is unlikely to be simply due to the disparity in gene lengths between miRNAs and protein-coding genes. We wanted to examine human CNVs to see if similar patterns could be found.

First we decided to see if the distributions of gained and lost miRNAs and protein-coding genes were different from random expectation because if they were then that would imply that there is a biological signal or process acting on their retention or loss in CNVs. We showed that the distributions are indeed significantly different from random expectation and moreover we also showed that there is a difference in the distributions with respect to pathogenic and healthy CNVs.

In healthy CNVs we showed that at frequencies in the distributions where there was a significant difference both miRNAs and protein-coding genes were gained and lost more often than expected. One explanation is that in order for organisms to remain healthy in the presence of copy number variations it is important that when protein-coding genes are gained, the miRNAs that buffer the dosage change in these genes are also gained. Conversely, when protein-coding genes are lost in healthy organisms the evolutionary pressure for the regulating miRNAs to be retained would also be lost and so the loss of the regulating miRNAs would be expected, as our results here imply.

When the same analysis was performed with pathogenic CNVs however a different pattern emerged: miRNAs and protein-coding genes were again gained more often than expected but they were both lost *less* often than expected. This is consistent with the idea that loss of protein-coding gene but retention of the regulating miRNAs is deleterious.

Next we wanted to see if the proportions of miRNAs and protein-coding genes that are gained or lost were significantly different since, given the expected regulatory role of miRNAs in buffering the effects of gene dosage, it could be expected that the gain of miRNAs would be less acceptable to viable organisms since gain of function has been shown to usually have noticeable phenotypical effects whereas loss of function often has no visible effect due to redundancy in miRNA targeting networks (Chen, et al., 2014).

We found no difference between the proportions of miRNAs and protein-coding genes that are lost in either pathogenic or healthy CNVs, consistent with our earlier observations of the frequency distributions. Interestingly however, we found that a significantly higher proportion of miRNAs than protein-coding genes were gained in both pathogenic and healthy CNVs. This is surprising given that previous studies have shown that gain of miRNA function usually leads to a phenotype (Chen, et al., 2014), though it is worth noting that in our results a significantly lower proportion of miRNAs than protein-coding genes overlap no pathogenic CNV suggesting that gain of miRNA function is more likely than not to be deleterious. One possible interpretation of this contradictory result is that the higher proportion of miRNAs gained is a relic of the two whole genome duplication events in human evolutionary history. A more likely explanation however is that the method of overlap analysis chosen (see methods) has an unexpected bias for over-reporting the proportion of miRNAs that overlap CNVs.

Finally, we wanted to see whether the protein-coding genes' propensities to overlap CNVs were influenced by the numbers of miRNAs that are predicted to target them to find out if this could explain the differences in frequency distributions and relative proportions of CNV overlap. We found that protein-coding genes that were both gained or lost in pathogenic CNVs were significantly less targeted by miRNAs than protein-coding genes that were not in pathogenic CNVs at all. This strongly suggests that the protein-coding genes that are less regulated by miRNAs are more likely to have deleterious effects when gained or lost, reinforcing the view that miRNAs are important for buffering changes in gene dosage.

When we compared the targeting densities of protein-coding genes that are gained or lost in healthy individuals we found that in contrast to the situation in pathogenic CNVs, gained protein-coding genes were no more likely to be targeted than protein-coding genes that overlap no CNVs, consistent with the idea that in healthy organisms it is the protein-coding genes that are regulated by miRNAs that can be safely gained since the miRNAs buffer the change in dosage.

When our results are considered in the round we can see that there are a lot of biological signals that can be detected

by analysing genomic datasets such as copy number variations in conjunction with gene models and it is clear that miRNAs have important and biologically significant effects in buffering the effects of gene dosage. Further work into the relationships between miRNAs and the protein-coding genes that they are predicted to target will give us greater insight into the effects of gain and loss of function in aneuploidy and the roles that miRNAs play in diseases such as cancer.

5 Acknowledgements

This study makes use of data generated by the DECIPHER community. A full list of centres who contributed to the generation of the data is available from http://decipher.sanger.ac.uk and via email from decipher@sanger.ac.uk.

6 References

Ambros, V., 2004. The functions of animal microRNAs. *Nature,* pp. 350-355.

Bartel, D. P., 2004. MicroRNAs: Genomics, biogenesis, mechanism and function. *Cell,* Volume 116, pp. 281-297.

Bartel, D. P., 2009. MicroRNAs: Target Recognition and Regulatory Functions. *Cell,* pp. 215-233.

Berthalot, C. et al., 2014. The rainbow trout genome provides novel insights into evolution after whole-genome duplication in vertebrates. *Nat. Commun.,* p. 5:3657 doi: 10.1038/ncomms4657.

Charif, D. & Lobry, J. R., 2007. Seqin{R} 1.0-2: a contributed package to the {R} project for statistical computing devoted to biological sequences retrieval and analysis. In: U. B. a. M. P. a. H. R. a. M. Vendruscolo, ed. *Structural approaches to sequence evolution: Molecules, networks, populations.* New York: Springer Verlag, pp. 207-232.

Chen, Y.-W.et al., 2014. Systematic Study of Drosophila MicroRNA Functions Using a Collection of Targeted Knockout Mutation. *Developmental Cell,* Volume 31, pp. 784-800.

Durinck, S., Spellman, P. T., Birney, E. & Huber, W., 2009. Mapping identifiers for the integration of genomic datasets with the R/Bioconductor package biomaRt. *Nature Protocols 4,* pp. 1184-1191, doi:10.1038/nprot.2009.

Firth, H. V. et al., 2009. DECIPHER: Database of Chromosomal Imbalance and Phenotype in Humans using Ensembl Resources. *Am.J.Hum.Genet,* Volume 84, pp. 524-533.

Flicek, P. et al., 2014. Ensemble 2014. *Nucleic Acids Research,* Volume 42, pp. 749-755.

Friedman, R. C., Farh, K. K.-H., Burge, C. B. & Bartel, D. P., 2009. Most mammalian mRNAs are conserved targets of microRNAs. *Genome Research,* pp. 92-105.

Grimson, A. et al., 2007. MicroRNA Targeting Specificity in Mammals: Determinants beyond Seed Pairing. *Molecular Cell,* Volume 27, pp. 91-105, doi:10.1016/j.molcel.2007.06.017.

Kozomara, A. & Griffiths-Jones, S., 2014. miRBase: annotating high confidence microRNAs using deep sequencing data.. *Nucleic Acids Research,* Volume 42, pp. 68-73.

Lawrence, M., Gentleman, R. & Carey, V., 2009. rtracklayer: an {R} package for interfacing with genome browsers. *Bioinformatics,* Volume 25, pp. 1841-1842, doi:10.1093/bioinformatics/btp328.

Lawrence, M. et al., 2013. Software for Computing and Annotating Genomic Ranges. *PLoS Comput Biol 9(8),* p. e1003118. doi:10.1371/journal.pcbi.1003118.

Lee, R. C., Feinbaum, R. L. & Ambros, V., 1993. The C. elegans heterochronic gene lin-4 encodes small RNAs with antisense complementarity to lin-14. *Cell,* Volume 75, pp. 843-854.

Lee, Y. et al., 2003. The nuclear RNase III Drosha initiates microRNA processing. *Nature,* pp. 415-419.

Lee, Y. et al., 2004. MicroRNA genes are transcribed by RNA polymerase II. *Embo J,* Volume 20, pp. 4051-4060.

MacDonald, J. R. et al., 2014. The Database of Genomic Variants: a curated collection of structural variation in the human genome. *Nucleic Acids Research,* Volume 42, pp. 986-992.

Marco, A. & Griffiths-Jones, S., 2009, personal communications. *Seed Vicious.* s.l.:s.n.

Murchison, E. & Hannon, G. J., 2004. miRNAs on the move: miRNA biogenesis and the RNAi machinery. *Curr. Opin. Cell Biol,* 16(3), pp. 223-229.

Musilova, K. & Mraz, M., 2015. MicroRNAs in B-cell lymphomas: how a complex biology gets more complex. *Leukemia,* Volume 29, pp. 1004-1017.

Pages, H., Aboyoun, P., Gentleman, R. & DebRoy, S., 2005. Biostrings: String objects representing biological sequences, and matching algorithms.

R Core Team, 2016. liftOver.

R Core Team, 2016. R: A Language and Environment for Statistical Computing.

Rosenbloom, K. R. et al., 2015. The UCSC Genome Browser database: 2015 update. *Nucleic Acids Research,* Volume 43, pp. 670-681.

Stankiewicz, P. & Lupski, J. R., 2010. Structural Variation in the Human Genome and its Role in Disease. *Annu. Rev. Med.,* Volume 61, pp. 437-455.

Tierney, L., Rossini, A. J., Li, N. & Sevcikova, H., 2015. *snow: Simple Network of Workstations.* [Online] Available at: https://CRAN.R-project.org/package=snow

Zheng, H. et al., 2013. Advances in the Techniques for the Prediction of microRNA Targets. *Int. J. Mol. Sci,* Volume 14, pp. 8179-8187, doi:10.3390/ijms14048179.

7 Appendix – Source code

Analysis/1 - How do the frequencies of miRNAs and protein-coding genes that are in copy number variants differ from those which would be expected given random genomic locations.r

```
# Question 1 - How do the frequencies of miRNAs and protein-coding genes that are in copy number variants differ from those which would
be
#              expected given random genomic locations?

require(GenomicRanges)
require(parallel)
require(snow)

setwd("C:/Users/mark/iCloudDrive/MSc/BIOL61230 Research Projects/1/Analysis")

# load miRNAs, protein-coding genes and CNVs
source(file = "Data/Loaders/pri-miRNAs.r")
source(file = "Data/Loaders/Protein-coding genes.r")
source(file = "Data/Loaders/Chromosomes.r")
source(file = "Data/Loaders/Decipher CNVs.r")
source(file = "Data/Loaders/DGV CNVs.r")

# returns significance asterisks for a P value
significance <- function(p)
{
  result <- ""
  if (!is.na(p) && p <= 0.1) result <- "."
  if (!is.na(p) && p <= 0.05) result <- "*"
  if (!is.na(p) && p <= 0.01) result <- "**"
  if (!is.na(p) && p <= 0.001) result <- "***"
  return(result)
}

# returns a data frame contain statistics about the simulations
summarise <- function(observations, simulations)
{
  # calculate averages and z-based p values
  simulation.mean <- apply(simulations, 2, function(column) mean(column))
  simulation.median <- apply(simulations, 2, function(column) median(column))
  simulation.sd <- apply(simulations, 2, function(column) sd(column))
  z <- (observations - simulation.mean) / simulation.sd
  p <- sapply(z, function(overlapZ) if (!is.na(overlapZ) && overlapZ < 0) return(pnorm(overlapZ)) else return(1 - pnorm(overlapZ)))

  # count how many simulations each overlap ocurred in and how many of each simulation are less than, equal to or more than the ob-
served
  simulation.in <- numeric(length(observations))
  less <- numeric(length(observations))
  equal <- numeric(length(observations))
  more <- numeric(length(observations))
  for (overlap in 1:length(observations))
  {
    simulation.in[overlap] <- sum(simulations[, overlap] > 0)
    less[overlap] <- sum(simulations[, overlap] < observations[overlap])
    equal[overlap] <- sum(simulations[, overlap] == observations[overlap])
    more[overlap] <- sum(simulations[, overlap] > observations[overlap])
  }

  # calculate significance based on less than and more than counts
  experimentCount <- length(simulations[ ,1])
  less.div <- less / experimentCount
  more.div <- more / experimentCount
  less.sig <- sapply(less / experimentCount, significance)
  more.sig <- sapply(more / experimentCount, significance)

  # return the summary
  return (data.frame(
    overlaps = 0:(length(simulations[1,]) - 1),
    observations, simulation.in,
    simulation.mean, simulation.median, simulation.sd, z, p, p.sig = sapply(p, significance),
    less, less.div, less.sig,
    equal,
    more, more.div, more.sig))
}

# writes the output files if filenames were specified
writeOutput <- function(simulations, simulationsFilename, summary, summaryFilename)
{
  # simulations
  if (!is.null(simulationsFilename))
  {
    write.table(simulations, file = simulationsFilename, sep = "\t", row.names = F, col.names = F)
  }

  # summary
  if (!is.null(summaryFilename))
  {
    write.table(summary, file = summaryFilename, sep = "\t", row.names = F)
  }
}

# creates a random variant of the query and returns a matrix containing its overlaps with the subject as a frequency table
randomiseOverlaps <- function(query, subject, chromosomes)
{
  require(GenomicRanges)
  # create vectors of randomly chosen chromosomes and positions within each
  randomChromosomes <- sample(seqnames(query), size = length(query), replace = T)
  randomStarts <- sapply(randomChromosomes, function(chr) sample.int(chromosomes$Length[chromosomes$Name == chr], 1))

  # create GRange holding random query-like ranges (lengths the same; chromosome, starts and strands random)
  randomRanges <- GRanges(
    seqnames = randomChromosomes,
    ranges = IRanges(
      start = randomStarts,
      width = width(query),
      names = names(query)),
    strand = sample(c("+", "-"), size = length(query), replace = T))
```

```
  # overlap the random ranges with the subject ranges and return as a frequency table
  randomOverlaps <- countOverlaps(query = randomRanges, subject = subject)
  return(matrix(data = table(factor(randomOverlaps, levels = min(randomOverlaps):max(randomOverlaps))), nrow = 1))
}

# performs the randomisation simulations
#  (intended to be run on cluster nodes and so requires variables 'observations, query, subject, chromosomes, simulationCount'
#   and function 'randomiseOverlaps' to be exported to the nodes)
performSimulations <- function()
{
  # create matrix to hold randomised simulations
  simulations <- matrix(data = 0, nrow = simulationCount, ncol = length(observations))

  # for each simulation...
  for(simulation in 1:simulationCount)
  {
    # randomise the query and summarise as a frequency table
    randomObservations <- randomiseOverlaps(query, subject, chromosomes)

    # add simulation columns if more are needed
    while(length(randomObservations) > length(simulations[1,]))
    {
      simulations <- cbind(simulations, rep.int(0, simulationCount))
    }

    # update the simulations
    for(randomObservation in 1:length(randomObservations))
    {
      simulations[simulation, randomObservation] <- randomObservations[1, randomObservation]
    }
  }

  # return the simulations
  return(simulations)
}

# overlaps the query over the subject and compares to the distribution obtained from random repositioning of the query
performExperiment <- function(pane, showLegend, queryName, subjectName, colour)
{
  # get the variables for the experiment
  query <- get(queryName)
  subject <- get(subjectName)
  chromosomes <- get("chromosomes")
  simulationCount <- get("simulationCount")
  totalSimulationCount <- simulationCount

  # abort if the simulation count is not divisible by the number of cores to simplify partitioning
  coreCount <- detectCores()
  if (simulationCount %% coreCount != 0)
  {
    stop(sprintf("Simulation count (%d) is not divisible by the number of cores (%d).", simulationCount, coreCount))
  }

  # overlap the query with the subject and summarise as frequency table
  overlaps <- countOverlaps(query, subject)
  observations <- matrix(data = table(factor(overlaps, levels = min(overlaps):max(overlaps))), nrow = 1)

  # if the summary does not already exist...
  experimentFilename <- sprintf("Results/1/Experiments/%s.%s.%d.tab", queryName, subjectName, totalSimulationCount)
  summaryFilename <- sprintf("Results/1/Summaries/%s.%s.%d.tab", queryName, subjectName, totalSimulationCount)
  if (!file.exists(summaryFilename))
  {

    # make cluster with maximum number of cores
    cluster <- makeCluster(coreCount)

    # partition the simulations
    simulationCount <- simulationCount / coreCount

    # export data, parameters and required functions to each node
    clusterExport(cluster, "observations", envir = environment())
    clusterExport(cluster, "query", envir = environment())
    clusterExport(cluster, "subject", envir = environment())
    clusterExport(cluster, "chromosomes", envir = environment())
    clusterExport(cluster, "simulationCount", envir = environment())
    clusterExport(cluster, "randomiseOverlaps", envir = environment())

    # perform the simulations in parallel
    nodeSimulations <- clusterCall(cl = cluster, fun = performSimulations)

    # stop the cluster
    stopCluster(cluster)

    # create matrix to hold simulations from all the nodes
    maxColumns <- 0
    for (node in 1:coreCount)
    {
      nodeSimulation <- nodeSimulations[[node]]
      nodeColumns <- length(nodeSimulation[1, ])
      if (nodeColumns > maxColumns)
      {
        maxColumns <- nodeColumns
      }
    }
    simulations <- matrix(data = 0, nrow = simulationCount * coreCount, ncol = maxColumns)

    # copy the simulations from each node's results
    simulationIndex <- 1
    for (node in 1:coreCount)
    {
      nodeSimulation <- nodeSimulations[[node]]
      for (nodeSimulationIndex in 1:simulationCount)
      {
        for (observation in 1:length(nodeSimulation[1, ]))
        {
          simulations[simulationIndex, observation] <- nodeSimulation[nodeSimulationIndex, observation]
        }
        simulationIndex <- simulationIndex + 1
      }
```

```
    }

    # adjust observation length to match simulations in case there are more values in simulations
    priorObservations <- length(observations)
    if (length(observations) < length(simulations[1,]))
    {
      length(observations) <- length(simulations[1,])
      # zero the new observations
      for (observation in priorObservations:length(observations))
      {
        observations[observation] <- 0
      }
    }

    # summarise the simulations
    summary <- summarise(observations, simulations)

    # output raw experimental data and results
    writeOutput(simulations, experimentFilename, summary, summaryFilename)
  }
  else
  {
    summary <- read.table(file = summaryFilename, header = T, sep = "\t")
  }

  # zero missing values
  summary$observations[!is.numeric(summary$observations)] <- 0
  summary$simulation.mean[!is.numeric(summary$simulation.mean)] <- 0

  # output the plot
  ymax = max(c(summary$observations, summary$simulation.mean)) * 1.1
  barplot(t(matrix(c(summary$observations, summary$simulation.mean), ncol = 2)),
          beside = T, space = c(0,1),
          main = pane,
          names.arg = 0:(length(summary$observations) - 1),
          xlab = "Overlap count", ylab = "Frequency",
          col = c(colour, "lightgrey"),
          cex.axis = 3, cex.names = 3, cex.main = 4, cex.lab = 3,
          ylim = c(0, ymax))
  if (showLegend)
  {
    legend("topright", legend = c("Observed", "Simulated"), fill = c(colour, "lightgrey"), bty = "n", cex = 3)
  }

  # output significance asterisks
  threshold <- 0.05
  for (i in 1:length(summary$p))
  {
    p <- summary$p[i]
    z <- summary$z[i]
    if (!is.na(p) & !is.na(z) & p < threshold)
    {
      sig <- ifelse(z < 0, "<", ">")
      y <- max(c(summary$observations[i], summary$simulation.mean[i]))
      text((3 * i) - 1, y, sig, cex = 3, pos = 3)
    }
  }
}

# number of randomised simulations
simulationCount <- 1000

marginSize <- 6
# miRNA gains
jpeg(filename = sprintf("Results/1/miRNAs.%d.jpg", simulationCount), width = 3000, height = 2000)
par(mfrow = c(2,2), mar = c(marginSize, marginSize, marginSize, marginSize))
performExperiment("A", T, "primiRNAMidpoints", "decipherCNVsGained", "darkgreen")
performExperiment("B", F, "primiRNAMidpoints", "dgvCNVsGained", "darkgreen")

# miRNA losses
performExperiment("C", F, "primiRNAMidpoints", "decipherCNVsLost", "darkgreen")
performExperiment("D", F, "primiRNAMidpoints", "dgvCNVsLost", "darkgreen")
dev.off()

# protein-coding gene gains
jpeg(filename = sprintf("Results/1/Protein-coding genes.%d.jpg", simulationCount), width = 3000, height = 2000)
par(mfrow = c(2,2), mar = c(marginSize, marginSize, marginSize, marginSize))
performExperiment("A", T, "proteinCodingGeneMidpoints", "decipherCNVsGained", "darkorange")
performExperiment("B", F, "proteinCodingGeneMidpoints", "dgvCNVsGained", "darkorange")

# protein-coding gene losses
performExperiment("C", F, "proteinCodingGeneMidpoints", "decipherCNVsLost", "darkorange")
performExperiment("D", F, "proteinCodingGeneMidpoints", "dgvCNVsLost", "darkorange")
dev.off()
```

Analysis/2 - Are miRNA genes more or less likely than protein-coding genes to overlap copy number variants.r

```
# Question 2 - Are miRNA genes more or less likely than protein-coding genes to overlap copy number variants?

require(GenomicRanges)
setwd("C:/Users/mark/iCloudDrive/MSc/BIOL61230 Research Projects/1/Analysis")

# load pri-miRNAs, protein-coding genes and CNVs
source(file = "Data/Loaders/pri-miRNAs.r")
source(file = "Data/Loaders/Protein-coding genes.r")
source(file = "Data/Loaders/Decipher CNVs.r")
source(file = "Data/Loaders/DGV CNVs.r")

# adds p-value text to a plot just above the proportion bars if significantly different
p.value.ifSignificant <- function(x, prop.test.result)
{
  threshold <- 0.05
  heightAdjust <- 0.03
  if (prop.test.result$p.value < threshold)
  {
    y <- max(prop.test.result$estimate) + heightAdjust
    text(x, y, sprintf("p = %1.2e", prop.test.result$p.value), cex = 3)
  }
}

# emits a plot comparing the proportions of primiRNAs and protein-coding genes that overlap each class of CNVs
CompareMiRNAsToProteinCodingGenes <- function(pane, showLegend, title, method, legendTitles, primiRNAs, proteinCodingGenes, allCNVs,
gainedCNVs, lostCNVs)
{
  # miRNA overlaps
  primiRNAsGained <- subsetByOverlaps(query = primiRNAs, subject = gainedCNVs)
  primiRNAsLost <- subsetByOverlaps(query = primiRNAs, subject = lostCNVs)
  primiRNAsNone <- subset(primiRNAs, !(primiRNAs@elementMetadata$attributes %in%
                                       subsetByOverlaps(query = primiRNAs, subject = allCNVs)@elementMetadata$attributes))
  # protein-coding gene overlaps
  proteinCodingGenesGained <- subsetByOverlaps(query = proteinCodingGenes, subject = gainedCNVs)
  proteinCodingGenesLost <- subsetByOverlaps(query = proteinCodingGenes, subject = lostCNVs)
  proteinCodingGenesNone <- subset(proteinCodingGenes, !(proteinCodingGenes@elementMetadata$ensembl_gene_id %in%
                                                           subsetByOverlaps(query = proteinCodingGenes, subject = allCNVs)@elementMe-
tadata$ensembl_gene_id))
  # proportions
  proportionGained <- prop.test(c(length(primiRNAsGained), length(proteinCodingGenesGained)), c(length(primiRNAs), length(proteinCod-
ingGenes)))
  proportionLost <- prop.test(c(length(primiRNAsLost), length(proteinCodingGenesLost)), c(length(primiRNAs), length(proteinCoding-
Genes)))
  proportionNone <- prop.test(c(length(primiRNAsNone), length(proteinCodingGenesNone)), c(length(primiRNAs), length(proteinCoding-
Genes)))

  # output statistucs
  cat(capture.output(proportionGained), file = sprintf("Results/2/%s (%s) gained.txt", title, method), sep = "\n")
  cat(capture.output(proportionLost), file = sprintf("Results/2/%s (%s) lost.txt", title, method), sep = "\n")
  cat(capture.output(proportionNone), file = sprintf("Results/2/%s (%s) none.txt", title, method), sep = "\n")

  # output plot
  geneColours <- c("darkgreen", "darkorange")
  barplot(
    matrix(c(proportionGained$estimate, proportionLost$estimate, proportionNone$estimate), ncol = 3),
    beside = T, space = c(0,1), ylim = 0:1,
    names.arg = c("Gain", "Loss", "None"),
    col = geneColours,
    main = pane,
    ylab = "Proportion overlapping", xlab = "CNV type",
    cex.axis = 3, cex.names = 3, cex.main = 4, cex.lab = 3)
  if (showLegend)
  {
    legend("topleft", legend = legendTitles, fill = geneColours, bty = "n", cex = 3)
  }
  p.value.ifSignificant(2, proportionGained)
  p.value.ifSignificant(5, proportionLost)
  p.value.ifSignificant(8, proportionNone)
}

# emits a plot comparing the proportions of genes that are gained or lost in the CNVs
CompareGainsToLosses <- function(title, primiRNAs, proteinCodingGenes, gainedCNVs, lostCNVs)
{
  # miRNA overlaps
  primiRNAsGained <- subsetByOverlaps(query = primiRNAs, subject = gainedCNVs)
  primiRNAsLost <- subsetByOverlaps(query = primiRNAs, subject = lostCNVs)
  # protein-coding gene overlaps
  proteinCodingGenesGained <- subsetByOverlaps(query = proteinCodingGenes, subject = gainedCNVs)
  proteinCodingGenesLost <- subsetByOverlaps(query = proteinCodingGenes, subject = lostCNVs)
  # proportions
  proportionMiRNAs <- prop.test(c(length(primiRNAsGained), length(primiRNAsLost)), c(length(primiRNAs), length(primiRNAs)))
  proportionProteinCodingGenes <- prop.test(c(length(proteinCodingGenesGained), length(proteinCodingGenesLost)), c(length(proteinCod-
ingGenes), length(proteinCodingGenes)))
  # output
  cat(capture.output(proportionMiRNAs), file = sprintf("Results/2/%s miRNAs.txt", title), sep = "\n")
  cat(capture.output(proportionProteinCodingGenes), file = sprintf("Results/2/%s protein-coding genes.txt", title), sep = "\n")
}

# emits a plot comparing the proportions of genes that are gained or lost in the CNVs
CompareCNVs <- function(title, elements, decipherGains, decipherLosses, dgvGains, dgvLosses)
{
  # Decipher
  elementsDecipherGained <- subsetByOverlaps(query = elements, subject = decipherGains)
  elementsDecipherLost <- subsetByOverlaps(query = elements, subject = decipherLosses)
  # DGV
  elementsDgvGained <- subsetByOverlaps(query = elements, subject = dgvGains)
  elementsDgvLost <- subsetByOverlaps(query = elements, subject = dgvLosses)
  # proportions
  proportionGained <- prop.test(c(length(elementsDecipherGained), length(elementsDgvGained)), c(length(elements), length(elements)))
  proportionLost <- prop.test(c(length(elementsDecipherLost), length(elementsDgvLost)), c(length(elements), length(elements)))
  # output
  cat(capture.output(proportionGained), file = sprintf("Results/2/%s gained.txt", title), sep = "\n")
  cat(capture.output(proportionLost), file = sprintf("Results/2/%s lost.txt", title), sep = "\n")
}

# 2A - Is there a significant difference in the proportions of each class of gene overlapping gained, lost, gained and lost,
#      any or no Decipher (pathogenic) or Database of Genomic Variation (healthy) copy number variants?
```

```
jpeg(filename = "Results/2/2AB.jpg", width = 3000, height = 2000)
par(mfrow = c(2,2), mar = c(5, 5, 5, 5))
CompareMiRNAsToProteinCodingGenes("A", T, "Pathogenic CNVs (Decipher)", "width-based", c("pri-miRNAs", "Protein-coding genes"), primiR-
NAs, proteinCodingGenes, decipherCNVs, decipherCNVsGained, decipherCNVsLost)
CompareMiRNAsToProteinCodingGenes("B", F, "Healthy CNVs (DGV)", "width-based", c("pri-miRNAs", "Protein-coding genes"), primiRNAs, pro-
teinCodingGenes, dgvCNVs, dgvCNVsGained, dgvCNVsLost)

# 2B - Does the result hold when only the midpoints of pri-miRNAs and protein-coding genes are considered (in case the
#      disparity in length between pri-miRNAs and protein-coding genes of around three orders of magnitude skews the result)?
CompareMiRNAsToProteinCodingGenes("C", F, "Pathogenic CNVs (Decipher)", "midpoint-based", c("pri-miRNA midpoints", "Protein-coding gene
midpoints"), primiRNAMidpoints, proteinCodingGeneMidpoints, decipherCNVs, decipherCNVsGained, decipherCNVsLost)
CompareMiRNAsToProteinCodingGenes("D", F, "Healthy CNVs (DGV)", "midpoint-based", c("pri-miRNA midpoints", "Protein-coding gene mid-
points"), primiRNAMidpoints, proteinCodingGeneMidpoints, dgvCNVs, dgvCNVsGained, dgvCNVsLost)
dev.off()

# 2C - Is there a significant difference in the number of each of miRNAs and protein-coding genes overlapping gained or lost
#      copy number variants?
CompareGainsToLosses("Pathogenic CNVs (Decipher)", primiRNAMidpoints, proteinCodingGeneMidpoints, decipherCNVsGained, decipherCNVsLost)
CompareGainsToLosses("Healthy CNVs (DGV)", primiRNAMidpoints, proteinCodingGeneMidpoints, dgvCNVsGained, dgvCNVsLost)

# 2D - Is there a significant difference in the proportions of miRNAs/protein-coding genes overlapping gained/lost Decipher or DGV CNVs
CompareCNVs("miRNAs", primiRNAMidpoints, decipherCNVsGained, decipherCNVsLost, dgvCNVsGained, dgvCNVsLost)
CompareCNVs("Protein-coding genes", proteinCodingGeneMidpoints, decipherCNVsGained, decipherCNVsLost, dgvCNVsGained, dgvCNVsLost)
```

Analysis/3 - What difference is there in the targeting density of protein-coding genes that are gained, lost or not in copy number variants.r

```
# Question 3 - What difference is there in the 3'UTR targeting density of protein-coding genes that are gained, lost or not in copy
number variants?

require(GenomicRanges)

setwd("C:/Users/mark/iCloudDrive/MSc/BIOL61230 Research Projects/1/Analysis")

# load protein-coding genes and CNVs
source(file = "Data/Loaders/Protein-coding genes.r")
source(file = "Data/Loaders/Decipher CNVs.r")
source(file = "Data/Loaders/DGV CNVs.r")

# count the targets on each 3'UTR
if (!exists("targetCounts"))
{
  if (!file.exists("Results/3/targetCounts.tab"))
  {
    source(file = "Data/Loaders/Targets.r")
    targetCounts <- as.data.frame(tapply(targets$miRNA, targets$EnsembleTranscriptID, length))
    targetCounts$EnsembleTranscriptID <- rownames(targetCounts)
    rownames(targetCounts) <- NULL
    colnames(targetCounts)[1] <- "TargetCount"
    write.table(targetCounts, file = "Results/3/targetCounts.tab", sep = "\t", row.names = F)
  }
  else
  {
    targetCounts <- read.table(file = "Results/3/targetCounts.tab", header = T, sep = "\t")
  }
}

# calculate the targeting density of each 3'UTR
if (!exists("targetedUTRs"))
{
  if (!file.exists("Results/3/targetedUTRs.tab"))
  {
    # load the UTRs and left join the target counts
    source(file = "Data/Loaders/3'UTRs.r")
    targetedUTRs <- merge(UTRs, targetCounts, by.x = "EnsembleTranscriptID", by.y = "EnsembleTranscriptID", all.x = T)
    targetedUTRs$TargetCount[is.na(targetedUTRs$TargetCount)] <- 0
    # remove UTRs less than 50bp as probably sequencing artefacts
    targetedUTRs$SequenceLength <- nchar(as.character(targetedUTRs$Sequence))
    targetedUTRs <- targetedUTRs[targetedUTRs$SequenceLength >= 50, ]
    # calculate UTR target densities
    targetedUTRs$TargetingDensity <- targetedUTRs$TargetCount / targetedUTRs$SequenceLength
    # cache the results
    write.table(targetedUTRs, file = "Results/3/targetedUTRs.tab", sep = "\t", row.names = F)
  }
  else
  {
    targetedUTRs <- read.table(file = "Results/3/targetedUTRs.tab", header = T, sep = "\t")
  }
}

# average the target densities for each gene
if (!exists("meanGeneTargetDensities"))
{
  if (!file.exists("Results/3/meanGeneTargetDensities.tab"))
  {
    meanGeneTargetDensities <- data.frame(
      MeanTargetingDensity = tapply(targetedUTRs$TargetingDensity, targetedUTRs$EnsembleGeneID, mean))
    meanGeneTargetDensities$EnsembleGeneID <- rownames(meanGeneTargetDensities)
    rownames(meanGeneTargetDensities) <- NULL
    write.table(meanGeneTargetDensities, file = "Results/3/meanGeneTargetDensities.tab", sep = "\t", row.names = F)
  }
  else
  {
    meanGeneTargetDensities <- read.table(file = "Results/3/meanGeneTargetDensities.tab", header = T, sep = "\t")
  }
}

# analyses protein-coding gene midpoint overlaps with CNVs with respect to their targeting densities
AnalyseTargetingDensities <- function(pane, showLegend, prefix, title, proteinCodingGeneMidpoints, meanGeneTargetDensities, CNVsGained,
CNVsLost, CNVs)
{
  # partition the genes by overlap of their midpoints with CNVs
  proteinCodingGenesGained <- subsetByOverlaps(query = proteinCodingGeneMidpoints, subject = CNVsGained)
  proteinCodingGenesLost <- subsetByOverlaps(query = proteinCodingGeneMidpoints, subject = CNVsLost)
  proteinCodingGenesNone <- subset(proteinCodingGenes, !(proteinCodingGenes@elementMetadata$ensembl_gene_id %in%
```

```
                                                    subsetByOverlaps(query = proteinCodingGenes, subject = CNVs)@elementMe-
tadata$ensembl_gene_id))

  # specify the order of the CNV levels to control the intercept choice in the later ANOVA step
  cnvLevels = c("neither gained nor lost", "gained", "lost")

  # add the mean targeting densities to the genes neither gained nor lost (ordered by genomic location)
  neitherGainedNorLost <- merge(
    data.frame(EnsembleGeneID = proteinCodingGenesNone@elementMetadata$ensembl_gene_id[
      order(as.numeric(substr(seqnames(proteinCodingGenesNone), 4, 6)), start(proteinCodingGenesNone))]),
    meanGeneTargetDensities,
    by.x = "EnsembleGeneID", by.y = "EnsembleGeneID", all.x = T)
  neitherGainedNorLost$MeanTargetingDensity[is.na(neitherGainedNorLost$MeanTargetingDensity)] <- 0
  neitherGainedNorLost$CNV <- factor("neither gained nor lost", levels = cnvLevels)
  neitherGainedNorLost$Colour <- "black"

  # add the mean targeting densities to the gained genes (ordered by genomic location)
  gained <- merge(
    data.frame(EnsembleGeneID = proteinCodingGenesGained@elementMetadata$ensembl_gene_id[
      order(as.numeric(substr(seqnames(proteinCodingGenesGained), 4, 6)), start(proteinCodingGenesGained))]),
    meanGeneTargetDensities,
    by.x = "EnsembleGeneID", by.y = "EnsembleGeneID", all.x = T)
  gained$MeanTargetingDensity[is.na(gained$MeanTargetingDensity)] <- 0
  gained$CNV <- factor("gained", levels = cnvLevels)
  gained$Colour <- "blue"

  # add the mean targeting densities to the lost genes (ordered by genomic location)
  lost <- merge(
    data.frame(EnsembleGeneID = proteinCodingGenesLost@elementMetadata$ensembl_gene_id[
      order(as.numeric(substr(seqnames(proteinCodingGenesLost), 4, 6)), start(proteinCodingGenesLost))]),
    meanGeneTargetDensities,
    by.x = "EnsembleGeneID", by.y = "EnsembleGeneID", all.x = T)
  lost$MeanTargetingDensity[is.na(lost$MeanTargetingDensity)] <- 0
  lost$CNV <- factor("lost", levels = cnvLevels)
  lost$Colour <- "red"

  # combined densities
  densities <- rbind(neitherGainedNorLost, gained, lost)
  write.table(densities, file = sprintf("Results/3/%s densities.tab", prefix), sep = "\t", row.names = F)

  # values and means
  plot(
    1:nrow(densities), densities$MeanTargetingDensity,
    main = pane,
    xlab = "CNV type/genomic location", ylab = "Mean 3'UTR targeting density",
    bg = densities$Colour, pch = 21,
    cex.axis = 2, cex.main = 2.5, cex.lab = 2)
  abline(h = mean(densities$MeanTargetingDensity[densities$CNV == "neither gained nor lost"]), col = "black", lwd = 3)
  abline(h = mean(densities$MeanTargetingDensity[densities$CNV == "gained"]), col = "blue", lwd = 3)
  abline(h = mean(densities$MeanTargetingDensity[densities$CNV == "lost"]), col = "red", lwd = 3)
  if (showLegend)
  {
    legend("topright", legend = c("neither gained nor lost", "gained", "lost"), fill = c("black", "blue", "red"), bty = "n", cex = 2)
  }

  # anova
  model <- aov(densities$MeanTargetingDensity ~ densities$CNV)
  filename <- sprintf("Results/3/%s.txt", prefix)
  cat("summary.lm\n==========\n", file = filename, sep = "\n")
  cat(capture.output(summary.lm(model)), file = filename, sep = "\n", append = T)
  cat("\nTukeyHSD\n========\n", file = filename, sep = "\n", append = T)
  cat(capture.output(TukeyHSD(model)), file = filename, sep = "\n", append = T)
  cat("\nmean\n====\n", file = filename, sep = "\n", append = T)
  cat(capture.output(tapply(densities$MeanTargetingDensity, densities$CNV, mean)), file = filename, sep = "\n", append = T)
  cat("\nsd\n==\n", file = filename, sep = "\n", append = T)
  cat(capture.output(tapply(densities$MeanTargetingDensity, densities$CNV, sd)), file = filename, sep = "\n", append = T)
  cat("\nlength\n======\n", file = filename, sep = "\n", append = T)
  cat(capture.output(tapply(densities$MeanTargetingDensity, densities$CNV, length)), file = filename, sep = "\n", append = T)
}

jpeg(filename = "Results/3/plots.jpg", width = 2000, height = 1000)
par(mfcol = c(1,2), mar = c(5, 5, 5, 5))
AnalyseTargetingDensities("A", T, "Decipher", "Pathogenic CNVs (Decipher)",
                          proteinCodingGeneMidpoints, meanGeneTargetDensities,
                          decipherCNVsGained, decipherCNVsLost, decipherCNVs)

AnalyseTargetingDensities("B", F, "DGV", "Healthy CNVs (DGV)",
                          proteinCodingGeneMidpoints, meanGeneTargetDensities,
                          dgvCNVsGained, dgvCNVsLost, dgvCNVs)
dev.off()
```

Dada/Loaders/3'UTRs.r

```
setwd("C:/Users/mark/iCloudDrive/MSc/BIOL61230 Research Projects/1/Analysis")

if (!exists("UTRs"))
{
  UTRs <- read.table(
    file = "Data/Sources/3'UTRs/3'UTRs.tab", header = T, sep = "\t")
}
```

Data/Loaders/Chromosomes.r

```
setwd("C:/Users/mark/iCloudDrive/MSc/BIOL61230 Research Projects/1/Analysis")

if (!exists("chromosomes"))
{
  chromosomes <- read.table(
    file = "Data/Sources/Chromosomes/hg38.chrom.sizes", sep = "\t",
    col.names = c("Name", "Length"))
  chromosomes <- chromosomes[chromosomes$Name %in% sprintf("chr%d", 1:22), ]
}
```

Data/Loaders/Decipher CNVs.r

```
require(GenomicRanges)
setwd("C:/Users/mark/iCloudDrive/MSc/BIOL61230 Research Projects/1/Analysis")

if (!exists("decipherCNVs"))
{
  decipherCNVs <- read.table(file = "Data/Sources/Decipher CNVs/Decipher.tab", header = T, sep = "\t")
  decipherCNVs <- decipherCNVs[decipherCNVs$seqnames != "chrX", ]
  decipherCNVs <- makeGRangesFromDataFrame(df = decipherCNVs, keep.extra.columns = T)
  decipherCNVsGained <- decipherCNVs[decipherCNVs$type >= 0, ]
  decipherCNVsLost <- decipherCNVs[decipherCNVs$type <= 0, ]
  decipherCNVsBoth <- decipherCNVs[decipherCNVs$type == 0, ]
}
```

Data/Loaders/DGV CNVs.r

```
require(GenomicRanges)
setwd("C:/Users/mark/iCloudDrive/MSc/BIOL61230 Research Projects/1/Analysis")

if (!exists("dgvCNVs"))
{
  dgvCNVs <- read.table(file = "Data/Sources/DGV CNVs/GRCh38_hg38_variants_2015-07-23.txt", header = T, sep = "\t")
  dgvCNVs$chr <- as.numeric(as.character(dgvCNVs$chr))
  dgvCNVs <- dgvCNVs[!is.na(dgvCNVs$chr), ]
  dgvCNVs <- transform(dgvCNVs, chr = sprintf("chr%d", chr))
  dgvCNVs <- makeGRangesFromDataFrame(df = dgvCNVs[dgvCNVs$varianttype == "CNV", ], keep.extra.columns = T)
  dgvCNVsGained <- dgvCNVs[dgvCNVs$variantsubtype %in% c("duplication", "gain", "gain+loss"), ]
  dgvCNVsLost <- dgvCNVs[dgvCNVs$variantsubtype %in% c("deletion", "loss", "gain+loss"), ]
  dgvCNVsBoth <- dgvCNVs[dgvCNVs$variantsubtype %in% c("gain+loss"), ]
}
```

Data/Loaders/pri-miRNAs.r

```
require(GenomicRanges)
setwd("C:/Users/mark/iCloudDrive/MSc/BIOL61230 Research Projects/1/Analysis")

if (!exists("primiRNAs"))
{
  # create GRanges
  primiRNAs = read.table(file = "Data/Sources/miRNAs/hsa.gff3", header = FALSE, sep = "\t")
  primiRNAs <- rename(primiRNAs, c("V1" = "sequence", "V2" = "source", "V3" = "feature", "V4" = "start", "V5" = "end",
                                   "V6" = "score", "V7" = "strand", "V8" = "frame", "V9" = "attributes"))
  primiRNAs <- makeGRangesFromDataFrame(df = primiRNAs[primiRNAs$feature == "miRNA_primary_transcript" & primiRNAs$sequence %in%
sprintf("chr%d", 1:22), ],
                                        keep.extra.columns = TRUE, seqnames.field = "sequence")
  # create midpoints variant
  primiRNAMidpoints <- primiRNAs
  attributes(primiRNAMidpoints@ranges)$start <- as.integer(attributes(primiRNAMidpoints@ranges)$start + (attributes(primiRNA-
Midpoints@ranges)$width / 2))
  attributes(primiRNAMidpoints@ranges)$width <- as.integer(rep.int(1, length(primiRNAMidpoints)))
}
```

Data/Loaders/Protein-coding genes.r

```
require(GenomicRanges)
setwd("C:/Users/mark/iCloudDrive/MSc/BIOL61230 Research Projects/1/Analysis")

if (!exists("proteinCodingGenes"))
{
  # create GRanges
  proteinCodingGenes <- makeGRangesFromDataFrame(df = read.table(file = "Data/Sources/Protein-coding genes/Protein-coding genes.tab",
                                                                 header = T, sep = "\t"), keep.extra.columns = T)
  # create midpoints variant
  proteinCodingGeneMidpoints <- proteinCodingGenes
  attributes(proteinCodingGeneMidpoints@ranges)$start <- as.integer(attributes(proteinCodingGeneMidpoints@ranges)$start + (attrib-
utes(proteinCodingGeneMidpoints@ranges)$width / 2))
  attributes(proteinCodingGeneMidpoints@ranges)$width <- as.integer(rep.int(1, length(proteinCodingGeneMidpoints)))
}
```

Data/Loaders/Targets.r

```
require(GenomicRanges)
setwd("C:/Users/mark/iCloudDrive/MSc/BIOL61230 Research Projects/1/Analysis")

if (!exists("proteinCodingGenes"))
{
  # create GRanges
  proteinCodingGenes <- makeGRangesFromDataFrame(df = read.table(file = "Data/Sources/Protein-coding genes/Protein-coding genes.tab",
                                                                 header = T, sep = "\t"), keep.extra.columns = T)
  # create midpoints variant
  proteinCodingGeneMidpoints <- proteinCodingGenes
  attributes(proteinCodingGeneMidpoints@ranges)$start <- as.integer(attributes(proteinCodingGeneMidpoints@ranges)$start + (attrib-
utes(proteinCodingGeneMidpoints@ranges)$width / 2))
  attributes(proteinCodingGeneMidpoints@ranges)$width <- as.integer(rep.int(1, length(proteinCodingGeneMidpoints)))
}
```

Data/Sources/3'UTRs/3'UTRs.r

```
#source("http://bioconductor.org/biocLite.R")
#biocLite("biomaRt")

library(biomaRt)

setwd("C:/Users/mark/iCloudDrive/MSc/BIOL61230 Research Projects/1/Analysis")

# GRCh38 human data source
ensembl = useEnsembl(biomart="ensembl", dataset="hsapiens_gene_ensembl")

# get all protein-coding transcripts for protein-coding genes on chromosomes 1..22
proteinCodingTranscripts <- getBM(mart = ensembl,
  attributes = c("ensembl_gene_id", "ensembl_transcript_id"),
  filters = c("biotype", "transcript_biotype", "chromosome_name"),
  values = list("protein_coding", "protein_coding", 1:22))
colnames(proteinCodingTranscripts) <- c("EnsembleGeneID", "EnsembleTranscriptID")

# 3'UTRs
utr3 <- getSequence(mart = ensembl, type = "ensembl_transcript_id", id = proteinCodingTranscripts$EnsembleTranscriptID, seqType =
"3utr")
colnames(utr3) <- c("Sequence", "EnsembleTranscriptID")

# add the gene IDs
geneUTRs <- merge(proteinCodingTranscripts, utr3, by.x = "EnsembleTranscriptID", by.y = "EnsembleTranscriptID", all.x = T)

# save only where sequence was available
write.table(geneUTRs[geneUTRs$Sequence != "Sequence unavailable", ], file = "Data/Sources/3'UTRs/3'UTRs.tab", sep = "\t", row.names =
F)
```

Data/Sources/3'UTRs/3'UTR tab to fasta.r

```
library(Biostrings)

setwd("C:/Users/mark/iCloudDrive/MSc/BIOL61230 Research Projects/1/Analysis")

# read the 3'UTR tab file
utr3 <- read.table(file = "Data/Sources/3'UTRs/3'UTRs.tab", header = T)

# build a DNA string set and set the names to the transcript IDs
dna <- DNAStringSet(utr3$X3utr)
names(dna) <- utr3$ensembl_transcript_id

# write to a file
writeXStringSet(dna, filepath = "Data/Sources/3'UTRs/3'UTRs.fa")
```

Data/Sources/Decipher CNVs/Decipher Liftover.r

```
# source("http://bioconductor.org/workflows.R")
# workflowInstall("liftOver")

require(rtracklayer)
require(GenomicRanges)
require(liftOver)

setwd("C:/Users/mark/iCloudDrive/MSc/BIOL61230 Research Projects/1/Analysis")

# load the CRCh37 to GRCh38 chain file (from http://hgdownload.soe.ucsc.edu/goldenPath/hg19/liftOver/)
hg19ToHg38 <- import.chain("Data/Sources/Decipher CNVs/hg19ToHg38.over.chain")

# load the Decipher CNVs
decipherCNVs <- read.table(file = "Data/Sources/Decipher CNVs/population_cnv.txt", header = T, sep = "\t")

# convert to GRanges (ignore chr23 as only Decipher has it)
decipherCNVs <- transform(decipherCNVs, chr = sprintf("chr%d", chr))
decipherCNVs <- makeGRangesFromDataFrame(df = decipherCNVs[decipherCNVs$chr != "chr23", ], keep.extra.columns = T)

# convert from CRCh37 to GRCh38
decipherCNVsGRC38 <- liftOver(decipherCNVs, chain = hg19ToHg38)

# save to file
write.table(x = decipherCNVsGRC38, file = "Data/Sources/Decipher CNVs/Decipher.tab", row.names = F, sep = "\t")
```

Data/Sources/miRNAs/Strip non-human fasta.r

```
library(seqinr)

setwd("C:/Users/mark/iCloudDrive/MSc/BIOL61230 Research Projects/1/Analysis")

# read the mature miRNA seqeunces (all species)
allMiRNAs <- read.fasta(file = "Data/Sources/miRNAs/mature.fa", as.string = T, forceDNAtolower = F)

# extract the human miRNAs
humanMiRNAs <- allMiRNAs[grep("hsa-*", attributes(allMiRNAs)$name)]

# write the human mature miRNAs to a file
write.fasta(sequences = humanMiRNAs, names = names(humanMiRNAs), file.out = "Data/Sources/miRNAs/mature.human.fa")
```

Data/Sources/Protein-coding genes/Protein-coding genes.r

```
#source("http://bioconductor.org/biocLite.R")
#biocLite("biomaRt")

library(biomaRt)

setwd("C:/Users/mark/iCloudDrive/MSc/BIOL61230 Research Projects/1/Analysis")

# GRCh38 human data source
ensembl = useEnsembl(biomart="ensembl", dataset="hsapiens_gene_ensembl")

# get all protein-coding genes on chromosomes 1..22 with protein-coding transcripts
proteinCodingGenes <- getBM(mart = ensembl,
  attributes = c("ensembl_gene_id", "chromosome_name", "start_position", "end_position", "strand", "transcript_count"),
  filters = c("biotype", "transcript_biotype", "chromosome_name"),
  values = list("protein_coding", "protein_coding", 1:22))

# rename columns for GRanges compatibility
colnames(proteinCodingGenes) <- c("ensembl_gene_id", "chr", "start", "end", "strand", "transcript_count")

# rename chromosomes and strands
proteinCodingGenes <- transform(proteinCodingGenes, chr = sprintf("chr%s", chr))
proteinCodingGenes$strand[proteinCodingGenes$strand == "1"] <- "+"
proteinCodingGenes$strand[proteinCodingGenes$strand == "-1"] <- "-"

# save file
write.table(proteinCodingGenes, file = "Data/Sources/Protein-coding genes/Protein-coding genes.tab", sep = "\t", row.names = F)
```