acorn Vignette

Tychele N. Turner, Ph.D., Washington University School of Medicine

2023-04-05

Contents

Load the acorn Package

```
library('acorn')
```

Load in the Test DNV Data

readDNV = Reads in a de novo variant (DNV) file in the format of sample, chromosome, genomic position, reference allele, alternate allele, and then any optional columns. File must be tab-delimited and the file must have the data in the order listed above (i.e., sample is field 1, chromosome is field 2, genomic position is field 3, reference allele is field 4, and alternate allele is field 5. The file can either be a uncompressed file or can be a gz compressed file. Please note that the chromosome data should take the form with a "chr" at the beginning (e.g., chr1).

Returns back a loaded in version of the DNV file that can be assigned to an object.

```
input <- readDNV(paste(path.package("acorn"),"/extdata/dnms_from_Ng_et_al_2022_Human_Mutation_paper.txt
head(input)
```

```
##    SAMPLE CHROM POS_B38 REFERENCE                    ALTERNATE
## 1 HG01928  chr1  913941         G                            T
## 2 HG03915  chr1  917676         C                            A
## 3 HG03915  chr1  918783         G                            C
## 4 HG00450  chr1 1216505         A                            G
## 5 HG02257  chr1 1217502         G                            A
## 6 HG00465  chr1 1366883         G GGTGTGAATTGGTGTAGTGTGAATGAGT
##                                          ID
## 1                            chr1_913941_G_T
## 2                            chr1_917676_C_A
## 3                            chr1_918783_G_C
## 4                           chr1_1216505_A_G
## 5                           chr1_1217502_G_A
## 6 chr1_1366883_G_GGTGTGAATTGGTGTAGTGTGAATGAGT
```

```
str(input)
```

```
## 'data.frame':    9741 obs. of  6 variables:
##  $ SAMPLE   : chr  "HG01928" "HG03915" "HG03915" "HG00450" ...
##  $ CHROM    : chr  "chr1" "chr1" "chr1" "chr1" ...
##  $ POS_B38  : int  913941 917676 918783 1216505 1217502 1366883 1765426 2332062 2645102 3355666 ...
##  $ REFERENCE: chr  "G" "C" "G" "A" ...
##  $ ALTERNATE: chr  "T" "A" "C" "G" ...
##  $ ID       : chr  "chr1_913941_G_T" "chr1_917676_C_A" "chr1_918783_G_C" "chr1_1216505_A_G" ...
```

Extract an Individual

extractIndividual = Extracts the DNVs out of a dnvObject from a particular individual. Returns a DNV object containing only DNVs in the specified individual.

```
ind <- extractIndividual(input, "HG01928")
head(ind)
```

```
##      SAMPLE CHROM   POS_B38 REFERENCE ALTERNATE                ID
## 1   HG01928  chr1    913941         G         T    chr1_913941_G_T
## 12  HG01928  chr1   3393842         G         A   chr1_3393842_G_A
## 166 HG01928  chr1  44230922         C         T  chr1_44230922_C_T
## 304 HG01928  chr1  94001171         C         T  chr1_94001171_C_T
## 405 HG01928  chr1 151473815         T         C chr1_151473815_T_C
## 422 HG01928  chr1 156638884         G         A chr1_156638884_G_A
```

```
nrow(ind)
```

```
## [1] 85
```

```
table(ind[,1])
```

```
##
## HG01928
##      85
```

Extract Multiple Individuals

```
ind <- extractIndividual(input, c("HG01928", "HG03915"))
head(ind)
```

```
##      SAMPLE CHROM  POS_B38 REFERENCE ALTERNATE                ID
## 1   HG01928  chr1   913941         G         T   chr1_913941_G_T
```

```
## 2   HG03915  chr1   917676         C         A   chr1_917676_C_A
## 3   HG03915  chr1   918783         G         C   chr1_918783_G_C
## 12  HG01928  chr1  3393842         G         A  chr1_3393842_G_A
## 74  HG03915  chr1 18766956         T         A chr1_18766956_T_A
## 166 HG01928  chr1 44230922         C         T chr1_44230922_C_T
```

```
nrow(ind)
```

```
## [1] 158
```

```
table(ind[,1])
```

```
##
## HG01928 HG03915
##      85      73
```

extract SNVs

extractSNVs = Extracts single-nucleotide variants (SNVs) out from a DNV object generated using the readDNV function. Returns a DNV object containing only SNVs.

```
snvs <- extractSNVs(input)
nrow(snvs)
```

```
## [1] 8558
```

Extract INDELs Only

extractINDELs = Extracts small insertions/deletions (INDELs) out from a DNV object generated using the readDNV function. Returns a DNV object containing only INDELs.

```
indels <- extractINDELs(input)
nrow(indels)
```

```
## [1] 1183
```

Extract MNVs Only (There are None in the Test Set)

extractMNVs = Extracts multi-nucleotide variants (MNVs) out from a DNV object generated using the readDNV function. Returns a DNV object containing only MNVs.

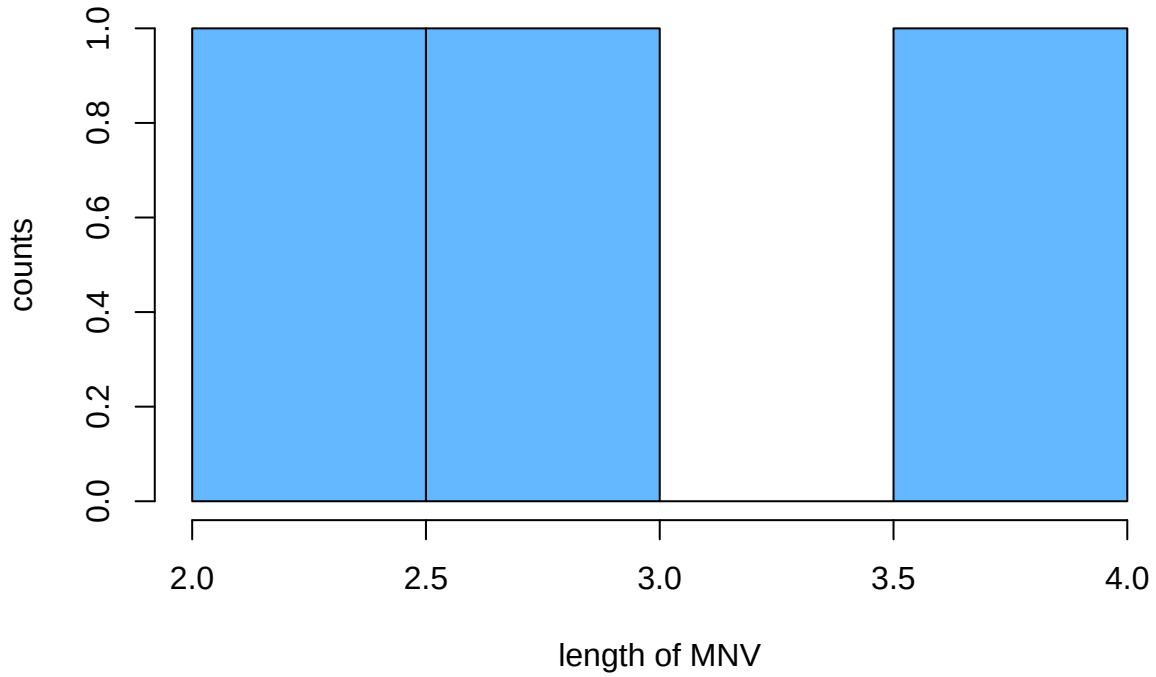
```
denovoMNVs <- readDNV(paste(path.package("acorn"),"/extdata/mnv_test.txt",sep=""))

mnvs <- extractMNVs(denovoMNVs)
nrow(mnvs)
```

```
## [1] 3
```

Calculate the MNV Lengths

calculateMNVLengths = This function will automatically grab only the MNVs from the DNV object for the calculation of the MNV lengths ratio. Returns the length of the MNVs, in the form of an object, observed in the DNV object. It also returns a barplot of the MNV lengths.

```
calculateMNVLengths(mnvs)
```



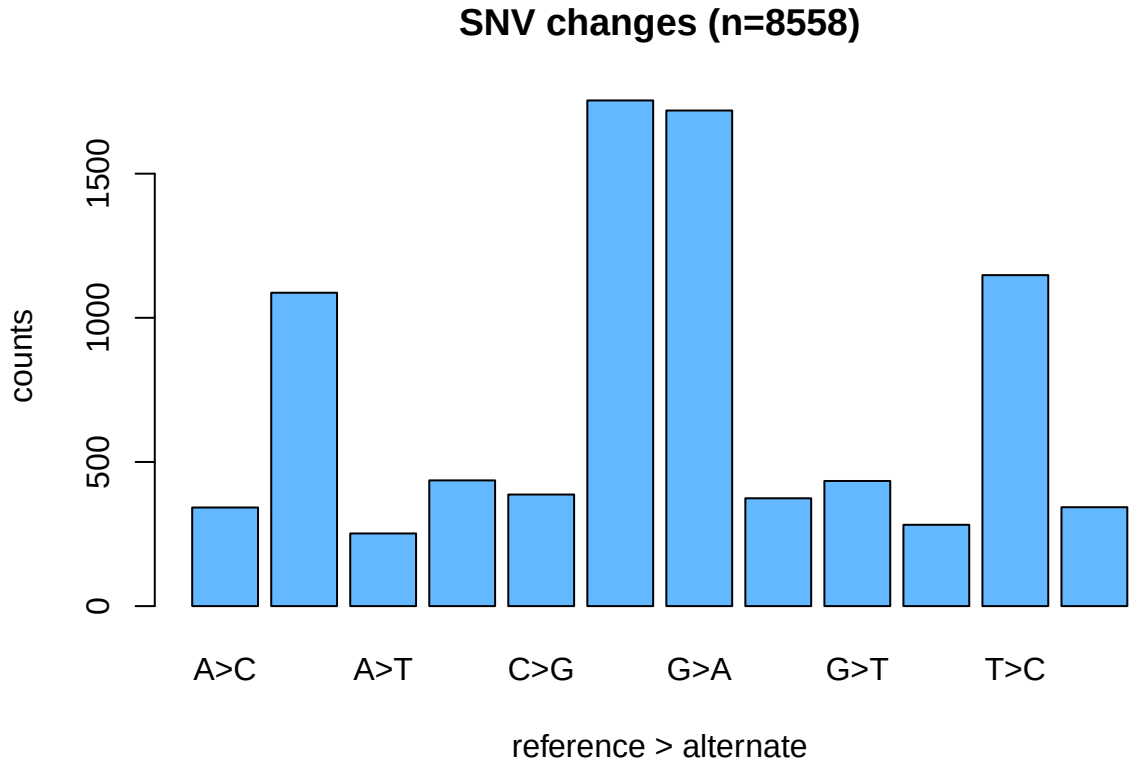

```
## [1] 2 3 4
```

Calculate the Transition/Transversion Ratio

calculateTiTvRatio = This function will automatically grab only the SNVs from the DNV object for the calculation of the transition/transversion (Ti/Tv) ratio. Returns the counts of transitions, the counts of transversions, the Ti/Tv ratio, and a barplot of the different types of SNV changes observed in the DNV object.

```
calculateTiTvRatio(input)
```

```
## number of transitions (A>G, C>T, G>A, T>C): 5708

## number of transversions (A>C, A>T, C>A, C>G, G>C, G>T, T>A, T>G): 2850

## Ti/Tv ratio: 2.00280701754386

## Plot of different nucleotide changes:
```



Calculate the Deletion/Insertion Ratio

calculateDeletionInsertionRatio = This function will automatically grab only the INDELs from the DNV object for the calculation of the deletion/insertion ratio. Returns the counts of deletions, the counts of insertions, and the deletion/insertion ratio.

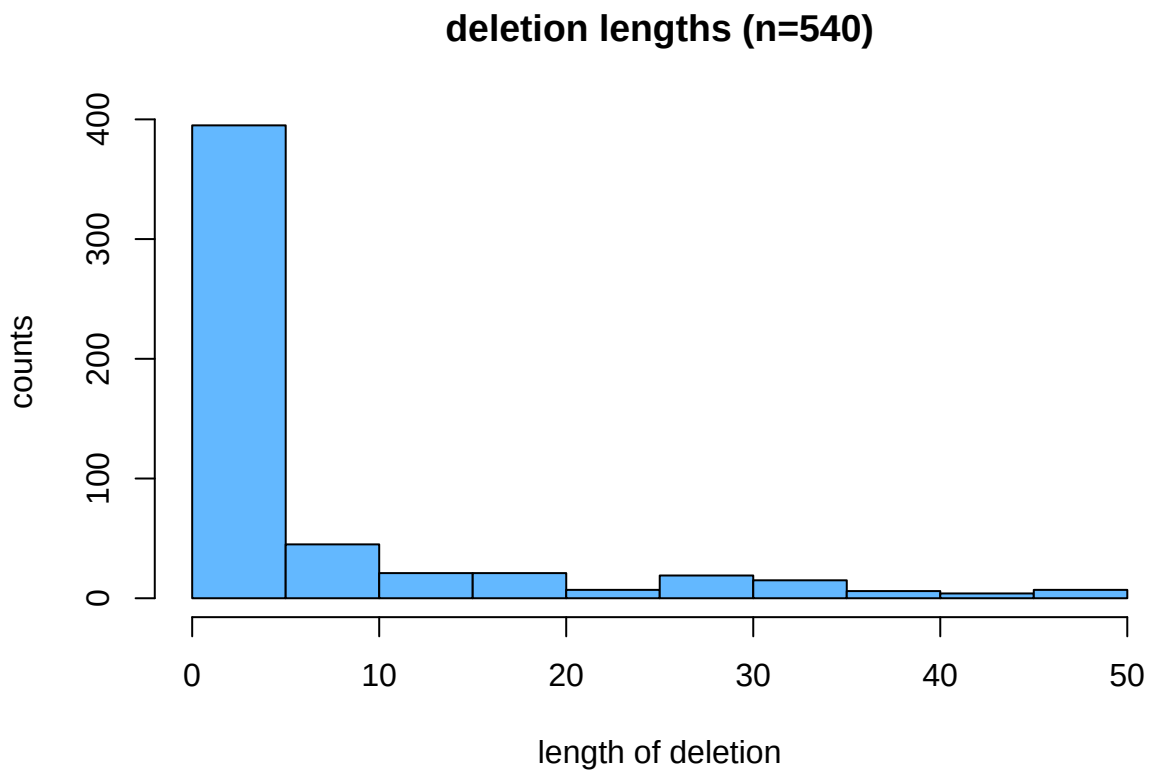
```
calculateDeletionInsertionRatio(input)
```

```
## number of deletions: 540
## number of insertions: 643
## deletion/insertion ratio: 0.839813374805599
```

Calculate Deletion Lengths

calculateDeletionLengths = This function will automatically grab only the deletions from the DNV object for the calculation of the length of the deletions. Returns the length of the deletions, in the form of an object, observed in the DNV object. It also returns a barplot of the deletion lengths.

```
dellengths <- calculateDeletionLengths(input)
```



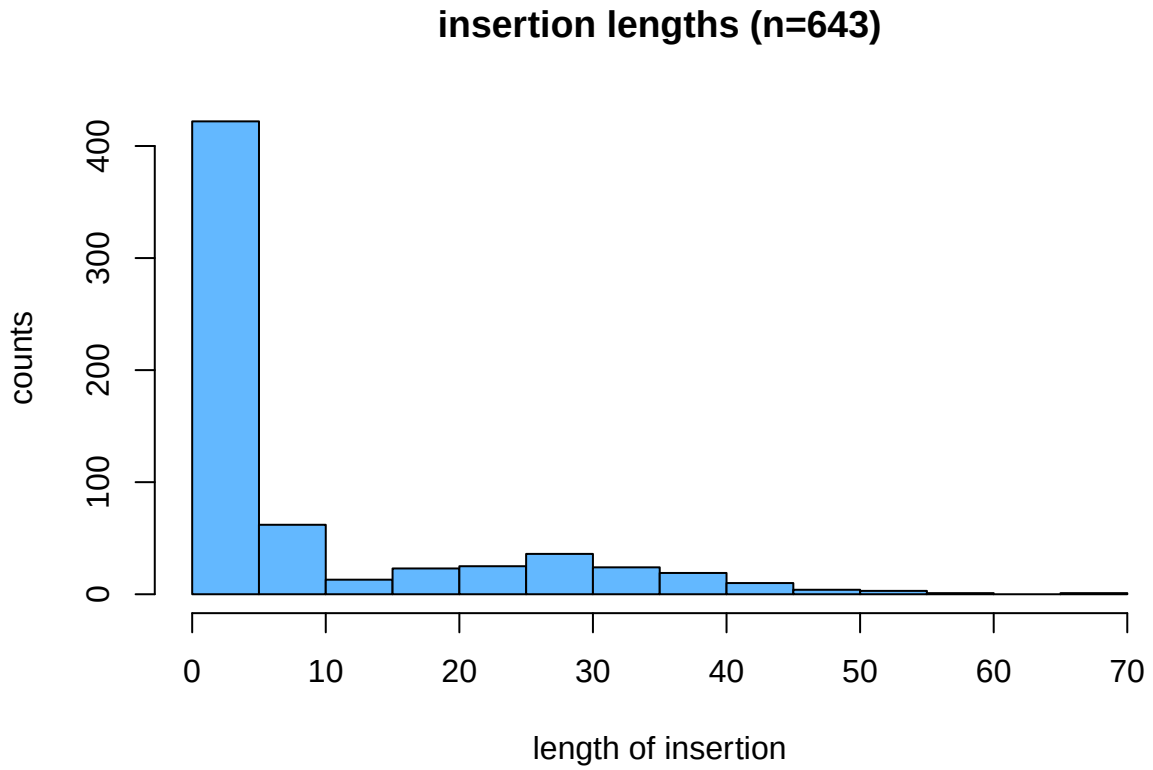

```
head(dellengths)
```

```
## [1]  2  2 27  2  2 15
```

Calculate Insertion Lengths

calculateInsertionLengths = This function will automatically grab only the insertions from the DNV object for the calculation of the length of the insertions. Returns the length of the insertions, in the form of an object, observed in the DNV object. It also returns a barplot of the insertion lengths.

```
inslengths <- calculateInsertionLengths(input)
```




```
head(inslengths)
```

```
## [1] 28  3 37 37 41  5
```

Keep Only the Autosomes

extractAutosomes = Extracts the autosomes (chromosomes 1 to 22) out from a DNV object originally generated using the readDNV function. You can also run this on objects generated from extractSNVs, extractINDELs, or extractMNVs. Returns a DNV object containing only DNVs on the autosomes.

```
aut <- extractAutosomes(input)
nrow(aut)
```

```
## [1] 9262
```

```
table(aut[,2])
```

```
##
##  chr1 chr10 chr11 chr12 chr13 chr14 chr15 chr16 chr17 chr18 chr19  chr2 chr20
##   707   408   477   408   338   479   251   273   233   232   206   833   257
## chr21 chr22  chr3  chr4  chr5  chr6  chr7  chr8  chr9
##   143   154   696   642   614   551   482   501   377
```

Keep Only the X Chromosome

extractX = Extracts the X chromosome out from a DNV object originally generated using the readDNV function. You can also run this on objects generated from extractSNVs, extractINDELs, or extractMNVs. Returns a DNV object containing only DNVs on the X chromosome.

```
X <- extractX(input)
nrow(X)
```

```
## [1] 479
```

```
table(X[,2])
```

```
##
## chrX
##  479
```

Keep Only the Y Chromosome (There are Mone on the Y in the Test Dataset)

extractY = Extracts the Y chromosome DNVs out from a DNV object originally generated using the readDNV function. You can also run this on objects generated from extractSNVs, extractINDELs, or extractMNVs. Returns a DNV object containing only DNVs on the Y chromosome.

```
Y <- extractY(input)
nrow(Y)
```

```
## [1] 0
```

Calculate Counts per Individual

countsPerIndividual = This function will count the DNVs from a DNV object originally generated using the readDNV function. You can also run this on objects generated from extractSNVs, extractINDELs, or extractMNVs. Returns the mean of the DNV counts per individual, the standard deviation of the DNV counts per individual, a plot of the density of the DNV counts per individual, and an object consisting of the sample name and the counts of their DNVs that can be assigned to another object.

```
counts <- countsPerIndividual(input)
```

```
## mean of the counts per individual:
```

```
## 79.1951219512195
```

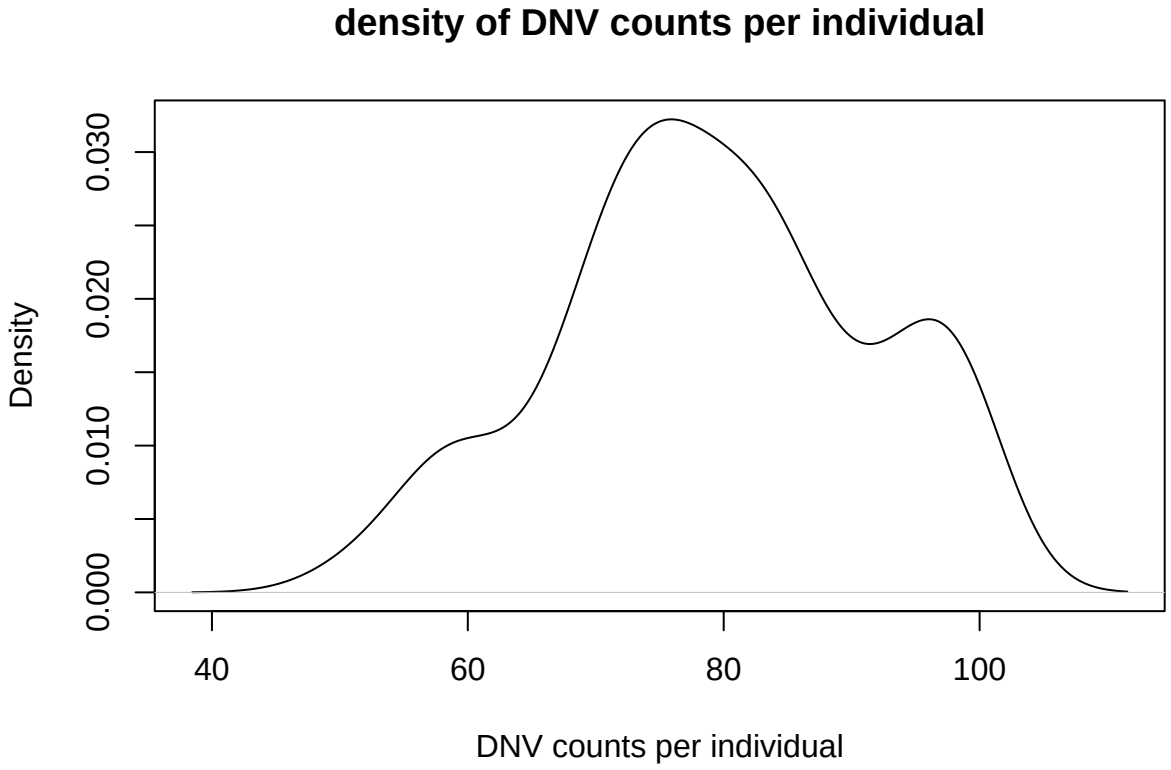
```
##
```

```
## standard deviation of the counts per individual:
```

```
## 12.2043116115263
```

```
##
```

```
## Plot generating of the density of the DNV counts.
```




```
head(counts)
```

```
##    sample dnv_count
## 1 HG00405        70
## 2 HG00423        78
## 3 HG00429        57
## 4 HG00438        66
## 5 HG00444        74
## 6 HG00447        75
```

Load in Example Data for Parental Age Analyses

```
input <- readDNV(paste(path.package("acorn"),"/extdata/dnms_from_Ng_et_al_2022_Human_Mutation_paper.txt


countExample <- read.delim(paste(path.package("acorn"),"/extdata/dnm_count_example.txt",sep=""))
parentExample <- read.delim(paste(path.package("acorn"),"/extdata/parental_age_example.txt",sep=""))
```

Make Parental Age Object

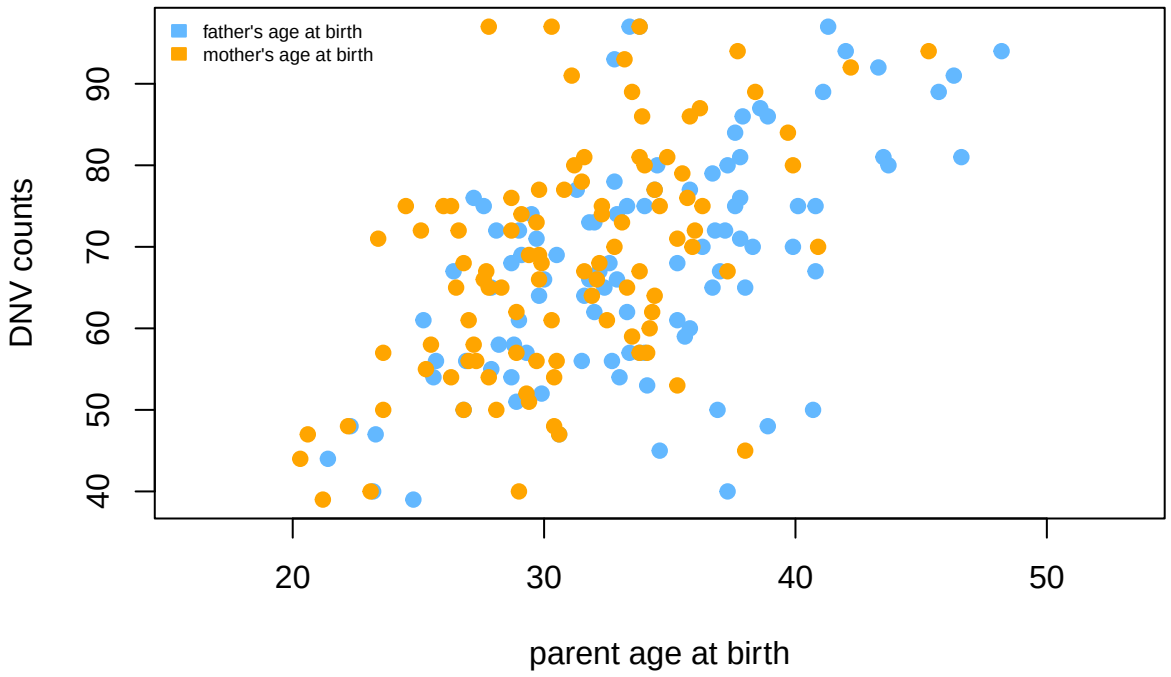
parentalAgeObject = Takes in a counts object that is either the result of countsPerIndividual() or is already read into an object from a file that contains the following two fields: sample and number of DNVs. The parental age object should be read in and contain the following fields: sample, father age at child's birth, and mother age at child's birth. Returns back an object with the de novo counts and parental age data together. The fields in this file are sample, dnm_counts, fatherAge, and motherAge.

```
parents <- parentalAgeObject(countExample, parentExample)
```

Run Parental Age Analyses Including Both Mother and Father

parentalAge = This function will calculate the correlation between father's and mother's age at birth and DNV counts per individual, the results of the linear model taking the form: lm(formula = dnm_counts ~ fatherAge+motherAge, data = parentalAgeObject) or the exponential model taking the form of lm(log(dnm_counts)~fatherAge+motherAge, data=parentalAgeObject). Input required is output from the parentalAgeObject function in this package. Returns the results of the linear model taking the form: lm(formula = dnm_counts ~ fatherAge + motherAge, data = parentalAgeObject) or the exponential model taking the form lm(log(dnm_counts)~fatherAge+motherAge, data=parentalAgeObject). It also returns a plot of father's and mother's age at birth and DNV counts. The default is the "linear" model.

```
parentalAge(parents)
```

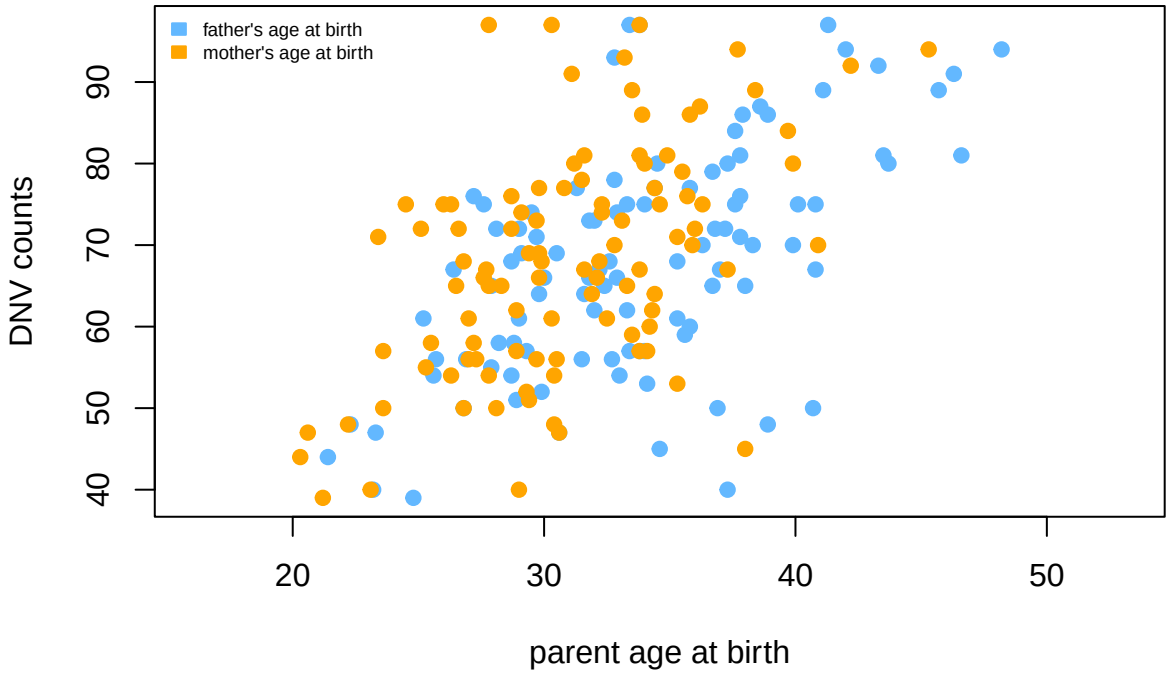



```
## $`summary of specified model type for father's and mother's age at birth and DNV counts (default mod
##
## Call:
## lm(formula = dnm_counts ~ fatherAge + motherAge, data = parentalAgeObject)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -30.190  -6.831  -0.477   6.975  31.700
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  10.8374     7.7691   1.395 0.166220
## fatherAge     1.0331     0.2669   3.871 0.000196 ***
## motherAge     0.7179     0.3174   2.262 0.025955 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.14 on 97 degrees of freedom
```

```
## Multiple R-squared:  0.3732, Adjusted R-squared:  0.3603
## F-statistic: 28.87 on 2 and 97 DF,  p-value: 1.45e-10
##
##
## $`confidence interval for specified model type for father's and mother's age at birth and DNV counts
##                   2.5 %    97.5 %
## (Intercept) -4.58207432 26.256945
## fatherAge    0.50343356  1.562695
## motherAge    0.08788934  1.347946
```

If you prefer to run the exponential model use:

```
parentalAge(parents, modelType="exponential")
```



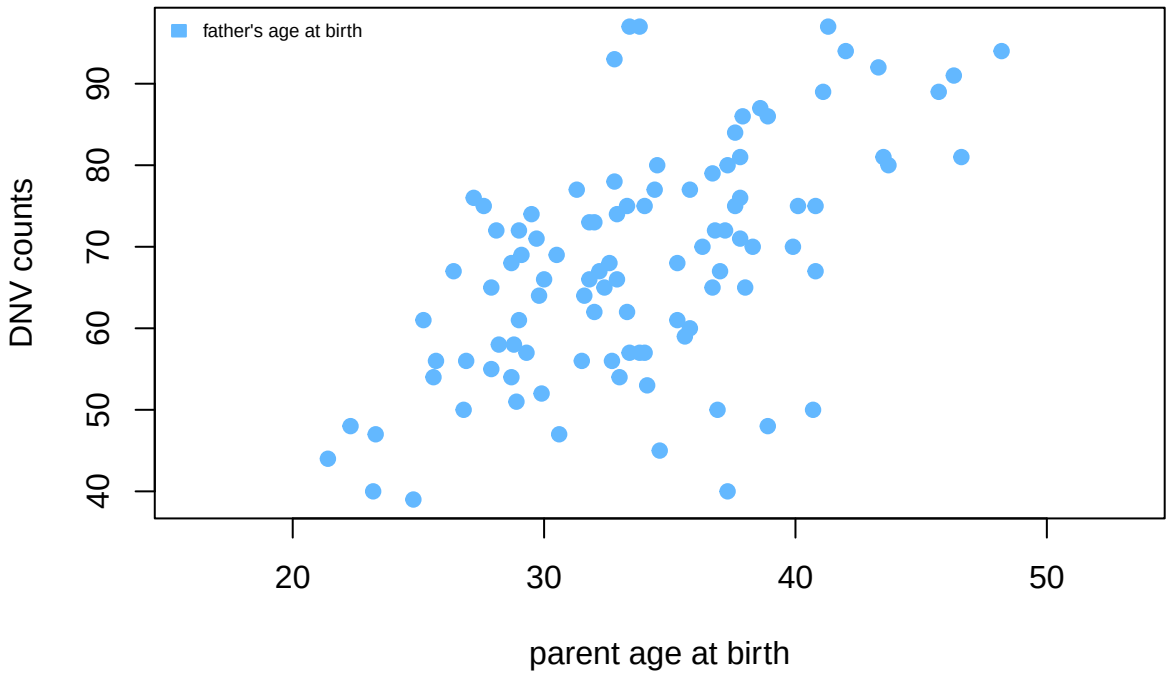

```
## $`summary of specified model type for father's and mother's age at birth and DNV counts (default mode
##
## Call:
## lm(formula = log(dnm_counts) ~ fatherAge + motherAge, data = parentalAgeObject)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.53649 -0.10197  0.00424  0.12486  0.42055
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.335858   0.119259  27.972  < 2e-16 ***
## fatherAge   0.014598   0.004096   3.564 0.000569 ***
## motherAge   0.011897   0.004873   2.442 0.016437 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.171 on 97 degrees of freedom
## Multiple R-squared:  0.3618, Adjusted R-squared:  0.3487
## F-statistic:  27.5 on 2 and 97 DF,  p-value: 3.457e-10
```

```
##
##
## $`confidence interval for specified model type for father's and mother's age at birth and DNV counts
##                   2.5 %     97.5 %
## (Intercept) 3.099162095 3.57255401
## fatherAge   0.006467636 0.02272774
## motherAge   0.002225955 0.02156835
```

Run Parental Age Analyses for Father Age Only

fatherAge = This function will calculate the correlation between father's age at birth and DNV counts per individual, the results of the linear model taking the form: lm(formula = dnm_counts ~ fatherAge, data = parentalAgeObject) or the exponential model taking the form lm(log(dnm_counts)~fatherAge, data=parentalAgeObject). Input required is output from the parentalAgeObject function in this package. Returns the correlation between father's age at birth and DNV counts per individual and the results of the linear model taking the form: lm(formula = dnm_counts ~ fatherAge, data = parentalAgeObject) or the exponetial model taking the form lm(log(dnm_counts)~fatherAge, data=parentalAgeObject). It also returns a plot of father's age at birth and DNV counts.

```
fatherAge(parents)
```




```
## $`summary of specified model type for father's and mother's age at birth and DNV counts (default mod
##
## Call:
## lm(formula = dnm_counts ~ fatherAge, data = parentalAgeObject)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -33.027  -6.690  -0.768   7.624  29.579
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  19.4131     6.9218   2.805  0.00608 **
```

```
## fatherAge     1.4374     0.2022   7.107 1.93e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.37 on 98 degrees of freedom
## Multiple R-squared:  0.3401, Adjusted R-squared:  0.3334
## F-statistic: 50.51 on 1 and 98 DF,  p-value: 1.927e-10
##
##
## $`confidence interval for specified model type for father's and mother's age at birth and DNV counts
##                 2.5 %    97.5 %
## (Intercept) 5.676962 33.149157
## fatherAge   1.036030  1.838704
```

If you prefer to run the exponential model use:

```
fatherAge(parents, modelType="exponential")
```


```
## $`summary of specified model type for father's and mother's age at birth and DNV counts (default mode
##
## Call:
## lm(formula = log(dnm_counts) ~ fatherAge, data = parentalAgeObject)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.58350 -0.10340  0.00716  0.13030  0.38540
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.477971   0.106692  32.598  < 2e-16 ***
## fatherAge   0.021298   0.003117   6.832 7.12e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.1752 on 98 degrees of freedom
## Multiple R-squared:  0.3226, Adjusted R-squared:  0.3157
## F-statistic: 46.68 on 1 and 98 DF,  p-value: 7.121e-10
##
##
## $`confidence interval for specified model type for father's and mother's age at birth and DNV counts
##                   2.5 %     97.5 %
## (Intercept) 3.26624439 3.68969800
## fatherAge   0.01511153 0.02748386
```

Run Parental Age Analyses for Mother Age Only

motherAge = This function will calculate the correlation between mother's age at birth and DNV counts per individual, the results of the linear model taking the form: lm(formula = dnm_counts ~ motherAge, data = parentalAgeObject) or the exponential model taking the form lm(log(dnm_counts)~motherAge, data=parentalAgeObject). Input required is output from the parentalAgeObject function in this package. Returns the correlation between mother's age at birth and DNV counts per individual and the results of the linear model taking the form: lm(formula = dnm_counts ~ motherAge, data = parentalAgeObject) or the exponential model taking the form lm(log(dnm_counts)~motherAge, data=parentalAgeObject). It also returns a plot of mother's age at birth and DNV counts.

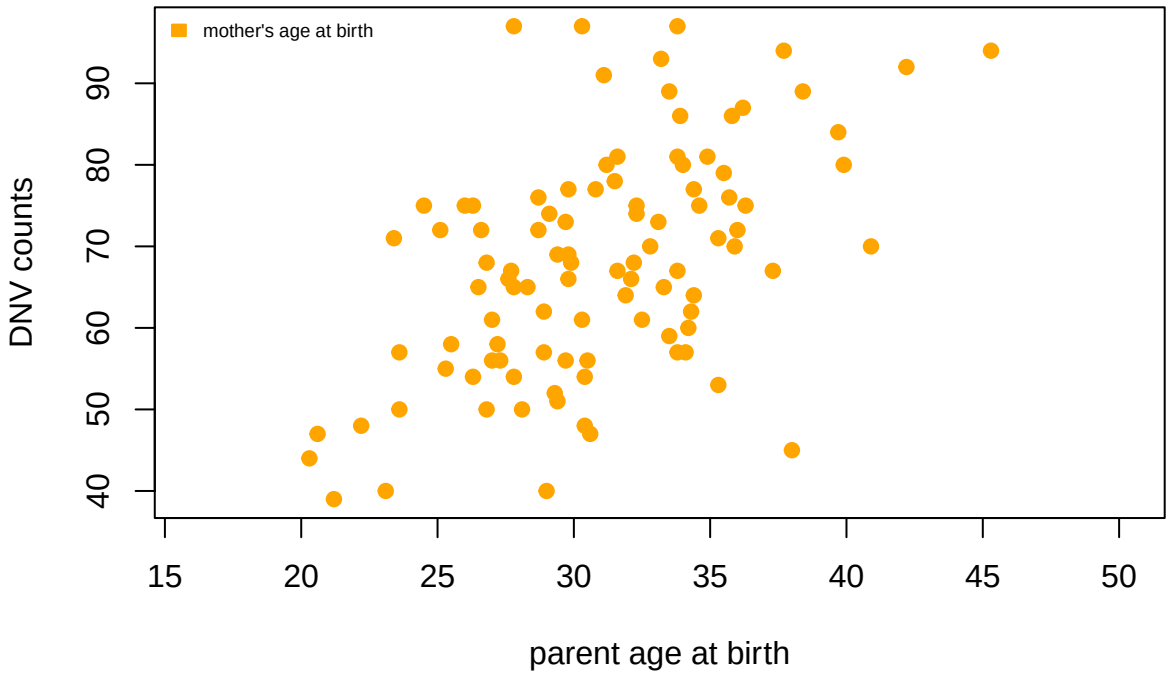
```
motherAge(parents)
```


```
## $`summary of specified model type for father's and mother's age at birth and DNV counts (default mode
##
## Call:
## lm(formula = dnm_counts ~ motherAge, data = parentalAgeObject)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -33.793  -7.578  -0.467   7.825  33.927
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  20.2284     7.8898   2.564   0.0119 *
## motherAge     1.5412     0.2519   6.117 1.96e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.91 on 98 degrees of freedom
## Multiple R-squared:  0.2763, Adjusted R-squared:  0.2689
## F-statistic: 37.42 on 1 and 98 DF,  p-value: 1.955e-08
##
##
## $`confidence interval for specified model type for father's and mother's age at birth and DNV counts
##                 2.5 %   97.5 %
## (Intercept) 4.571389 35.88550
## motherAge   1.041211  2.04113
```

If you prefer to run the exponential model use:

```
motherAge(parents, modelType="exponential")
```




```
## $`summary of specified model type for father's and mother's age at birth and DNV counts (default mod
##
## Call:
## lm(formula = log(dnm_counts) ~ motherAge, data = parentalAgeObject)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.55604 -0.11926  0.00849  0.12494  0.45202
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.468558   0.119869  28.936  < 2e-16 ***
## motherAge   0.023530   0.003828   6.147 1.71e-08 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1809 on 98 degrees of freedom
## Multiple R-squared:  0.2783, Adjusted R-squared:  0.2709
## F-statistic: 37.79 on 1 and 98 DF,  p-value: 1.705e-08
##
##
## $`confidence interval for specified model type for father's and mother's age at birth and DNV counts
##                 2.5 %     97.5 %
## (Intercept) 3.2306824 3.70643266
## motherAge   0.0159343 0.03112591
```

Print the Session Information

```
sessionInfo()
```

```
## R version 4.1.0 (2021-05-18)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Catalina 10.15.7
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] acorn_0.99.8
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.1.0  fastmap_1.1.1   cli_3.6.0       tools_4.1.0
##  [5] htmltools_0.5.4 rstudioapi_0.14 yaml_2.3.7      rmarkdown_2.20
##  [9] highr_0.10      knitr_1.42      xfun_0.37       digest_0.6.31
## [13] rlang_1.0.6     evaluate_0.20
```