1 Maternal Age Effect and Severe Germline bottleneck in the Inheritance of mitochondrial DNA heteroplasmy

This notebook replicates the analyses shown in Rebolledo-Jaramillo, Su et al (2014) Maternal Age Effect and Severe Germline bottleneck in the Inheritance of mitochondrial DNA heteroplasmy PNAS October 28, 2014 vol. 111 no. 43 15474-15479

This analysis uses the following datasets as inputs:

- Allele counts produced with Galaxy pipeline (TODO: Provide link to Galaxy history here);
- GenBank file containing sequence and annotation for human mitochondrial genome (accession NC_012920.1);
- Tab-delimited file of ages for individuals analyzed here
- Known list of problematic sites to be excluded from the analysis

2 Define inputs

This notebook requires three input datasets: * List of variable sites * List of ages for all individuals * List of bad sites (see “Define problematic sites and regions” below)

In [1]: # Replace '1413' with the number of Galaxy history item
    # containing **Variable sites**
    var_sites = 1413

    # Replace '1414' with the number of Galaxy history item
    # containing **Sample ages**
    ages = 1414

    # Replace '1415' with the number of Galaxy history item
    # containing **Bad sites**
    bad_sites = 1415

2.1 Import necessary python modules

- pandas - A library providing high-performance, easy-to-use data structures and data analysis tools
- numpy - A package for scientific computing with Python
- itertools - Functions for creation of iterators for efficient looping
- biopython - A set of Python modules for biological computation

In [2]: import pandas as pd
    import numpy as np
    import itertools
    from Bio import SeqIO
from Bio.Seq import Seq
from Bio import Entrez
from Bio.Alphabet import IUPAC

2.2 Load R extensions and install necessary R modules

In [3]: # Load R magic, which will allow running R directly in the notebook
   %load_ext rpy2.ipython

In [4]: # Make a directory where R modules will be installed
   !mkdir R

mkdir: cannot create directory 'R': File exists

In [5]: %%R
   install.packages("shape", lib="R", repos="http://cran.cnr.berkeley.edu")

The downloaded source packages are in
’/tmp/RtmpZwnpqq/downloaded_packages’

In [6]: %%R
   install.packages("sm", lib="R", repos="http://cran.cnr.berkeley.edu")
The downloaded source packages are in '/tmp/RtmpZwnpqq/downloaded_packages'

In [7]: %R
   install.packages("vioplot", lib="R", repos="http://cran.cnr.berkeley.edu")

   /opt/conda/envs/python2/lib/python2.7/site-packages/rpy2/robjects/functions.py:106: UserWarning: trying URL 'http://cran.cnr.berkeley.edu/src/contrib/vioplot_0.2.tar.gz'
   res = super(Function, self)._call_(*new_args, **new_kwargs)
   res = super(Function, self)._call_(*new_args, **new_kwargs)
   res = super(Function, self)._call_(*new_args, **new_kwargs)

The downloaded source packages are in '/tmp/RtmpZwnpqq/downloaded_packages'

In [8]: %R
   require(shape, lib.loc="R")
   require(sm, lib.loc="R")
   require(vioplot, lib.loc="R")

   res = super(Function, self)._call_(*new_args, **new_kwargs)
   res = super(Function, self)._call_(*new_args, **new_kwargs)
   /opt/conda/envs/python2/lib/python2.7/site-packages/rpy2/robjects/functions.py:106: UserWarning: Package 'sm', version 2.2-5.4: type help(sm) for summary information
   res = super(Function, self)._call_(*new_args, **new_kwargs)
   res = super(Function, self)._call_(*new_args, **new_kwargs)

2.3 Load datasets

In this example all necessary data are located in Galaxy’s history. They can be accessed using the get() function. For example, to load data in the first history item into Jupyter environment simply use get(1), where 1 is the history item number.

   Obviously, if your history looks different, change the numbers in the cells below.

In [9]: # Load Allele Counts

   with open(get(var_sites)) as ac:
       first_line = ac.readline()
       if first_line.startswith("#"):
2.4 Define problematic sites and regions

- Problematic sites are defined as heteroplasmic sites that failed to be validated by experimental means. In particular, there is an additional screening step not shown here, where we calculate the cycle bias of the site, i.e., whether the alternative allele is supported primarily by nucleotides within 25 bp of the read ends. There are 9 such cases, and two additional cases of sites we could not replicate with a new long range PCR (deemed PCR errors). These 11 sites are provided as an input dataset for this analysis.
- Problematic reagions include:
• mtDNA homopolymeres
• region around the artificial “N” at position 3107
• regions within 50 bp of the long range PCR primers

In [12]: # Read in bad (problematic) sites dataset from history

    knownBadhqSites = pd.read_table(get(bad_sites), header=None)

In [13]: # Define problematic regions

    mask = [(66, 71), (303, 311), (514, 523), (12418, 12425), (16184, 16193),
            (3105, 3109), (2817, 2868), (3320, 3370), (10796, 10846), (11520, 11570)]

    maskRegions = list()
    for start, end in mask:
        maskRegions += range(start, end + 1)

2.5 Prepare data

If a header was present in the allele counts input dataset, Pandas assigned the column names automatically. However we will standardize the column names so they can be easily accessed later.

In [14]: df.columns=['sample', 'reference', 'position', 'A', 'C', 'G', 'T', 'a', 'c', 'g', 't', 'cvrg', 'nalleles',
                     'major', 'minor', 'maf', 'sb']

In [15]: # Let’s take a look at the first two lines in the data frame

    df.head(2)

Out[15]:

<table>
<thead>
<tr>
<th></th>
<th>sample</th>
<th>reference</th>
<th>position</th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th>a</th>
<th>c</th>
<th>g</th>
<th>t</th>
<th>cvrg</th>
<th>nalleles</th>
<th>major</th>
<th>minor</th>
<th>maf</th>
<th>sb</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>M117-b1</td>
<td>chrM</td>
<td>2</td>
<td>4343</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5955</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10298</td>
<td>1</td>
<td>A</td>
<td>.</td>
<td>0.0</td>
<td>.</td>
</tr>
<tr>
<td>1</td>
<td>M117-b1</td>
<td>chrM</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4385</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5888</td>
<td>10273</td>
<td></td>
<td>A</td>
<td>.</td>
<td>0.0</td>
<td>.</td>
</tr>
</tbody>
</table>

In our data, all but one mother-child pair conforms to the naming convention:

<table>
<thead>
<tr>
<th></th>
<th>mother</th>
<th>child</th>
</tr>
</thead>
<tbody>
<tr>
<td>family-tissue</td>
<td>familyChild#-tissue</td>
<td></td>
</tr>
<tr>
<td>M477-ch</td>
<td>M477C1-ch</td>
<td></td>
</tr>
</tbody>
</table>

However, the pair M502G (grandmother) and M501 (mother) break the rule. So, we adjusted their ids accordingly:

In [16]: old = ['M502G-ch', 'M502G-b1', 'M501-ch', 'M501-b1']
    new = ['M502-ch', 'M502-b1', 'M502C1-ch', 'M502C1-b1']
    df.replace(to_replace=old, value=new, inplace=True)

2.6 Plot sequencing depth distribution (Fig. S7)

At this point we can calculate the coverage distribution of each sample, as shown in Figure S7 in the PNAS paper. To do so, we need to split the dataframe into blood and cheek dataframes, and make the object available to R (via Rpy2).
In [17]: # Here we split the dataframe into blood and cheek samples
blood = df[df['sample'].str.contains("-bl")]
cheek = df[df['sample'].str.contains("-ch")]  
In [18]: # Let's look at blood data frame
blood.head(2)

Out[18]:
<table>
<thead>
<tr>
<th>sample</th>
<th>reference</th>
<th>position</th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th>a</th>
<th>c</th>
<th>g</th>
<th>t</th>
<th>cvrg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>M117-bl</td>
<td>chrM</td>
<td>2</td>
<td>4343</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5955</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>M117-bl</td>
<td>chrM</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4385</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

nalleles major minor maf sb
0 1 A . 0.0 .
1 1 T . 0.0 .

In [19]: # And at the cheek data frame
cheek.head(2)

Out[19]:
<table>
<thead>
<tr>
<th>sample</th>
<th>reference</th>
<th>position</th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th>a</th>
<th>c</th>
<th>g</th>
<th>t</th>
<th>cvrg</th>
</tr>
</thead>
<tbody>
<tr>
<td>16560</td>
<td>M117-ch</td>
<td>chrM</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1829</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3239</td>
<td>0</td>
</tr>
<tr>
<td>16561</td>
<td>M117-ch</td>
<td>chrM</td>
<td>2</td>
<td>1829</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3291</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

nalleles major minor maf sb
16560 1 G . 0.0 .
16561 1 A . 0.0 .

In [20]: # Use Rmagic to load data into R using the -i flag
# This step will take a bit (~2 min)

%%R -i cheek,blood

Let's peek at the R version of the blood dataframe:

In [21]: %%R
head(blood,2)

sample reference position A C G T a c g t cvrg nalleles major
0 M117-bl chrM 2 4343 0 0 0 5955 0 0 0 10298 1 A
1 M117-bl chrM 3 0 0 4385 0 0 0 5888 0 0 0 10273 1 T

minor maf sb
0 . 0 .
1 . 0 .

Transform numeric looking columns into actual numeric columns to guarantee the value types:

In [22]: %%R
tonumeric = c(3:13,16)
blood[,tonumeric] = apply(blood[,tonumeric], 2, function(x) as.numeric(as.character(x)))
cheek[,tonumeric] = apply(cheek[,tonumeric], 2, function(x) as.numeric(as.character(x)))

Define custom R function to generate Figure S7:

In [23]: %%R
boxPlotCvrg = function(data,tissue){
names = sort(unique(data["sample"])))
data["sample"] = factor(data["sample"], levels=names)

boxplot(log10(cvrg) ~ sample, data=data, whisklty="solid", outline=F, whisklwd=0.5, boxlwd=1, medlwd=1, medcol="red", main="", ylab="log10(coverage)", bty="n", frame=F, boxcol="white", boxfill="black", medlwd=3, whiskcol="grey", staplecol="grey", ylim=c(2, 6))

mtext(tissue, adj=0, side=3, las=1, at=length(names)/2, font=2, cex=1.25)

2.6.1 Plot the figure

You can adjust the size of the plotting image by adjusting:

- `-w` = width
- `-h` = height
- `-u` = units
- `-r` = resolution

In [24]: ```
%%R
-w 18 -h 10 -u in -r 72
``` par(las=2)
par(mar=c(4,4,4,1))
par oma=c(2, 2, 0, 0))
par(mfrow=c(2,1))
par(cex.lab=1.25)
par(cex.axis=0.75)

boxPlotCvrg(cheek,"cheek")
boxPlotCvrg(blood,"blood")
2.7 Define high quality heteroplasmic sites

We define high quality (HQ) sites as:

1. minor allele frequency (maf) ≥ 1%
2. coverage ≥ 1000
3. maf balance (1% in forward and reverse strands)
4. no strand bias
5. outside “problematic sites”:
   - mtDNA homopolymeres
   - around the artificial “N” at position 3107
   - within 50 bp of the long range PCR primers

In [25]: 
   # Filter sites on minor allele frequency (maf), coverage (cvrg) and whether the sites are located in problematic regions 
   # The list of problematic region (maskRegions) is defined in cell 12 above 
   hq_sites = df[(df.maf>0.01) & (df.cvrg>=1000) & ~df.position.isin(maskRegions)]

In [26]: len(hq_sites)

Out[26]: 559

By applying these initial filters, we reduced the dataframe from ~2 million lines to 572 lines only, which is much more manageable. Next, we calculate strand bias and maf balance for these 572 sites. The strand bias calculation is performed according to Guo Y et al. 2012

In [27]: 
   # Compute strand and minor allele frequency bias

   def strand_stats(x, mafThreshold=0.01):
       falleles = ['A','C','G','T']
       ralleles = ['a','c','g','t']
       sample,position,major,minor,coverage,maf = x[['sample','position','major','minor','coverage','maf']]
       fcounts = x[falleles]
       rcounts = x[ralleles]
       if minor!='.':
           index_major = falleles.index(major)
           index_minor = falleles.index(minor)

           fcount_minor = float(fcounts[index_minor])
           ftotal = fcount_minor + fcounts[index_major]

           rcount_minor = float(rcounts[index_minor])
           rtotal = rcount_minor + rcounts[index_major]

           minor_total = float(fcount_minor + rcount_minor)
           site_total = ftotal + rtotal

           try:
               strandBias = abs( (fcount_minor/ftotal) - (rcount_minor/rtotal) ) / (minor_total/site_total)
           except:
               strandBias = np.nan

           try:
               maf_frwd = fcount_minor/sum(fcounts)
           except:
maf_frwd = np.nan
try:
    maf_rvrs = rcount_minor/sum(rcounts)
except:
    maf_rvrs = np.nan

if (maf_frwd>=mafThreshold) and (maf_rvrs>=mafThreshold):
    mafBalance = 1
else:
    mafBalance = 0
else:
    strandBias = float(2)
    mafBalance = 0

return pd.Series([strandBias,mafBalance])

In [28]: # Apply strand calculations to the data

biasCols = hq_sites.apply(strand_stats,axis=1, args=(0.01,))
biasCols.columns = ["strandBias", "mafBalance"]
hq_sites = pd.concat([hq_sites, biasCols], axis=1)

In [29]: # Filter on strand and maf balance

hq_sites = hq_sites[(hq_sites.strandBias<=1) & (hq_sites.mafBalance==1)]
len(hq_sites)

Out[29]: 190

In [30]: # Set bad sites column names
knownBadhqSites.columns=["sample", "position"]

# Adjust naming convention for the anomalous grandmother-mother pair
knownBadhqSites.replace(to_replace=old, value=new, inplace=True)

# Transform bad sites into a hashable object
bad = set(knownBadhqSites.itertuples(index=False))

# Get a boolean array to filter high quality sites
good = [x not in bad for x in hq_sites[['sample', 'position']].itertuples(index=False)]

# Finally, filter high quality sites
hq_sites = hq_sites[good]

In [31]: len(hq_sites)
#hq_sites.to_csv("hq173.txt", sep="\t", index=False)

Out[31]: 181

2.8 Test statistical significance of high quality sites

Finally, we calculate the significance of the minor allele frequency of a site provided the error rate at that position. The error rate is estimated from the remaining 155 samples, and the expected allele accounts are compared to the observed allele counts:

In [32]: from scipy.stats import poisson
# We define a poisson function that will take a single high quality site, and explore the variability of # the position among the remaining samples

def poisson_pval(current_df,sample):
    alleles = ['A','C','G','T','a','c','g','t']
    sample_counts = list(current_df.loc[current_df['sample']==sample, alleles].iloc[0,:])
    others_counts = list(current_df.loc[current_df['sample']!=sample, alleles].apply(sum,axis=0)
    sample_coverage = sum(sample_counts)
    observed_error = (sum(others_counts) - max(others_counts))/float(sum(others_counts))
    sample_nonMajor_counts = int(sample_coverage - max(sample_counts))
    pvalue = poisson.pmf(sample_nonMajor_counts, observed_error*sample_coverage)
    return pvalue

In [34]: poisson_pvalues = []

      for sample,position in hq_sites["sample","position"].itertuples(index=False):
          poisson_pvalues.append(poisson_pval(df[df['position']==position],sample))

      hq_sites["poisson"] = poisson_pvalues
      hq_sites = hq_sites[hq_sites.poisson<=0.05]
      len(hq_sites)

Out[34]: 181

As described in the paper, all sites were statistically significant under the Poisson and Likelihood (not shown here) frameworks.

### 3 Screening for contamination

In our previous publication, Dickins, Rebolloedo-Jaramillo, et al (2014) Controlling for contamination in resequencing studies with a reproducible web-based phylogenetic approach BioTechniques, 56(3):134–141, we described warning signs of a potential contamination. They include: 1. Excess heteroplasmic sites (>5 per sample) 2. Tight minor allele frequency distribution 3. Non-family related positions of heteroplasmic sites

We routinely apply our contamination detection pipeline, so we are confident our sites in the PNAS paper were not artifacts. As an example of the screening for contamination, we can plot the number of sites and the minor allele frequency distribution of all samples in the high quality sites set:

In [35]: # Make R aware of the hq_sites dataframe
     %R -i hq_sites

In [36]: %%%R

# Adjust value types in the hq_sites dataframe

tonumeric = c(3:13,16:18)
hq_sites[,tonumeric] = apply(hq_sites[,tonumeric], 2, function(x) as.numeric(as.character(x)))
head(hq_sites,2)
In [37]: %%R 

# Plot hq_sites number of sites and minor allele frequency distribution

par(mar=c(4,15,4,0))
boxplot(
  maf~sample,
  data=hq_sites,
  pch=16,cex=0.75,
  outcol="azure3",outline=T,
  whisklty=1,whiskwd=1.5,
  staplewd=1.5,
  boxwex=0.75,boxcol="white",boxfill="cornflowerblue",
  horizontal=T,cex.axis=0.75,las=2,
  frame=F,xaxt="n",ylim=c(-0.01,0.5)
)

axis(1,at=seq(0,5,0.5)/10,lab=100*(seq(0,5,0.5)/10),cex.axis=0.75)
title(xlab="minor allele frequency (\%)",line=2.5,cex.axis=0.75)

nsites = table(hq_sites["sample"])
for (i in 1:length(nsites)){
  text(-0.01,i,lab=nsites[i],cex=0.75)
}

mtext("Nsites:",side=3,line=-2,at=0,adj=1.25)
3.1 Placing high quality sites into *quartets*

For each high quality site, we can retrieve the minor allele frequency information for the remaining 3 samples in the family collection. So, a quartet is simply a tabulation of the minor allele frequency for the mother blood and cheek, and her child blood and cheek, for the same site. Below is an example of a quarted for family **M494**:

<table>
<thead>
<tr>
<th>family</th>
<th>position</th>
<th>major</th>
<th>minor</th>
<th>mother_cheek</th>
<th>mother_blood</th>
<th>child_cheek</th>
<th>child_blood</th>
</tr>
</thead>
<tbody>
<tr>
<td>M494</td>
<td>9196</td>
<td>G</td>
<td>A</td>
<td>0.032</td>
<td>0.030</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

However, before we can do that, we need to add family information to the high quality sites. We will do that by extracting the family id from each sample’s id. It is also useful to have a way to split the data by tissue or member of the pair, so we will add the columns *family*, *tissue* and *member*, accordingly.

**In [38]:**  # Get family id from sample id  
# i.e. M512C1-ch returns M512

```python
def getfamlables(samplename):
    nameparts = [''.join(x) for _, x in itertools.groupby(samplename, key=str.isdigit)]
    family = ''.join(nameparts[:2])
    if "-ch" in nameparts:
        tissue = "cheek"
    else:
        tissue = "blood"
    if len(nameparts)>3:
        pairclass = "child"
    else:
        pairclass = "mother"
    return pd.Series([family, tissue, pairclass])
```

**In [39]:**  # Apply the above function to the data

```python
hq_sites["family","tissue","member"] = hq_sites["sample"].apply(getfamlables)
```

**In [40]:**  hq_sites.head(2)

```
<table>
<thead>
<tr>
<th>sample</th>
<th>reference</th>
<th>position</th>
<th>chrM</th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th>a</th>
<th>c</th>
<th>g</th>
<th>maf</th>
<th>sb strandBias</th>
<th>mafBalance</th>
</tr>
</thead>
<tbody>
<tr>
<td>49743 M117C1-ch</td>
<td>chrM</td>
<td>214</td>
<td>1234</td>
<td>0</td>
<td>22</td>
<td>0</td>
<td>1581</td>
<td>0</td>
<td>36</td>
<td></td>
<td>0.02019</td>
<td>0.23517</td>
<td>0.235166</td>
</tr>
<tr>
<td>80535 M132-bl</td>
<td>chrM</td>
<td>14461</td>
<td>0</td>
<td>195</td>
<td>0</td>
<td>4356</td>
<td>0</td>
<td>183</td>
<td>0</td>
<td></td>
<td>0.04041</td>
<td>0.11746</td>
<td>0.117458</td>
</tr>
</tbody>
</table>
```

```python
poisson family tissue member
49743 0.011098 M117 cheek child
80535 0.000005 M132 blood mother
```

[2 rows x 23 columns]
In [41]: # Now we can extract the unique quartets by selecting the "family" and "position" columns

    unique_quartets = hq_sites[['family', 'position']].drop_duplicates()
    len(unique_quartets)

Out[41]: 108

In [42]: unique_quartets.head(2)

Out[42]:
                  family  position
    49743         M117       214
    80535         M132      14461

In [43]: # For each family_id/position combination,
    # retrieve the information for all 4 members of the quartets
    # from the original data dataframe

def getQuartets(hqsite):
    position = hqsite['position']
    familyid = hqsite['family']
    pos_data = df[df['position'] == position]
    allmembers = [s for s in pos_data['sample'].drop_duplicates() if s.startswith(familyid)]
    mother = min([len(x) for x in allmembers])
    child = max([len(x) for x in allmembers])
    if len(allmembers) == 4:
        for member in allmembers:
            if len(member) == mother and member.endswith('-ch'):
                motherCheek = df[(df['sample'] == member) & (df['position'] == position)][['major', 'minor', 'maf']]
            elif len(member) == mother and member.endswith('-bl'):
                motherBlood = df[(df['sample'] == member) & (df['position'] == position)][['major', 'minor', 'maf']]
            elif len(member) == child and member.endswith('-ch'):
                childCheek = df[(df['sample'] == member) & (df['position'] == position)][['major', 'minor', 'maf']]
            else:
                childBlood = df[(df['sample'] == member) & (df['position'] == position)][['major', 'minor', 'maf']]
    else:
        pass

In [44]: # Apply getQuartets to data

    quartets = unique_quartets.apply(getQuartets, axis=1)

In [45]: # The following is necessary to remove empty rows from the dataframe

    quartets = quartets.dropna()

In [46]: quartets

Out[46]:
   0  1   2  3   4  5  6  7   8   9  10  11  12
--- --- --- --- --- --- --- -- --- --- --- --- ---
49743 M117  214.0  A  G  0.00181  A  G  0.00043  A  G  0.02019  A  G
80535 M132 14461.0  T  C  0.04461  T  C  0.04041  T  C  0.00145  T  C
82828 M132  185.0  A  G  0.01535  A  G  0.00569  A  G  0.00522  A  G
<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2502781</td>
<td>SC16</td>
<td>16170.0</td>
<td>A</td>
<td>G</td>
<td>0.06300</td>
<td>A</td>
<td>G</td>
</tr>
<tr>
<td>2503365</td>
<td>SC16</td>
<td>185.0</td>
<td>A</td>
<td>G</td>
<td>0.01450</td>
<td>A</td>
<td>G</td>
</tr>
<tr>
<td>2512743</td>
<td>SC16</td>
<td>9565.0</td>
<td>G</td>
<td>A</td>
<td>0.01445</td>
<td>G</td>
<td>A</td>
</tr>
<tr>
<td>2519899</td>
<td>SC16</td>
<td>152.0</td>
<td>T</td>
<td>C</td>
<td>0.00178</td>
<td>T</td>
<td>C</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>49743</td>
<td>0.00215</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80535</td>
<td>0.00089</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>82828</td>
<td>0.00506</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>115798</td>
<td>0.00019</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>140360</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>145325</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>147727</td>
<td>0.00036</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>191703</td>
<td>0.00599</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>308891</td>
<td>0.06739</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>328908</td>
<td>0.00013</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>330811</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>342450</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>365134</td>
<td>0.02330</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>366061</td>
<td>0.01087</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>375236</td>
<td>0.02052</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>474020</td>
<td>0.00044</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>507120</td>
<td>0.02656</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>539922</td>
<td>0.00050</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>558655</td>
<td>0.01555</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>589762</td>
<td>0.00158</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>597746</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>602733</td>
<td>0.00049</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>623866</td>
<td>0.01150</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>631050</td>
<td>0.02196</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>634146</td>
<td>0.02859</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>649833</td>
<td>0.00044</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>666023</td>
<td>0.00037</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>666180</td>
<td>0.00183</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>666832</td>
<td>0.00062</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>666887</td>
<td>0.00043</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1975402</td>
<td>0.22615</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000080</td>
<td>0.04190</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012839</td>
<td>0.04255</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2046767</td>
<td>0.00127</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2046777</td>
<td>0.00027</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2067319</td>
<td>0.05447</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2090706</td>
<td>0.00028</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2113750</td>
<td>0.00053</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2156362</td>
<td>0.00141</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2163961</td>
<td>0.40417</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2166255</td>
<td>0.36141</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2179847</td>
<td>0.00040</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2228303</td>
<td>0.00055</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2242245</td>
<td>0.00451</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2245112</td>
<td>0.00044</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2291624</td>
<td>0.00117</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2293155</td>
<td>0.00047</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In [47]: # Set column names
# mc: mother cheek
# mb: mother blood
# cc: child cheek
# cb: child blood

quartets.columns = ["family","position","mcMajor","mcMinor","mcMAF","mbMajor","mbMinor","mbMAF","ccMajor","ccMinor","ccMAF","cbMajor","cbMinor","cbMAF"]

In [48]: quartets.head(2)

Out[48]:

<table>
<thead>
<tr>
<th>family</th>
<th>position</th>
<th>mcMajor</th>
<th>mcMinor</th>
<th>mcMAF</th>
<th>mbMajor</th>
<th>mbMinor</th>
<th>mbMAF</th>
<th>ccMajor</th>
<th>ccMinor</th>
<th>ccMAF</th>
<th>cbMajor</th>
<th>cbMinor</th>
<th>cbMAF</th>
</tr>
</thead>
<tbody>
<tr>
<td>49743</td>
<td>M117</td>
<td>214.0</td>
<td>A</td>
<td>G</td>
<td>0.00181</td>
<td>A</td>
<td>G</td>
<td>0.00043</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80535</td>
<td>M132</td>
<td>14461.0</td>
<td>T</td>
<td>C</td>
<td>0.04461</td>
<td>T</td>
<td>C</td>
<td>0.04041</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We can add even more information to the quartets table. For instance, the impact of the alternative allele and the nucleotide change class:

In [49]: # Define function for generating protein translation

```python
def translate(sequence,gene):
    if len(str(sequence))%3!=0:
        add=3 - (len(str(sequence))%3)
    else:
        add=0

    if genedb[gene]["strand"]==1:
        modseq=str(sequence)+add*'A'
    else:
        modseq=str(sequence.reverse_complement())+add*'A'

    try:
        translation=str(Seq(modseq,IUPAC.unambiguous_dna).translate(table=2,cds=True))
    except:
        translation=[]
```
def evoImpact(quartet):
    try:
        het, major, minor = quartet
        pos = int(het)-1
        gene = [g for g in genedb if genedb[g]['end']>=pos>=genedb[g]['start']][0]

        if gene in [feature.qualifiers['gene'][0] for feature in rCRS.features if feature.type in ['rRNA', 'tRNA', 'CDS']]:
name = feature.qualifiers['gene'][0]
start = int(feature.location.start)
end = int(feature.location.end)
strand = int(feature.location.strand)
genedb[name] = dict(zip(labs, [ftype, start, end, strand]))
genedb['D-loop1'] = dict(zip(labs, ['Dloop', 0, 576, 1]))
genedb['D-loop2'] = dict(zip(labs, ['Dloop', 16023, 16569, 1]))

In [52]: a = quartets.loc[[282115:328908]]
a.head(20)
# a[['position', 'mbMajor', 'mbMinor']].apply(evoImpact, axis=1)

Out[52]:

<table>
<thead>
<tr>
<th>position</th>
<th>mbMajor</th>
<th>mbMinor</th>
<th>mcMAF</th>
<th>mbMAF</th>
</tr>
</thead>
<tbody>
<tr>
<td>308891</td>
<td>M188</td>
<td>16240.0</td>
<td>A</td>
<td>G</td>
</tr>
<tr>
<td>328908</td>
<td>M190</td>
<td>3202.0</td>
<td>T</td>
<td>C</td>
</tr>
</tbody>
</table>

ccMajor ccMinor ccMAF cbMajor cbMinor cbMAF
308891 A G 0.10872 A G 0.06739
328908 T C 0.00043 T C 0.00013

In [53]: # We set the ancestral state to the alleles found in the mother's blood sample.

quartets[['ptchange', 'class', 'ntchange']] = quartets[['position', 'mbMajor', 'mbMinor']].apply(evoImpact, axis=1)

Finalized quartets table:

In [54]: quartets.head(2)

Out[54]:

<table>
<thead>
<tr>
<th>family</th>
<th>position</th>
<th>mcMajor</th>
<th>mcMinor</th>
<th>mcMAF</th>
<th>mbMajor</th>
<th>mbMinor</th>
<th>mbMAF</th>
</tr>
</thead>
<tbody>
<tr>
<td>49743</td>
<td>M117</td>
<td>214.0</td>
<td>A</td>
<td>G</td>
<td>0.00181</td>
<td>A</td>
<td>G</td>
</tr>
<tr>
<td>80535</td>
<td>M132</td>
<td>14461.0</td>
<td>T</td>
<td>C</td>
<td>0.04461</td>
<td>T</td>
<td>C</td>
</tr>
</tbody>
</table>

ccMajor ccMinor ccMAF cbMajor cbMinor cbMAF
49743 A G 0.02019 A G 0.00215
80535 T C 0.00145 T C 0.00089

ntchange
49743 ts
80535 ts

3.2 Plot the number of heteroplasmic sites per individual or family (Fig. S11)

In [55]: # Since we modified the hq_sites dataframe, we have to reload it in R

%%R

Load modified hq_sites and quartets

In [56]:

# Adjust value types in the hq_sites dataframe
tonumeric = c(3:13, 16:18)
hq_sites[, tonumeric] = apply(hq_sites[, tonumeric], 2, function(x) as.numeric(as.character(x)))
head(hq_sites, 2)

<table>
<thead>
<tr>
<th>sample reference</th>
<th>position</th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th>a</th>
<th>c</th>
<th>g</th>
<th>t</th>
<th>cvrg</th>
</tr>
</thead>
<tbody>
<tr>
<td>49743 M117C1-ch</td>
<td>chrM</td>
<td>214</td>
<td>1234</td>
<td>0</td>
<td>22</td>
<td>0</td>
<td>1581</td>
<td>0</td>
<td>36</td>
<td>0 2873</td>
</tr>
<tr>
<td>80535 M132-b1</td>
<td>chrM</td>
<td>14461</td>
<td>0</td>
<td>195</td>
<td>0</td>
<td>4356</td>
<td>0</td>
<td>183</td>
<td>0</td>
<td>4620 9354</td>
</tr>
</tbody>
</table>
nalleles major minor maf sb strandBias mafBalance poisson
49743 2 A G 0.02019 0.23517 0.23516 63 1 1.109776e-02
80535 2 T C 0.04041 0.11746 0.11745 80 1 4.761880e-06

family tissue member
49743 M117 cheek child
80535 M132 blood mother

In [57]: %%R

# Frequency (number of sites per individual)

getFreq = function(data,tissue,member) {
    siteFreq = data.frame(table(table(as.character(data[(data["tissue"]==tissue) & (data["member"]==member),1])))
    siteFreq = unlist(apply(siteFreq,1,FUN=function(x) rep(x[1],x[2])))
    siteFreq = as.numeric(c(rep(0,39-length(siteFreq)),siteFreq))

    return(siteFreq)
}

In [58]: %%R

# Size of circles
symbolPlot = function(data,pos) {
    symbols(rep(pos,length(unique(data))),
    sort(unique(data)),circles=(data.frame(table(data))$Freq)*0.01,
    add=T,inches=F,bg="black")
}

In [59]: %%R

# Backbone boxplot
boxPlotNsites = function(data,pos,addOpt="False"){
    boxplot(data,ylim=c(-2,maxSites),frame=F,axes=F,xlim=c(1,7),at=pos,col=rgb(0,0,0,0),
    boxlwd=2,boxcol="coral3",medcol="coral3",whisklty="solid",whiskcol="coral3",
    staplecol="coral3",add=as.logical(addOpt),outline=F)
}

In [60]: %%R -w 11 -h 8 -u in -r 72
mc = getFreq(hq_sites,"cheek","mother")
mh = getFreq(hq_sites,"blood","mother")
cc = getFreq(hq_sites,"cheek","child")
ch = getFreq(hq_sites,"blood","child")

fam = data.frame(table(table(quartets["family"])))
fam = unlist(apply(fam,1,FUN=function(x) rep(as.numeric(x[1]),x[2])))
fam = as.numeric(c(rep(0,39-length(fam)),fam))
maxSites = max(c(mc, mb, cc, cb, fam))
par(mar=c(2, 2, 2, 1))
par oma=c(0, 0, 0, 0))

plot(1:7, 1:7, type="n", ylim=c(-2, maxSites), frame=F, axes=F)

symbolPlot(mc, 2)
symbolPlot(mb, 3)
symbolPlot(cc, 4)
symbolPlot(cb, 5)
symbolPlot(fam, 6)

Lab = c("Maternal \ncheek", "Maternal \nblood", "Child \ncheek", "Child \nblood", "Family")

axis(1, at=2:6, lab=Lab, pos=-1.5, las=1, cex.axis=0.8, tck=-0.01)
axis(2, at=0:maxSites, lab=0:maxSites, pos=1, las=2, cex.axis=0.8)

mtext("Number of point heteroplasmies", 2, 1, cex=0.8, adj=0.7)

par(new=T)
boxPlotNsites(mc, 2)
boxPlotNsites(mb, 3, "True")
boxPlotNsites(cc, 4, "True")
boxPlotNsites(cb, 5, "True")
boxPlotNsites(fam, 6, "True")
3.3 Plot distribution of high quality heteroplasmy (Fig. S10)

In [61]: %

```r
data = quartets
uniqueFamilies = sort(unique(as.character(data$family)))
faid = cbind(1:length(uniqueFamilies),uniqueFamilies)

plotid=c()
for (s in as.character(data$family)){
    plotid=c(plotid,faid[faid[,2]==s,1])
}
data$id = as.numeric(plotid)
data$position = as.numeric(as.character(data$position))
head(data)
```

<table>
<thead>
<tr>
<th>family</th>
<th>position</th>
<th>mcMajor</th>
<th>mcMinor</th>
<th>mcMAF</th>
<th>mbMajor</th>
<th>mbMinor</th>
<th>mbMAF</th>
<th>ccMajor</th>
</tr>
</thead>
<tbody>
<tr>
<td>49743</td>
<td>M117</td>
<td>A</td>
<td>G</td>
<td>0.00181</td>
<td>A</td>
<td>G</td>
<td>0.00043</td>
<td>A</td>
</tr>
<tr>
<td>80535</td>
<td>M132</td>
<td>T</td>
<td>C</td>
<td>0.04461</td>
<td>T</td>
<td>C</td>
<td>0.04041</td>
<td>T</td>
</tr>
<tr>
<td>82828</td>
<td>M132</td>
<td>A</td>
<td>G</td>
<td>0.01535</td>
<td>A</td>
<td>G</td>
<td>0.00569</td>
<td>A</td>
</tr>
<tr>
<td>115798</td>
<td>M132</td>
<td>C</td>
<td>T</td>
<td>0.00307</td>
<td>C</td>
<td>T</td>
<td>0.00030</td>
<td>C</td>
</tr>
<tr>
<td>140360</td>
<td>M137</td>
<td>A</td>
<td>.</td>
<td>0.00000</td>
<td>A</td>
<td>G</td>
<td>0.01451</td>
<td>A</td>
</tr>
<tr>
<td>145325</td>
<td>M137</td>
<td>T</td>
<td>C</td>
<td>0.00105</td>
<td>T</td>
<td>C</td>
<td>0.01591</td>
<td>T</td>
</tr>
</tbody>
</table>

ccMinor | ccMAF | cbMajor | cbMinor | cbMAF | ptchange | class | ntchange | id |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>49743</td>
<td>G</td>
<td>0.02019</td>
<td>A</td>
<td>G</td>
<td>0.00215</td>
<td>-</td>
<td>Dloop</td>
<td>ts 1</td>
</tr>
<tr>
<td>80535</td>
<td>C</td>
<td>0.00145</td>
<td>T</td>
<td>C</td>
<td>0.00089</td>
<td>nonsyn</td>
<td>CDS</td>
<td>ts 2</td>
</tr>
<tr>
<td>82828</td>
<td>G</td>
<td>0.00522</td>
<td>A</td>
<td>G</td>
<td>0.00506</td>
<td>-</td>
<td>Dloop</td>
<td>ts 2</td>
</tr>
<tr>
<td>115798</td>
<td>T</td>
<td>0.01945</td>
<td>C</td>
<td>T</td>
<td>0.00019</td>
<td>-</td>
<td>Dloop</td>
<td>ts 2</td>
</tr>
<tr>
<td>140360</td>
<td>T</td>
<td>0.00037</td>
<td>A</td>
<td>.</td>
<td>0.00000</td>
<td>nonsyn</td>
<td>CDS</td>
<td>ts 3</td>
</tr>
<tr>
<td>145325</td>
<td>.</td>
<td>0.00000</td>
<td>T</td>
<td>.</td>
<td>0.00000</td>
<td>nonsyn</td>
<td>CDS</td>
<td>ts 3</td>
</tr>
</tbody>
</table>

In [62]: %

```r
# Define colors for each mitochondrial genome features

alp =200
trna ="blue"
rrna ="lightseagreen"
prot ="orange"
dloop="red"

colors=c()
for (c in data[["class"]]) if (c=="Dloop") {
    colors=c(colors,dloop)
} else if (c=="tRNA") {
    colors=c(colors,trna)
} else if (c=="rRNA") {
    colors=c(colors,rrna)
} else {
    colors=c(colors,prot)
```

22
# Define symbols depending on whether a site is syn/nonsyn or ts/tv

```r
symbol = c()
for (i in 1:nrow(data)) {
  if (data["ntchange"] [i] == "tv" & data["ptchange"] [i] == "nonsyn") {symbol = c(symbol, 17)
  } else if (data["ntchange"] [i] == "tv" & data["ptchange"] [i] != "nonsyn") {symbol = c(symbol, 2)
  } else if (data["ptchange"] [i] == "syn") {
    symbol = c(symbol, 16)
  } else {
    symbol = c(symbol, 1)
  }
}
data$symbol = symbol
```

In [64]: %%%

```r
-w 5 -h 4 -u in -r 144
```

# Grid

```r
for (i in data$id) {segments(1,i*5+5,16569,i*5+5,col="grey90")}
for (i in seq(500,16500,500)) {segments(i,5,i,162,col="grey90")}
for (i in seq(500,16500,1000)) {text(i,163,lab=i,col="black",cex=0.25)}
mtext("Family",3,-1.2,las=1,adj=-0.07,cex=0.5)
mtext("coord:",3,-1.35,las=1,adj=0.01,cex=0.3)
```

# Actual plot

```r
points(data$position,data$id*5+5,xlim=c(1,16569), ylim=c(-80,170),pch=data$symbol,ylab="",col=data$colors, xlab="")
```

# Axes

```r
abline(h=5)
lab = unique(data[,c(1,18)])
axis(2, pos=-500, at=sort(unique(data$id*5+5)), labels=lab[order(lab$id),][,1], cex.axis=0.4)
```

# Legend

```r
legend(1,180,legend=c("D-loop","tRNA","rRNA","cds-syn","cds-nonSyn","transversion"),
fill=c(NA,NA,NA,NA,NA,NA),border=c(rep("white",4),NA,NA),pch=c(1,1,1,1,16,2),
col=c(dloop,trna,rrna,prot,prot,"black"),cex=0.5,bty="n",pt.cex=0.7,horiz=T, x.intersp=c(0.7,0.7,0.7,0.7,0.7,1),text.width=1350)
```

# mtDNA genes

```r
Arrows(1,0,576,0,arr.length=0.05,arr.type='simple')
text(576,0,labels='DLOOP',cex=0.5,pos=4)
Arrows(576,-5,647,-5,arr.length=0.05,arr.type='simple')
text(647,-5,labels='TRNF',cex=0.5,pos=4)
Arrows(647,-10,1601,-10,arr.length=0.05,arr.type='simple')
```
3.4 Plot correlations in minor allele frequencies (Fig. 1)

In [65]: %R

```r
#head(quartets,3)
quartets[as.character(quartets$cbMinor)!=as.character(quartets$ccMinor),]
```

family position mcMajor mcMinor mcMAF mbMajor mbMinor mbMAF ccMajor ccMAF
There are a few cases of reversal of minor allele frequencies between two tissues of the same individual, or between a mother and her child. Consequently, it is necessary to fix the “ancestral” allele, and we arbitrarily decided to use the maternal blood as the ancestral state.

In [66]: %%%R

```
# mb = maternal blood
# mc = maternal cheek
# cb = child blood
# cc = child cheek

adjustMAF = function(row){
  mbMajor = row["mbMajor"]
  ...
```
mbMinor = row["mbMinor"]
mcMajor = row["mcMajor"]
mcMinor = row["mcMinor"]
ccMajor = row["ccMajor"]
ccMinor = row["ccMinor"]
cbMajor = row["cbMajor"]
cbMinor = row["cbMinor"]

if ((c(mbMajor, mbMinor) == c(mcMinor, mcMajor)) & (mcMinor != ".")){
    mcMAFadj = 1 - as.numeric(row["mcMAF"])
} else{
    mcMAFadj = as.numeric(row["mcMAF"])
}

if ((c(mbMajor, mbMinor) == c(ccMinor, ccMajor)) & (ccMinor != ".")){
    ccMAFadj = 1 - as.numeric(row["ccMAF"])
} else{
    ccMAFadj = as.numeric(row["ccMAF"])
}

if ((c(mbMajor, mbMinor) == c(cbMinor, cbMajor)) & (cbMinor != ".")){
    cbMAFadj = 1 - as.numeric(row["cbMAF"])
} else{
    cbMAFadj = as.numeric(row["cbMAF"])
}

return(c(mcMAFadj, ccMAFadj, cbMAFadj))

In [67]: %R

adjustedMAF = data.frame(t(apply(quartets, 1, adjustMAF)))
colnames(adjustedMAF) = c("mc", "cc", "cb")

head(adjustedMAF, 2)

<table>
<thead>
<tr>
<th>mc</th>
<th>cc</th>
<th>cb</th>
</tr>
</thead>
<tbody>
<tr>
<td>49743</td>
<td>0.00181</td>
<td>0.02019</td>
</tr>
<tr>
<td>80535</td>
<td>0.04461</td>
<td>0.00145</td>
</tr>
</tbody>
</table>

Due to the adjustment of MAF based on the maternal state, comparing the child tissues independently of the mother’s, require an additional adjustment.

In [68]: %R

ccx=c()
for (maf in adjustedMAF$cc) {if (maf>0.5) ccx=c(ccx, (1-maf)) else ccx=c(ccx, maf)}
cbx=c()
for (maf in adjustedMAF$cb) {if (maf>0.5) cbx=c(cbx, (1-maf)) else cbx=c(cbx, maf)}

In [69]: %R
xyplot = function(x,y,sub,xtissue,ytissue,case){
  xLab = paste("het. allele frequency ",xtissue,"",sep="")
  yLab = paste("het. allele frequency ",ytissue,"",sep="")

  plot(x,y,pch=20,col="#00000078",axes=F,xlab=xLab, ylab=yLab,cex.lab=0.85,cex=2,xlim=c(0,1),ylim=c(0,1))

  abline(lm(x~y), col="darkgrey",lwd=1)
  mylabel = bquote(italic(R)^2 == .(round(summary(lm(x~y))$r.squared,2)))

  text(-0.05,0.7, pos=4,labels=mylabel,font=2,cex=1)
  text(-0.05,0.9,labels=case,cex=1,pos=4)
  mtext(sub,3,0.5,at=0,cex=1,font=1)
  axis(1,at=c(0,0.5,1))
  axis(2,at=c(0,0.5,1))
}

In [70]: %%R -w 4 -h 4 -u in -r 144

par(mfrow=c(2,2))
par(oma=c(0,0,0,0))
par(mar=c(3,2.5,2,1))

par(mgp=c(1.5,0.25,0.25))
par(tck=-0.05)

xyplot(ccx,cbx,"A","cheek","blood","CHILD")
xyplot(adjustedMAF$mc,quartets$mbMAF,"B","cheek","blood","MOTHER")
xyplot(adjustedMAF$mc,ccx,"C","mother","child","CHEEK")
xyplot(quartets$mbMAF,cbx,"D","mother","child","BLOOD")
3.5 Plot the bottleneck size (Fig. S15)

In [71]: %%%R

# We calculated the bottleneck size by comparing
# the allele frequency of the minor allele in the mother and her child

bottleneckData = data.frame(
  mc=adjustedMAF$mc,
  mb=quartets[["mbMAF"]],
  cc=adjustedMAF$cc,
  cb=adjustedMAF$cb)

# We used the average of the two tissues in an individual

bottleneckData["meanM"] = apply(bottleneckData[,1:2], 1,mean)
bottleneckData["meanC"] = apply(bottleneckData[,3:4], 1,mean)

# And modeled the bottleneck as in Millar et al, 2008

bottleneckData$bn1 = (bottleneckData$meanM*(1-bottleneckData$meanM))/(bottleneckData$meanC-bottleneckData$meanM)**2
bottleneckData$bn1.cheek = (bottleneckData$mc*(1-bottleneckData$mc))/(bottleneckData$mc-bottleneckData$cc)**2
bottleneckData$bn1.blood = (bottleneckData$mb*(1-bottleneckData$mb))/(bottleneckData$mb-bottleneckData$cb)**2
# Select cases where there is evidence of the minor allele in the maternal lineage (i.e. the minor allele is present in both tissues of the mother, at least 1% in one of the tissues, and 0.2% in the other tissue)

```
bn1.m = bottleneckData[(bottleneckData$mc>=0.01|bottleneckData$mb>=0.01) & (bottleneckData$mc>=0.002 & bottleneckData$mb>=0.002),]["bn1"]
```

```
b1.cheek = bottleneckData[bottleneckData$mc>=0.01,]["bn1.cheek"]
b1.blood = bottleneckData[bottleneckData$mb>=0.01,]["bn1.blood"]
```

In [72]: %%R

```
length(bn1.m)
```

[1] 50

In [73]: %%R

# Accounting for mitotic segregation

```
mitotic = function(row){
  mc = row[1]
  mb = row[2]
  cc = row[3]
  cb = row[4]
  variance = (((mc-cc)^2+(mc-cb)^2+(mb-cc)^2+(mb-cc)^2-2*(mc-mb)^2-2*(cc-cb)^2)/4
  return(variance)
}
```

```
bottleneckData[["mitotvar"]]=apply(bottleneckData,1,mitotic)
bottleneckData$bn2 = (bottleneckData$meanM*(1-bottleneckData$meanM))/(bottleneckData$mitotvar)
bn2.m = bottleneckData[(bottleneckData$mc>=0.01|bottleneckData$mb>=0.01) & (bottleneckData$mc>=0.002 & bottleneckData$mb>=0.002),]["bn2"]
```

In [74]: %%R

# We removed negative or indetermined estimates of the bottleneck

```
bn2.m = bn2.m[bn2.m>0]
```

In [75]: %%R

```
length(bn2.m)
```

[1] 45

In [76]: %%R

```
bn1.m
```

```
[16] 4576.759055 87.090388 7.145162 543.823984 681.303195
```
In [77]: %%R

# Get the actual stats before transforming the data
data_tmp=list(bn1.m, bn2.m)
medians=c()
first=c()
third=c()

for (i in 1:2){
    medians=c(medians, median(unlist(data_tmp[i])))
    first=c(first, summary(unlist(data_tmp[i]))[2])
    third=c(third, summary(unlist(data_tmp[i]))[5])
}

bn1.m = log10(bn1.m)
bn2.m = log10(bn2.m)

In [78]: %%R -w 6 -h 5 -u in -r 144

par(oma=c(0,0,4,0))
par(bty="n")
par(xpd=TRUE)
par(lwd=1.5)
par(pch=20)

# blank boxplot
boxplot(bn1.m,xlim=c(0,5), ylim=c(-1,3), at=1, frame=F, axes=F,
       ylab="", medcol="white", whiskcol="white", boxcol="white", staplecol="white")

# actual drawings
vioplot(bn1.m, ylim=c(-1,3), at=1, col="royalblue", add=T, border=NA)
vioplot(bn2.m, ylim=c(-1,3), at=2, col="tomato", add=T, border=NA, outline=F)

# y-axis
axis(2, at=seq(0,4,1), lab=seq(0,4,1), pos=0, las=2, lwd=1.5, cex.axis=1.5)

# labs
labx = expression(paste("N=", "p(1-p)/\sigma^2", "[gen]"))
laby = expression(paste("log"[10], "(N)"))
mtext(labx,2,1,cex=1.5,adj=1)
legend(3,4,legend=c(labx,"Mitot. Segreg."), fill=c("royalblue", "tomato"), bty="n", border="white")

# Add boxplot stats to vioplot
data_tmp=list(bn1.m, bn2.m)

for (i in c(1,2)) {
  y=round(median(unlist(data_tmp[i])),2)
  text(i+0.175,y,lab=round(medians[i],1),col="white",cex=0.7,font=2)
  y=round(summary(unlist(data_tmp[i]))[2],2)
  text(i,y,lab=round(first[i],1),col="white",cex=0.7,font=2)
  y=round(summary(unlist(data_tmp[i]))[5],2)
  text(i,y,lab=round(third[i],1),col="white",cex=0.7,font=2)
}

3.6 Plot age correlations (Fig. 2)

In [79]: %R

```
countHq = function(row){
  cheek = row[1]
  blood = row[2]
  if(cheek>=0.01|blood>=0.01){
    return(1)
  }else{
```

![Graph showing log10(N) against countHq with points labeled 102.9, 43.4, 13.4, 110.3, 38.7, 10 for Mitot. Segreg. and N=p(1-p)/σ²_gen.]
# count the number of heteroplasmic sites with MAF ≥ 1% (in either tissue) per individual
data = quartets
data$Nmother = apply(quartets[,c(5,8)],1,FUN=countHq)
data$Nchild = apply(quartets[,c(11,14)],1,FUN=countHq)
nsites = aggregate(cbind(Nmother,Nchild) ~ family, data=data,FUN=sum)

# age in days
head(sampAges)

# the age of the mother at the time of conception of the child is assumed to be
# the current age of the mother, less the current age of the child, less nine months (in days 9*30)
ageEffect$motherAgeFertilization = ageEffect$motherAgeCollection - (ageEffect$childAgeCollection + (9*30))
# colors
black = "black"
m_col = "royalblue1"
c_col = "tomato1"

# transparent colors
mother = rgb(matrix(col2rgb(m_col),1,3),alpha=120,maxColorValue=255)
child = rgb(matrix(col2rgb(c_col),1,3),alpha=120,maxColorValue=255)
borders = c(m_col,c_col)

# plot margins
par(oma=c(0,0,0,0))
par(mar=c(0.5,2.1,0,0))

# Mother data only
d = ageEffect[,c("Nmother","motherAgeCollection")]
plot(1:10,1:10,xlim=c(15,60),ylim=c(-1.5,5),type="n",frame=F,axes=F,xlab="",ylab="",main="")
points(d["motherAgeCollection"]/365,d["Nmother"],pch=23,col=borders[1],lwd=1,cex=1,bg=mother)
r3=glm(Nmother~motherAgeCollection,data=d,family="poisson")
p3=round(summary(r3)$coefficients[2,4],3)
fit3=data.frame(age=r3$data$motherAgeCollection/365,f=r3$fitted.values)
fit3=fit3[order(fit3$age),]
lines(fit3$age,fit3$f,col=m_col,lwd=2.5)
x1=min(fit3$age)
x2=max(fit3$age)
pmother=round(summary(r3)$coefficients[2,4],2)

# Child data only
par(new=T)
c = ageEffect[,c("Nchild","motherAgeFertilization")]
points(c["motherAgeFertilization"]/365,c["Nchild"],pch=21,col=borders[2],lwd=1,cex=1,bg=child)
r3=glm(Nchild~motherAgeFertilization,data=c,family="poisson")
fit3=data.frame(con=r3$data$motherAgeFertilization/365,f=r3$fitted.values)
fit3=fit3[order(fit3$con),]
lines(fit3$con,fit3$f,col=c_col,lwd=2.5)
pchild=round(summary(r3)$coefficients[2,4],3)

# labs
axis(side=1,at=seq(15,60,by=5),lab=NA,lwd=2,pos=-0.25,cex.axis=0.75)
axis(side=2,at=0:5,lwd=2,pos=14.5,las=2,cex.axis=0.75)
mtext("number of point heteroplasmies",2,1.3,at=2.5,cex=0.75,font=2)
mtext("maternal age (years)",1,-0.5,cex=0.75,font=2)
for (i in seq(15,60,by=5)){text(i,-0.75,lab=i,cex=0.75)}
M=paste(paste("mother","p=","p="),pmother,sep="")
C=paste(paste("child","p="),pchild,sep="")
legend(15,4.6,legend=c(M,C),col=borders,
      pt.bg=c(mother,child),pch=c(23,21),pt.cex=1.2,pt.lwd=1.2,
      cex=0.6,title="Poisson model",box.col="gray",box.lwd=1.5,bg=NA)
x3 = min(fit3$con)
x4 = max(fit3$con)

lines(c(x1,x2), c(-1,-1), lwd=1.5, col=m_col)
lines(c(x3,x4), c(-1.1,-1.1), lwd=1.5, col=c_col)
mtext("fertilization", 1, -1.25, at=27, cex=0.75, col=c_col, font=2)
mtext("collection", 1, -1.25, at=45, cex=0.75, col=m_col, font=2)