Here is an example of imputation using Beagle 4.0 (reference manual here). This version was chosen because it is possible to use pedigree data to improve imputation accuracy.
The imputation part is pretty straightforward, I would say that the most annoying part is the conversion of your data to the Beagle format, so here I share the scripts I used for that too.

# Data format

### Genotypic data

Beagle requires the VCF format for genotypes. If your data is in the plink format, the conversion is easy but I found the conversion from an Excel or txt file to VCF difficult. Fortunately, the creators of Beagle also wrote a script to convert a txt file to VCF (it is called beagle2vcf, see here).
If your data has the following form :

| | ind1 | ind2 | ... |
|---|---|---|---|
| mk1 | AA | AB | ... |
| mk2 | AA | BB | ... |

Then the required input data for **beagle2vcf** must have this form :

| I | id | ind1 | ind1.1 | ind2 | ind2.1 |
|---|---|---|---|---|---|
| M | mk1 | A | A | A | A |
| M | mk2 | A | B | B | B |
| ... | ... | ... | ... | ... | ... |

I used the script **geno_to_vcf** to obtain this format, which is the Beagle 3.3 genotypes format. I replaced my data with a toy example so you can see how it works and adapt it to your data. You will need one file per chromosome.
In addition to these files, you also need a file per chromosome that indicates the markers of each chromosome, their physical position and the reference/alternative allele, something like that :

| mk | pos | ref | alt |
|---|---|---|---|
| mk1 | 7556 | A | C |
| mk2 | 8635 | T | G |
| ... | ... | ... | ... |

The format is the Beagle 3.3 markers format, more info here.
Your data may be coded in 0/1/2, I'm pretty sure that in this case you will first have to convert the values to the actual alleles otherwise Beagle doesn't work.

## Conversion to the VCF format

Once your files are ready, the conversion to VCF with **beagle2vcf** is easy and takes a few seconds :

```
for chr in $(seq 1 1 17)
do
java -jar beagle2vcf.jar \
chrom=$chr \                              # chromosome number
markers=markers_format_chr$chr.txt \   # the genotypes with the Beagle format
bgl=genotypes_format_chr$chr.txt \     # the markers with the Beagle format
missing=NA \                              # the code for missing data in your files
> out=genotypes_to_impute_chr$chr.vcf \  # the output name
done
```

## Using a reference set for imputation

When imputing from low or medium density to high density, a reference set that has been genotyped at high density has to be specified (if not, Beagle will only impute the missing markers of your data).
This reference set has to be phased and must have no missing data, so a first phasing of the reference set may be required. If the phasing is done with Beagle, missing data will be imputed at the same time.
You can phase your reference set with the following lines of code :

```
for chr in $(seq 1 1 17)
do
java -Xmx2g -jar beagle.r1399.jar \ # max 2Gb of memory allowed for the phasing
gt=your_reference_genotypes_chr$chr.vcf \
ped=pedigree_data.txt \
out=phased_reference_chr$chr &
# The & at the end allows to run the phasing
# for the 17 chromosomes simultaneously
done
```

Your input data has to be in the VCF format and you may use a pedigree file to help Beagle phase the data more accurately.
If you use the **ped** argument, the input pedigree file should have 4 columns corresponding to the Pedigree Id (I don't remember what that means , we set it to -9 for every entry), to name of the individual, the name of the father and the name of the mother (the order doesn't matter for these two columns). For example :

| Id | Ind | P1 | P2 |
|---|---|---|---|
| -9 | 70 | 789 | 25 |
| -9 | 658 | 2658 | 1256 |
| ... | ... | ... | ... |

We observed that using pedigree data improved the imputation accuracy a lot but if you don't use it for phasing or if your pedigree is too shallow I would recommend using Beagle 5, which doesn't use pedigrees and is much faster than the other Beagle versions.
The phasing part is the longest process when doing imputation (I would say between 3 and 6 hours per chromosome).

## Imputation of the data

We used the following command lines for the imputation part :

```
for chr in $(seq 1 1 17)
do
java -Xmx32g -jar beagle.r1399.jar \
ref=phased_reference_chr$chr.vcf \
gt=genotypes_to_impute_chr$chr.vcf \
ped=pedigree_data.txt \    # only if there are duos/trios in your data to impute
out=imputed_results_chr$chr &
done
```

Some other options may be useful, you can check the list of options here

## Output files

After the imputation step, you should obtain one VCF file per chromosome that looks like that :

| CHROM | POS | ID | ... | Ind1 | Ind2 |
|---|---|---|---|---|---|
| 1 | 5688 | mk1 | ... | 0\|1:0.99:0.021,0.967,0.012 | 1\|1:1.956:0.001,0.043,0.957 |
| 1 | 8974 | mk2 | ... | 0\|1:0.992:0.02,0.968,0.012 | 1\|1:1.959:0,0.04,0.959 |
| ... | ... | ... | ... | ... | ... |

So for each imputed individual you obtain the most likely haplotype and the associated probability. If you are not interested in obtaining these probabilities, you can use the **gprobs=false** argument in the imputation step. If you don't have a way to check to accuracy of the imputation, I would recommend getting these probabilities to assess the uncertainty around your imputation.
If you obtain such an output format, you may want to get only the genotypic value at each marker, only the phase at each marker or only the max probability at each marker. To do so, I use these three short functions to extract the results :

```
library(tidyverse)

# Extract the genotype at each marker
phase_to_value <- function(df){

  apply(df, 2, function(x) as.numeric(str_sub(x,
                                              end = 1)) + as.numeric(str_sub(x,
                                                                             start = 3,
                                                                             end = 3)))
}

# Extract the phase at each marker
get_phase <- function(df){

  apply(df, 2, function(x) str_sub(x,
                                   end = 3))
}
```

```r
# Extract probabilities at each marker
get_proba <- function(df){

  temp_df <- apply(df,
                   2,
                   function(x) str_split(x, ":") %>% unlist)
  temp_df <- temp_df[3 * 1:nrow(df), ] %>% apply(.,
                                                 2,
                                                 function(x) str_split(x, ","))

  return(lapply(temp_df,
                sapply,
                function(x) x %>%
                            as.numeric %>%
                            max) %>%
         Reduce(cbind,
                .))

}

toy_df <- data.frame(Ind1 = c("0|1:0.99:0.021,0.967,0.012",
                              "0|1:0.992:0.02,0.968,0.012"),
                     Ind2 = c("1|1:1.956:0.001,0.043,0.957",
                              "1|1:1.959:0,0.04,0.959"))

phase_to_value(toy_df)
```

```
##      Ind1 Ind2
## [1,]    1    2
## [2,]    1    2
```

```r
get_phase(toy_df)
```

```
##      Ind1  Ind2
## [1,] "0|1" "1|1"
## [2,] "0|1" "1|1"
```

```r
get_proba(toy_df)
```

```
##       init
## [1,] 0.967 0.957
## [2,] 0.968 0.959
```