In [1]:
```
import os
from itertools import *
from collections import *
import random
from sys import *

sample_names = 'sample_stacks_codes_noBGMN.txt' #a file with all the sample names, populations and
STACKS codes
infile = 'noBG_20x.fa' #input fasta file from stacks
MissThrs = 4 #minimum number of regions per locus

#create dictionary for the sample, population and region names
#CAUTION. FOR EACH FA FILE PRODUCED BY STACKS (IN CASE DIFFERENT INDIDUALS ARE EXCLUDED), A NEW SAM
PLE ID MAP SHOULD BE CREATED
with open(sample_names) as f:
    sample_id = {}
    pop_id = {}
    pop_indx = {}
    reg_id = {}
    for line in f:
        a = line.split()
        sample_id[a[0]] = a[1]
        pop_id[a[0]] = a[3]
        reg_id[a[0]] = a[4]

#create a dictionary to store the dictionaries of samples and sequences (both alleles)
loci_d = {}

#create a dictionary to store allele frequencies per region and Locus and distinct alleles per Locu
s
lociDall = {}

PrevAlleleState = ""
with open(infile) as f:
    while True:
        next_n_lines = list(islice(f, 2))
        if not next_n_lines:
            break
        #get only the Locus ID
        locus_ID = next_n_lines[0].split('_')[1]
        #get only the Sample ID
        sample_ID = next_n_lines[0].split('_')[3]
        #get only region/population name
        RegionName = reg_id[str(sample_ID)]
        #get only allele state
        AlleleState = next_n_lines[0].rstrip().split('_')[7]
        #get only the locus sequence
        seq = next_n_lines[1].rstrip()
        readLen = len(seq)


        #now store the sequence to the dictionary of dictionaries of sets (for unique alleles)
        loci_d.setdefault(locus_ID,{}).setdefault(RegionName, set()).add(seq)
        #store all different alleles per Locus
        lociDall.setdefault(locus_ID,{}).setdefault('AllAlleles', set()).add(seq)

        if PrevAlleleState == '0' and AlleleState == '1':
            #Locus heterozygous
            lociDall.setdefault(locus_ID,{}).setdefault(RegionName, Counter())[PrevSeq] += 1
            lociDall.setdefault(locus_ID,{}).setdefault(RegionName, Counter())[seq] += 1
        elif PrevAlleleState == '0' and AlleleState == '0':
            #previous locus homozygous
            lociDall.setdefault(PrevLocus_ID,{}).setdefault(PrevRegion, Counter())[PrevSeq] += 2

        #make current names previous for the storage
        PrevAlleleState = AlleleState
        PrevSeq = seq
        PrevRegion = RegionName
        PrevLocus_ID = locus_ID


    #end of file
    if AlleleState == '0':
        lociDall.setdefault(locus_ID,{}).setdefault(RegionName, Counter())[seq] += 2
```

Apply rarefaction function (Kalinowski 2004)

- Number of copies of the *i*th allele in the sample from the *j*th population:

$$N_{ij}$$

- Total number of genes/haploid individuals sampled from the *j*th population:

$$N_j$$

- Total number of distinct alleles observed at the given locus:

$$m \quad \ni \quad N_j = \sum_{i=1}^{m} N_{ij}$$

- Probability that a sample of *g* genes/haploid individuals taken from a sample of N_j genes/haploid individuals will not contain allele *i*:

$$Q_{ijg} = \frac{\binom{N_j - N_{ij}}{g}}{\binom{N_j}{g}} = \prod_{u=0}^{g-1} \frac{N_j - N_{ij} - u}{N_j - u}$$

and

$$P_{ijg} = 1 - Q_{ijg}$$

- Allelic richness of the *j*th population for a particular locus, $\alpha_g^{(j)}$:

$$\hat{\alpha}_g^{(j)} = \alpha_g^{(j)} = \sum_{i=1}^{m} P_{ijg}$$

- Private allelic richness

$$\hat{\pi}_g^{(j)} = \sum_{i=1}^{m} \left[P_{ijg} \left(\prod_{\substack{j'=1 \\ j' \neq j}}^{J} Q_{ij'g} \right) \right]$$

In [2]:
```
import numpy

#get the allelic richness per Locus for each region for the range 2 to Maximum G (the minimum sampl
e size per locus and region)
#terminology: g = standardized sample size, u = sample size range, i = allele type, Nregion = total
sample size of the region
def RegionRichVector(SampleThres, AlleleFreq):
    AlphaDict = {}
    for g in range(2,SampleThres+1):
        for regio in AlleleFreq.keys():
            Nregion = sum(list(allFreqRegion[regio]))
            alphaReg = 0
            for i, alleleFreq in enumerate(AlleleFreq[regio]):
                Qi = numpy.prod([(Nregion-alleleFreq-u)/(Nregion-u) for u in range(g)])
                Pi = 1 - Qi
                alphaReg += Pi
            AlphaDict.setdefault(g,{})[regio] = alphaReg
    return(AlphaDict)
#get the private allelic richness
def PrivateAllelicRich(SampleThres, RegAlleles):
    GlobalPI = {}
    for g in range(2, SampleThres+1):
        RegionQ = Counter()
        for i in RegAlleles['AllAlleles']:
            QDict = {}
            for region, AllFreq in RegAlleles.items():
                if region != 'AllAlleles':
                    Nregion = sum([freq for freq in AllFreq.values()])
                    alleleFreq  = AllFreq[i]
                    #calculate Q, the probabilty that the region doesn't contain allele x
                    Qi = numpy.prod([(Nregion-alleleFreq-u)/(Nregion-u) for u in range(g)])
                    QDict[region] = Qi
            #iterate over the dictionary of Q probabilities to calculate PI for the particular samp
le size and allele
            for region, Q in QDict.items():
                P = 1 - Q
                Qproduct = numpy.prod([Q for reg, Q in QDict.items() if reg != region])
                pi = P * Qproduct
                RegionQ[region] += pi
        GlobalPI[g] = RegionQ
    return(GlobalPI)
```

Run the two functions

In [3]:
```
SampleThres = 20
MissThrs = 4
GlobalAlpha = {}
GlobalPI = {}
GlobalS = {}
LocN = 0
LocList = []
for locus, RegAlleles in lociDall.items():
    allFreqRegion = {}
    for region, v in RegAlleles.items():
        if region != 'AllAlleles':
            allFreqRegion[region] = list(v.values())
    #get loci that occur in all regions and with a minimum sample size (chromosomes) above the thre
shold
    if min([sum(i) for i in list(allFreqRegion.values())]) >= SampleThres and len(allFreqRegion) >=
MissThrs:
        LocN += 1
        LocList.append(locus)
        AlleleFreq = allFreqRegion
        alpha = RegionRichVector(SampleThres, AlleleFreq)
        GlobalAlpha[locus] = alpha

        pi = PrivateAllelicRich(SampleThres, RegAlleles)
        GlobalPI[locus] = pi
```

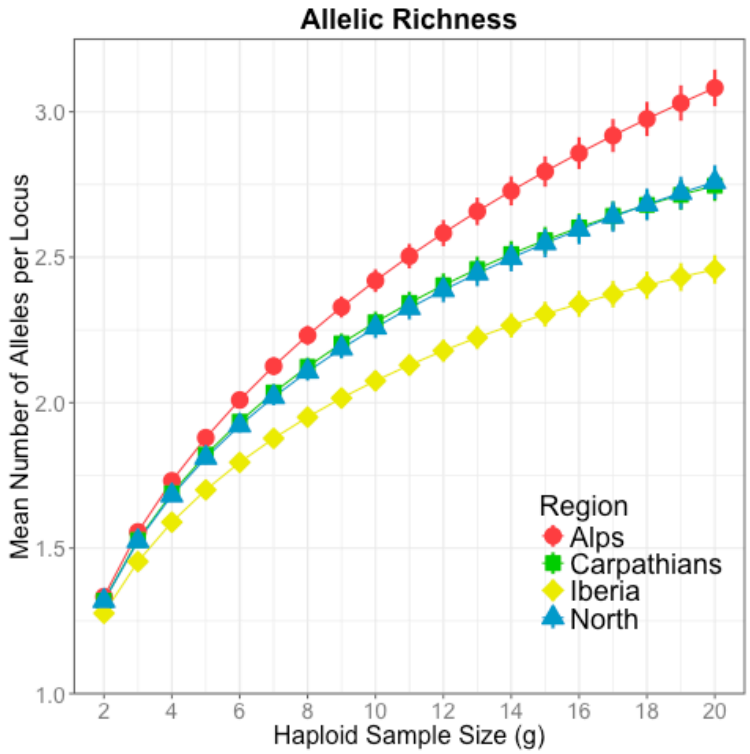
In [4]:
```
from scipy import stats
#create data file for visualization
groups = list(set(list(reg_id.values())))
groups.sort()
GroupComp = list(combinations(groups, 2))
All_Alpha = []
All_AlphaSE = []

All_Pi = []
All_PiSE = []

#All_S = []
for Gname in groups:
    for g in range(2, SampleThres+1):
        la = []
        laSE = []
        lpi = []
        lpiSE = []

        la.append(Gname)
        laSE.append(Gname)
        lpi.append(Gname)
        lpiSE.append(Gname)
        la.append(g)
        laSE.append(g)
        lpi.append(g)
        lpiSE.append(g)

        a = numpy.mean([alpha for locus, v in GlobalAlpha.items() for number, regio in v.items() fo
r name, alpha in regio.items() if name == Gname  and number == g])
        aSE = stats.sem([alpha for locus, v in GlobalAlpha.items() for number, regio in v.items() f
or name, alpha in regio.items() if name == Gname  and number == g])

        pi = numpy.mean([alpha for locus, v in GlobalPI.items() for number, regio in v.items() for
name, alpha in regio.items() if name == Gname  and number == g])
        piSE = stats.sem([alpha for locus, v in GlobalPI.items() for number, regio in v.items() for
name, alpha in regio.items() if name == Gname  and number == g])

        la.append(a)
        laSE.append(aSE)
        lpi.append(pi)
        lpiSE.append(piSE)

        All_Alpha.append(la)
        All_AlphaSE.append(laSE)

        All_Pi.append(lpi)
        All_PiSE.append(lpiSE)
```

In [5]:
```
import pandas as pd
dfA = pd.DataFrame(All_Alpha, columns=['Region', 'G', 'Alpha'])
dfA.to_csv('AllRichness.txt', sep='\t', index = False)

dfA_SE = pd.DataFrame(All_AlphaSE, columns=['Region', 'G', 'AlphaSE'])
dfA_SE.to_csv('AllRichnessSE.txt', sep='\t', index = False)

dfPi = pd.DataFrame(All_Pi, columns=['Region', 'G', 'Pi'])
dfPi.to_csv('PrivAll.txt', sep='\t', index = False)

dfPiSE = pd.DataFrame(All_PiSE, columns=['Region', 'G', 'PiSE'])
dfPiSE.to_csv('PrivAllSE.txt', sep='\t', index = False)
```

Visualize data using R

In [6]:
```
%load_ext rmagic
```

In [7]:
```
%%R
library(ggplot2)
data = read.table('AllRichness.txt', sep = '\t', h = T)
dataSE = read.table('AllRichnessSE.txt', sep = '\t', h = T)

colours = c('brown1', 'green3', 'yellow2', 'DeepSkyBlue3', 'grey30',' grey60')
#get number of populations for the colour assignment in the plot
pop_num = length(unique(data$Region))

dataComb = cbind(data, dataSE[3])

#plot data
p = ggplot(dataComb,aes(G,Alpha)) + geom_line(aes(colour = Region)) + geom_pointrange(aes(x=G, ymin
=Alpha-AlphaSE, ymax=Alpha+AlphaSE, colour = Region, shape = Region, fill = Region), size = 1) +
scale_color_manual(values = colours[1:pop_num]) + scale_fill_manual(values = colours[1:pop_num]) +
scale_shape_manual(values=c(21:24)) + scale_x_continuous("Haploid Sample Size (g)", breaks = c(seq(
2, 20, by = 2))) + scale_y_continuous("Mean Number of Alleles per Locus", breaks = c(seq(0, 3, by =
0.5)), limits = c(1,3.25), expand = c(0.00125,0)) + theme(plot.title = element_text(vjust=1, size =
18), axis.title.y=element_text(vjust=0.35, size = 16), axis.title.x=element_text(vjust=0.35, size =
16) ,axis.text.x = element_text(size=14, colour = "gray50"), axis.text.y = element_text(size=14,
colour = "gray50"), axis.ticks = element_line(colour = 'gray40')) + theme(plot.title = element_text
(size = 18, face="bold"), panel.border = element_rect(fill =NA, colour = "gray40", size=1),
panel.grid.major=element_line(colour = "gray93"), panel.grid.minor = element_line(colour = "gray93"
), panel.background = element_blank(), plot.background = element_blank()) + labs(title='Allelic Ric
hness') + theme(legend.position=c(0.83,0.2),legend.title = element_text(size = 18), legend.text =
element_text(size = 18), legend.background = element_blank(), legend.key = element_blank())

ggsave(p,file = "AllRich.pdf", bg = "transparent", height=7, width=7)

p
```



In [8]:
```
%%R
library(ggplot2)
data = read.table('PrivAll.txt', sep = '\t', h = T)
dataSE = read.table('PrivAllSE.txt', sep = '\t', h = T)

colours = c('brown1', 'green3', 'yellow2', 'DeepSkyBlue3', 'grey30',' grey60')
#get number of populations for the colour assignment in the plot
pop_num = length(unique(data$Region))

dataComb = cbind(data, dataSE[3])

#plot data
p = ggplot(dataComb,aes(G,Pi))+ geom_line(aes(colour = Region)) + scale_color_manual(values =
colours[1:pop_num]) + geom_pointrange(aes(x=G, ymin=Pi-PiSE, ymax=Pi+PiSE, colour = Region, shape =
Region, fill = Region), size=1) + scale_fill_manual(values = colours[1:pop_num]) +
scale_shape_manual(values=c(21:24)) + scale_x_continuous("Haploid Sample Size (g)", breaks = c(seq(
2, 20, by = 2))) + scale_y_continuous("Mean Number of Private Alleles per Locus", breaks = c(seq(0.
2, 1, by = 0.2)), limits = c(0.2,1.1), expand = c(0.00125,0)) + theme(plot.title = element_text(vju
st=1, size = 16), axis.title.y=element_text(vjust=0.35, size = 16), axis.title.x=element_text(vjust
=0.35, size = 16) ,axis.text.x = element_text(size=14, colour = "gray50"), axis.text.y =
element_text(size=14, colour = "gray50"), axis.ticks = element_line(colour = 'gray40')) + theme(plo
t.title = element_text(size = 18, face="bold"), panel.border = element_rect(fill =NA, colour = "gra
y40", size = 1), panel.grid.major=element_line(colour = "gray93"), panel.grid.minor = element_line(
colour = "gray93"), panel.background = element_blank(), plot.background = element_blank()) + labs(t
itle='Private Allelic Richness') + theme(legend.position=c(0.83,0.2),legend.title = element_text(si
ze = 18), legend.text = element_text(size = 18), legend.background = element_blank(), legend.key =
element_blank())
ggsave(p,file = "PrivRich.pdf", bg = "transparent", height=7, width=7)
p
```