Demultiplexing and data processing were performed on CentOS 7.9.2009
Data analyses were performed on Ubuntu 22.04 and MacOS

Demultiplexing

Sequencing was done at the University of Wisconsin, Madison Biotechnology Center. We used double-digest restriction site-associated DNA sequencing (ddRADseq) with the enzymes EcoRI and MseI.

FASTQ files were partially demultiplexed into rows A-H, with each file containing columns 1-10 (11 and 12 were not filled in our plate). Each of the eight FASTQ files was demultiplexed using the same set of column barcodes to identify individual samples. We used `stacks 2.5 process_radtags` to demultiplex the files with the `--inline_null` flag. Files were demultiplexed into rows with combinatorial barcodes in the FASTQ headers, and then demultiplexed into individual samples with inline barcodes unique to each column on the forward reads.

```
for i in {A..H}
do

/location/of/stacks-2.5/process_radtags -P -p /path/to/multiplexed/files/ -o
/path/to/fastqs/ -b ./row$i-barcodes.txt -c -q -r --inline-null --renz-1 ecoRI
--renz-2 bfaI

done
```

To import the demultiplexed FASTQ files into ipyrad, they need to renamed from `file.1.fq.gz` to `file_R1_.fq.gz`. The `rename` command on Linux systems is a good way to do this.

```
rename .1. _R1_. *.fq.gz
rename .2. _R2_. *.fq.gz
```

Data processing

Following demultiplexing, we used the software package ipyrad 0.9.84 to process raw reads.

Note:
Conda has had some issues (at least on Ubuntu) with environment solving. This update has been very helpful:

```
conda update -n base conda
conda install -n base conda-libmamba-solver
```

```
conda config --set solver libmamba
```

It can be helpful to install ipyrad in a new conda environment, to avoid conflicts with other programs.

```
conda init
conda create -n ipyrad
conda activate ipyrad
conda install ipyrad -c bioconda -c conda-forge
```

We ran the data processing on 12 cores using the multithread option in ipyrad.

```
ipyrad -p params-sa-pter.txt -s 1234567 -c 12 --MPI
```

We used the following parameters for data processing. A majority are the default parameters suggested by the ipyrad documentation. A few notable changes are:

- [14] clustering threshold
- [21] minimum samples per locus - we altered this number based to slightly reduce the amount of missing data retained in the final dataset [add more details later]

```
------- ipyrad params file (v.0.9.84)-------------------------------------
sa-pter                        ## [0] [assembly_name]: Assembly name. Used
                               to name output directories for assembly
                               steps
/data/sylvia/sa-pteridium/pteridium/ipyrad  ## [1] [project_dir]: Project
                               dir (made in curdir if not present)
                               ## [2] [raw_fastq_path]: Location of raw
                               non-demultiplexed fastq files
                               ## [3] [barcodes_path]: Location of barcodes
                               file
/data/sylvia/sa-pteridium/fastqs/*.fq.gz ## [4] [sorted_fastq_path]:
                               Location of demultiplexed/sorted fastq files
denovo                         ## [5] [assembly_method]: Assembly method
                               (denovo, reference)
                               ## [6] [reference_sequence]: Location of
                               reference sequence file
pairddrad                      ## [7] [datatype]: Datatype (see docs): rad,
                               gbs, ddrad, etc.
AATT, ATA                      ## [8] [restriction_overhang]: Restriction
                               overhang (cut1,) or (cut1, cut2)
5                              ## [9] [max_low_qual_bases]: Max low quality
                               base calls (Q<20) in a read
```

```
33                              ## [10] [phred_Qscore_offset]: phred Q score
                                offset (33 is default and very standard)
6                               ## [11] [mindepth_statistical]: Min depth
                                for statistical base calling
6                               ## [12] [mindepth_majrule]: Min depth for
                                majority-rule base calling
10000                           ## [13] [maxdepth]: Max cluster depth within
                                samples
0.85                            ## [14] [clust_threshold]: Clustering
                                threshold for de novo assembly
0                               ## [15] [max_barcode_mismatch]: Max number
                                of allowable mismatches in barcodes
2                               ## [16] [filter_adapters]: Filter for
                                adapters/primers (1 or 2=stricter)
35                              ## [17] [filter_min_trim_len]: Min length of
                                reads after adapter trim
2                               ## [18] [max_alleles_consens]: Max alleles
                                per site in consensus sequences
0.05                            ## [19] [max_Ns_consens]: Max N's (uncalled
                                bases) in consensus
0.05                            ## [20] [max_Hs_consens]: Max Hs
                                (heterozygotes) in consensus
4                               ## [21] [min_samples_locus]: Min # samples
                                per locus for output
0.2                             ## [22] [max_SNPs_locus]: Max # SNPs per
                                locus
8                               ## [23] [max_Indels_locus]: Max # of indels
                                per locus
0.5                             ## [24] [max_shared_Hs_locus]: Max #
                                heterozygous sites per locus
0, 0, 0, 0                      ## [25] [trim_reads]: Trim raw read edges
                                (R1>, <R1, R2>, <R2) (see docs)
0, 0, 0, 0                      ## [26] [trim_loci]: Trim locus edges (see
                                docs) (R1>, <R1, R2>, <R2)
p, s, l                         ## [27] [output_formats]: Output formats
                                (see docs)
                                ## [28] [pop_assign_file]: Path to
                                population assignment file
                                ## [29] [reference_as_filter]: Reads mapped
                                to this reference are removed in step 3
```

After processing each row, I removed samples that had less that 1000 loci retiatined. I think combined all samples and re-ran ipyrad, and removed samples that had less than ~3000 loci retained. This higher threshold was due to the fact that most other samples had very good recovery, of well over 10,000 loci.

Combining worldwide dataset from Wolf et al. 2019

We used very similar parameters for ipyrad with this worldwide dataset as the South American samples, with a few exceptions:

- [7] datatype: these data were sequenced as double digest radseq, but the reads have already been paired using a custom script. Therefore, to read them into ipyrad we had to process them like a single-end read
- [8] restriction overhang: As stated above, the reads have already been paired and trimmed by custom scripts. So, we did not need to account for restriction overhang and it was left out.

We first loaded in the worldwide dataset, and then merged it with the South American data. Then, we removed all samples that had fewer than ~1000 loci in the final clustering.

```
ipyrad -p params-worldwide.txt -s 12

ipyrad -m ww-sa params-worldwide.txt params-southam.txt

ipyrad -p params-ww-sa.txt -s 34567

# removing all samples with fewer than ~1000 loci
ipyrad -p params-ww-sa.txt -b ww-sa2 - 007KLPH Wolf376 Wolf387 Wolf795
001AMSL 040CONT 031BRFR 141PWLT 251PFNT 293CPHW 321KISA AZ01 HM05 IJ_2048
MALW N2nR -f

ipyrad -p params-ww-sa2.txt -s 34567

# merged south american fastq samples, as the worldwide samples were
already merged. We were having some issue with the data and wanted to try
this
parallel "vsearch --fastq_mergepairs {}_R1_.fastq.gz --reverse
{}_R2_.fastq.gz --fastqout {}.fastq --fastq_allowmergestagger" :::::
spp-list.txt

ipyrad -p params-worldwide2.txt -b worldwide3 - 001AMSL 031BRFR 040CONT
085YSCH 141PWLT 226LBSP 251PFNT 293CPHW 321KISA AZ01 HM05 IJ_2048 MALW N2nR
Wolf_1002
```

Phylogenetic reconstruction

We used RAxML to build a phylogeny for the samples. In short, geography was a better predictor of phylogeny than morphological identification.

```
raxmlHPC -s sa-pter.phy -n sa-pter.out -m GTRCAT -f a -x $RANDOM -N 100 -p
$RANDOM -T 16
```

Structure

We ran 50 chains each for K=2-6
Mainparams and extraparams files were left at default setting
-L # loci
-N # inds

```
for i in {2..6}
do
    for j in {1..50}
do
        structure -K $i -L 42147 -N 47 -i sa-pter.ustr -o
out/sa-pter$i_$j.txt
    done
done
```

Structure output files were run through the processing software CLUMPAK

Plotting

```
# load in ks from clummpack files
k2 <- read.csv("k2.txt", sep = '', header = F)
k3 <- read.csv("k3.txt", sep = '', header = F)
k4 <- read.csv("k4.txt", sep = '', header = F)
k5 <- read.csv("k5.txt", sep = '', header = F)
k6 <- read.csv("k6.txt", sep = '', header = F)
k7 <- read.csv("k7.txt", sep = '', header = F)

# names file includes individual ids, species names, and geographic
locations
names <- read.csv("names.csv", sep = ',', header = T)

x <- as.data.frame(matrix(ncol = 17, nrow = 47))
x[,1:2] <- k2
```

```
x[,3:5] <- k3
x[,6:9] <- k4
x[,10:14] <- k5
x[,15] <- names[,1]
x[,16] <- names[,2]

# order by species
x <- x[order(x[,16]),]

## find locations for each column, each species
## variables
# labels - species names
# xlabels - specimen labels
# ninds - number of individuals
# klist - list of x values

labels <- x[,16]
x_labels <- x[,15]
ninds <- 47
klist <- list(x[,1:2], x[,3:5], x[,6:9], x[,10:14])

# each unique species names
sp.names <- as.character(unique(labels))

# locations of each column, found via barplot column locations
b <- as.data.frame(matrix(ncol = 1, nrow = ninds))
b[,1] <- barplot(t(klist[[1]][1]), beside = F, col = c('black', 'white'),
cex.name = 1, cex.axis = 1.2, border = 1, space = 0.05, xaxt = 'n', yaxt =
'n', cex.lab = 1, cex.main = 2)

# find locations for labels in the barplot
my.mean <- tapply(X = b[,1], INDEX = labels, mean)
my.min <- tapply(X = b[,1], INDEX = labels, min)
my.max <- tapply(X = b[,1], INDEX = labels, max)

# data frame for plotting
d <- sp_labels(names = sp.names, min = my.min, mean = my.mean, max =
my.max)

plot_chains_species(kqlist = klist, xlabel = x_labels)

plot_chains_ids(kqlist = klist, xlabel = labels)
```

```
#########################################################
# functions needed
#########################################################

# plotting and labeling function

# create labels
sp_labels <- function(names, min, mean, max, ...){
    d <- as.data.frame(matrix(nrow = length(names), ncol = 4))
    colnames(d) <- c("names", "min", "mean", "max")
    for (j in 1:length(names)){
            d[j,1] <- names[j]
            d[j,3] <- min[[j]][1]
            d[j,2] <- mean[[j]][1]
            d[j,4] <- max[[j]][1]
    }
    return(d)
}

# plot chains
plot_chains_ids <- function(kqlist, xlabel){
    # define colors
    cols <- c('pink','#5928ED', '#00BF7D','#FFFF00', '#69C261', '#FF59AC',
'#26CDCD',  '#C1C6FF')

    par(mfrow = c(length(kqlist),1), mar = c(1,3,3,1) + 0.1, oma =
c(15,0,0,0), mgp = c(1,1,0))
    chain <- seq(1, length(kqlist), 1)

    # plot ks
    for(i in 1:(length(kqlist)-1)){
        barplot(t(kqlist[[i]]), beside = F, col = cols, border = 1, space =
0.05, xaxt = 'n', yaxt = 'n', ylab = paste("k =", chain[i]+1, sep = ' '),
cex.lab = 1.2, cex.main = 1.6)
        # y axis
        axis(2, at = c(0, 0.25, 0.5, 0.75, 1), cex.axis = 1, las = 2, pos =
-0.2)

        # lines
        for (i in 1:length(d[,1])){
            lines(x = d[i,3:4] , y = rep(-0.09, 2), lwd = 2.5, col =
"black", xpd = NA)
            #lines(x = d[i,3:4] , y = rep(1.1, 2), lwd = 2.5, col =
```

```
"black", xpd = NA, )
        }
    }

    # plot last k with labels
    for(i in length(kqlist)){
        barplot(t(kqlist[[i]]), beside = F, col = cols, border = 1, space =
0.05, yaxt = 'n', ylab = paste("k =", chain[i]+1, sep = ' '), cex.lab =
1.1, cex.main = 1.6, names.arg = xlabel, las = 2)
        # y axis
        axis(2, at = c(0, 0.25, 0.5, 0.75, 1), cex.axis = 1, las = 2, pos =
-0.2)

        # lines
        for (i in 1:length(d[,1])){
            lines(x = d[i,3:4] , y = rep(-0.09, 2), lwd = 2.5, col =
"black", xpd = NA, )
        }
    }
}

plot_chains_species <- function(kqlist, xlabel){

    # define colors
    cols <- c('#A8FFFD','#B862D3', '#A39D9D','#FFFF00', '#69C261',
'#FF59AC', '#26CDCD',  '#C1C6FF')
    #cols <- c('#000075', '#E6194B', '#AAFFC3', '#FFE119', '#F58231',
'#3CB44B')

    par(mfrow = c(length(kqlist),1), mar = c(1,3,3,1) + 0.1, oma =
c(15,0,0,0), mgp = c(1,1,0))
    chain <- seq(1, length(kqlist), 1)

    for(i in 1:length(kqlist)){
        barplot(t(kqlist[[i]]), beside = F, col = cols, border = 1, space =
0.05, xaxt = 'n', yaxt = 'n', ylab = paste("k =", chain[i]+1, sep = ' '),
cex.lab = 1.2, cex.main = 1.6)
        # y axis
        axis(2, at = c(0, 0.25, 0.5, 0.75, 1), cex.axis = 1, las = 2, pos =
-0.2)

        # lines
        for (i in 1:length(d[,1])){
```

```
            lines(x = d[i,3:4] , y = rep(-0.1, 2), lwd = 2.5, col =
"black", xpd = NA)
        }
    }
    # labels
    text(cex = 1.3, x = (d[,2]-0.3), y = -0.7, labels = d[,1], xpd=NA,
srt=50, font=3)
}

################################
# saving pdfs
################################

pdf('k2-6_ids.pdf', width = 10, height = 9)
plot_chains_ids(kqlist = klist, xlabel = x_labels)
dev.off()
```

Principle component analysis

Code taken from the [ipyrad analysis toolkit]

```
import ipyrad.analysis as ipa
import pandas as pd
import toyplot

################################
# South American samples
################################

data = "sa-pter.snps.hdf5"

imap = {
"pop1": ["arach169-4", "arach169-8"],
"pop4": ["arach1685-4", "arach1685-6"],
"pop5": ["arach1629-1", "arach1670-1"],
"pop6": ["arach1611-1", "arach1611-3"],
"pop7": ["arach205-10", "arach205-7"],
"pop8": ["arach1293-10"],
"pop9": ["gry509-10"],
"pop10": ["arach1133-2", "arach1141-6"],
"pop11": ["arach410-1", "arach379-1", "arach379-8"],
```

```
"pop12": ["gry404-3", "gry404-5"],
"pop14": ["arach437-1", "arach437-9"],
"pop15": ["arach457-6"],
"pop19": ["gry290"],
"pop27": ["harp1623", "harpP2"],
"pop35": ["arach292"],
"pop36": ["arach905"],
"pop40": ["arach2128", "paedo3844"],
"pop47": ["arach8241"],
"pop49": ["paedo2570"],
}

# removed because these were very different than the rest of the samples
"pop13": ["camp3785"],
"Out1": ["083WAWA"],
"Out3": ["wolf-1002"],

minmap = {i: 0.5 for i in imap}

pca = ipa.pca(
    data=data,
    imap=imap,
    minmap=minmap,
    mincov=0.75,
    impute_method="sample",
)

pca.run()

df = pd.DataFrame(pca.pcaxes[0], index=pca.names)

# write the PC axes to a CSV file
df.to_csv("pca_analysis2.csv")

# show the first ten samples and the first 10 PC axes
df.iloc[:10, :10].round(2)

# plot PC axes 0 and 2
# this doesnt work for me for some reason
pca.draw(0, 2);

# save to file
pca.draw(outfile="2pca-sa-pter.pdf")

################################
# Worldwide + South American samples
################################

data = "ww-sa2.snps.hdf5"
# all samples
imap = {
"Caudatum": ["238HECR", "274QCOL", "328CJPN", "MD2_2VENZ", "Wolf_1002"],
```

```
"Semihaustatum": ["278KMAL", "281MLNT"],
"Aquilinum": ["017FZSW", "029TTWN", "065CLYW", "068SERI", "071AIJP", "085YSCH", "096GNCH",
"099EDSC", "100AOUS", "103ASSP", "104MOGJ", "106OXLN", "110KUFZ", "112ZAMB_CH3Z", "113YOJP",
"143YCCM", "147WMCH", "148BRMN", "164KUKR", "169FLUS", "182PMAL", "188CORF", "191GSAF",
"194AMTK", "202RVIT", "203HFLA", "217GCOM", "218ACAW", "226LBSP", "228MCAM", "256AZOR",
"280CHJP", "292MAHI", "305YGIN", "307BRGW", "316SHJP", "325OWUS", "336RAMD", "350NIFR",
"353BBSA", "354WFNQ", "362HKSL", "368NDNG", "416TREV", "503CAPM_ZOMA", "AD02", "AKFZ",
"Barrington2287", "CAF4", "CP73", "Der66", "Der67", "Der68", "HM04", "M1FE", "M3FE", "M5FE",
"N6nR", "NIS01", "RU4K", "TAUR", "Wolf921", "Wolf921"],
"SouthAmEsculentum": ["arach169-4", "arach169-8", "arach1685-4", "arach1685-6",
"arach1629-1", "arach1670-1", "arach1611-1", "arach1611-3", "arach205-10", "arach205-7",
"arach1293-10", "gry509-10", "arach1133-2", "arach1141-6", "arach410-1", "arach379-1",
"arach379-8", "gry404-3", "gry404-5", "arach437-1", "arach437-9", "arach457-6", "gry290",
"harp1623", "harpP2", "arach292", "arach905", "arach2128", "paedo3844", "arach8241",
"paedo2570"],
"Esculentum": ["026OPBR", "037TUBN", "083WAWA", "127KEWA", "144RMEX", "213CRAN", "275KONC",
"317SPBR", "324HNNZ", "332SVNZ", "387CRDQ", "401RYNA", "IJ_1245", "ME2_1VNZA", "Wolf_1001",
"Wolf638"],
}


# subset of samples, including all south american samples

data = "subset.snps.hdf5"

imap = {
"e-SouthAm": ["arach169-4", "arach169-8", "arach1685-4", "arach1685-6", "arach1629-1",
"arach1670-1", "arach1611-1", "arach1611-3", "arach205-10", "arach205-7", "arach1293-10",
"gry509-10", "arach1133-2", "arach1141-6", "arach410-1", "arach379-1", "arach379-8",
"gry404-3", "gry404-5", "arach437-1", "arach437-9", "arach457-6", "gry290", "harp1623",
"harpP2", "arach292", "arach905", "arach2128", "paedo3844", "arach8241", "paedo2570",
"026OPBR", "317SPBR", "144RMEX", "387CRDQ", "401RYNA", "IJ_1245", "Wolf_1001", "ME2_1VNZA"],
"caudatum": ["238HECR","274QCOL", "328CJPN", "MD2_2VENZ", "Wolf_1002"],
"Aquilinum": ["354WFNQ", "M1FE", "M3FE", "M5FE"],
}

"e-SouthAm": ["arach169-4", "arach169-8", "arach1685-4", "arach1685-6", "arach1629-1",
"arach1670-1", "arach1611-1", "arach1611-3", "arach205-10", "arach205-7", "arach1293-10",
"gry509-10", "arach1133-2", "arach1141-6", "arach410-1", "arach379-1", "arach379-8",
"gry404-3", "gry404-5", "arach437-1", "arach437-9", "arach457-6", "gry290", "harp1623",
"harpP2", "arach292", "arach905", "arach2128", "paedo3844", "arach8241", "paedo2570"],
"e-brazil": ["026OPBR", "317SPBR"],
"e-mexico": ["144RMEX"],
"e-australia": ["387CRDQ", "401RYNA"],
"e-bolivia": ["IJ_1245"],
"e-ecuador": ["Wolf_1001"],
"e-venezuela": ["ME2_1VNZA"],

minmap = {i: 0.1 for i in imap}

pca = ipa.pca(
    data=data,
```

```
    imap=imap,
    minmap=minmap,
    mincov=0.25,
    impute_method=3,
)

pca.run()
pca.draw(outfile="sapter2-all.pdf")
```

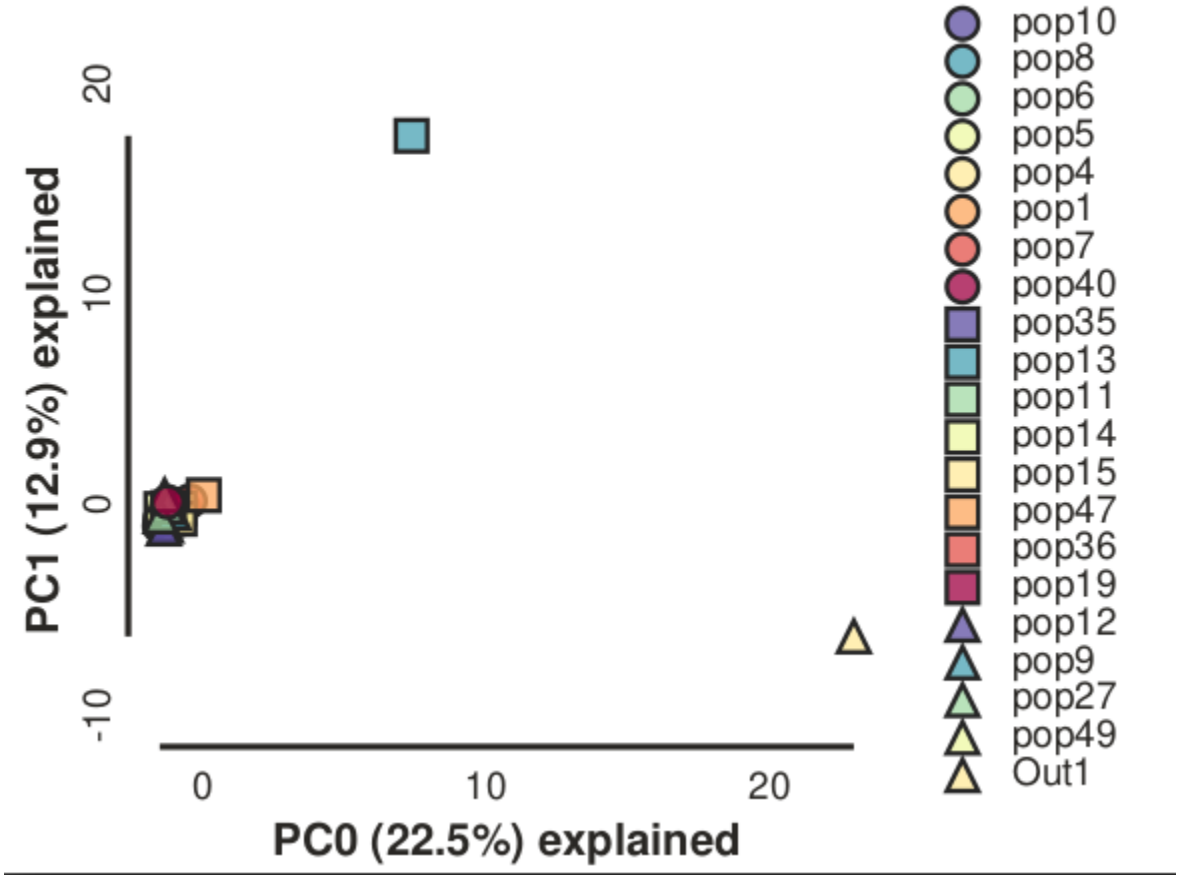
```
imap = {
"arachnoideum": ["arach169-4", "arach169-8", "arach1685-4", "arach1685-6", "arach1629-1",
"arach1670-1", "arach1611-1", "arach1611-3", "arach205-10", "arach205-7", "arach1293-10",
"arach1133-2", "arach1141-6", "arach410-1", "arach379-1", "arach379-8", "arach437-1",
"arach437-9", "arach457-6", "arach292", "arach905", "arach2128", "arach8241"],
"gryphus": ["gry509-10", "gry404-3", "gry404-5", "gry290"],
"harpianum": ["harp1623", "harpP2"],
"paedomorficum": ["paedo3844", "paedo2570"],

"campestre": ["camp3785"],
"esculentum-aus": ["083WAWA"],
"caudatum": ["wolf-1002"],
}
```

Tldr; i don't think we can combine these datasets

Imap .5
Samples: 31
Sites before filtering: 121988
Filtered (indels): 782
Filtered (bi-allel): 1377
Filtered (mincov): 101201
Filtered (minmap): 121253
Filtered (subsample invariant): 18629
Filtered (minor allele frequency): 0
Filtered (combined): 121347
Sites after filtering: 653
Sites containing missing values: 402 (61.56%)
Missing values in SNP matrix: 680 (3.36%)
SNPs (total): 653
SNPs (unlinked): 214
Imputation: 'sampled'; (0, 1, 2) = 98.4%, 1.2%, 0.4%

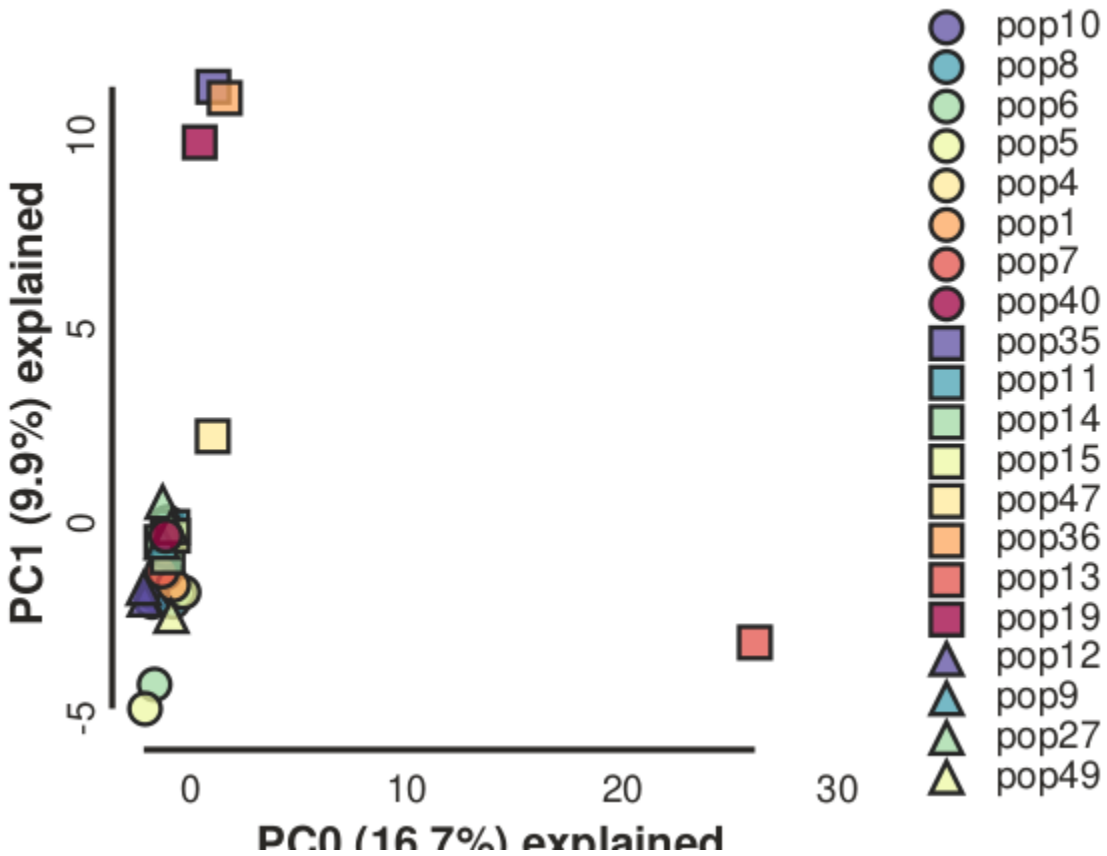
imap .25
Samples: 30

Sites before filtering: 121988
Filtered (indels): 695
Filtered (bi-allel): 1042
Filtered (mincov): 98818
Filtered (minmap): 119509
Filtered (subsample invariant): 43151
Filtered (minor allele frequency): 0
Filtered (combined): 120075
Sites after filtering: 1887
Sites containing missing values: 1142 (60.52%)
Missing values in SNP matrix: 2068 (3.65%)
SNPs (total): 1887
SNPs (unlinked): 781
Imputation: 'sampled'; (0, 1, 2) = 92.2%, 4.0%, 3.8%



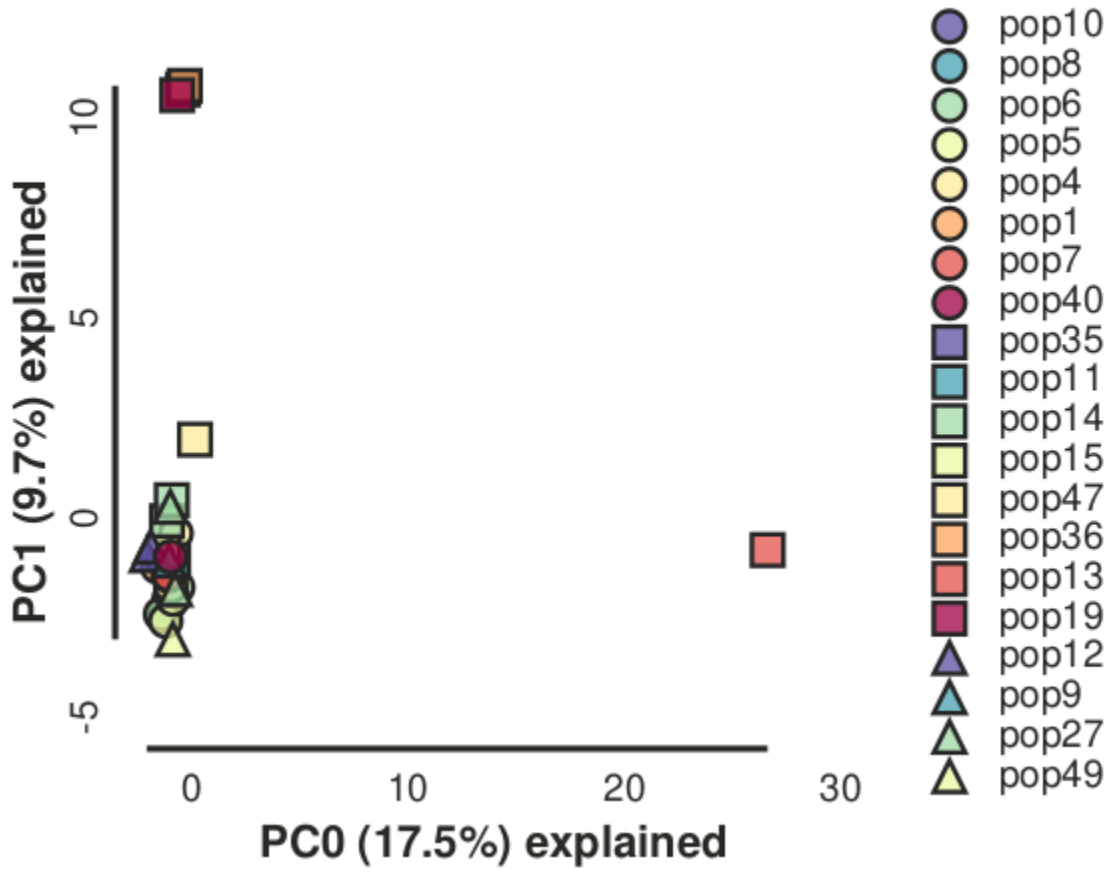
Removing out1 and pop13
Samples: 29
Sites before filtering: 121988

Filtered (indels): 674
Filtered (bi-allel): 898
Filtered (mincov): 96548
Filtered (minmap): 117623
Filtered (subsample invariant): 56201
Filtered (minor allele frequency): 0
Filtered (combined): 119839
Sites after filtering: 2083
Sites containing missing values: 1318 (63.27%)
Missing values in SNP matrix: 2486 (4.12%)
SNPs (total): 2083
SNPs (unlinked): 1075
Imputation: 'sampled'; (0, 1, 2) = 89.3%, 5.3%, 5.4%



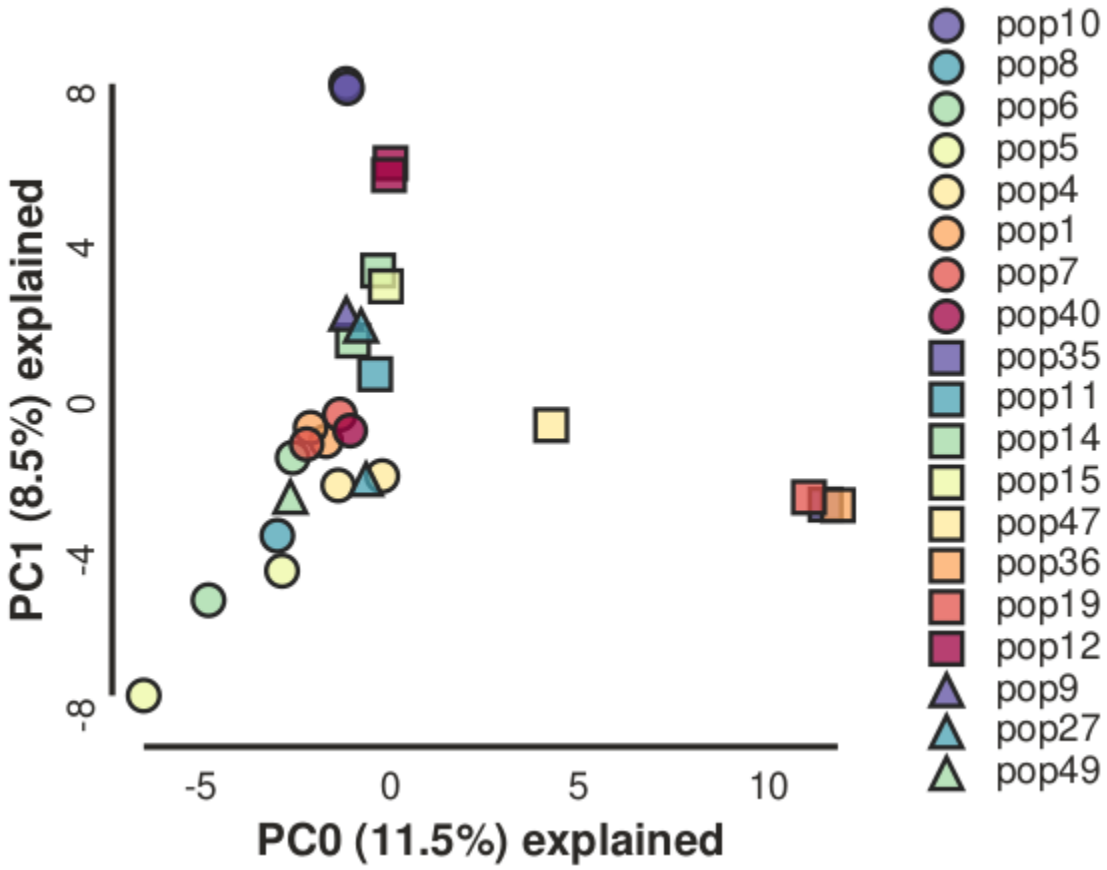
Minmap .1
Samples: 29
Sites before filtering: 121988
Filtered (indels): 674
Filtered (bi-allel): 898

Filtered (mincov): 96548
Filtered (minmap): 117623
Filtered (subsample invariant): 56201
Filtered (minor allele frequency): 0
Filtered (combined): 119839
Sites after filtering: 2083
Sites containing missing values: 1318 (63.27%)
Missing values in SNP matrix: 2486 (4.12%)
SNPs (total): 2083
SNPs (unlinked): 1075
Imputation: 'sampled'; (0, 1, 2) = 89.3%, 5.3%, 5.4%



Removed pop13 (there were 2 labeled as such)
Samples: 28
Sites before filtering: 121988
Filtered (indels): 617
Filtered (bi-allel): 825
Filtered (mincov): 95000
Filtered (minmap): 116934
Filtered (subsample invariant): 60641

Filtered (minor allele frequency): 0
Filtered (combined): 119777
Sites after filtering: 2139
Sites containing missing values: 1396 (65.26%)
Missing values in SNP matrix: 2773 (4.63%)
SNPs (total): 2139
SNPs (unlinked): 1186
Imputation: 'sampled'; (0, 1, 2) = 87.8%, 6.3%, 5.9%



Genetic - geographic distance

Code is from this landscape population genetics tutorial, and this article on Mantel tests

```
library(poppr)
library(ape)
library(magrittr)
library(letsR)
```

```
library(vegan)

# all samples
sa2 <- read.structure("sa-pter2_u.str", n.ind=47, n.loc=42161,
onerowperind=FALSE, col.lab=1)

sa2 <- read.structure("sa-pter2.str", n.ind=47, n.loc=121892,
onerowperind=FALSE, col.lab=1)
# esculentum only
sa2 <- read.structure("esc-pter2.str", n.ind=45, n.loc=121892,
onerowperind=FALSE, col.lab=1)

# south american esculentum only
sa2 <- read.structure("esc-sa-pter2.str", n.ind=44, n.loc=121892,
onerowperind=FALSE, col.lab=1)

# 0 or enter for any prompts when loading in data

# pairwise distance using the provesti distance (for SNP data and can
handle missing data)
sadist <- provesti.dist(sa2)

# plot neighbor-joining tree
tr <- sadist %>%
nj() %>%
ladderize()
plot(tr)

coord <- read.csv("coords.csv")
coord <- read.csv("coords-esculentum.csv")
coord <- read.csv("coords-esculentum-sa.csv")

# order is long, lat
my_points <- coord[,4:3]

# create a dissimilarity matrix for geographic locations as class dist
dist_mat <- lets.distmat(my_points, asdist=TRUE)

# compare matrices using a mantel test
mantel(sadist, dist_mat)

### for all samples ###
# Mantel statistic based on Pearson's product-moment correlation
```

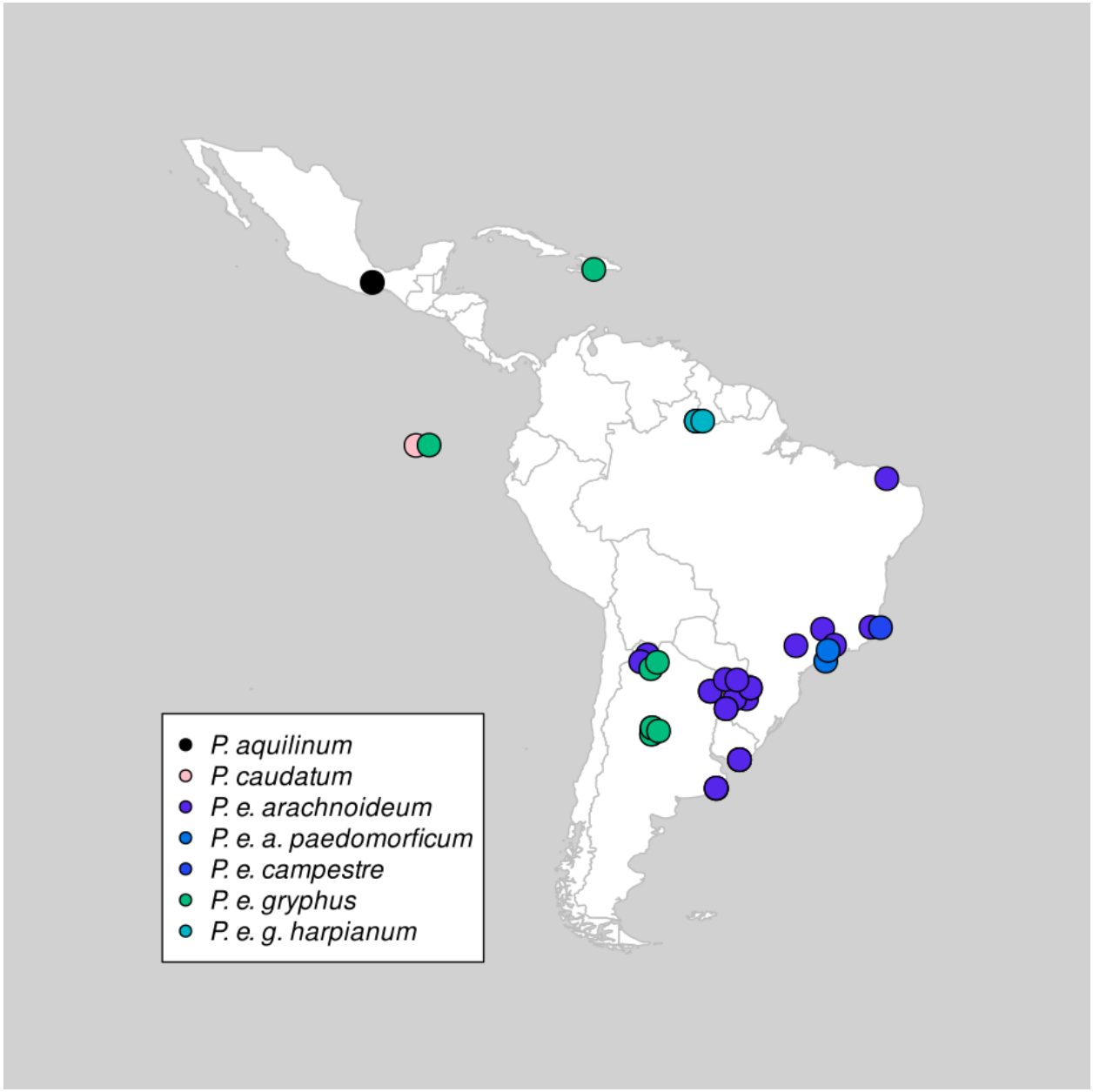
```
Call:
mantel(xdis = sadist, ydis = dist_mat)

Mantel statistic r: 0.2567
      Significance: 0.016

Upper quantiles of permutations (null model):
  90%   95% 97.5%   99%
0.105 0.149 0.191 0.577
Permutation: free
Number of permutations: 999

### without outgroups ###

Mantel statistic based on Pearson's product-moment correlation

Call:
mantel(xdis = sadist, ydis = dist_mat)

Mantel statistic r: -0.1013
      Significance: 0.754

Upper quantiles of permutations (null model):
  90%   95% 97.5%   99%
0.160 0.202 0.245 0.302
Permutation: free
Number of permutations: 999

### south american samples only ###

Call:
mantel(xdis = sadist, ydis = dist_mat)

Mantel statistic r: 0.1422
      Significance: 0.123

Upper quantiles of permutations (null model):
  90%   95% 97.5%   99%
0.152 0.183 0.212 0.236
Permutation: free
Number of permutations: 999
```

```
# geographic and genetic variation/distance are not correlated with P.
esculentum
```

Map

```
library(maps)

p <- read.csv("coords.csv", header = T, sep = ",")
coords <- p[1:47, 3:4]

map(database = 'world', regions = c('Brazil', 'Uruguay', 'Argentina', 'Chile',
'Colombia', 'Peru', 'Venezuela', 'Ecuador', 'Paraguay', 'Guyana', 'Suriname',
'French Guiana', 'Falkland Islands', 'Bolivia', 'Mexico', 'Panama', 'Costa Rica',
'Belize', 'El Salvador', 'Guatemala', 'Honduras', 'Nicaragua', 'Cuba', 'Dominican
Republic', 'Haiti'), fill =T, col= 'white', border="grey", bg="lightgrey")

# aquilinum
points(coords[1,2], coords[1,1], col = "#000000", bg = "black", cex = 2, pch = 21)

# arachnoideum
points(coords[2:30,2], coords[2:30,1], col = "#000000", bg = "#5928ED", cex = 2,
pch = 21)
# a. paedomorficum
points(coords[46:47,2], coords[46:47,1], col = "#000000", bg = "#0073E6", cex = 2,
pch = 21)

# campestre
points(coords[31,2], coords[31,1], col = "#000000", bg = "#2546F0", cex = 2, pch =
21)

# caudatum
points(coords[32,2], coords[32,1], col = "#000000", bg = "pink", cex = 2, pch = 21)

# esculentum
# points(coords[33,2], coords[33,1], col = "#000000", bg = "blue", cex = 2, pch =
21)

# gryphus
points(coords[34:41,2], coords[34:41,1], col = "#000000", bg = "#00BF7D", cex = 2,
pch = 21)

# g. harpianum
```

```
points(coords[42:44,2], coords[42:44,1], col = "#000000", bg = "#00B4C5", cex = 2,
pch = 21)

# indet
# points(coords[45,2], coords[45,1], col = "#000000", bg = "blue", cex = 2, pch =
21)


legend(x = "bottomleft", legend = c("P. aquilinum", "P. caudatum", "P. e.
arachnoideum", "P. e. a. paedomorficum", "P. e. campestre", "P. e. gryphus", "P. e.
g. harpianum"), bg = "white", pch = 19, cex = 1, col = c("black", "pink",
"#5928ED", "#0073E6", "#2546F0", "#00BF7D", "#00B4C5"), text.font = 3)
```



http://blog.phytools.org/2019/03/projecting-phylogeny-onto-geographic.html

Plot map and phylogeny together