Spatial detection of outlier loci with Moran eigenvector maps (MEM)

Introduction

The purpose of this vignette is to illustrate a new outlier detection method that uses genetic data and spatial coordinates to detect loci potentially under selection (Wagner et al. 2016). This method, which we call **Moran spectral outlier detection (MSOD)**, uses the power spectrum of **Moran eigenvector maps (MEM)** to quantify how variation in the frequency of an allele is distributed across a range of spatial scales defined by MEM spatial eigenvectors. MSOD can be used on individual or population based data sets.

If environmental predictors are available, MSOD can be followed by a second step that uses **Moran spectral randomization (MSR)** to test associations between outlier loci and environmental predictors, while accounting for spatial autocorrelation.

Assumptions

MSOD relies on an assumption of normality under the null hypothesis that the distributions of loci are shaped by the same process (gene flow). Deviations from normality are likely under non-equilibrium and isolation by distance scenarios, which may increase false positive rates. MSR relies on stationarity assumptions, which means that p-values may be inflated if both variables have a linear trend. See Wagner et al. 2016 for more discussion of these assumptions, and how they impact outlier detection in a variety of simulation scenarios.

Data & packages

In this vignette, we'll walk through an MSOD/MSR example using part of an individual-based simulation data set from Forester et al. 2016. In this case, the genetic data are input as allele counts (i.e. 0/1/2) for each locus. For population-based data, you can input the genetic data as allele frequencies within demes.

Begin by installing the necessary packages, if you don't already have them:

```
# install.packages(c("spdep","adespatial"), dependencies=TRUE)

# Load packages
# -------------
require(spdep)
require(adespatial)
```

Analysis

Section 1: Load and inspect the data

The dimensions of our data frame are 500 rows and 103 columns. Every row in the data frame contains information for one individual, so in this example we will be analyzing 500 individuals. Columns 1 and 2 are the X and Y UTM coordinates for each individual. Column 3 is a factor ("Env") indicating what type of habitat an individual is located in. Columns 4 through 103 are genotypes at 100 loci for each individual. These genotypes are provided as allele counts (0/1/2). Locus 1 (L1) is the locus under selection (responsive to the habitat factor "Env"). The remainder of the loci are neutral (L2 - L100).

```
dat   <- read.csv("data/MSOD_vignette_data.csv") # All the data

Coord <- data.matrix(dat[,1:2])   # The UTM coordinates of the 500 individuals
Env   <- as.data.frame(dat[,3])   # Habitat factor for the 500 individuals
Loci  <- data.matrix(dat[,4:103]) # Genotypes at 100 loci for the 500 individuals
```

The example data provided here are simplified from a CDPOP (Landguth & Cushman 2010) simulation output file. This particular simulation used an aggregated habitat surface, a selection coefficient of 5%, and individual dispersal of 5% of the surface per generation. These data are from the first replicate of 10 for these simulations. The full simulation data set is available at Dryad doi: 10.5061/dryad.v0c77, with more information on the simulation scenarios in Forester et al. 2016.

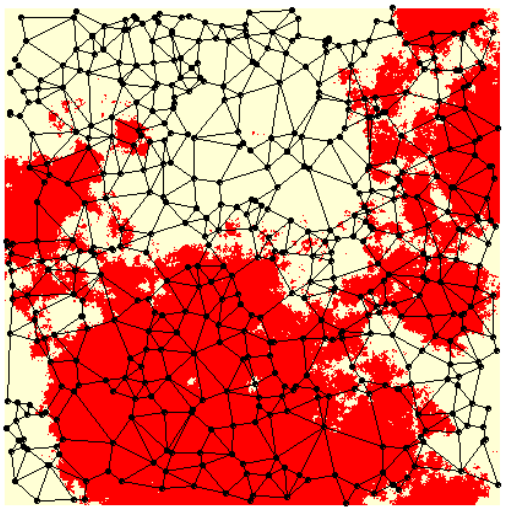
Now that we have loaded the data, our next task is to calculate the **Moran eigenvector maps (MEM)** from the XY coordinates of our 500 individuals. MEM are orthogonal synthetic variables that provide a decomposition of the spatial relationships among sampling sites based on a spatial weighting matrix (see Wagner et al. 2016 for more information and relevant citations).

```
# Calculate MEM vectors and values
# --------------------------------
nb      <- graph2nb(gabrielneigh(Coord), sym=TRUE)  # Gabriel graph: neighbor definition
listW   <- nb2listw(nb, style="W")                  # Spatial weights matrix
disttri <- nbdists(nb, Coord)                       # Add longlat=T for lat/long coordinates
fdist   <- lapply(disttri, function(x) x^(-1))      # Use inverse distance weights
listW   <- nb2listw(nb, glist=fdist, style="W")     # Revised spatial weights matrix
tmp     <- scores.listw(listW, MEM.autocor = "all") # Eigenanalysis

mem     <- list(vectors = as.matrix(tmp), values = attr(tmp, "values")) # MEM eigenvectors and eigenval
mem$values <- mem$values / abs(sum(mem$values))     # Rescale eigenvalues to Moran's I
```

In the above code, we begin by defining neighbors with a Gabriel graph. Then we create a spatial weights matrix so that neighbors receive a weight that is proportional to the inverse distance between a location and that neighbor. MEM are then extracted by eigenanalysis. This produces a set of eigenvalues (one for each MEM axis) that quantifies the spatial scale of that MEM. We also obtain the MEM eigenvectors, which are the MEM axes we'll use for our analysis. In this case, we have 500 individuals sampled at 500 sets of XY coordinates, so we end up with 499 MEM axes.

This plot shows the habitat selection surface, the 500 sampled individuals, and the Gabriel graph used to calculate the spatial weights matrix and MEM variables:

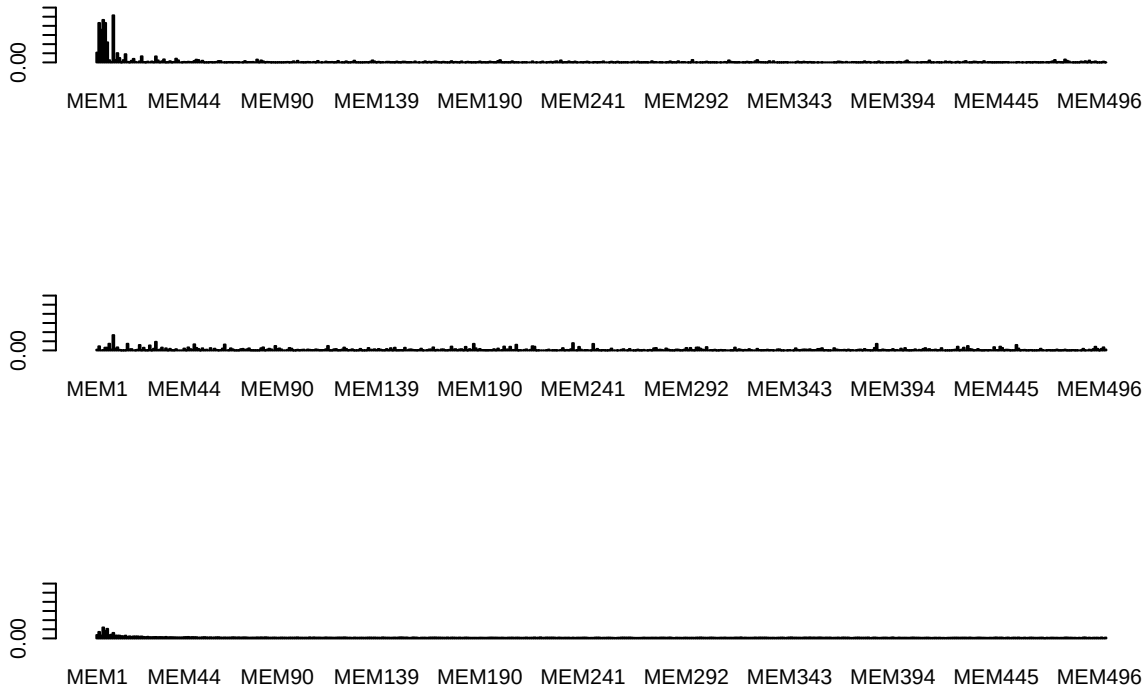


Section 2: Moran spectral outlier detection (MSOD)

Now that we have all of the necessary data, we'll calculate an MEM power spectrum for each of our 100 loci. We do this by calculating correlations R.YV between allele counts (or frequencies) at each locus and all 499 MEM eigenvectors. Note that the square of R.YV for each locus is its power spectrum.

```
# Correlations between the Loci and MEM axes
# ------------------------------------------
# Calculate R.YV, which contains for each locus the vector of its correlations with all MEM axes.
R.YV <- cor(Loci, mem$vectors, use="pairwise.complete.obs") # R.YV = Correlation with MEM axes
S    <- apply(R.YV^2, 2, mean)                              # S = Average power spectrum

# Plot power spectra
# ------------------
barplot((R.YV^2)[1,], ylim=c(0, 0.12))  # First locus (under selection)
barplot((R.YV^2)[2,], ylim=c(0, 0.12))  # Second locus (neutral)
barplot(S, ylim=c(0, 0.12))             # Average power spectrum (all 100 loci)
```



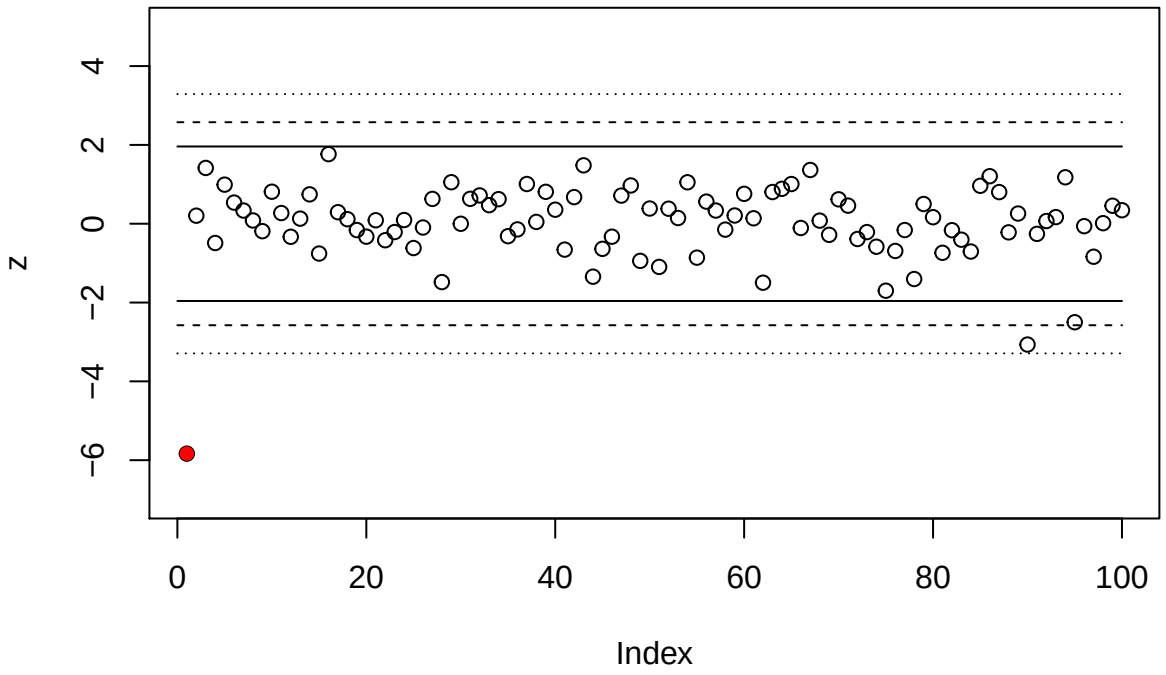
Compare the power spectra plots for the locus under selection (top) and a neutral locus (middle). The power spectrum is just the squared correlation coefficient obtained by correlation (R.YV) of each locus with the 499 MEM variables. The average power spectrum across all 100 loci is shown at the bottom.

Next, we'll calculate z-scores and compare detection based on three cutoff options.

```
cutoffs <- abs(qnorm(c(0.05, 0.01, 0.001)/2))  # Cutoffs (can be modified!)

# Calculate z-scores for power spectra
# ------------------------------------
Dev <- sweep(R.YV^2, 2, S, "/") - 1            # Subtract average power spectrum from each locus.
Dev[Dev > 0] <- 0                              # Set positive deviations to zero.
Dev <- apply(Dev, 1, sum)                      # Sum of negative deviations
z <- scale(Dev)                                # Standardize

# Plot z-scores for power spectra
# -------------------------------
plot(z, ylim=c(-7,5))                          # Plot the z-scores
points(1, z[1], pch=16, col="red")             # The locus under selection
for(h in 1:length(cutoffs))                    # Add lines for the three cutoffs
{
  lines(c(0,100), rep(cutoffs[h],2), lty=h)
  lines(c(0,100), rep(-cutoffs[h],2), lty=h)
}
```



This plot shows the standardized z-scores for the 100 loci (locus under selection in red), along with the three outlier cutoffs. You can see that two loci would be detected as false positives using the most liberal cutoff of 0.05. We'll continue and just use the middle cutoff of 0.01:

```
# Apply cutoff to determine outliers
# ----------------------------------
cutoff.msod <- cutoffs[2]                                # Just the middle cutoff of 0.01

Candidates.msod <- c(1:100)[abs(z)>cutoff.msod]          # Candidate loci at this cutoff
TPR.msod <- as.numeric(is.element(1, Candidates.msod)) # True positive rate (is the first locus detecte
print(paste("True positive rate: ", TPR.msod))           # Yes!
```

```
## [1] "True positive rate:  1"
```

```
FPR.msod <- (length(Candidates.msod) - TPR.msod)/99     # False pos. rate (# false detections/99 neutral
print(paste("False positive rate: ", round(FPR.msod, 3)))
```

```
## [1] "False positive rate:  0.01"
```

```
print(paste("Falsly identified outlier loci (ID): ",
      rownames(R.YV)[Candidates.msod[Candidates.msod > 1]])) # One false positive
```

```
## [1] "Falsly identified outlier loci (ID):  L90"
```

Notice how true and false positive rates change based on the cutoff. Setting a cutoff and deciding on a correction for multiple tests depends on your question and your tolerance for false positives and false negatives.

Section 3: Moran spectral randomization (MSR)

This step is used as an optional post-MSOD test for association between candidate outliers and environmental predictors. MSR calculates correlations between loci and environmental predictors while accounting for spatial autocorrelation in both data sets. A randomization approach is used to build a null hypothesis of no correlation, given the power spectra of both variables. See Wagner et al. 2016 for more information on MSR and relevant citations.

```
# Set a cutoff & number of permutations for MSR
# ---------------------------------------------
cutoff.msr <- 0.05    # Set a less stringent cutoff
nPerm <- 199          # Set number of permutations for MSR test (may choose e.g. 499 or 999)

# MEM correlations for Env and coordinates (as spurious predictors)
# -----------------------------------------------------------------
R.XV.Env <- cor(Env, mem$vectors)
R.XV.xcoord <- cor(Coord[,1], mem$vectors)
R.XV.ycoord <- cor(Coord[,2], mem$vectors)
```

For these simulations, we use the UTM coordinates as spurious environmental predictors. You can think of them as environmental variables that are correlated with latitude and longitude. The correct predictor driving selection at the locus under selection (L1) is "Env".

```
# Function to perform MSR test
# ----------------------------
get.pvalue.msr <- function(r.XV=R.XV, r.YV=R.YV, nPerm=199)
{
  R.XV.rand <- matrix(r.XV, nPerm, ncol(r.XV), byrow=TRUE)
  R.XV.rand <- R.XV.rand * sample(c(-1,1), length(R.XV.rand), replace=TRUE)
  Cor.obs <- abs(as.vector(r.YV %*% t(r.XV)))
  Cor.rand <- abs(r.YV %*% t(R.XV.rand))
  P.values.MSR <- apply((cbind(Cor.obs,Cor.rand) >= Cor.obs), 1, mean)
  P.values.MSR
}

# MSR test for candidate outlier loci detected by MSOD
# ----------------------------------------------------
b.Env <- get.pvalue.msr(r.XV=R.XV.Env, r.YV=R.YV[Candidates.msod,], nPerm=nPerm)
b.X <- get.pvalue.msr(r.XV=R.XV.xcoord, r.YV=R.YV[Candidates.msod,], nPerm=nPerm)
b.Y <- get.pvalue.msr(r.XV=R.XV.ycoord, r.YV=R.YV[Candidates.msod,], nPerm=nPerm)

print(paste("Loci significantly associated with Env: ", names(b.Env)[b.Env < cutoff.msr]))
```

```
## [1] "Loci significantly associated with Env:  L1"
```

```
print(paste("Loci significantly associated with X: ", names(b.X)[b.X < cutoff.msr]))
```

```
## [1] "Loci significantly associated with X:  L90"
```

```
print(paste("Loci significantly associated with Y: ", names(b.Y)[b.Y < cutoff.msr]))
```

```
## [1] "Loci significantly associated with Y:  L1"
```

For this simulation and cutoff, MSOD (Section 2) detected two loci as potentially under selection - one was a correct true positive detection (L1), while the other was a false positive detection (L90). Using MSR (Section 3), we found that our false positive was significantly associated with the spurious "X" predictor, while the true positive detection (L1) was significantly associated with both the driving environmental predictor, "Env", as well as the spurious "Y" predictor.

Conclusions

In summary, **Moran spectral outlier detection (MSOD)** can be used to reliably identify outlier loci based on their unusual spatial signature, without reference to environmental data. Follow-up analyses using **Moran spectral randomization (MSR)** can be used to identify potential drivers of selection at these

candidate loci, while accounting for spatial autocorrelation. These methods can be used on either individual or deme-level samples, and are robust to a variety of sampling designs and sample sizes (Wagner et al. 2016).

Contributors

- Brenna R. Forester (Author)
- Helene H. Wagner (Author)
- Mariana Chávez-Pesqueira (Author)

References

Forester BR, Jones MR, Joost S, Landguth EL, Lasky JR (2016) Detecting spatial genetic signatures of local adaptation in heterogeneous landscapes. Molecular Ecology, 25, 104-120.

Landguth EL, Cushman SA (2010) cdpop: a spatially explicit cost distance population genetics program. Molecular Ecology Resources, 10, 156-161.

Wagner HH, Chávez-Pesqueira M, Forester BR (In press) Spatial detection of outlier loci with Morana eigenvector maps (MEM). Molecular Ecology Resources.

Session Information

```
options(width = 100)
devtools::session_info()
```

```
## Session info -------------------------------------------------------------------------------------
```

```
##  setting  value
##  version  R version 3.3.2 (2016-10-31)
##  system   x86_64, darwin13.4.0
##  ui       X11
##  language (EN)
##  collate  en_US.UTF-8
##  tz       America/Los_Angeles
##  date     2016-12-15
```

```
## Packages -----------------------------------------------------------------------------------------
```

```
##  package      * version     date       source
##  ade4           1.7-5       2016-12-13 CRAN (R 3.3.2)
##  adegraphics    1.0-6       2016-12-13 CRAN (R 3.3.2)
##  adespatial   * 0.0-7       2016-12-13 CRAN (R 3.3.2)
##  backports      1.0.4       2016-10-24 cran (@1.0.4)
##  boot           1.3-18      2016-02-23 CRAN (R 3.2.3)
##  coda           0.18-1      2015-10-16 CRAN (R 3.2.0)
##  deldir         0.1-12      2016-03-06 CRAN (R 3.2.4)
##  devtools       1.12.0      2016-06-24 CRAN (R 3.3.0)
##  digest         0.6.10      2016-08-02 CRAN (R 3.3.0)
##  evaluate       0.10        2016-10-11 cran (@0.10)
##  gdata          2.17.0      2015-07-04 CRAN (R 3.2.0)
##  gmodels        2.16.2      2015-07-22 CRAN (R 3.2.0)
##  gtools         3.5.0       2015-05-29 CRAN (R 3.2.0)
##  htmltools      0.3.5       2016-03-21 CRAN (R 3.2.4)
##  httpuv         1.3.3       2015-08-04 CRAN (R 3.2.0)
```

```
##  KernSmooth     2.23-15     2015-06-29 CRAN (R 3.2.0)
##  knitr          1.14.15     2016-11-04 Github (yihui/knitr@56faff4)
##  lattice        0.20-33     2015-07-14 CRAN (R 3.2.0)
##  latticeExtra   0.6-28      2016-02-09 CRAN (R 3.2.3)
##  LearnBayes     2.15        2014-05-29 CRAN (R 3.2.0)
##  magrittr       1.5         2014-11-22 CRAN (R 3.2.0)
##  MASS           7.3-45      2015-11-10 CRAN (R 3.2.2)
##  Matrix       * 1.2-6       2016-05-02 CRAN (R 3.3.0)
##  memoise        1.0.0       2016-01-29 CRAN (R 3.2.3)
##  mime           0.5         2016-07-07 cran (@0.5)
##  nlme           3.1-128     2016-05-10 CRAN (R 3.3.0)
##  R6             2.2.0       2016-10-05 cran (@2.2.0)
##  RColorBrewer   1.1-2       2014-12-07 CRAN (R 3.2.0)
##  Rcpp           0.12.8      2016-11-17 cran (@0.12.8)
##  rmarkdown      1.1.9014    2016-11-04 Github (rstudio/rmarkdown@b36c67e)
##  rprojroot      1.1         2016-10-29 cran (@1.1)
##  shiny          0.14.2.9001 2016-12-12 Github (rstudio/shiny@907b9a9)
##  sp           * 1.2-3       2016-04-14 CRAN (R 3.3.0)
##  spdep        * 0.6-8       2016-09-21 cran (@0.6-8)
##  stringi        1.1.1       2016-05-27 CRAN (R 3.3.0)
##  stringr        1.1.0       2016-08-19 cran (@1.1.0)
##  withr          1.0.2       2016-06-20 cran (@1.0.2)
##  xtable         1.8-2       2016-02-05 CRAN (R 3.2.3)
##  yaml           2.1.13      2014-06-12 CRAN (R 3.2.0)
```