Re-working Clair Barnes' MSc dissertation

Wilfrid Kendall

9 March 2016 then 13-18 January 2017

1 Purpose

Verifying calculations of Barnes [2015].

2 Preparation

2.1 Initialization

Define constants for figure dimensions.

```
# Run script within R using
#       rmarkdown::render("Barnes.Rmd")
knitr::opts_chunk$set(fig.path='figure/')
MFH <- 3 # Medium figure height
MFW <- 5 # Medium figure width
```

2.2 Libraries

Load useful libraries.

```
library(fpc)     # Library of clustering algorithms including DBSCAN.
library(igraph)  # Library for network analysis.
library(ggplot2) # Library for graphics.
```

Load Clair Barnes' libraries (see `README.md` for details on how to prepare, download, and load into `R`).

```
library(AS.preprocessing)
library(AS.angles)
library(AS.circular)
```

2.3 Genlis image

The original Genlis image is defined by a `JPEG` file.

```
genlis <- import.map("Genlis.jpg", threshold = 0.2, plot = T)
```

3 Image processing

Identify scale marker and axis. (Requires user intervention, so set to be inactive in this script. Presumed that the equivalent commands have been carried out already as specified in `README.md`.)

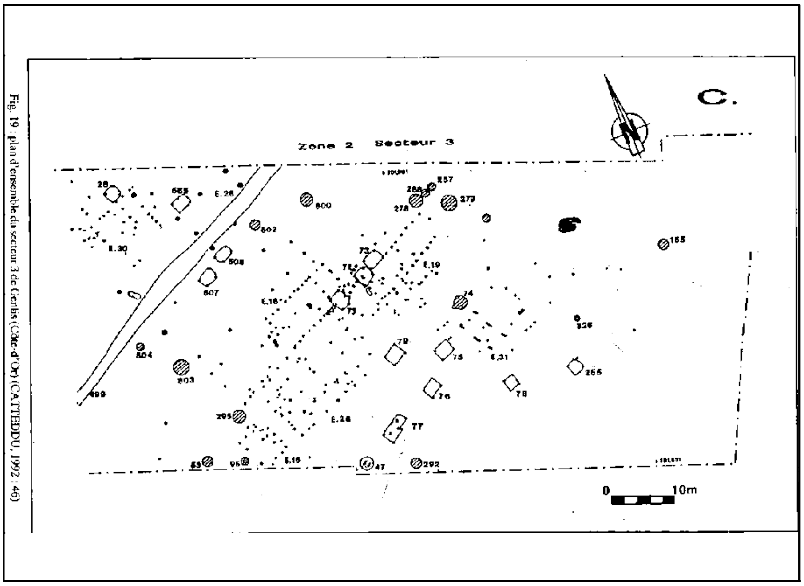


Figure 1: Genlis map in original form.

```
get.scale(genlis)
genlis.NS <- get.NS.axis(rescaled)
genlis <- NS.marked
remove(rescaled, NS.marked)
save.features(genlis, "Genlis-Initial-Cleansing")
```

4 Analysis using von Mises and Jones-Pewsey models

Load initially cleansed data structure.

```
genlis <- load.features("Genlis-Initial-Cleansing")
```

Exclude sparse shapes from image in raster form. Display resulting post-holes.

```
exclude.sparse.shapes(genlis, density = 0.55, lower = 3, plot = F)
genlis.closing <- feature.closing(sparse.shapes.classified, plot.progress = F)
fill.broken.boundary(features.closed, s = 0.2, plot.progress = F)
genlis <- boundary.filled
get.postholes(genlis)
```

Write and reload features to avoid having to re-clean image later on.

```
save.features(genlis, "Genlis-final")
write.csv(centres, "Genlis-posthole-centres.csv", row.names = F)
genlis <- load.features("Genlis-final")
```

Preserve row-names, rather than loading centres from `csv` file. Further cleaning to exclude "distant points", remaining points placed in `pts`.

```
dist.filter <- filter.by.distance(centres)
aspect <- (max(centres$ym) - min(centres$ym)) / (max(centres$xm) - min(centres$xm))
qplot(data=centres, xm,ym, colour=dist.filter, xlab="x", ylab="y", asp=aspect)
```

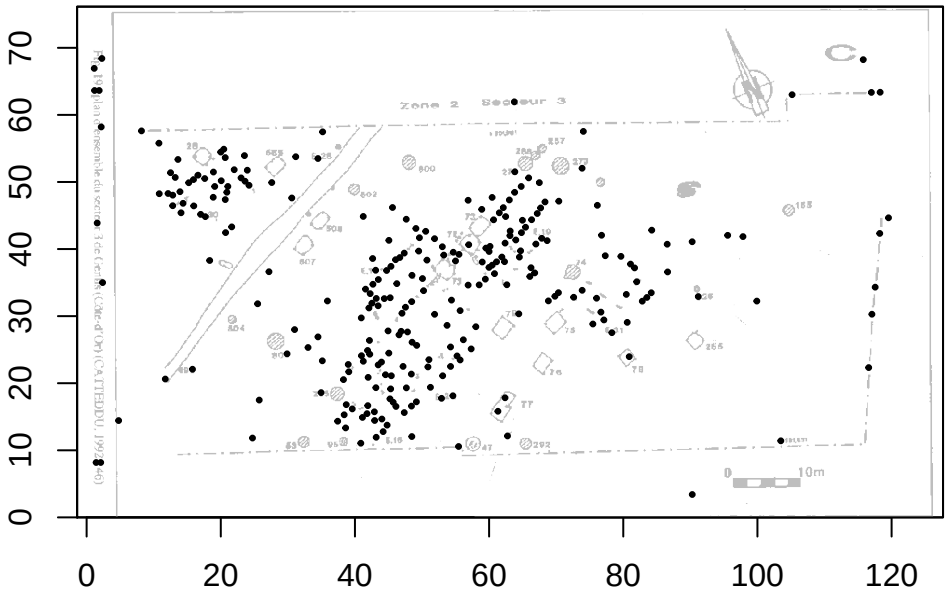


Figure 2: Genlis map with post-holes identified.

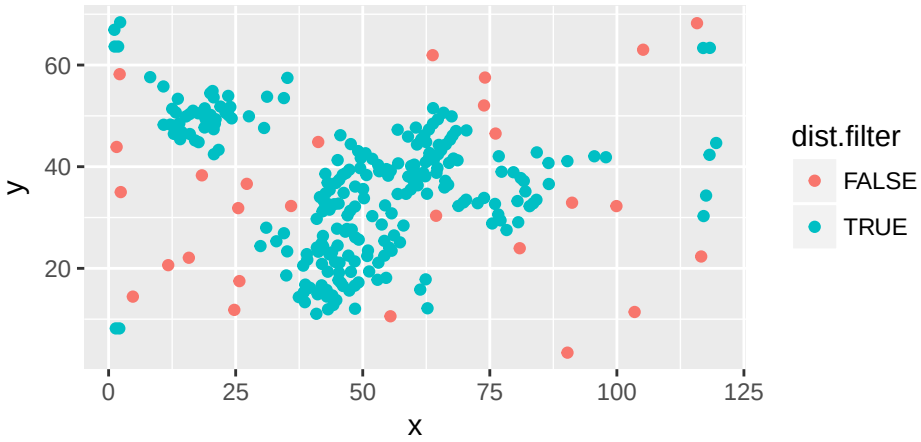


Figure 3: Genlis map excluding distant post-holes.

```
pts <- centres[dist.filter,]
```

Extract angles of segments between each point and its first nearest-neighbour (third column of `k.1`), multiply by 4 and reduce modulo $2/\pi$ to obtain a measure of the angle of the perpendicular grid to which each point might thus be thought to belong.

```
k.1 <- k.nearest.angles(pts, 1)
q <- circular(4 * k.1[,-c(1,2)]) %% (2*pi)
```

Carry out statistical tests for uniformity on resulting data structure `q`. Conclusion: data clearly not uniform, as indeed is indicated by the histogram.

```
qplot((90 / (2*pi)) * q, xlab="Orientation in degrees")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

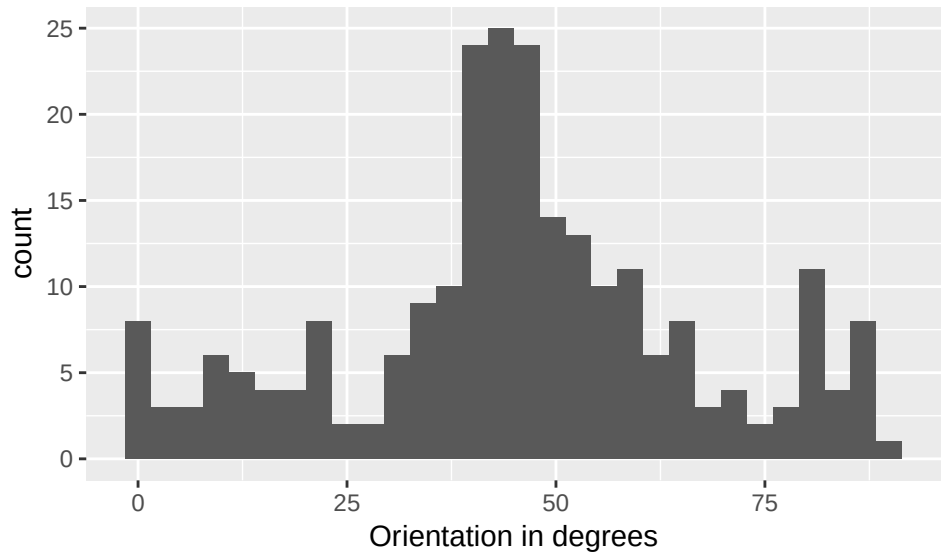


Figure 4: Histogram of `perpendicularity` angles.

```
rayleigh.test(q)
```

```
##
##        Rayleigh Test of Uniformity
##        General Unimodal Alternative
##
## Test Statistic:  0.3421
## P-value:  0
```

```
kuiper.test(q)
```

```
##
##        Kuiper's Test of Uniformity
##
## Test Statistic:  4.5416
## P-value < 0.01
##
```

```
watson.test(q)
```

```
##
```

```
##        Watson's Test for Circular Uniformity
##
## Test Statistic: 1.7652
## P-value < 0.01
##
```

Test for symmetry. Conclusion: does not give grounds for supposing data not to be symmetric!

```
r.symm.test.stat(q)
```

```
## $test.statistic
## [1] 0.5730578
##
## $p.val
## [1] 0.5666055
```

Estimation of various relevant statistics. First, spread of estimate of location (degrees) using biased-corrected estimate: variation is $\pm 12^o$.

```
bc <- bc.ci.LS(q, alpha = 0.05)
(bc$mu[3] - bc$mu[1]) * 180 / pi
```

```
##   upper
## 12.4979
```

Bias-corrected skewness estimate does not differ significantly from zero.

```
bc$beta2
```

```
##    estimate       lower       upper
## -0.03494808 -0.17315954  0.10326339
```

Bias-corrected excess kurtosis is appreciable, so we may need a model which is more peaked than a von Mises model.

```
    (bc$alpha2[1] - (bc$rho[c(1)]^4))
```

```
##  estimate
## 0.2929723
```

```
    (bc$alpha2[2:3] - (bc$rho[c(3,2)]^4))
```

```
##     lower     upper
## 0.1861651 0.3878370
```

Von Mises maximum-likelihood estimate with confidence intervals – need to reduce the μ estimate modulo 2π.

```
vm.mle <- mle.vonmises(q)
q95 <- qnorm(0.975)
vm.mle$mu <- vm.mle$mu %% (2 * pi)
list(mu = c(est = vm.mle$mu,
            lower = (vm.mle$mu - q95*vm.mle$se.mu) %% (2*pi),
            upper = (vm.mle$mu +   q95*vm.mle$se.mu)) %% (2*pi),
     kappa = c(est = vm.mle$kappa,
               lower = vm.mle$kappa - q95*vm.mle$se.kappa,
               upper = vm.mle$kappa + q95*vm.mle$se.kappa)
     )
```

```
## $mu
## Circular Data:
## Type = angles
```

```
## Units = radians
## Template = none
## Modulo = asis
## Zero = 0
## Rotation = counter
##      est    lower    upper
## 3.245095 2.992054 3.498135
##
## $kappa
##       est     lower     upper
## 0.7281306 0.5318086 0.9244527
```

Jones-Pewsey maximum-likelihood estimate with confidence intervals. Also computing normalizing constant.

```
jp.mle <- JP.mle(q)
jp.ncon <- JP.NCon(jp.mle$kappa, jp.mle$psi)
JP.ci.nt(jp.mle, alpha=0.05)
```

```
## $alpha
## [1] 0.05
##
## $mu
##      est    lower    upper
## 3.128822 2.996659 3.260986
##
## $kappa
##       est     lower     upper
## 0.9022438 0.7057558 1.0987318
##
## $psi
##       est     lower     upper
## -2.448444 -3.407135 -1.489754
```

Goodness-of-fit. Jones-Pewsey is a much better fit than von Mises!

```
vM.GoF.output <- vM.GoF.boot(q, vm.mle$mu, B=999)
JP.GoF.output <- JP.GoF.boot(q, jp.mle$mu, B=999)
```

```
vM.GoF.output
```

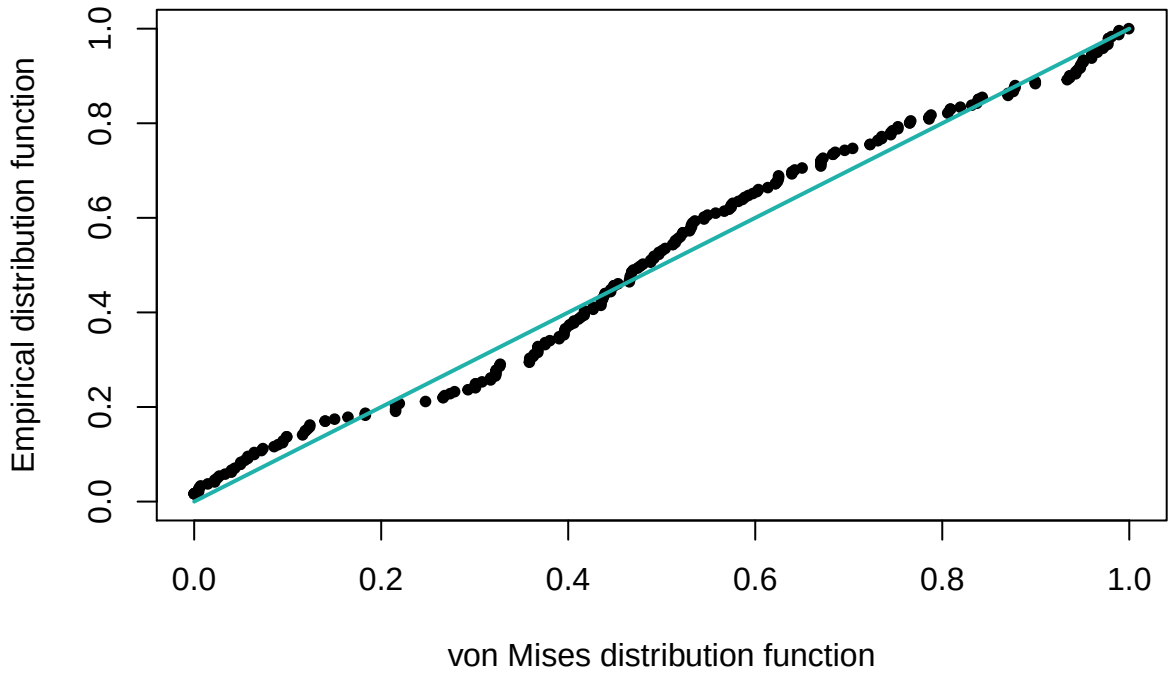
```
##   kuiper   watson      rao rayleigh
##    0.001    0.001    0.001    0.001
```

```
JP.GoF.output
```

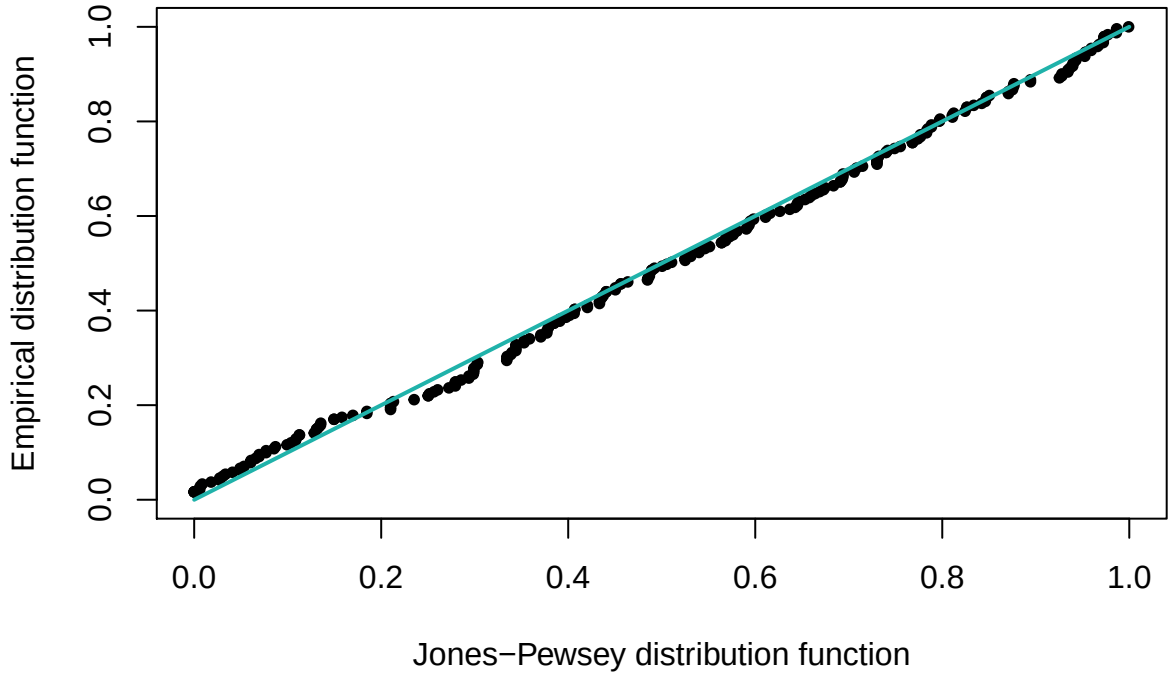
```
##   kuiper   watson      rao rayleigh
##    0.176    0.109    0.001    0.162
```

Plot and calculate residuals: Jones-Pewsey residuals look better!

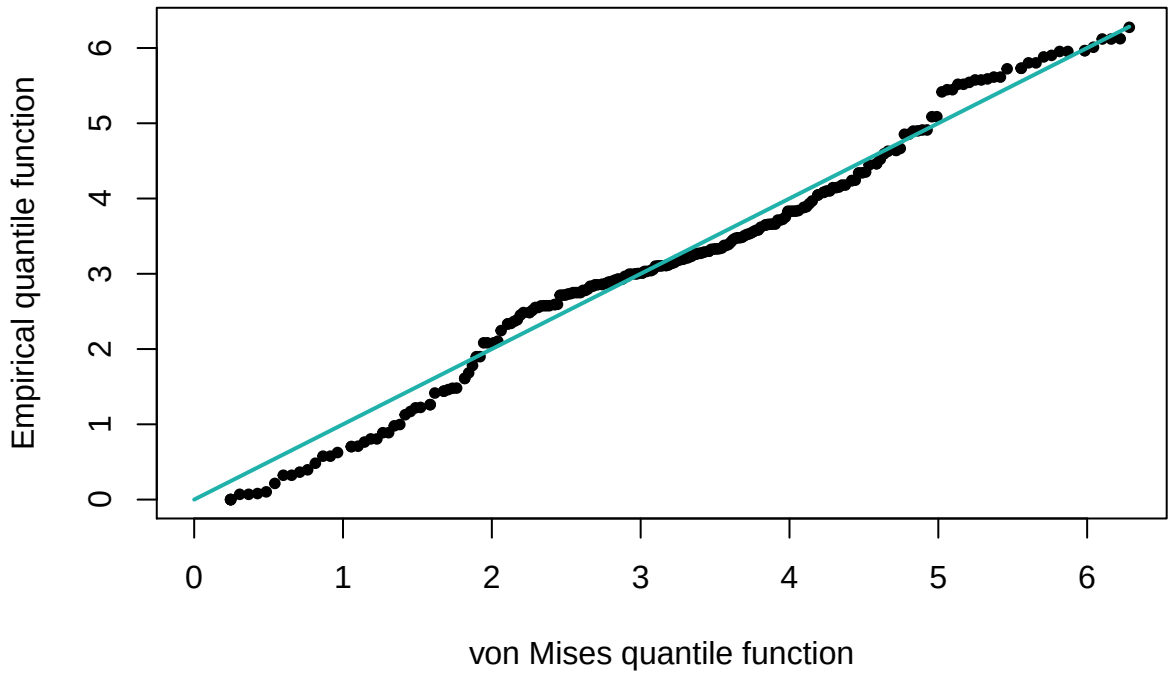
```
vm.pp.res <- vM.PP(q, vm.mle$mu, vm.mle$kappa)
```



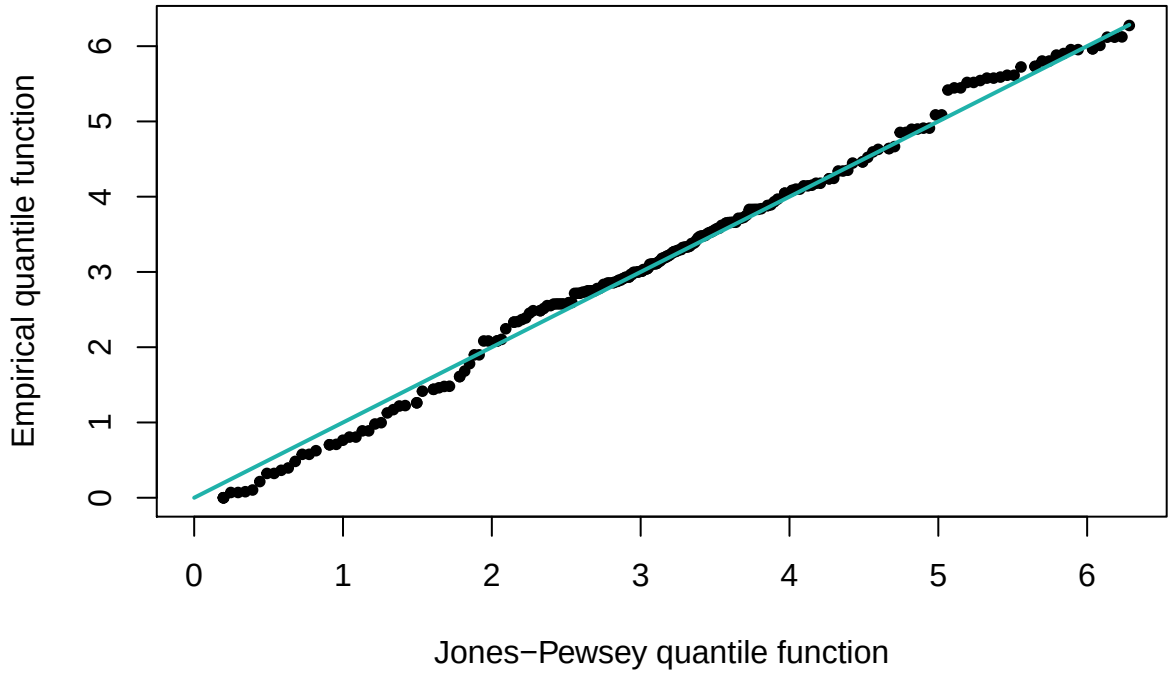

```
jp.pp.res <- JP.PP(q, jp.mle$mu, jp.mle$kappa, jp.mle$psi)
```




```
vm.qq.res <- vM.QQ(q, vm.mle$mu, vm.mle$kappa)
```




```
jp.qq.res <- JP.QQ(q, jp.mle$mu, jp.mle$kappa, jp.mle$psi)
```



Compute mean squared error and standard deviation: again Jones-Pewsey residuals look better.

```
mean(vm.pp.res^2); sd(vm.pp.res^2)
```

```
## [1] 0.001264196
## [1] 0.001075323
```

```
mean(jp.pp.res^2); sd(jp.pp.res^2)
```

```
## [1] 0.0002916625
```

```
## [1] 0.0003083872
```

```
mean(vm.qq.res^2); sd(vm.qq.res^2)
```

```
## Circular Data:
## Type = angles
## Units = radians
## Template = none
## Modulo = asis
## Zero = 0
## Rotation = counter
## [1] 0.04036465
```

```
## [1] 0.03968721
```

```
mean(jp.qq.res^2); sd(jp.qq.res^2)
```

```
## Circular Data:
## Type = angles
## Units = radians
## Template = none
## Modulo = asis
## Zero = 0
## Rotation = counter
## [1] 0.01498657
```

```
## [1] 0.0226363
```

4.1 Testing for perpendicularity versus mere collinearity

Now we test for perpendicularity, by slicing the data into corresponding quadrants. First we find the approximate modal angle, and from this we compute the cut-points and produce a corresponding factor for the data.

```
q.original <- circular(k.1[,-c(1,2)]) %% (2*pi)
mx <- circular(as.numeric(names(which.max(table(round(q.original, 1))))))
cutpoints <- sort(circular(mx + c(pi/4, 3*pi/4, 5*pi/4, 7*pi/4)) %% (2*pi))
axial <- findInterval(q.original, cutpoints) %in% c(1,3)
q.axial <- circular(2 * k.1[,-c(1,2)]) %% (2*pi)
```

We now carry out estimation *etc* for each of the two axial components, using angles reduced by perpendicularity.

```
bc.A <- bc.sample.statistics(q[axial])
vm.A <- mle.vonmises(q[axial], bias = T)
vm.A$mu <- vm.A$mu %% (2*pi)
jp.A <- JP.mle(q[axial])
jp.ci.A <- JP.ci.nt(jp.A, alpha = 0.05)
```

```
bc.B <- bc.sample.statistics(q[!axial])
vm.B <- mle.vonmises(q[!axial], bias = T)
vm.B$mu <- vm.B$mu %% (2*pi)
jp.B <- JP.mle(q[!axial])
jp.ci.B <- JP.ci.nt(jp.B, alpha = 0.05)
```

Uniformity and symmetry tests produce predictable results

```
rayleigh.test(q[axial]); rayleigh.test(q[!axial])
```

```
##
##        Rayleigh Test of Uniformity
##        General Unimodal Alternative
##
## Test Statistic:  0.3537
## P-value:  0
```

```
##
##        Rayleigh Test of Uniformity
##        General Unimodal Alternative
##
## Test Statistic:  0.3337
## P-value:  0
```

```
kuiper.test(q[axial]); kuiper.test(q[!axial])
```

```
##
##        Kuiper's Test of Uniformity
##
## Test Statistic:  3.1206
## P-value < 0.01
##
```

```
##
##        Kuiper's Test of Uniformity
##
## Test Statistic:  3.8999
## P-value < 0.01
##
```

```
watson.test(q[axial]); watson.test(q[!axial])
```

```
##
##        Watson's Test for Circular Uniformity
##
## Test Statistic: 0.7984
## P-value < 0.01
##
```

```
##
##        Watson's Test for Circular Uniformity
##
## Test Statistic: 1.0378
## P-value < 0.01
##
```

```
r.symm.test.stat(q[axial]); r.symm.test.stat(q[!axial])
```

```
## $test.statistic
## [1] 0.3154254
##
## $p.val
## [1] 0.7524387
```

```
## $test.statistic
## [1] 1.027515
##
## $p.val
```

```
## [1] 0.3041781
```

Now compare distributions for the two axes.

```
q.samples <- list(q[axial], q[!axial]);
q.sizes <- c(length(q[axial]), length(q[!axial]))
watson.common.mean.test(q.samples)
```

```
## $Y.g
## [1] 0.178242
##
## $p.val
## [1] 0.6728884
##
## $disp.ratio
## [1] 1.087085
```

```
wallraff.concentration.test(q.samples)
```

```
## $p.val
## [1] 0.5905906
##
## $result
##
##  Kruskal-Wallis rank sum test
##
## data:  distdat and g.id
## Kruskal-Wallis chi-squared = 0.28942, df = 1, p-value = 0.5906
```

```
mww.common.dist.LS(cs.unif.scores(q.samples), q.sizes)
```

```
## $statistic
## [1] 0.002721365
##
## $p.val
## [1] 0.9986402
```

```
watson.two.test(q[axial], q[!axial])
```

```
##
##       Watson's Two-Sample Test of Homogeneity
##
## Test Statistic: 0.0565
## P-value > 0.10
##
```

```
wtt.output <- watson.two.test.rand(q[axial], q[!axial], NR = 999)

JP.GoF.output1 <- JP.GoF.boot(q[axial], jp.mle$mu, B=999)
JP.GoF.output2 <- JP.GoF.boot(q[!axial], jp.mle$mu, B=999)
```

```
wtt.output
```

```
## $observed
## [1] 0.05647039
##
## $p.val
## [1] 0.623
```

```
JP.GoF.output1
```

```
##   kuiper   watson      rao rayleigh
##    0.666    0.501    0.001    0.382
```

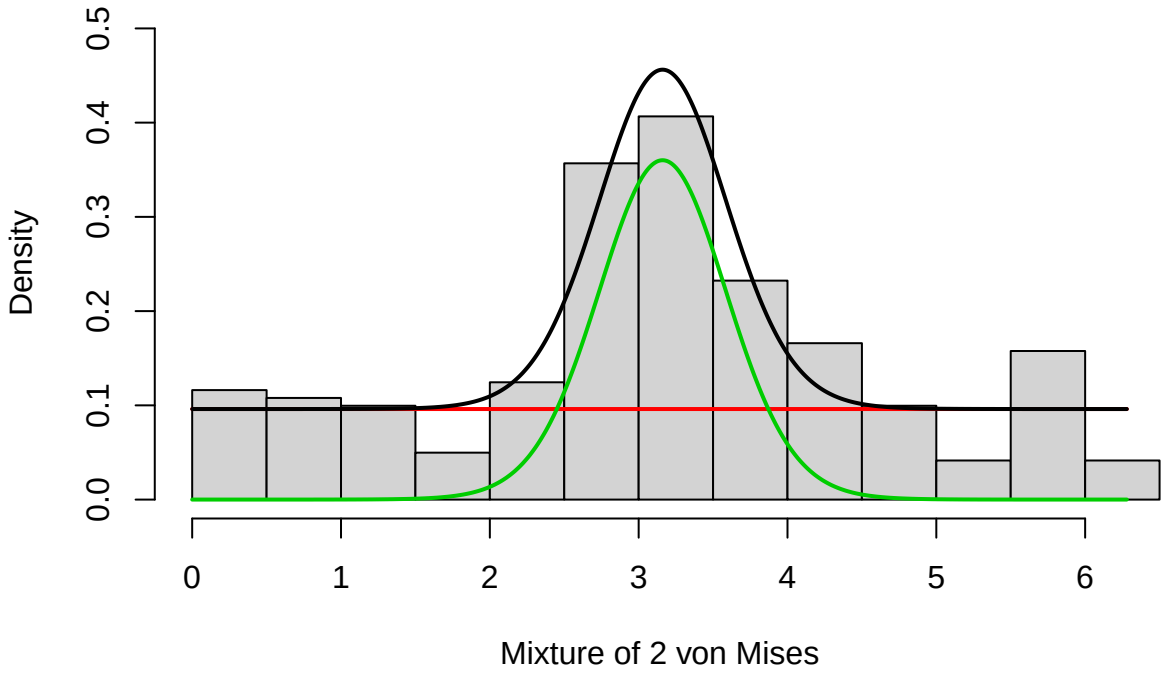
```
JP.GoF.output2
```

```
##   kuiper   watson      rao rayleigh
##    0.053    0.035    0.001    0.065
```

4.2 Fitting mixture of von Mises and Uniform

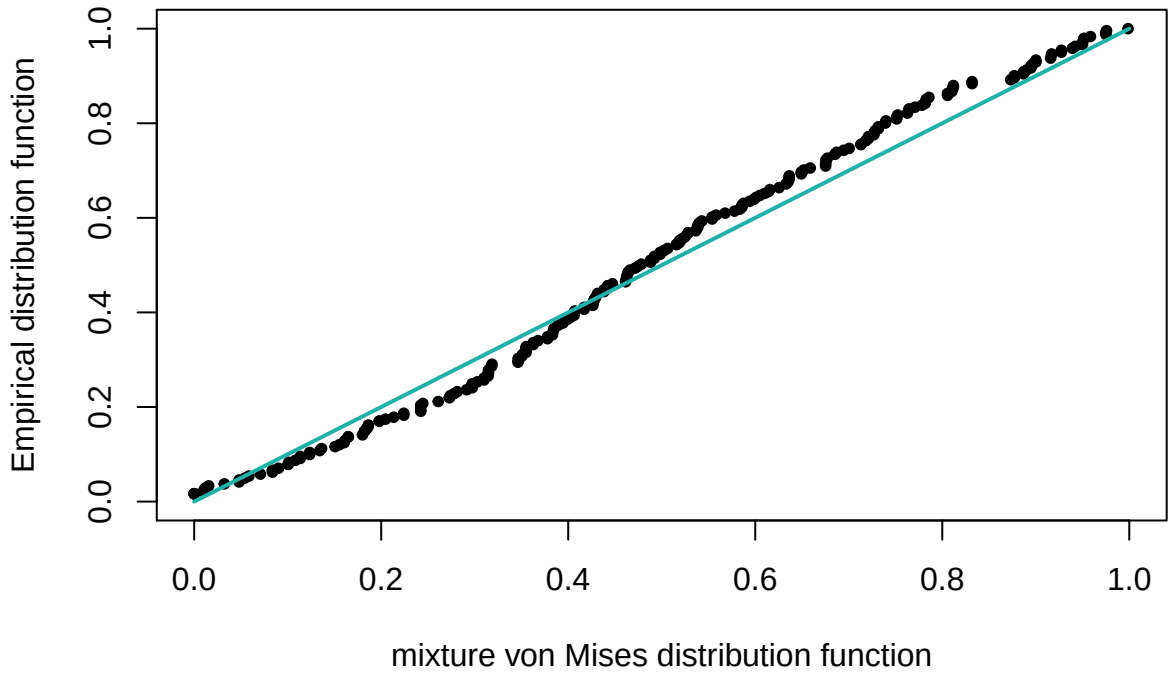
A uniform-von Mises mixture model is a natural (if slightly over-parametrized) way to adapt von Mises distributions to produce densities which are non-negligible everywhere. Note that we actually fit a mixture of two von Mises distributions: however one of them pretty much *has* to have zero concentration because observations are scattered over the entire range!

```
em.vm <- EM.vonmises(q, k=2)
em.u.vm <- EM.u.vonmises(q, k=2)
plot.EM.vonmises(q, em.u.vm)
```

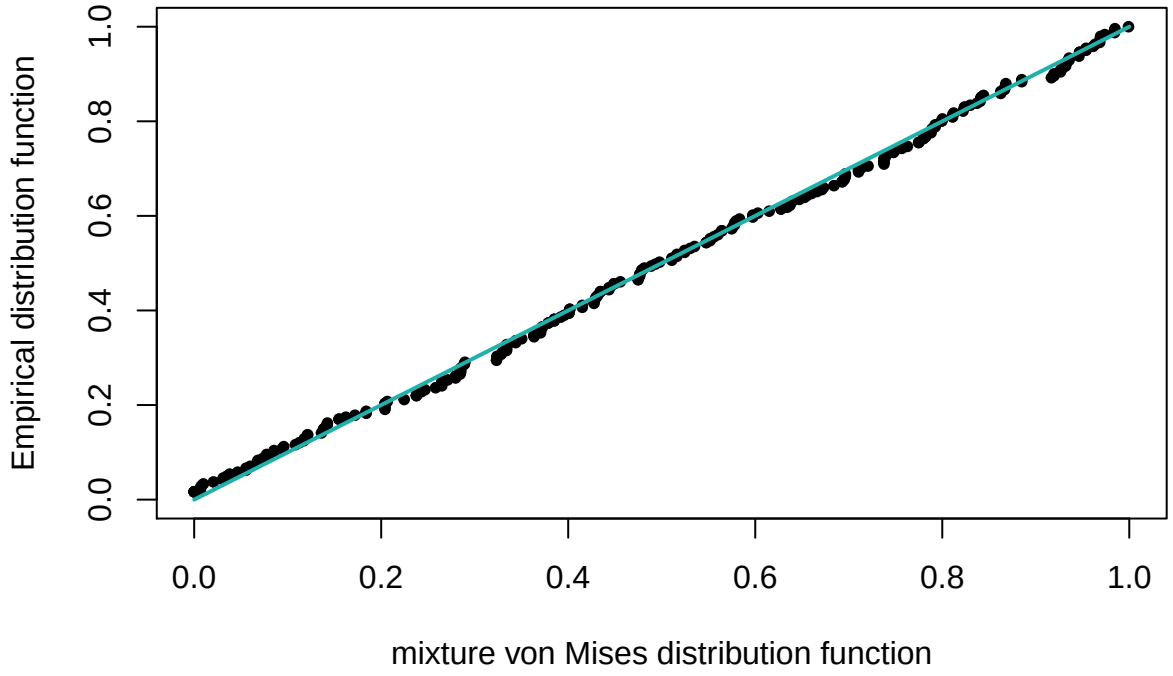


We now compare residuals

```
mvm.pp.res <- mvM.PP(q, em.vm$mu[1], em.vm$kappa[1], em.vm$mu[2],
                     em.vm$kappa[2], em.u.vm$alpha[1])
```




```
uvm.pp.res <- mvM.PP(q, em.u.vm$mu[1], em.u.vm$kappa[1], em.u.vm$mu[2],
                     em.u.vm$kappa[2], em.u.vm$alpha[1])
```




```
mean(uvm.pp.res^2); sd(uvm.pp.res^2)
```

```
## [1] 0.0001442629
```

```
## [1] 0.00015642
```

```
mean(mvm.pp.res^2); sd(mvm.pp.res^2)
```

```
## [1] 0.001391985
```

```
## [1] 0.001165677
```

```
mean(jp.pp.res^2); sd(jp.pp.res^2)
```

```
## [1] 0.0002916625
```

```
## [1] 0.0003083872
```

We compute AIC for each model, obtaining a marginal preference for the Uniform / von Mises mixture, which also has a satisfying interpretation (postholes either randomly placed or orientated with respect to nearest neighbour according to a von Mises distribution).

```
n <- length(q)
```

Model	parameters	AIC
Jones-Pewsey	3	807
von Mises mixture (2)	5	804
Uniform / von Mises	3	802

Apply winner-takes-all clustering based on uniform-von Mises mixture .

```
mvM.clusters <- function(data, model) {
    components <- matrix(nrow = model$k, ncol = length(data))
    for (i in 1:model$k) {
        components[i, ] <- dvonmises(data, circular(model$mu[i]), model$kappa[i])
    }
    apply(components, 2, which.max)
}
em.clusts <- mvM.clusters(q, em.u.vm)
```

Test von Mises component for perpendicularity.

```
clust.a <- q[axial    & (em.clusts == 2)]
clust.b <- q[(!axial) & (em.clusts == 2)]
c.samples <- list(clust.a, clust.b); c.sizes <- c(length(clust.a), length(clust.b))

watson.common.mean.test(c.samples)
```

```
## $Y.g
## [1] 1.193124
##
## $p.val
## [1] 0.2747003
##
## $disp.ratio
## [1] 1.094136
```

```
wallraff.concentration.test(c.samples)
```

```
## $p.val
## [1] 0.5972423
##
## $result
##
##  Kruskal-Wallis rank sum test
##
## data:  distdat and g.id
```

```
## Kruskal-Wallis chi-squared = 0.27918, df = 1, p-value = 0.5972
```

```
mww.common.dist.LS(cs.unif.scores(c.samples), c.sizes)
```

```
## $statistic
## [1] 3.069965
##
## $p.val
## [1] 0.2154595
```

```
watson.two.test(clust.a, clust.b)
```

```
##
##       Watson's Two-Sample Test of Homogeneity
##
## Test Statistic: 0.1176
## P-value > 0.10
##
```

```
output <- watson.two.test.rand(clust.a, clust.b, NR = 999)
```

```
wtt.output
```

```
## $observed
## [1] 0.05647039
##
## $p.val
## [1] 0.623
```

```
# check confidence intervals for rho
bc.vm.a <- bc.ci.LS(clust.a, alpha = 0.05)
bc.vm.b <- bc.ci.LS(clust.b, alpha = 0.05)
bc.vm.a; bc.vm.b
```

```
## $alpha
## [1] 0.05
##
## $symmetric
## [1] FALSE
##
## $mu
## estimate    lower    upper
## 3.185210 3.075392 3.295029
##
## $rho
##  estimate     lower     upper
## 0.9160136 0.8933017 0.9387254
##
## $beta2
##     estimate        lower        upper
##  0.002978988 -0.018365441  0.024323417
##
## $alpha2
##  estimate     lower     upper
## 0.6935936 0.6138944 0.7732928
```

```
## $alpha
## [1] 0.05
```

```
##
## $symmetric
## [1] FALSE
##
## $mu
## estimate    lower    upper
## 3.104609 3.009851 3.199366
##
## $rho
##  estimate     lower     upper
## 0.9226513 0.9028983 0.9424044
##
## $beta2
##     estimate        lower        upper
## -0.013469033 -0.031201858  0.004263793
##
## $alpha2
##  estimate     lower     upper
## 0.7169409 0.6472601 0.7866217
```

Now we assess the degree to which there is gridding, and use `dbscan` to split into spatial clusters.

```
plot(pts[em.clusts == 1,], pch = 20, col = "grey", xlab="x", ylab="y")
points(pts[em.clusts == 2,], pch = 20, col = "black")

db.clust <- dbscan(pts, MinPts = 4, eps = 4.65)$cluster
points(pts, col = db.clust + 1)
```

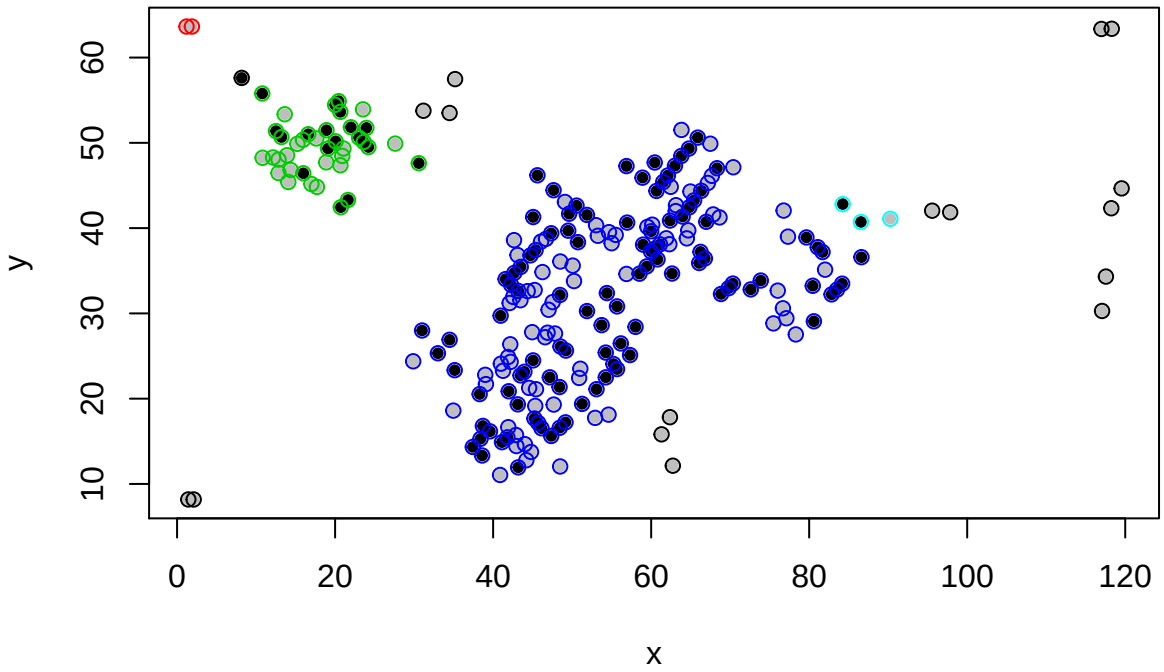


Figure 5: Black dots show points which appear to be part of aligned grid structure.

```
plot(pts, lwd = 2, xlab = "", ylab = "",
     col = c("grey", "red", "blue", "black", "green3", "magenta")[db.clust + 1],
     pch = c(25,25,22,21,23,24)[db.clust+1],
```

```
    bg = c("white", "black")[em.clusts])
```

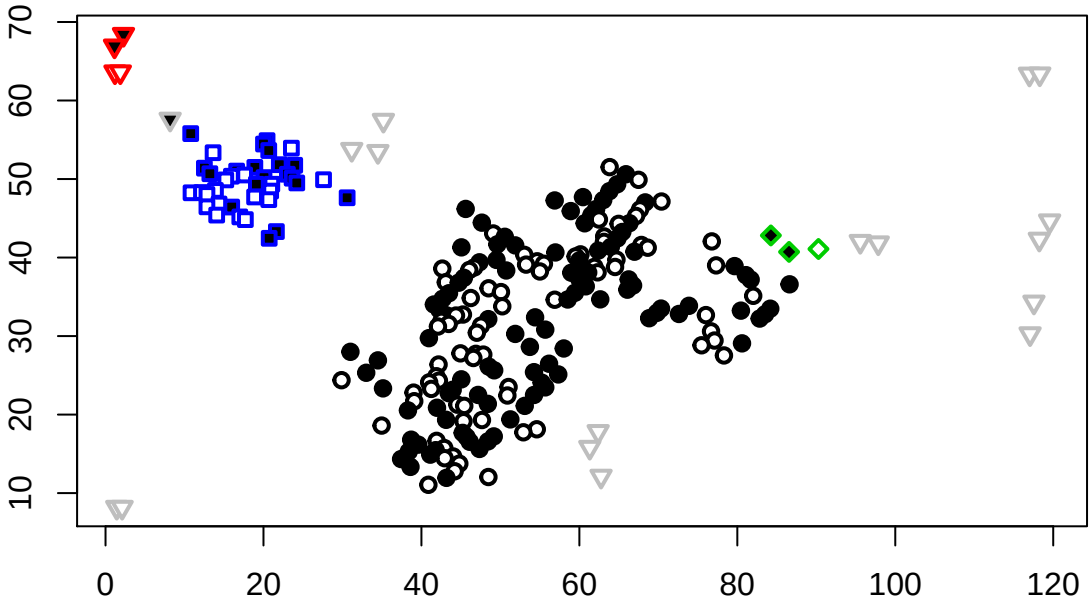


Figure 6: Black dots show points which appear to be part of aligned grid structure.

Subset into density-based clusters and test for common orientation.

```
q.db1 <- q[db.clust == 1]
q.db3 <- q[db.clust == 3]
db.samples <- list(q.db1, q.db3); db.sizes <- c(length(q.db1), length(q.db3))
watson.common.mean.test(db.samples)
```

```
## $Y.g
## [1] 5.420073
##
## $p.val
## [1] 0.01990653
##
## $disp.ratio
## [1] 1.932678
```

```
wallraff.concentration.test(db.samples)
```

```
## $p.val
## [1] 0.2009292
##
## $result
##
##  Kruskal-Wallis rank sum test
##
## data:  distdat and g.id
## Kruskal-Wallis chi-squared = 1.6356, df = 1, p-value = 0.2009
```

```
mww.common.dist.LS(cs.unif.scores(db.samples), db.sizes)
```

```
## $statistic
## [1] 3.355248
```

```
##
## $p.val
## [1] 0.1868173
```

```
watson.two.test(q.db1, q.db3)
```

```
##
##       Watson's Two-Sample Test of Homogeneity
##
## Test Statistic: 0.1325
## P-value > 0.10
##
```

```
wtt.output <- watson.two.test.rand(q.db1, q.db3, NR = 999)
```

```
wtt.output
```

```
## $observed
## [1] 0.1324959
##
## $p.val
## [1] 0.153
```

References

Clair Barnes. *Statistics in Anglo-Saxon Archaeology*. Msc dissertation, University of Warwick, 2015.