- In movie.peru viewing time is proportional to the time measured between earthquakes. How was that done in the code?? (comment that section)
- mystery9 and mystery2 are in fungraphs.pck
- Try everything but lattice to search for interesting plots in both of them (animate first no more than 5 minutes, then pairs)
- For mystery2 use the proactive mvntest then orthogonalize and use explore (also the asymmetry search)
- If anything like an animal appears google the name of that animal with the words “effect”
- Describe what you saw, describe what you found when you googled the … effect, and describe how you think the curse of dimensionality may have kept you from seeing it with animate quickly without starting values.

Movie.peru comments:

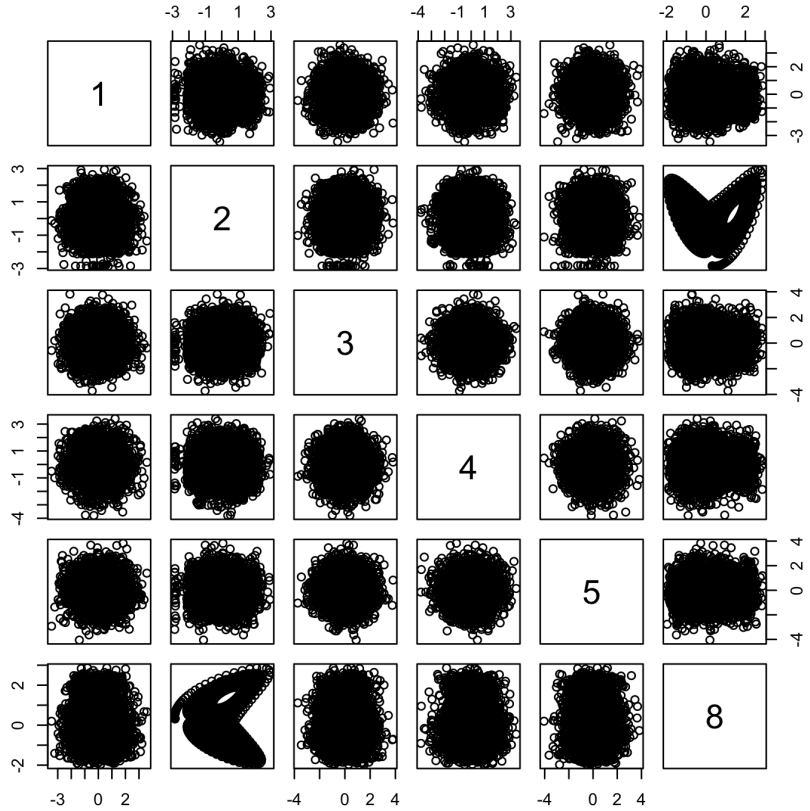
```
function(depth=350,depth.nw=200, dis.vec=c(8:13))
{
library(scatterplot3d)
sctstr<-scatterplot3d(Peru[,c(2,3,4)],type="n")
IDmat<-Peru[,24:27]
IDmat[,2]<-IDmat[,2]*2
IDmat[,3]<-IDmat[,3]*3
IDmat[,4]<-IDmat[,4]*4
ztime<-Peru[,1]
nlag=20
#loop through time with window, id points pi
nz<-length(ztime)-nlag
for(i in 1:nz){
v9<-c(i:(i+nlag))
icolvec<-apply(IDmat[v9,],1,sum)
#this loop sets delay in movie

#the time between earth quakes is stored in the object time.count and it is calculated by subtracting the
 time of the current earthquake by the time of the prior earthquake
time.count<-floor(1000*(ztime[i+nlag]-ztime[i+(nlag-1)]))
#the following nested loops delay the i loop (that runs to plot each sequential earthquake ) essentially
 putting it on hold for an amount of time proportional to the time between earthquakes since the first
 nested for loop j runs 2*timecount and runs the loop k 100 times every iteration.
for(j in 1:(2*time.count)){
v1<-rnorm(100)
for(k in 1:100){
 v2<-sample(v1,replace=T)
 mean(v2)
}
}
#the following plots the earthquake after the nested forloops j and k have completed their iterations
 causing the proportional delay to have been completed
#Perumat<-cbind(Peru[,c(1,2,3)],sqrt(Peru[,4]))
sctstr<-scatterplot3d(Peru[,c(2,3,4)],type="n")
sctstr$points3d(Peru[v9,c(2,3,4)],col=icolvec,type="h",lwd=((exp(Peru[v9,5]-4))))
#print(c(i,nz,ztime[i],ztime[nz]))

}
print("DONE")
}
```

Exploration of data frame mystery9:

Console Prompts:

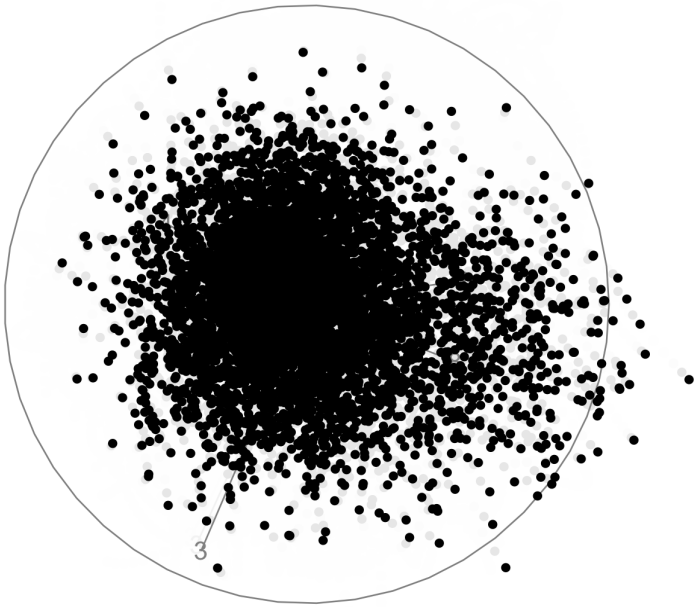
source("~/Downloads/fungraphs.pck")

pairs(mystery9)



library(tourr)

animate(mystery9)

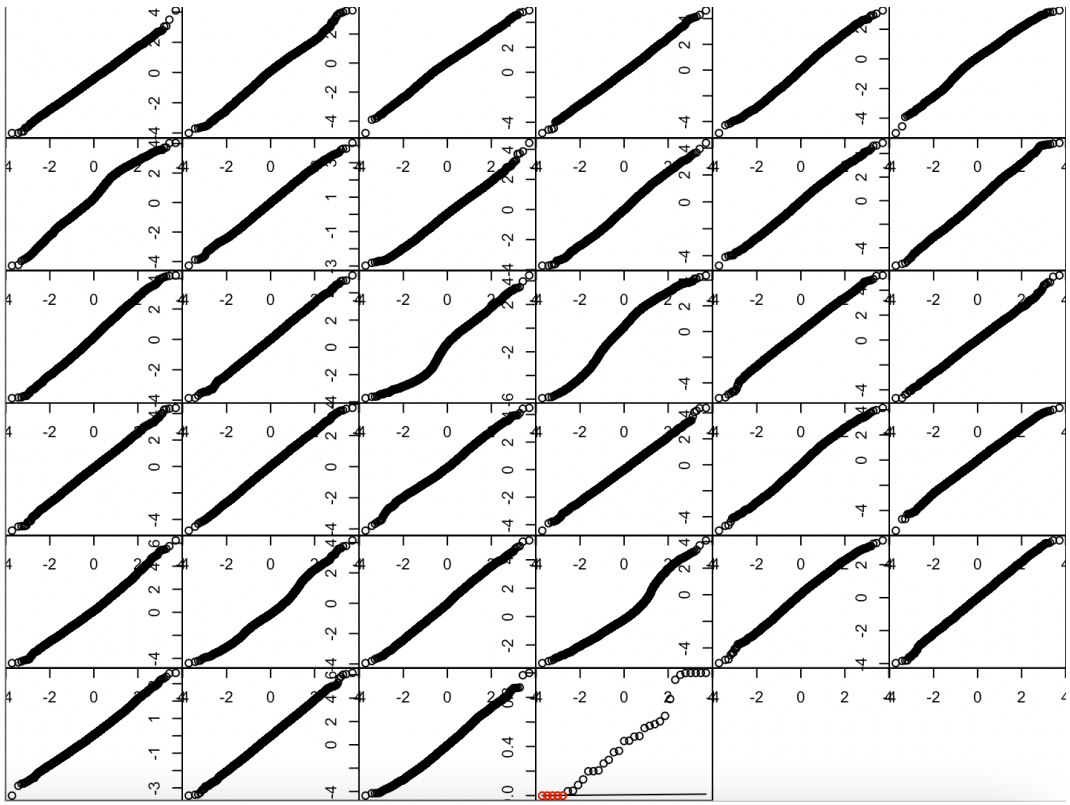


Exploration of data frame mystery2:

Console Prompts:

source("~/Downloads/fungraphs.pck")
mvntest<-Multivariate.normal.test(mystery2,35,0.05)
mvntest

Plot:



Console Output:

$Interesting.directions

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]
u1	0.30712968	0.71334238	0.41967864	0.14379413	-0.2782679	0.2163839	0.2696959	0.05491946
u1	-0.29306857	0.09755281	-0.07706256	0.63256189	0.1678926	-0.3611475	0.4973772	0.30417371
u1	-0.40497863	0.53073177	-0.22156204	-0.04190968	0.3736964	0.1022572	0.2175083	0.55322197
u1	0.08121823	0.23215888	0.15668125	-0.08838338	0.6377514	-0.1571264		
u1	-0.11447560	-0.26415027	-0.64956619	0.15585522	-0.4774561	-0.3042832		

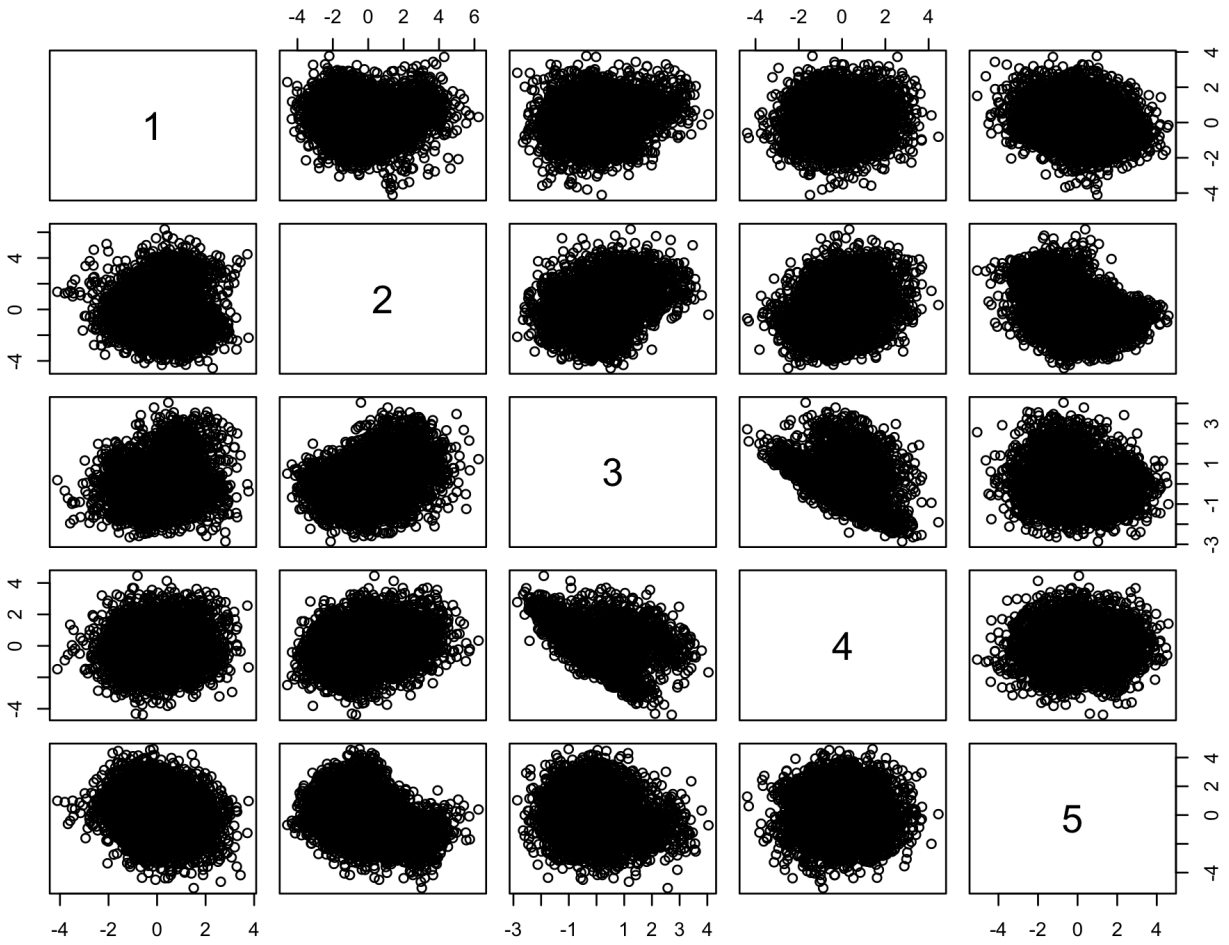
u1 -0.6358728 0.26719932
u1 -0.2774496 -0.27085417

$rejectingplots
[1] 21 26 28 25 7

$pvals
 [1] 0.4447 1.0000 0.9987 0.4476 0.2914 1.0000 0.0002 0.5823 0.9440 0.0865
[11] 0.6046 0.3649 0.1979 0.1993 1.0000 0.9995 0.3538 0.6501 0.5692 0.7869
[21] 0.0000 0.2052 0.4806 0.5503 0.0001 0.0000 0.1319 0.0000 0.9847 0.4853
[31] 0.0395 0.2622 0.0358

Console Prompts:
orthmat<-orthogonalize(mvntest$Int[1:5,])
expl<-explore(orthmat,mystery2,T,F,F)

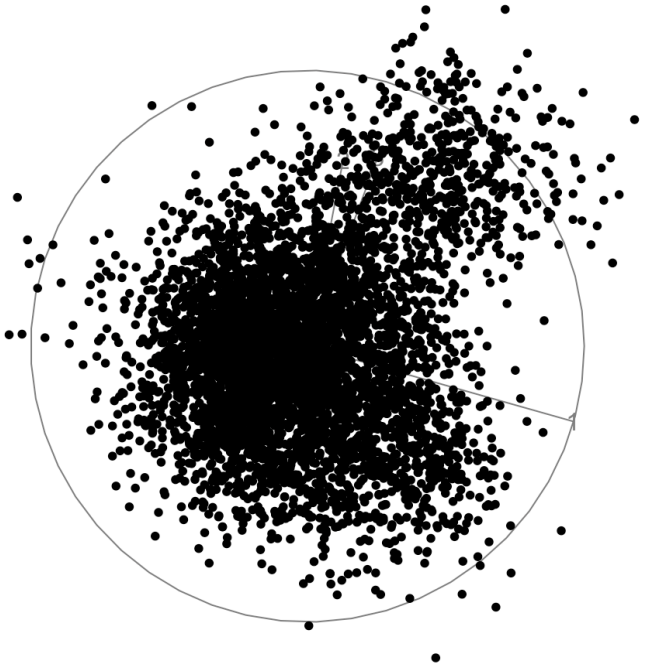
Pairs Plot:



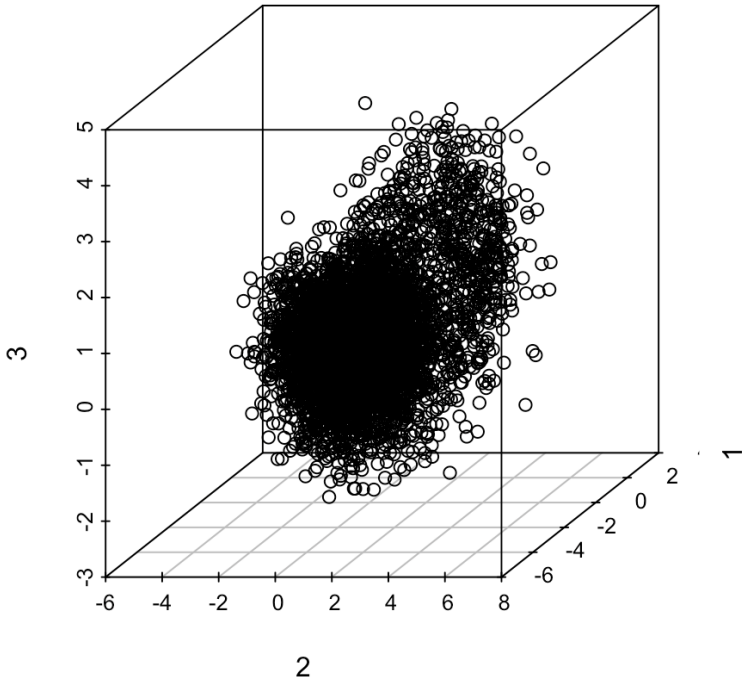
Console Prompts:

```
orthmat<-orthogonalize(mvntest$Int[1:3,])
expl<-explore(orthmat,mystery2,F,F,T)
expl<-explore(orthmat,mystery2,F,T,F,c(2,1,3))
```

Animated Plot:



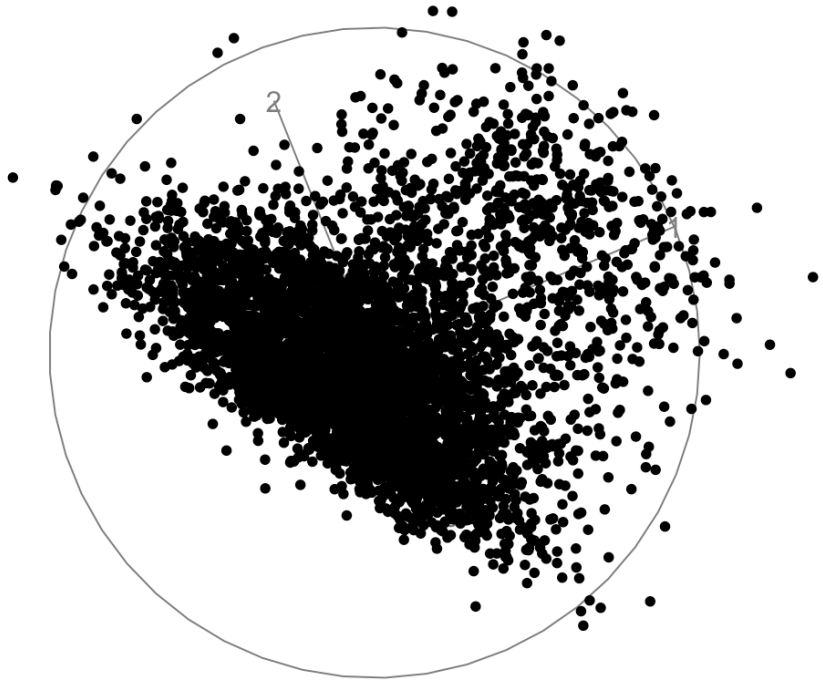
3d Plot:



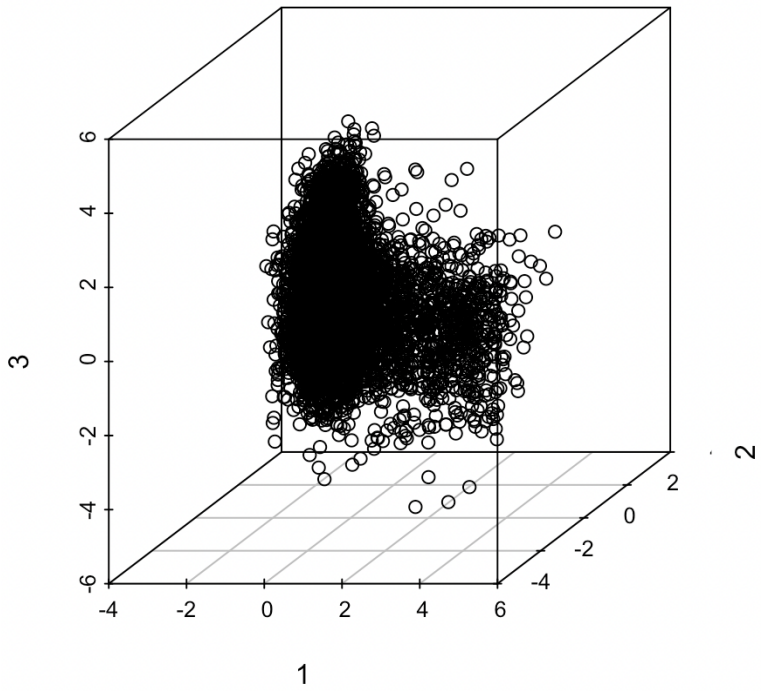
Console Prompts:

```
orthmat<-orthogonalize(mvntest$Int[3:5,])
expl<-explore(orthmat,mystery2,F,F,T)
expl<-explore(orthmat,mystery2,F,T,F,c(1,2,3))
```

Animated Plot:



3d Plot:



Observations:

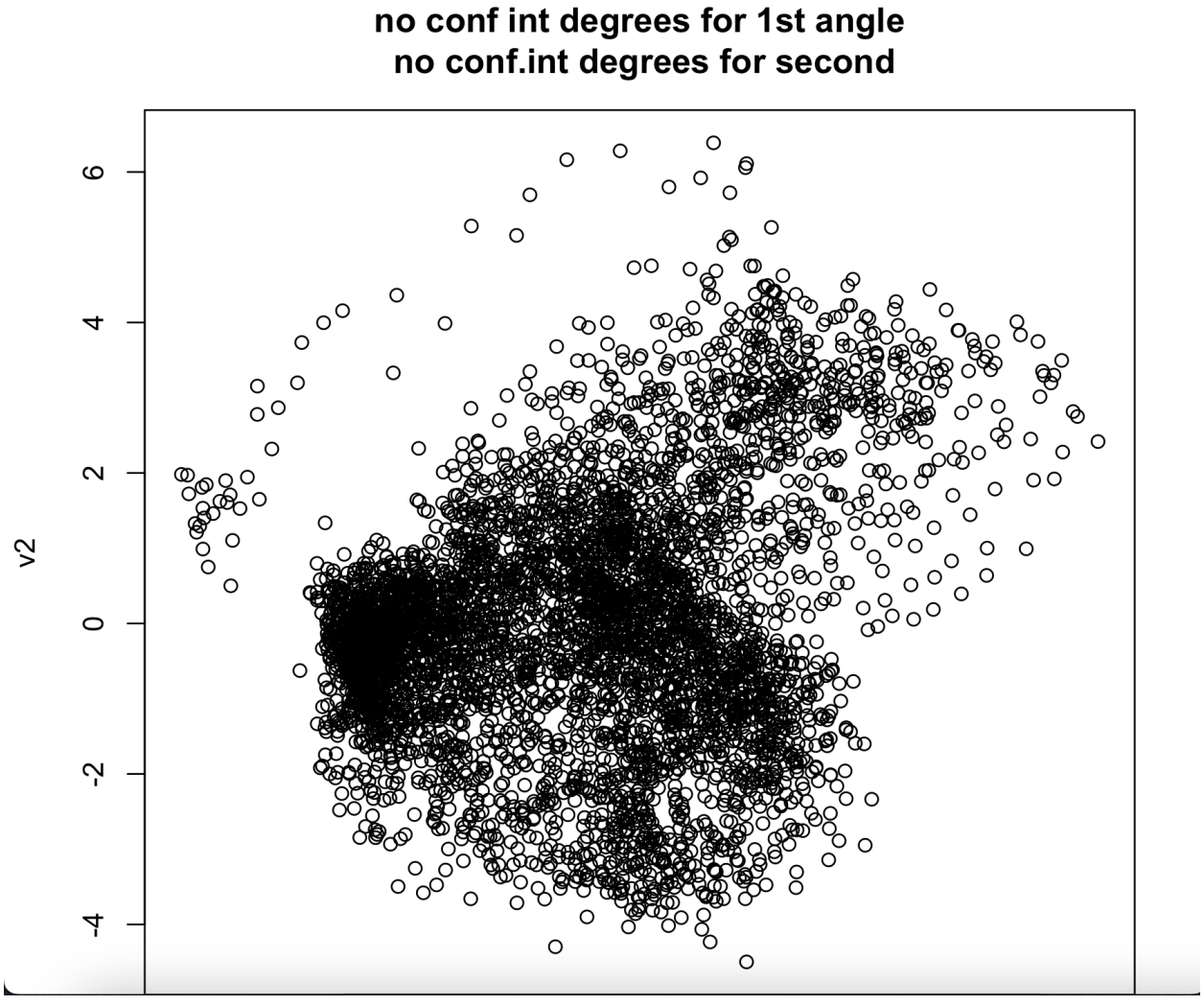
The pairs, animated, and 3d scatter plots that are produced by exploring the orthogonalized vectors of the 5 interesting directions in Riya's 33 multivariate normal test qq plots, are particularly interesting because of their unique shapes, specifically the dimpled plots and harsh flat plane where the points are seemingly contained by when plotting the 3,4 and 5 orthogonalized vectors. As the 3d animated plots rotate they never seem to look like a multivariate normal distribution, they don't resemble ellipses, most of the pair plots are dimpled and abstract.

The following is Asym plots part of the quiz

Console Prompts:

asym.contamination.plot(mystery2,0,c(.3.8),0.95,T,F,F)

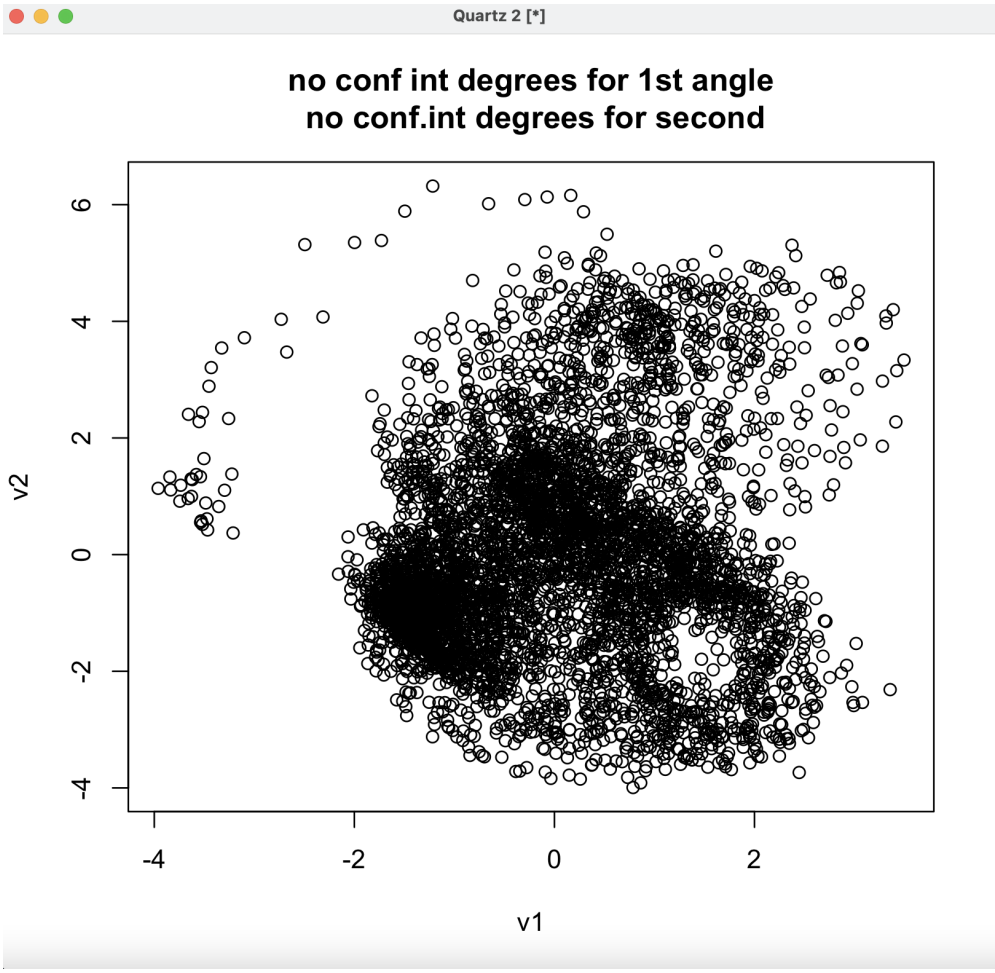
Plot:



Console Prompts:

asym.contamination.plot(mystery2,0,c(.2,.7),0.95,T,F,F)

Original Plot before outlier removal and reanalysis:



Console Prompts:

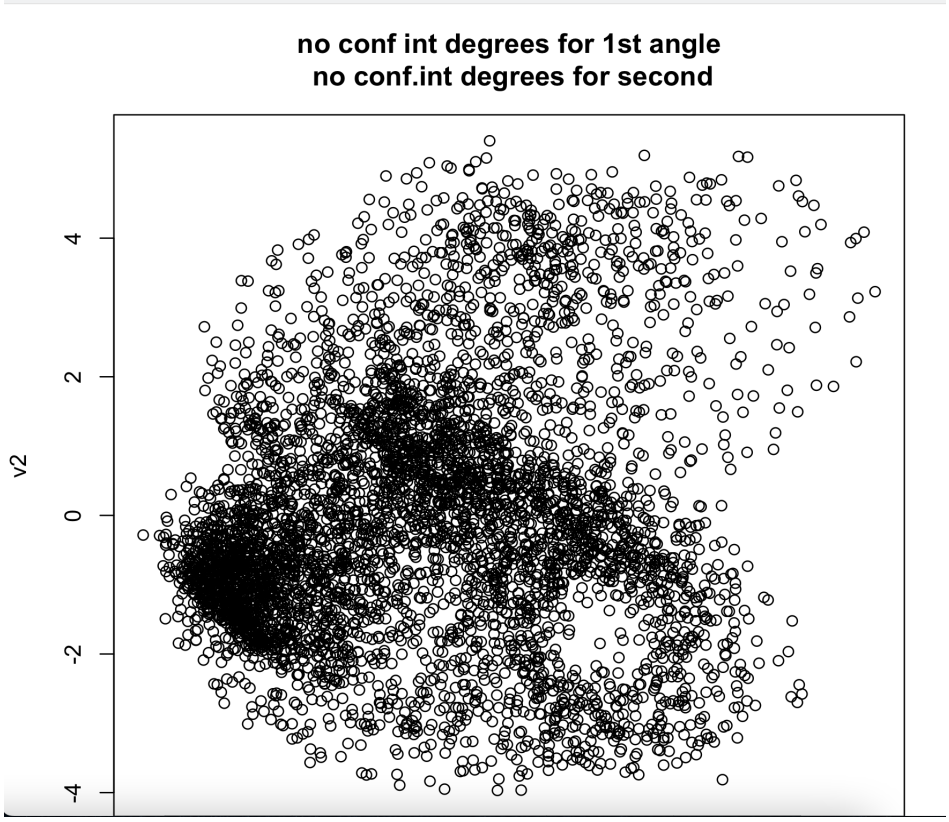
> asym.contamination.plot(mystery2,0,c(.2,.7),0.95,T,F,F)
[1] "Would you like to identify the points to remove, then reanalyze? (Y/N)"
1: Y
2:
Read 1 item
[1] "How many points?"
1: 10
2:
Read 1 item
 [1] 2 3 5 11 14 16 18 20 22 26
 [1] 0.013711967 0.030862223 0.058003982 0.001773928 0.027179230 0.005816341
 [7] 0.083702428 0.019587917 0.008757164 0.029282589
[1] "Would you like to identify the points to remove, then reanalyze? (Y/N)"

1: Y
2:
Read 1 item
[1] "How many points?"
1: 10
2:
Read 1 item
 [1] 4 5 6 7 10 12 13 14 15 16
 [1] 0.073294492 0.035265250 0.041120786 0.042349727 0.054262670 0.048015035
 [7] 0.047444191 0.001118382 0.062572459 0.026313785
[1] "Would you like to identify the points to remove, then reanalyze? (Y/N)"
1: Y
2:
Read 1 item
[1] "How many points?"
1: 10
2:
Read 1 item

 [1] 1 3 5 6 7 8 9 10 11 12
 [1] 0.002391798 0.043400945 0.039004783 0.036496441 0.014576318 0.019126503
 [7] 0.011305204 0.009812422 0.002081828 0.006790994
[1] "Would you like to identify the points to remove, then reanalyze? (Y/N)"
1: Y
2:
Read 1 item
[1] "How many points?"
1: 10
2:
Read 1 item
 [1] 1 2 3 4 5 6 7 8 9 10
 [1] 0.0314576585 0.0307166791 0.0013724388 0.0003307611 0.0001052552
 [6] 0.0050423727 0.0019248893 0.0055497448 0.0107145275 0.0158115804
[1] "Would you like to identify the points to remove, then reanalyze? (Y/N)"
1: Y
2:
Read 1 item
[1] "How many points?"
1: 4
2:

Read 1 item
[1] 1 2 3 1426
[1] 0.003632323 0.001469993 0.005946708 0.111917909
[1] "Would you like to identify the points to remove, then reanalyze? (Y/N)"
1: N

Final Plot:



Observations:

Changing the trims seems to rotate the function slightly after looking after different plots with different trims, and the larger the trim the less outliers there appear to be. Either way the shape produced reminds me of a butterfly. After the outlier removal and reanalysis, the plot showed a more distinctive image. The plots produced via the mvn test method vs the asym method are both non normal distributions, however the asym plots produce a more distinct image and the ability to reanalyze with the removal of outliers. After searching up the Butterfly effect, i found that “in chaos theory, the butterfly effect is the sensitive dependence on initial conditions in which a small change in one state of a deterministic nonlinear system can result in large differences in a later state.” The curse of dimensionality meant that if we attempted the mvntest and only plotted the normal directions we would have been unable to see the asymmetry inherent in the

image produced by the asym plot and by the mvntest when we selectively plotted the non normal interesting direction!