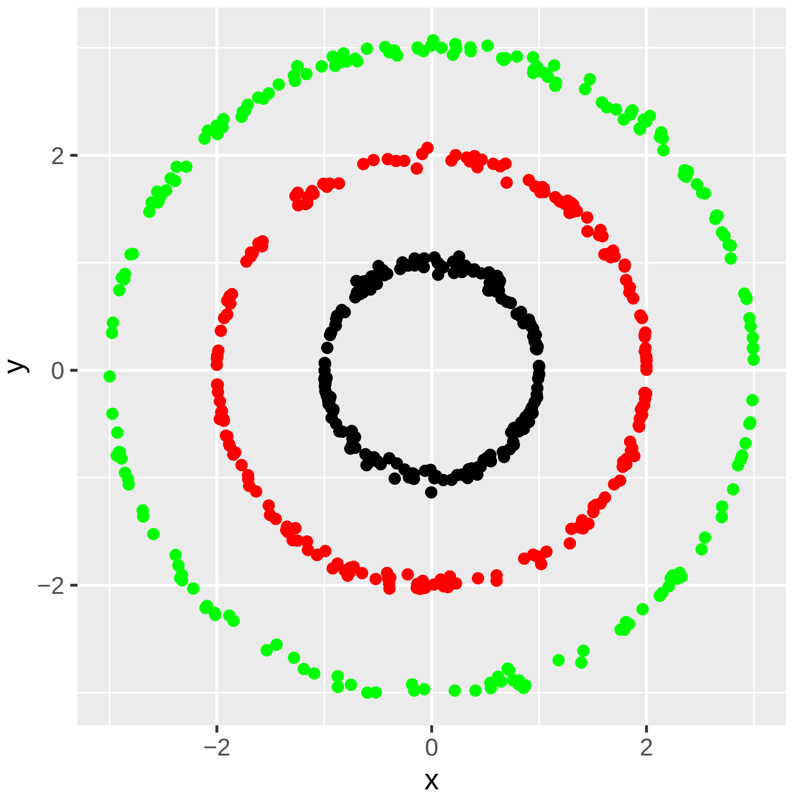
Project1

Anton Holm

2021-02-02

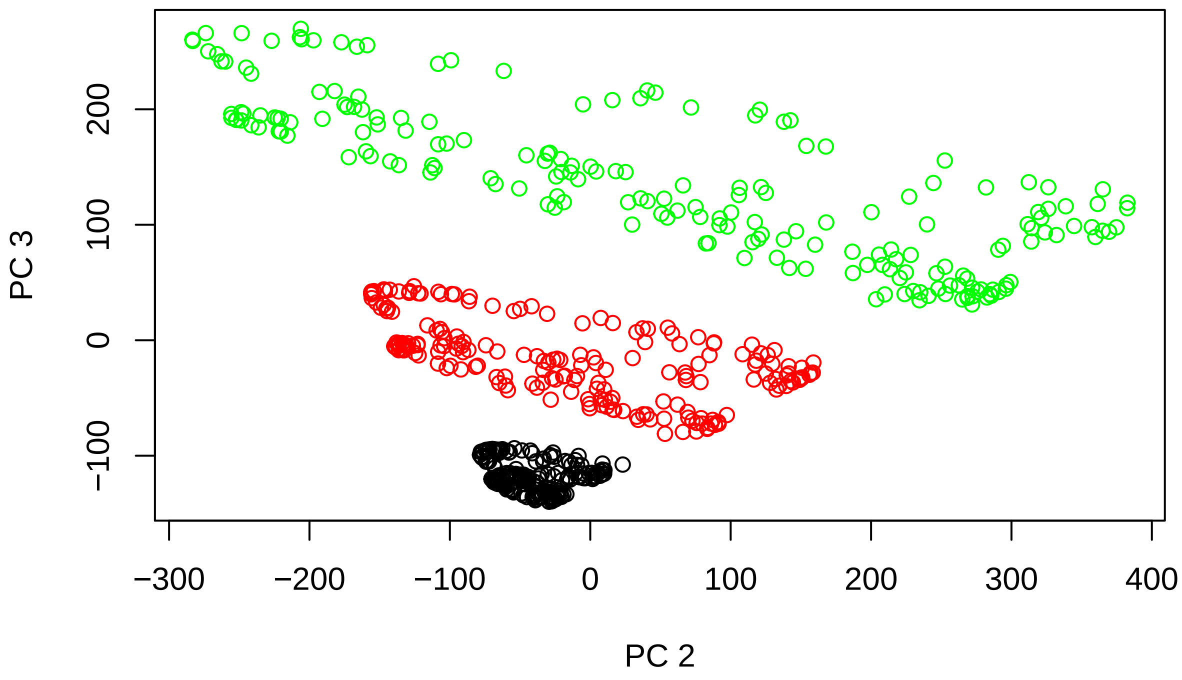
Below is a figure showing a dataset och 600 datapoints in a concentric circle. The data was generated using polar coordinates with three different radi, 1, 2, and 3 with equal distribution of points between each circle.

Figure 1: 2−Dim concentric dataset illustrated



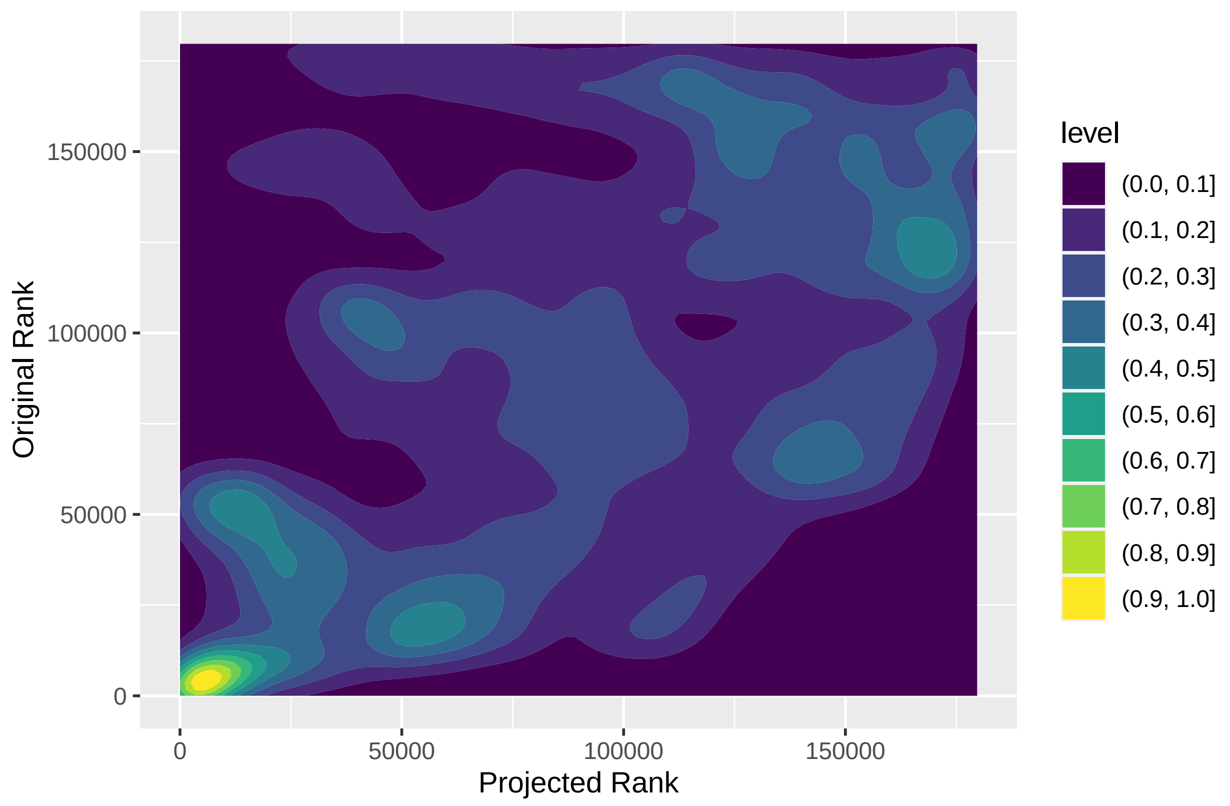
Below is the plot for the data when plotted on the second and third principle component after performing kernel PCA with a polynomial kernel with degree 2. Here I also set `scale` to be equal to 2 since this made the linear seperability more clear while `offset` is equal to 1. The `scale` variable is simply a scalar infront of the scalar products, i.e. if k is the polynomial kernel we have $k(x, x') = (\text{scale} < x, x' > + \text{offset})^{\text{degree}}$. We can see that the data clearly is linearly seperable in this case.

Figure 2: Data on projected space



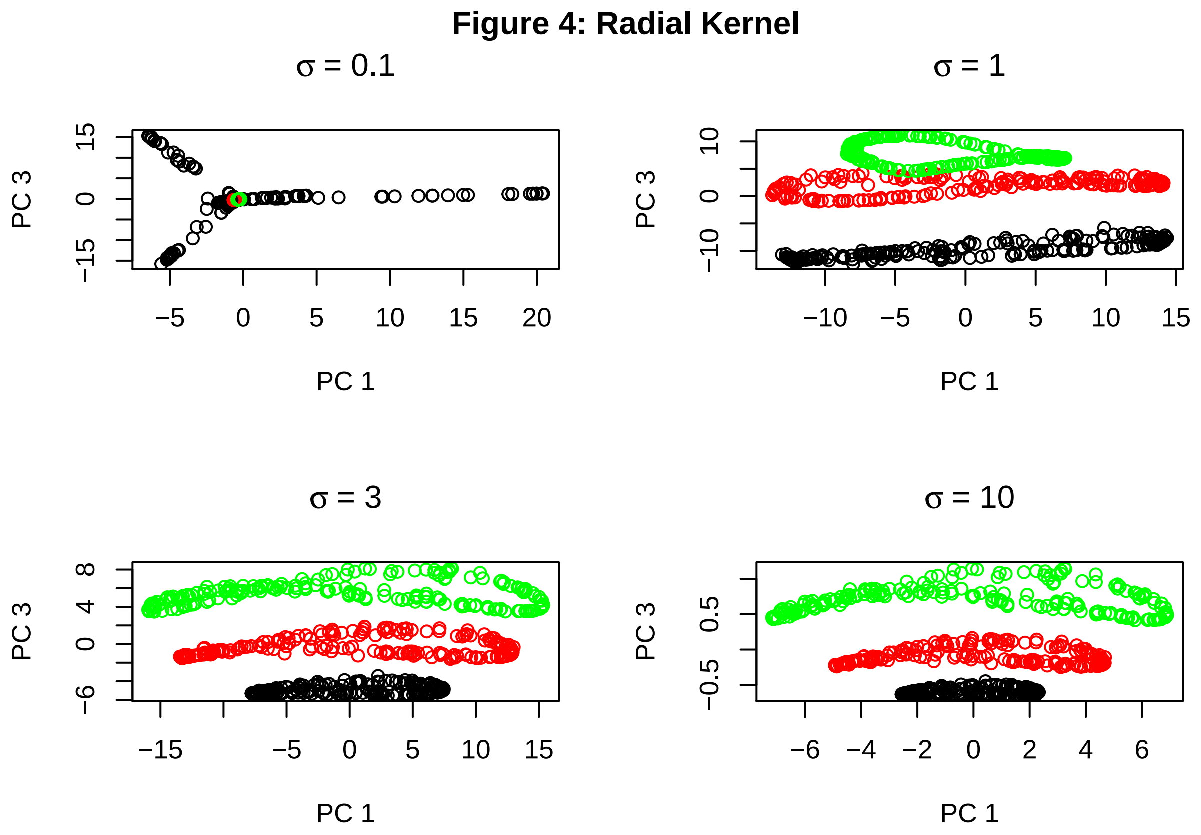
In order to validate our results we use the Shepard rank diagram shown in Figure 3. Due to the immense number of datapoints to plot I use a 2-D density plot where the density levels are scaled to lie between 0 and 1 instead of being around $k10^{-10}$ for some integer k to make the graph more pleasent for the eyes. If the ranks would have been exactly the same in the original and projected space, all of the density in Figure 3 would have been on the $y = x$ axis. The more density we have close to this line, the better the projection respects the rank while if we have more density far away from this line the ranks are changed more. If all points lie on the $y = -x + m$ for some fitting intercept m, then we would have a correlation coefficient of -1 and as such, the projection is very bad. We can see that the points which were the most similar in the original space (low rank) are also those with the smallest euclidean distance in the projected space and as such the projection seems to respect the pairwise distance well for the points being most similar in the original space. For ranks above 25.000 (maybe even smaller), we can see that the density is very spread out, showing us that the ranks does not remain the same in the projected space.

Figure 3: Density plot of ranks in actual vs projected space



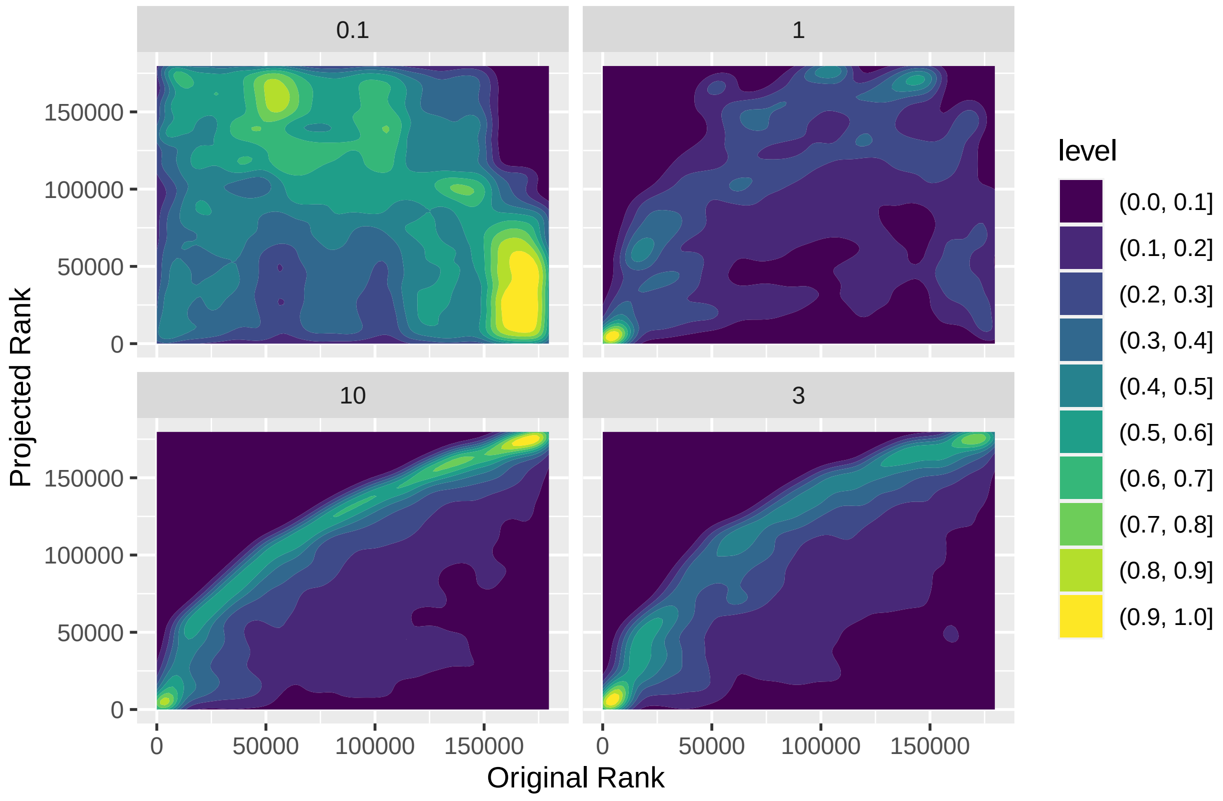
We now move on and try to perform kpca using a radial kernel instead. One of the main focus is going to be to test out different values of σ and see how this affects the results. In Figure 4 we see the data plotted on the principle component 2 and 3 after performing kpca using a radial kernel with σ to be 4 different values. The radial kernel is defined as $K(x, x') = exp(-\frac{||x-x'||^2}{2\sigma^2})$. So, if we have a large σ, that would mean that we have a high variance in the kernel and as such, even though two points in the projected space lie far away from eachother, they might be similar and only be far away due to this variance. We can also see this in the expression of the kernel. If σ is very large, then this term will have so much weight that the distance between two points in the original space will have very little say in deciding if two points are similar in the projected space. If we send σ to infinity, then the kernel between two points will always be one, and as such, we can no longer distinguish if points were similar or not in the original space by looking at the points projected in the new space, i.e. the variance is too high and the points will look uniformly distributed in the projected space. On the other hand, if σ is small, that mean that the variance in the kernel is now small. So only if two points are close in the projected space they will be deemed similar. Since σ is low, we see in the definition of the kernel that the distance in the original space now has more weight instead. However, if we pick σ too small we will run into problems. If we send σ to zero, this mean that for any two points, the kernel will be zero unless two points have the same coordinates in the original space (then the kernel is equal to one). In our case, we see that σ being around 3 or 10 works quite well if we want the data to be linearly seperable in the projected space spanned by the PC 2 and 3. We see that when σ is small, the points starts to all converge towards zero in the projected space.

There is another phenomenon that occurs when σ grows larger. Since the variance in the kernel becomes bigger, there is more variance in the data to explain. As such, the `kpca` function in `R` picks up more and more principle components that has a non-zero eigenvalue. The eigenvalues of those PC's found for smaller σ grows smaller while the more of eigenvalues that previously were zero grows larger and all the non-zero eigenvalues becomes more similar. This mean that as we pick larger values for σ, we introduce more variance that the principle components needs to explain while the PC's that explained much of the variance before, now explain much less thus resulting in us needing more principle components to explain the variance in the data on the projected space.



We now perform validation on the above kpca using Shepard rank diagrams. We can see in Figure 5 that for $\sigma = 3$ or $\sigma = 10$, the rank seem to be preserved better than for the smaller σ's. The majority of the data lie around the $y = x$ line altho having sort of a logarithmic structure. However, it does a much better job than when using a polynomial degree 2 kernel. When we decrease σ to be equal to 3 we can see that some of the structure from the when σ was equal to 10 is preserved, especially for the lower ranks. As σ becomes smaller we see that we are getting close to the $y = -x + m$ line discussed earlier, showing us that we probably have a negative correlation here. So for the smallest σ, the ranks are not preserved and even looks to be flipped, i.e. two points that had a high rank in the original space now have a low rank in the projected space and vice versa.

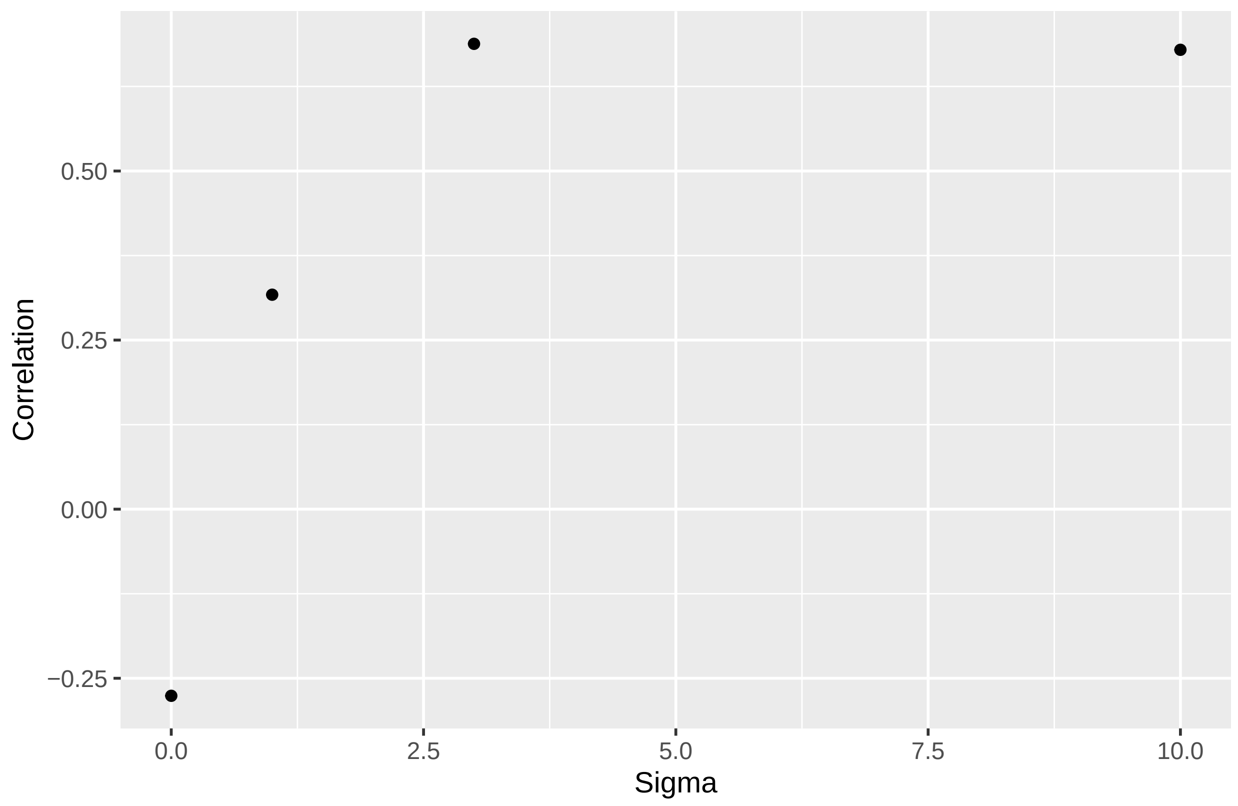
Figure 5: Density plot of ranks in actual vs projected space



Finally, we plot the correlation as a function of the value of σ. As mentioned above, we can see that for σ at 3 or 10, we have a much better correlation than for the smaller σ's. This makes sense considering Figure 5 showing this positive correlation, especially for the smallest and largest ranks. When σ becomes too small, we get a negative correlation instead which mean that the more similar two points are in the original space, the less similar they are in the projected space and vice versa. This is quite visible in Figure 5 upper left panel where we see that the points more resemble the $y = -x + m$ line than the $y = x$ line. This is a very bad property.

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

Figure 6: Correlation vs sigma



```r
set.seed(931031)
# Function to simulate data on a circle with noise using polar coordinates
circular_data <- function(n, r, sd){
radius = r
angle = runif(n, min = 0, max = 2*pi)
x = radius * cos(angle)
y = radius * sin(angle) + rnorm(n, 0, sd)
cbind(x,y)
}

# Simulate data from 3 concentric circles with radius 1,2, and 3
r1 <- as.data.frame(circular_data(200,1,0.05)) %>%
  mutate(radius = 1,
         color = "Black")
r2 <- as.data.frame(circular_data(200,2,0.05))%>%
  mutate(radius = 2,
         color = "Red")
r3 <- as.data.frame(circular_data(200,3,0.05))%>%
  mutate(radius = 3,
         color = "Green")

# Join all data
data <- full_join(r1,r2, by = c("x", "y", "radius", "color")) %>%
  full_join(r3, by = c("x", "y", "radius", "color")) %>%
  mutate(radius = as.factor(radius))
```

```r
#Plot the data
data %>%
  ggplot(aes(x = x, y = y, color = color)) +
  geom_point() +
  coord_fixed() +
  scale_color_identity(labels = c(1,2,3)) +
  labs(title = "Figure 1: 2-Dim concentric dataset illustrated")
```

```r
#Center the data
data_cent <- data %>%
  mutate(x = x - mean(x),
         y = y - mean(y))

#Perform kernel PCA with a polynomial kernel with degree 2
kpca_poly2 <- kpca(x = as.matrix(data_cent[,1:2]), kernel = "polydot", kpar = list(degree = 2, scale = 

#Plot the data on the principle component 1 and 3 to show linear seperability
plot(rotated(kpca_poly2)[,c(2,3)], col = data$color, main = "Figure 2: Data on projected space", xlab =
```

```r
#Creates the distance rank matrices using euclidean distance
dRank_true <- dist(data_cent[,1:2], method = "euclidean", upper = FALSE) %>%
  rank()
dRank_PC <- dist(rotated(kpca_poly2)[,c(1,3)], method = "euclidean", upper = FALSE) %>%
  rank()
full_rank <- cbind(dRank_true, dRank_PC) %>%
  as.data.frame()

#Plot the density scaled to lie between 0 and 1
full_rank %>%
  ggplot(aes(x = dRank_PC, y = dRank_true)) +
  geom_density_2d_filled(contour_var = 'ndensity') +
  labs(title = "Figure 3: Density plot of ranks in actual vs projected space", x = "Projected Rank", y 
```

```r
#Kernlab use kernel exp(-sigma * euclidean dist). This function transform our prefered sigma
#To fit to the kernlab function I.e. they use the inverse kernel width for radial kernel.
Sigma_trans <- function(x){
  return(1/(2*x^2))
}

#Perform KPCA using the radial kernal with different sigma
kpca_rad1 <- kpca(~., data = data_cent[,1:2], kernel = "rbfdot", kpar = list(sigma = Sigma_trans(0.1)),
kpca_rad2 <- kpca(~., data = data_cent[,1:2], kernel = "rbfdot", kpar = list(sigma = Sigma_trans(1)), f
kpca_rad3 <- kpca(~., data = data_cent[,1:2], kernel = "rbfdot", kpar = list(sigma = Sigma_trans(3)), f
kpca_rad4 <- kpca(~., data = data_cent[,1:2], kernel = "rbfdot", kpar = list(sigma = Sigma_trans(10)), 

#Plot observations on the second and third principle component for different sigma
par(mfrow = c(2,2))
plot(rotated(kpca_rad1)[,c(2,3)], col = data$color, xlab = "PC 1", ylab = "PC 3", main = bquote(sigma ~
plot(rotated(kpca_rad2)[,c(2,3)], col = data$color, xlab = "PC 1", ylab = "PC 3", main = bquote(sigma ~
plot(rotated(kpca_rad3)[,c(2,3)], col = data$color, xlab = "PC 1", ylab = "PC 3", main = bquote(sigma ~
plot(rotated(kpca_rad4)[,c(2,3)], col = data$color, xlab = "PC 1", ylab = "PC 3", main = bquote(sigma ~
mtext(expression(bold("Figure 4: Radial Kernel")), outer=TRUE,  cex=1, line=-1.5)
```

```r
#Creates the distance rank matrices using euclidean distance
dRank_PC2 <- dist(rotated(kpca_rad1)[,c(1,3)], method = "euclidean") %>%
  rank()
dRank_PC3 <- dist(rotated(kpca_rad2)[,c(1,3)], method = "euclidean") %>%
  rank()
dRank_PC4 <- dist(rotated(kpca_rad3)[,c(1,3)], method = "euclidean") %>%
  rank()
dRank_PC5 <- dist(rotated(kpca_rad4)[,c(1,3)], method = "euclidean") %>%
  rank()
full_rank2 <- tibble(True = dRank_true, "0.1" = dRank_PC2, "1" = dRank_PC3, "3" = dRank_PC4, "10" = dRan
  gather(key = "Sigma", value = "Rank", -True)

#Plot the density scaled to lie between 0 and 1
full_rank2 %>%
  ggplot(aes(x = True, y = Rank)) +
  geom_density_2d_filled(contour_var = 'ndensity') +
  facet_wrap(~Sigma, ncol = 2) +
  labs(title = "Figure 5: Density plot of ranks in actual vs projected space", x = "Original Rank", y =
```

```r
full_rank2 %>%
  group_by(Sigma) %>%
  summarise(corr = cor(True, Rank)) %>%
  mutate(Sigma = as.integer(Sigma)) %>%
  arrange(Sigma) %>%
  ggplot(aes(x = Sigma, y = corr)) +
  geom_point() +
  labs(title = "Figure 6: Correlation vs sigma", x = "Sigma", y = "Correlation")
```