Examples of principal nested torii (general d) – Part I

Eduardo García-Portugués, Huiling Le, James S. Marron, and Andrew Wood

2018-04-18, v1.0

Contents

Examples in $d = 2$

We optimize iteratively the objective function

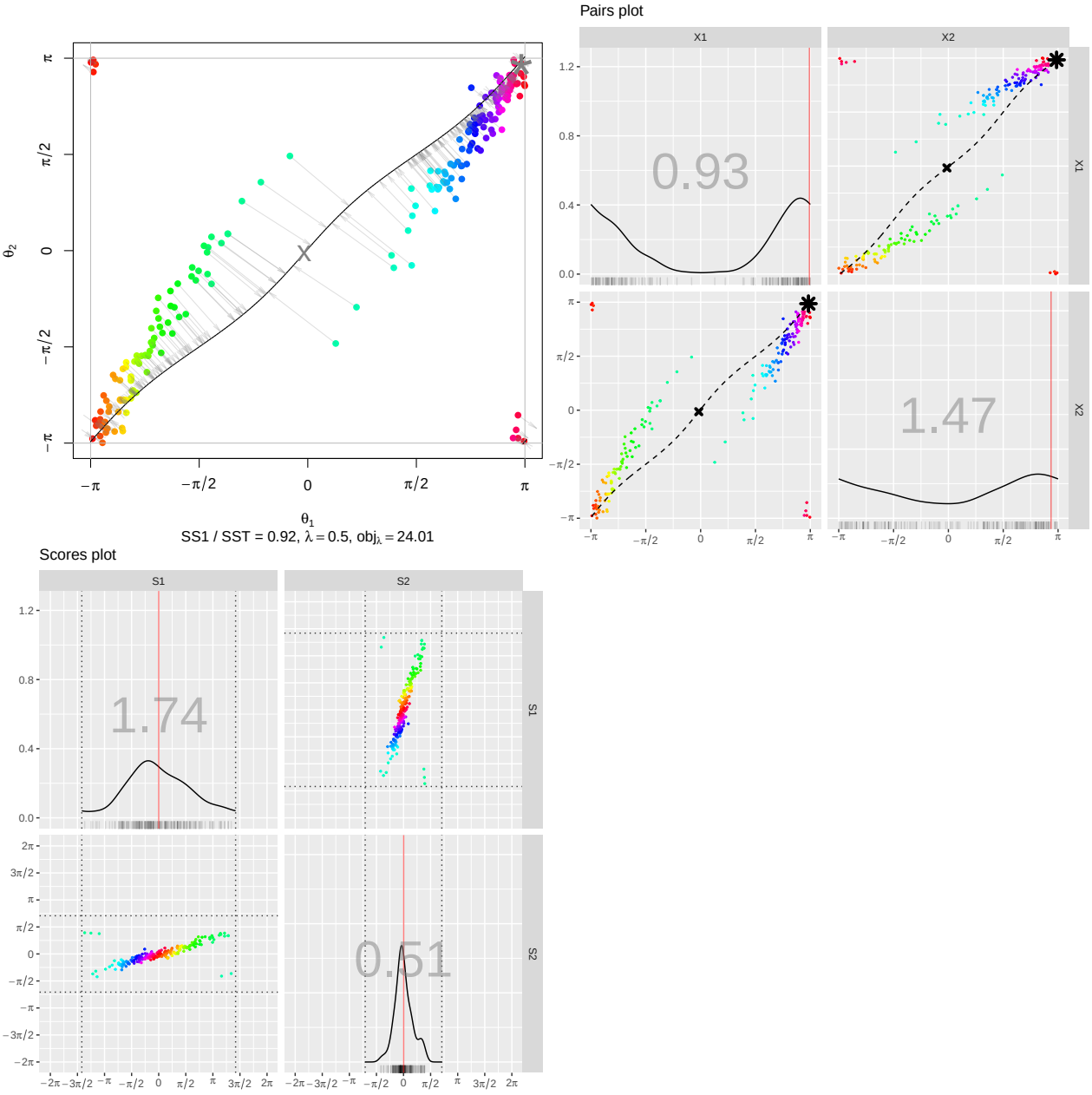
$$\lambda \underbrace{\sum_{i=1}^{n}\sum_{k=1}^{d-1}\left(\frac{1}{\sqrt{d}}\|\mathbf{P}_k\mathbf{X}_i\| - 1\right)^2}_{\substack{\text{Proj's distance to}\\ \sqrt{d}\times\mathbb{S}^{2d-1}\supset(\mathbb{S}^1)^d}} + (1-\lambda)\underbrace{\sum_{i=1}^{n}\|(\mathbf{I}-\mathbf{P})\mathbf{X}_i\|^2}_{\substack{\text{Complement to}\\ \text{proj's variation}}}$$

We set $\lambda = 0.5$ in the next plots. We use the *raw scores* (no rescaling, see Issue 1). The legend for the next plots:

- *Red vertical lines*: Fréchet circular means of the marginals.
- *Overlaid gray numbers*: Fréchet standard deviations (with circular distance) with respect to the Fréchet circular mean.
- *Dashed horizontal/vertical black lines*: theoretical support of the scores (changes with the level of the scores). Periodicity is assumed in this support. Present if `distanceScaled = FALSE` (the default). See Issue 1.
- *Dashed black curves*: projections of the PC1 curve in pairs of variables. The backwards mean is represented with "*", whereas the backwards antimean is represented by an"x".

Steve's dataset

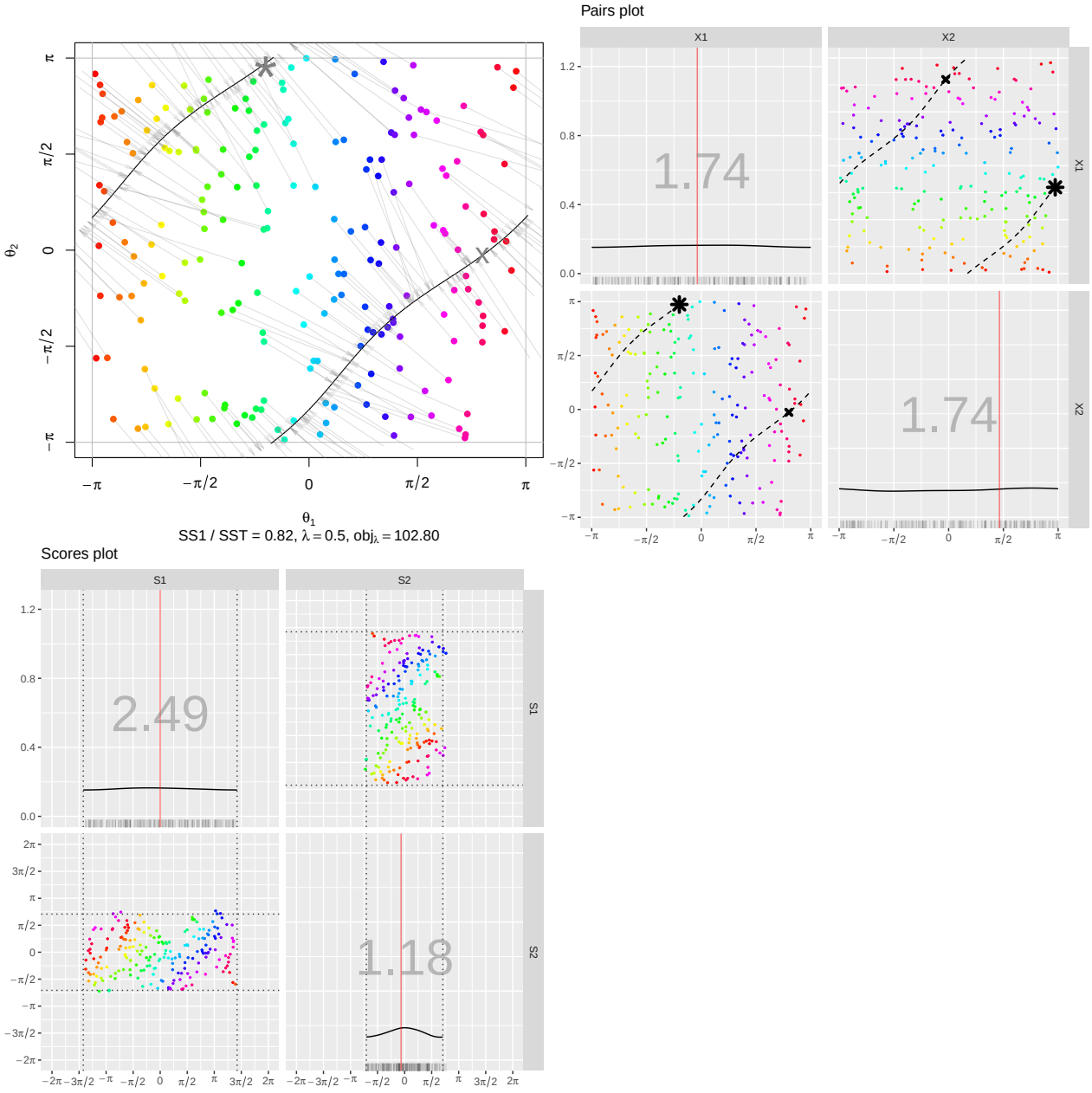
```
## Reduction to dimension d = 2. Time: 0.129 seconds.
## Reduction to dimension d = 1. Time: 0.36 seconds.
```



Uniform-like

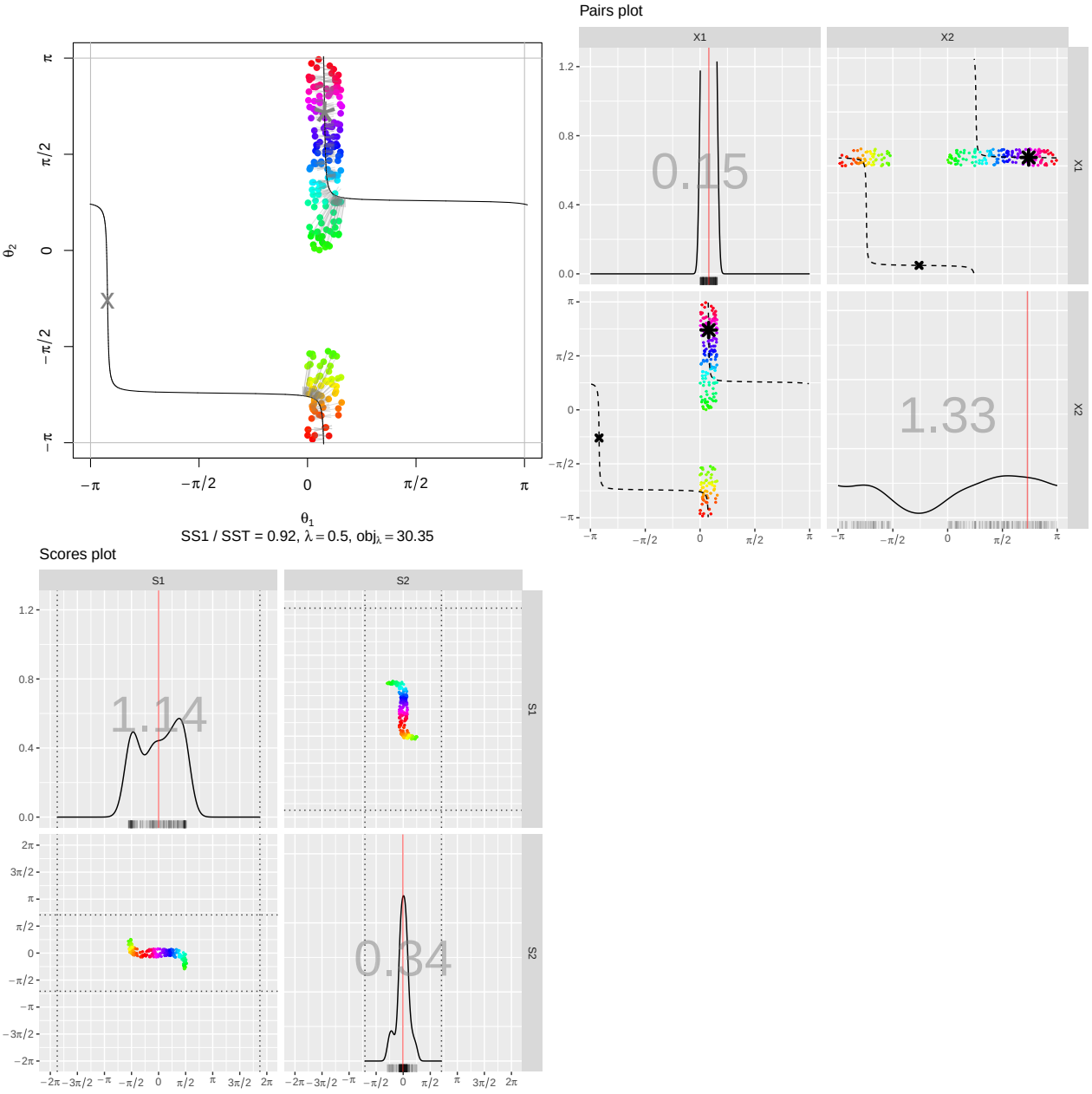
Uniform in $[-\pi, \pi)^2$

```
## Reduction to dimension d = 2. Time: 0.275 seconds.
## Reduction to dimension d = 1. Time: 0.641 seconds.
```



Uniform vertical band

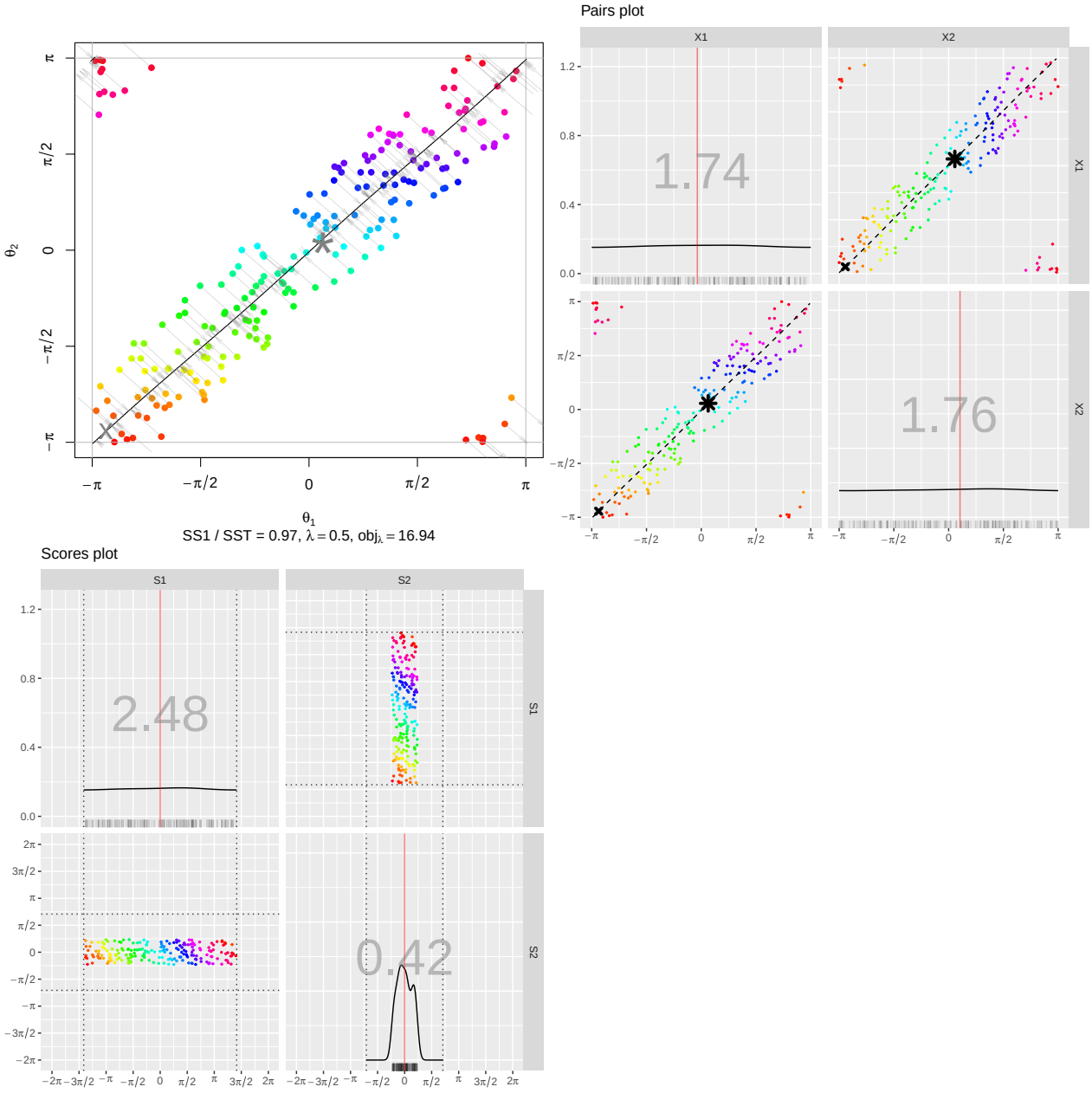
```
## Reduction to dimension d = 2. Time: 0.201 seconds.
## Reduction to dimension d = 1. Time: 0.42 seconds.
```



Diagonal-like

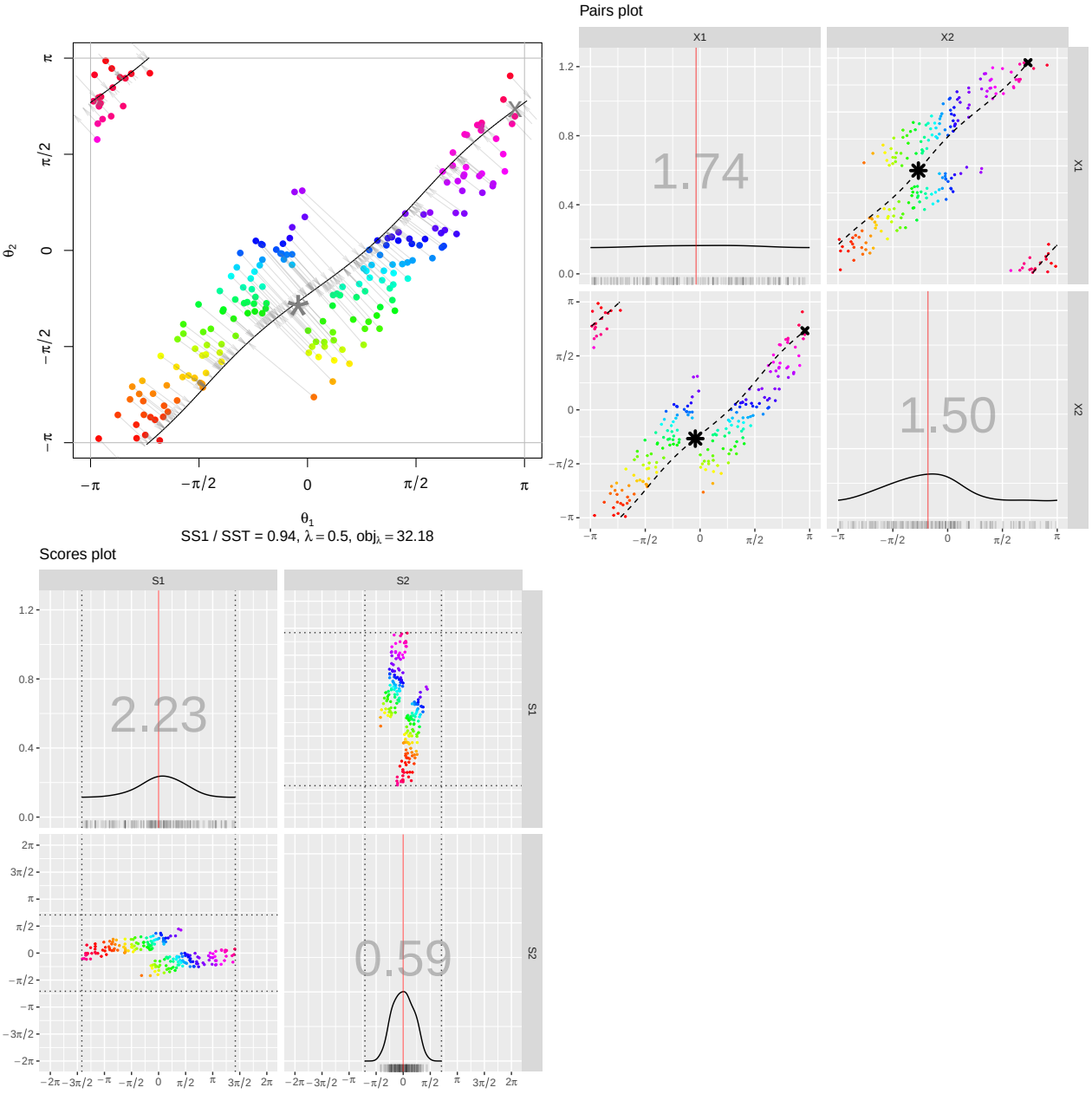
Diagonal $y=x

```
## Reduction to dimension d = 2. Time: 0.086 seconds.
## Reduction to dimension d = 1. Time: 0.378 seconds.
```



Diagonal $y = x - 2$

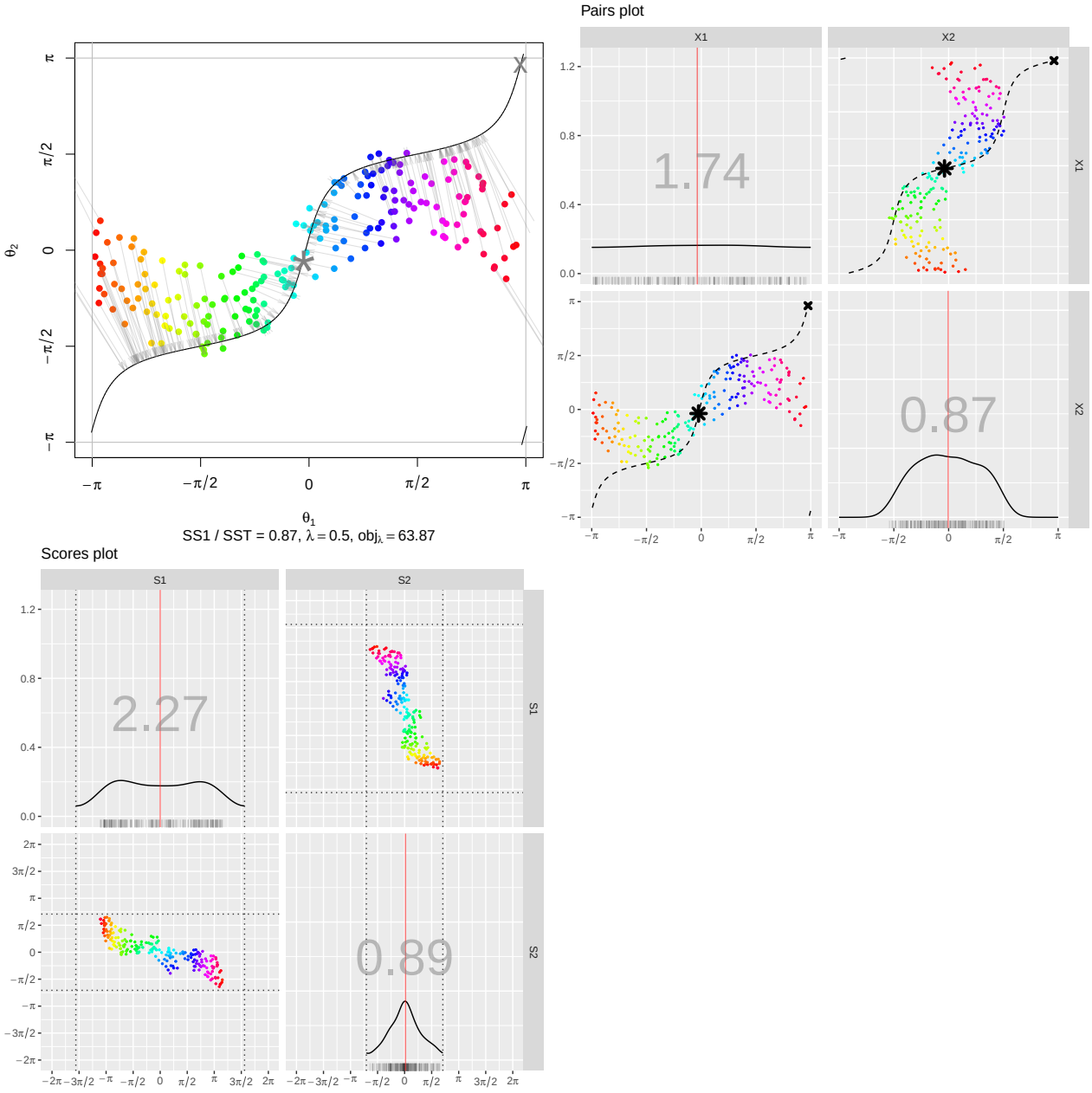
```
## Reduction to dimension d = 2. Time: 0.119 seconds.
## Reduction to dimension d = 1. Time: 0.373 seconds.
```



Trigonometric-like

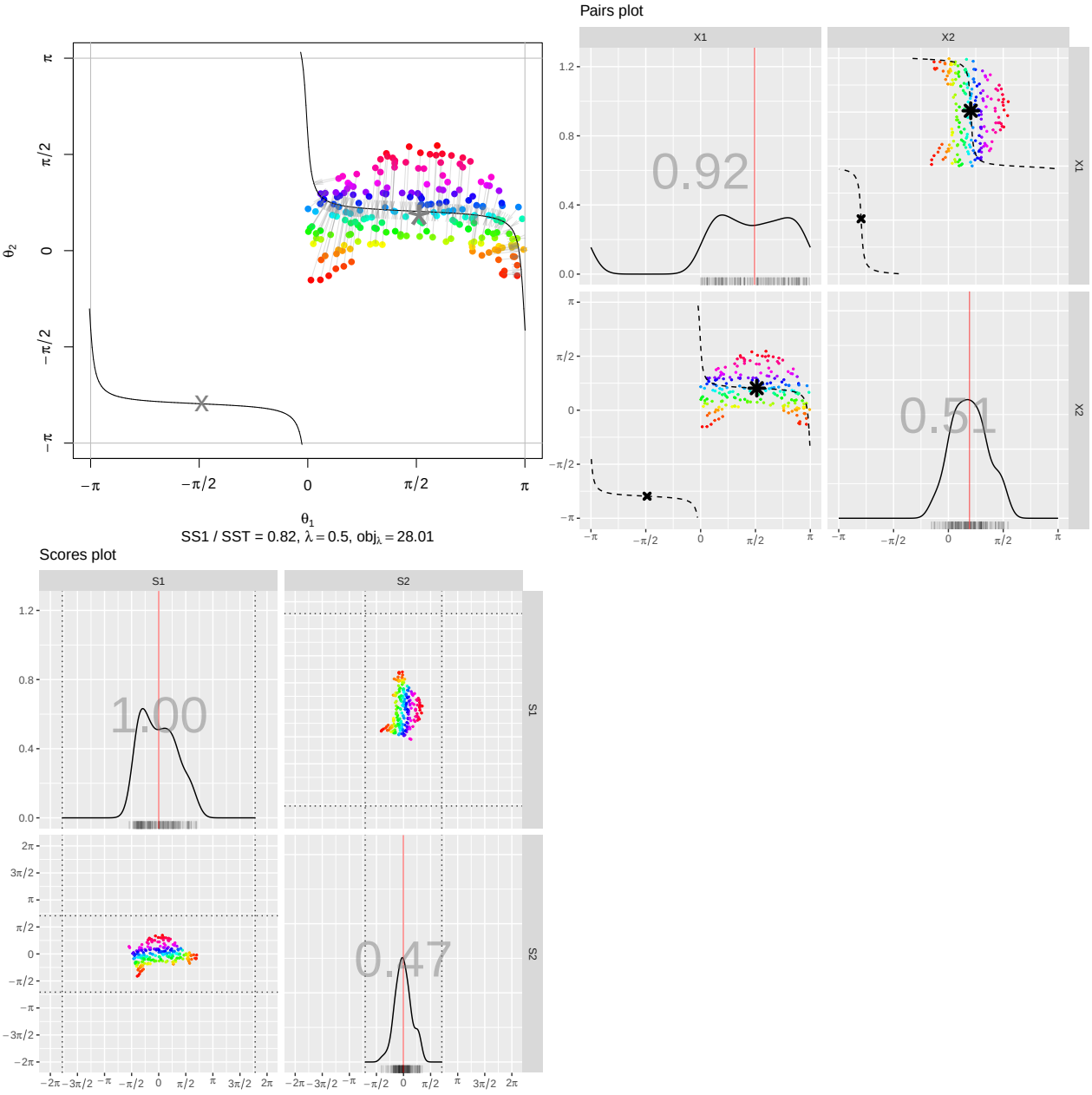
$\sin(x)$

```
## Reduction to dimension d = 2. Time: 0.067 seconds.
## Reduction to dimension d = 1. Time: 0.293 seconds.
```



Shorter-support $\sin(x)$

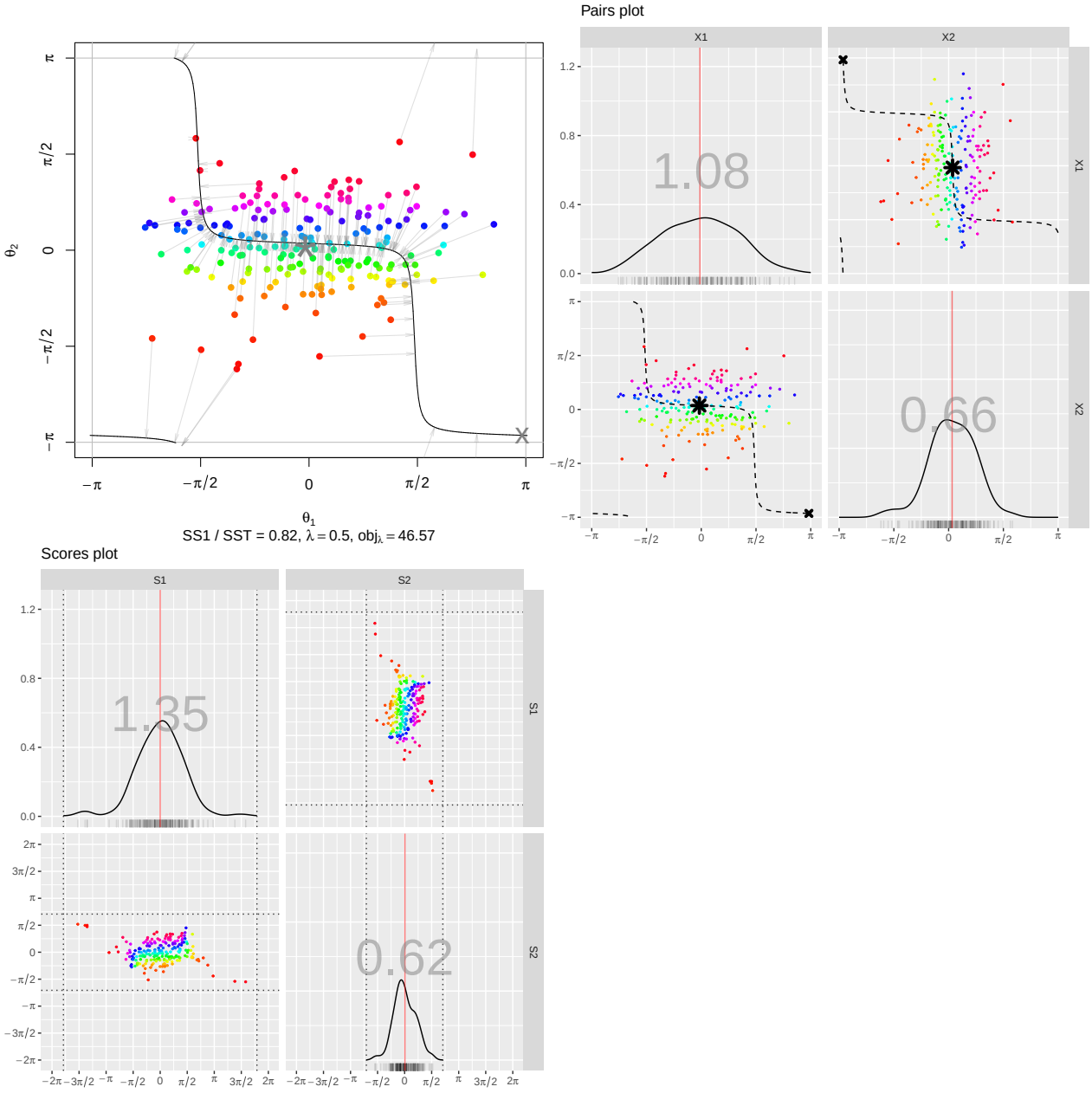
```
## Reduction to dimension d = 2. Time: 0.074 seconds.
## Reduction to dimension d = 1. Time: 0.3 seconds.
```



Gaussian-like

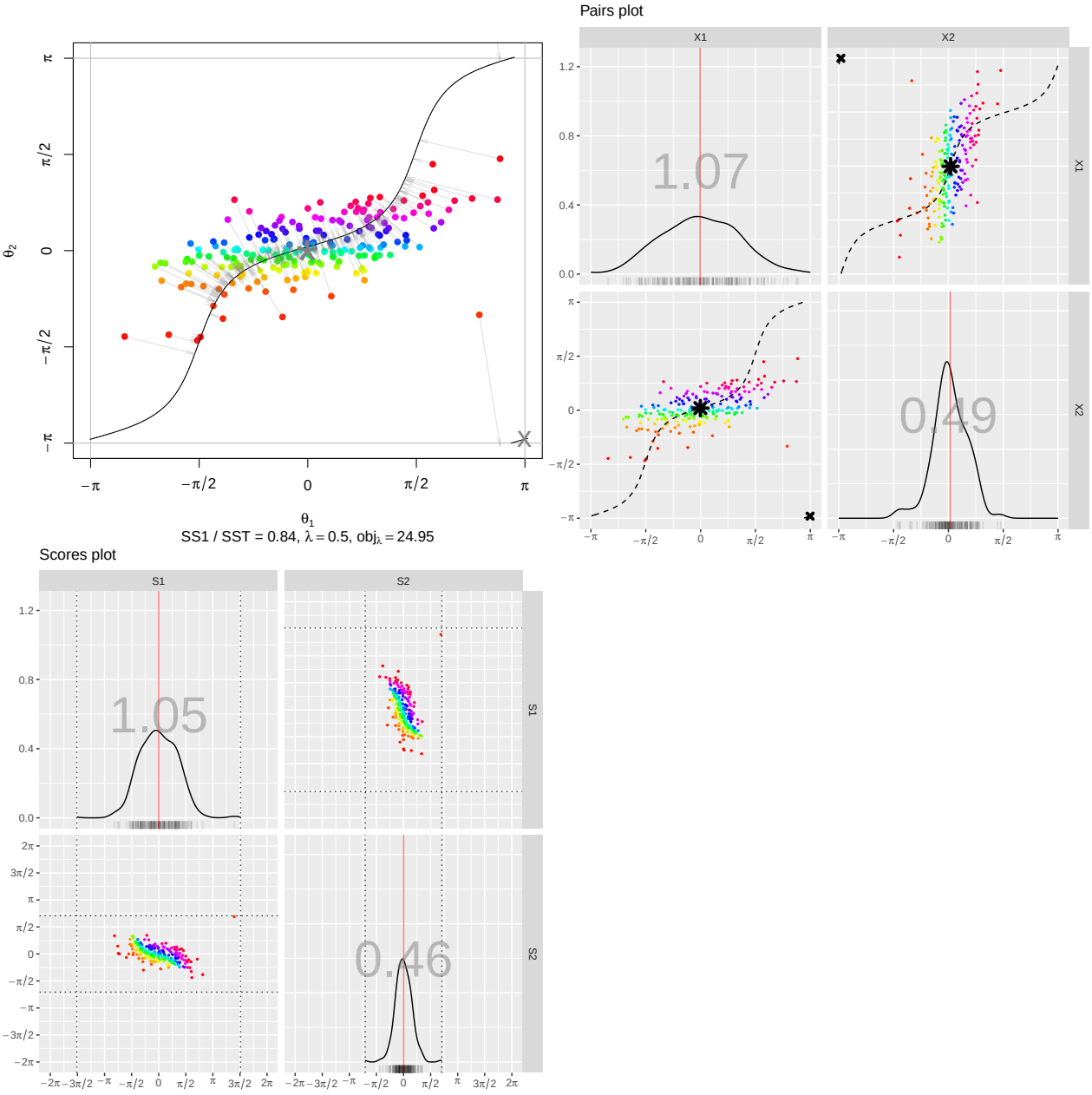
Non-isotropic

```
## Reduction to dimension d = 2. Time: 0.118 seconds.
## Reduction to dimension d = 1. Time: 0.338 seconds.
```



Rotated

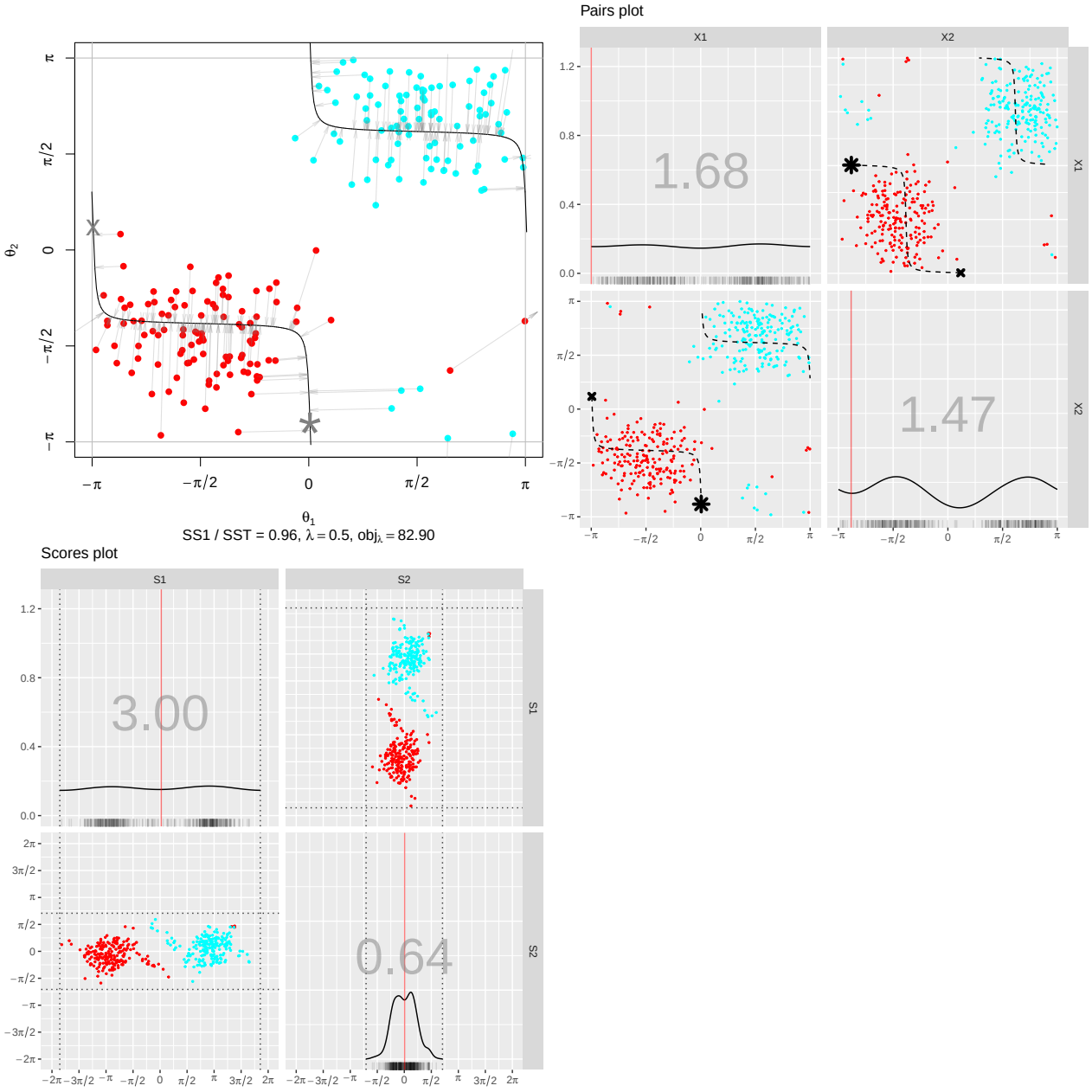
```
## Reduction to dimension d = 2. Time: 0.084 seconds.
## Reduction to dimension d = 1. Time: 0.319 seconds.
```



Clusters

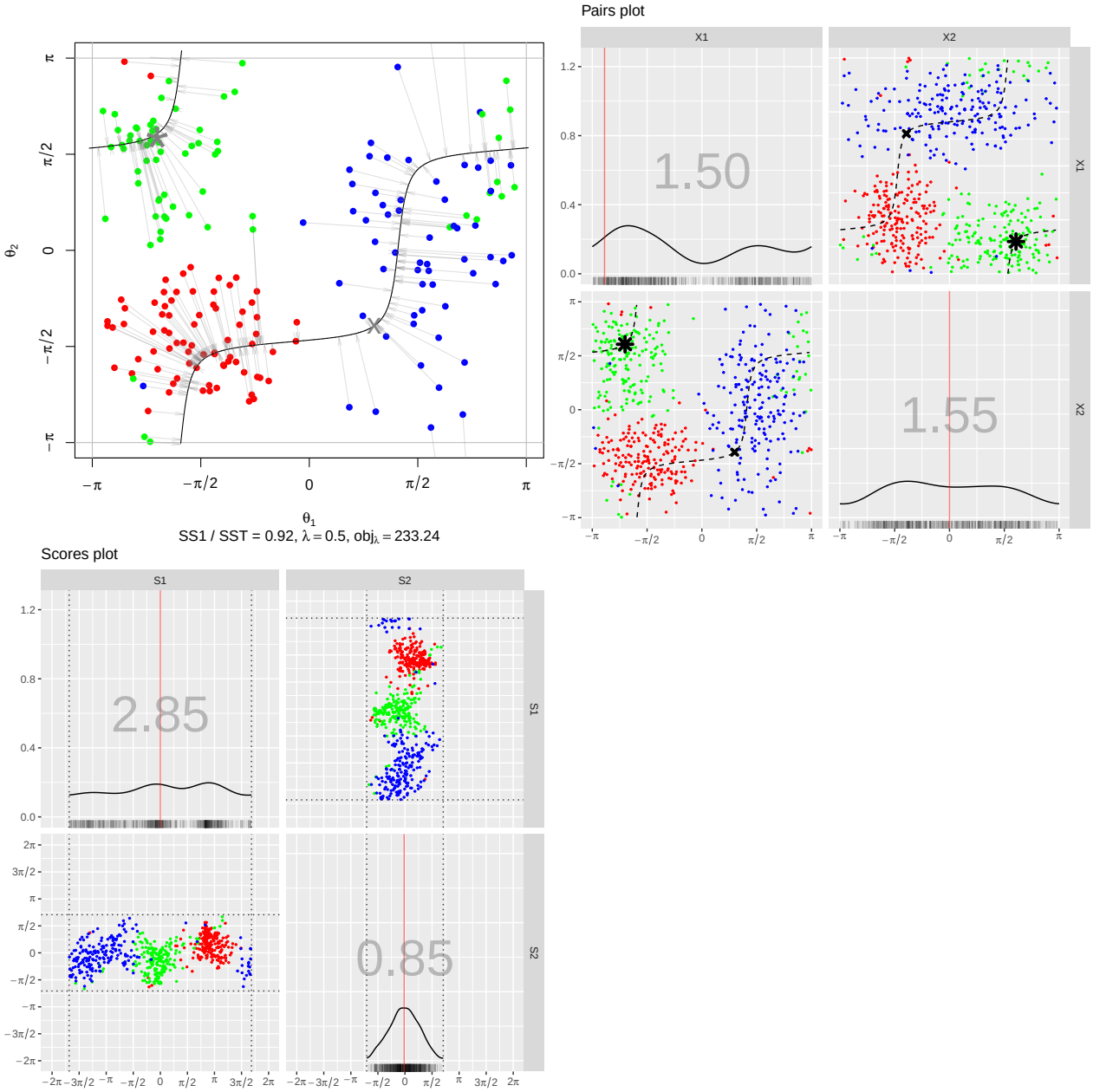
Two separated clusters

```
## Reduction to dimension d = 2. Time: 0.14 seconds.
## Reduction to dimension d = 1. Time: 0.497 seconds.
```



Three clusters

```
## Reduction to dimension d = 2. Time: 0.278 seconds.
## Reduction to dimension d = 1. Time: 0.615 seconds.
```



λ effect

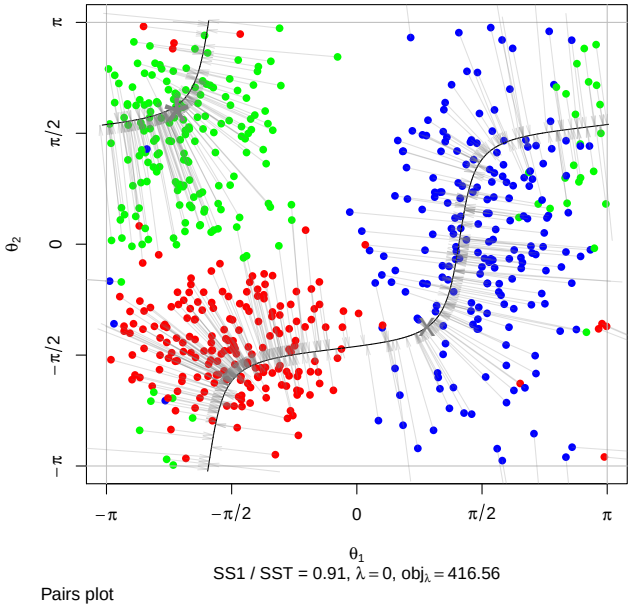
Legend for the next plots:

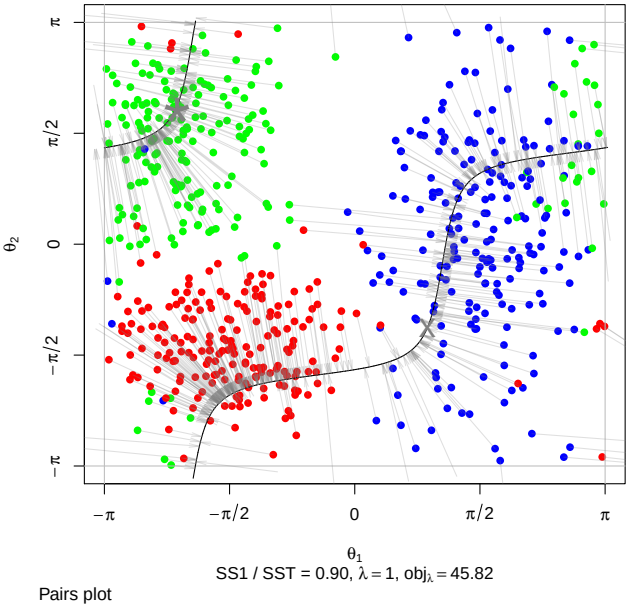
- *Left* plots corresponds to fits with $\lambda = 0$.
- *Right* to fits with $\lambda = 1$.

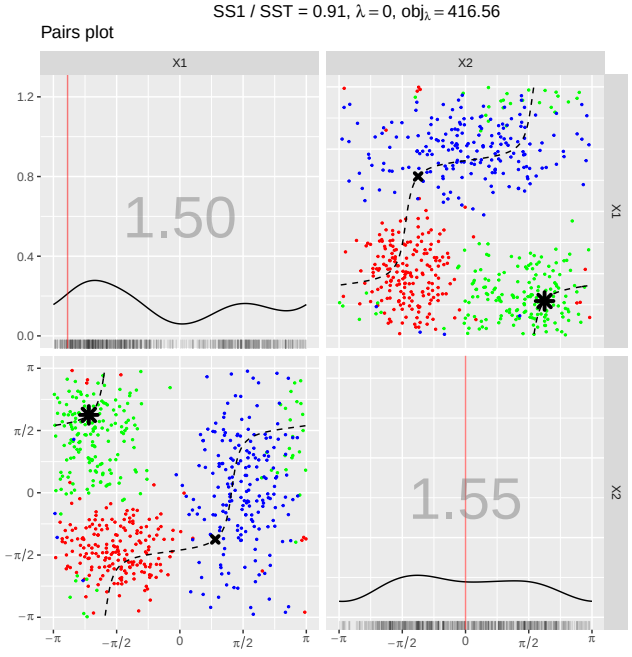
Three sparse separated clusters

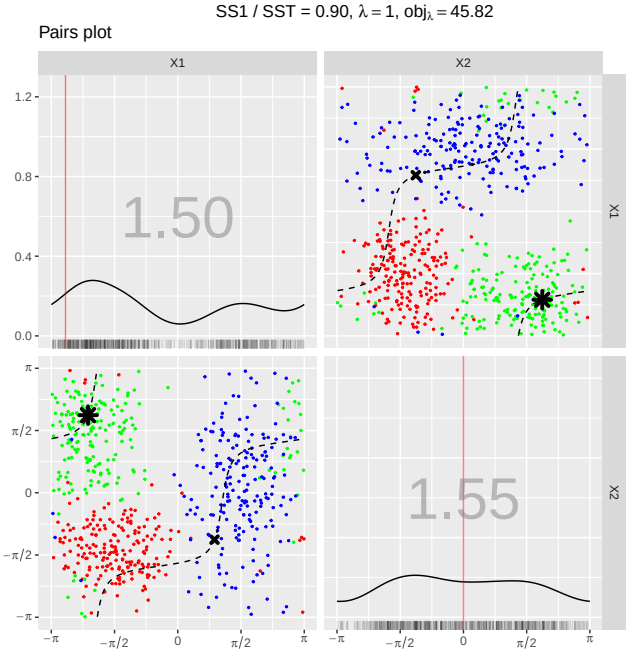
```
## Reduction to dimension d = 2. Time: 0.257 seconds.
## Reduction to dimension d = 1. Time: 0.592 seconds.

## Reduction to dimension d = 2. Time: 0.25 seconds.
```

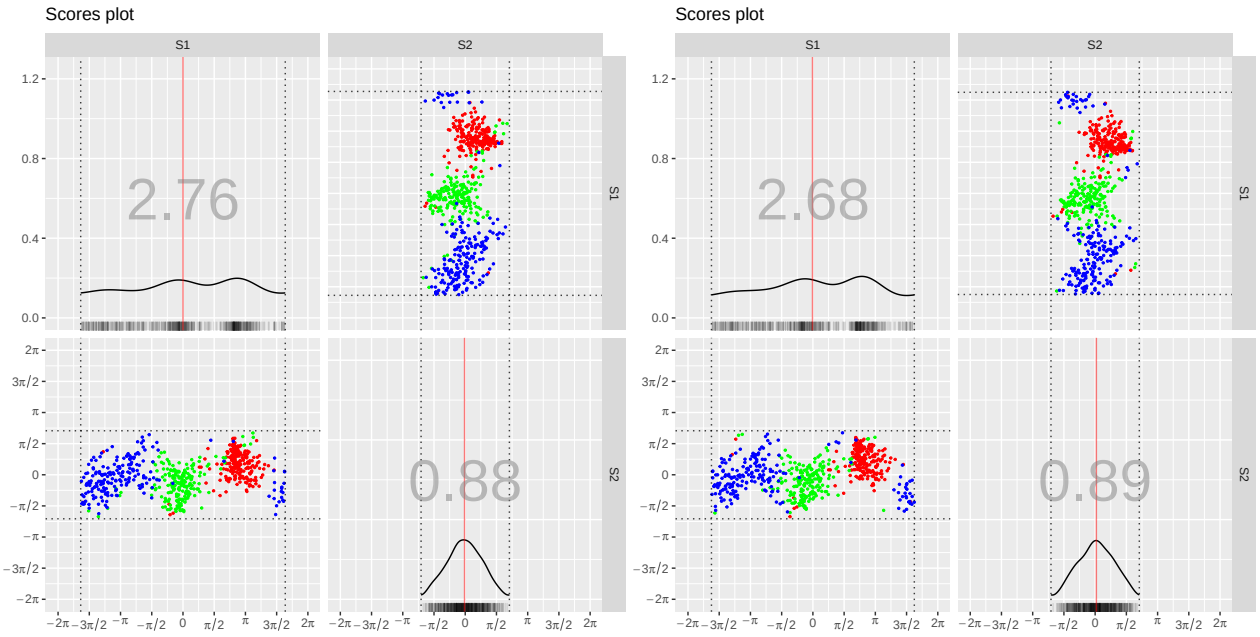
```
## Reduction to dimension d = 1. Time: 0.576 seconds.
```





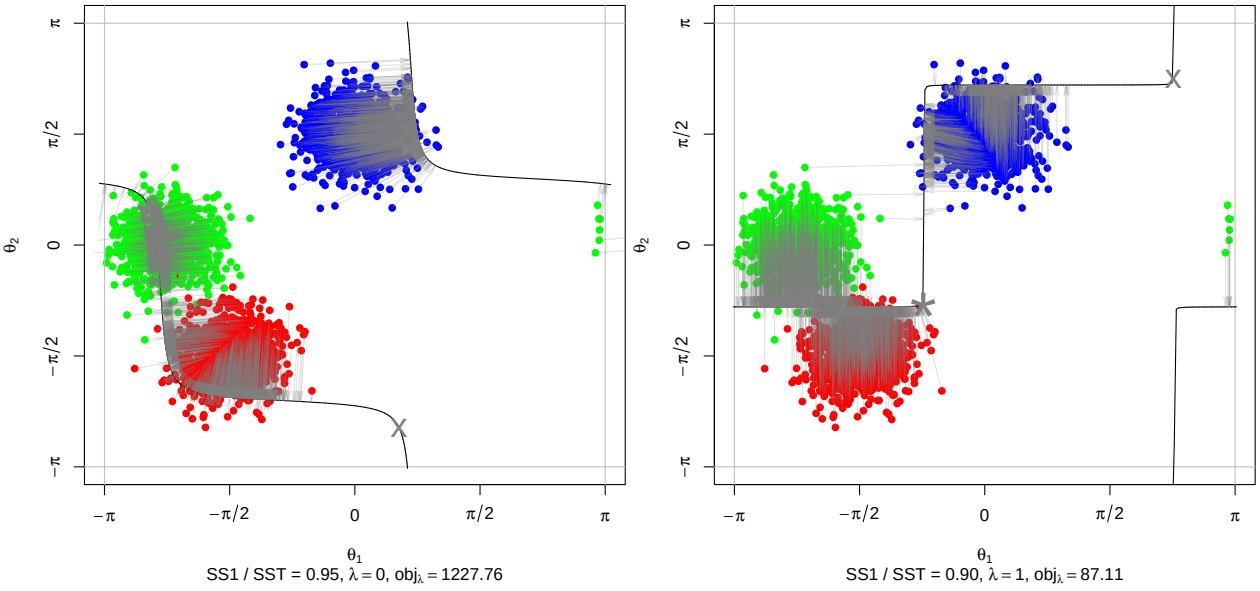


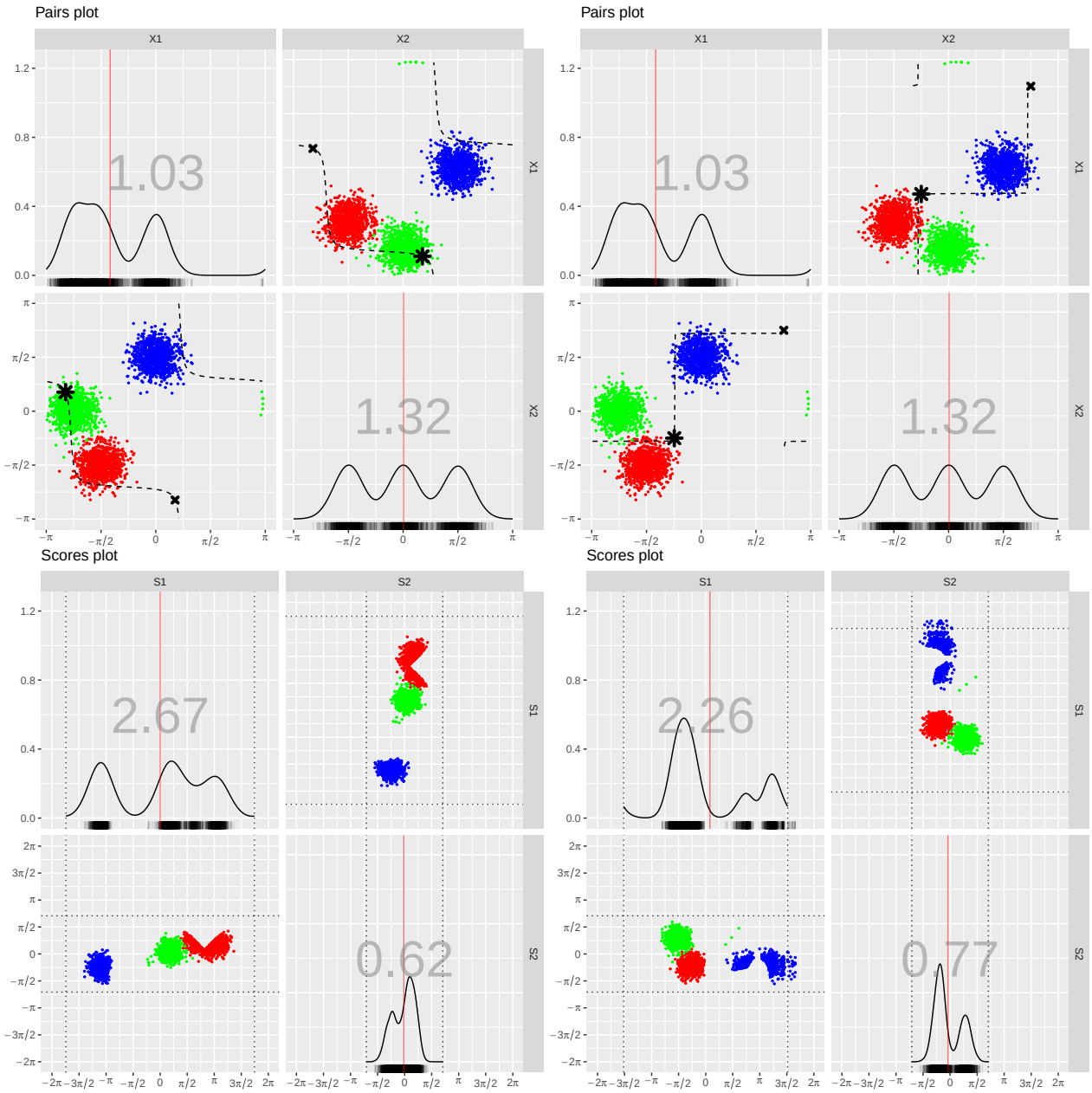




Three concentrated separated clusters

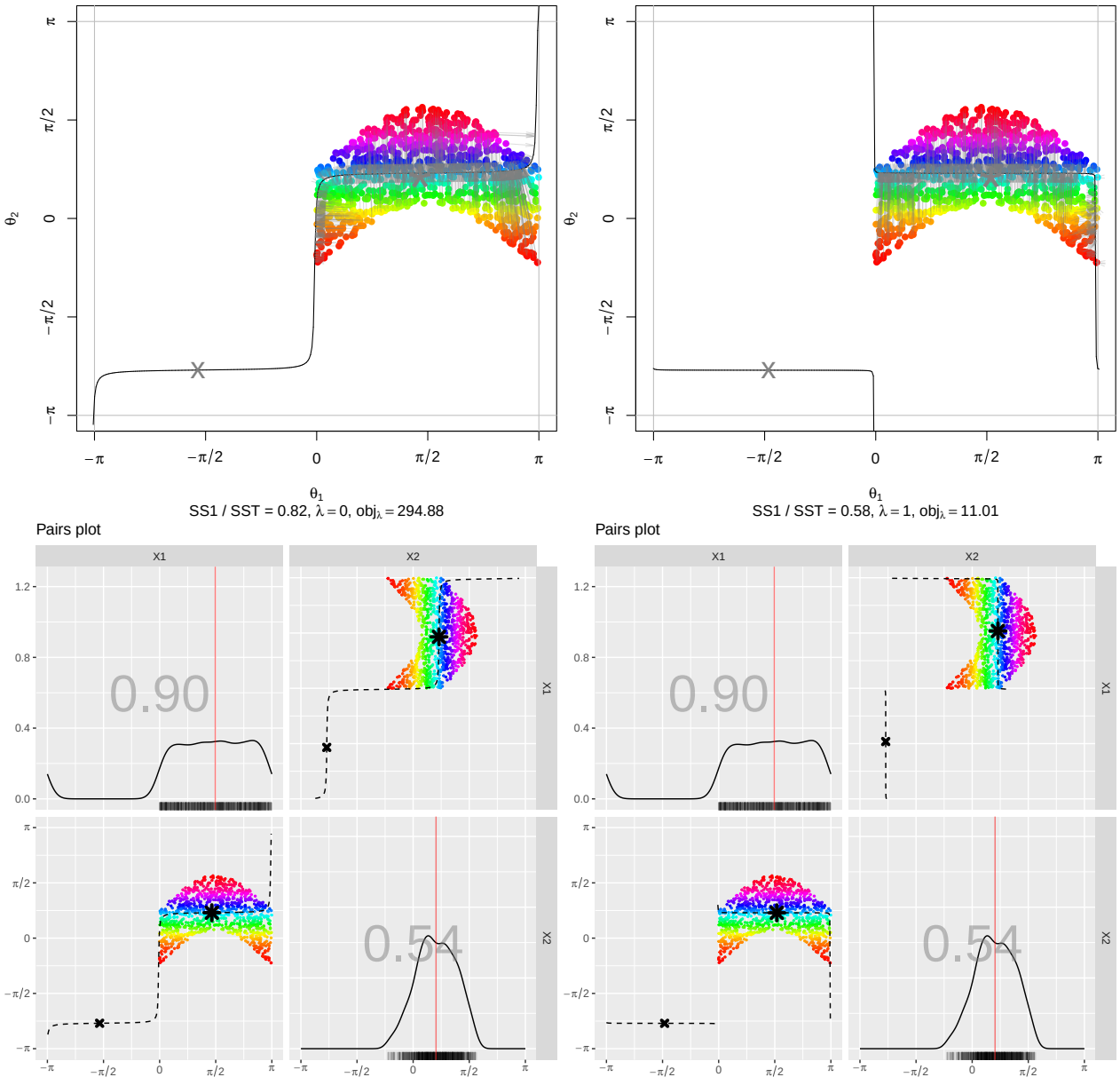
```
## Reduction to dimension d = 2. Time: 1.347 seconds.
## Reduction to dimension d = 1. Time: 3.044 seconds.

## Reduction to dimension d = 2. Time: 1.725 seconds.
## Reduction to dimension d = 1. Time: 2.904 seconds.
```

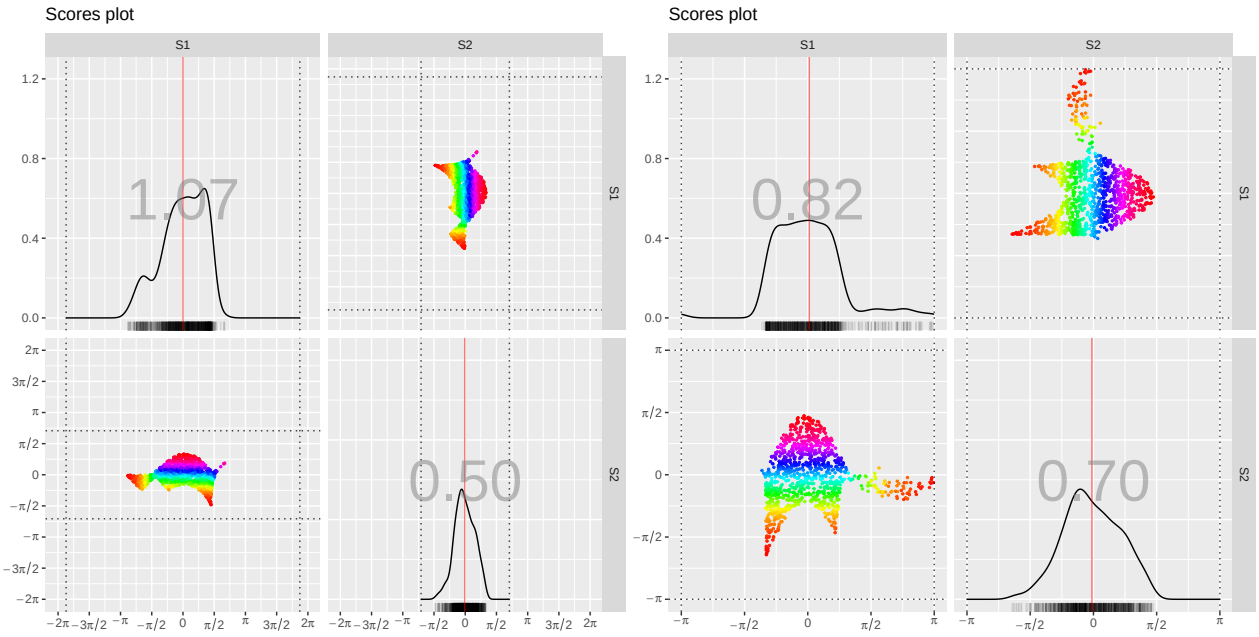




Shorter-support $\sin(x)$

```
## Reduction to dimension d = 2. Time: 0.406 seconds.
## Reduction to dimension d = 1. Time: 1.128 seconds.

## Reduction to dimension d = 2. Time: 0.462 seconds.
## Reduction to dimension d = 1. Time: 1.203 seconds.
```


θ2
θ1
SS1 / SST = 0.82, λ = 0, objλ = 294.88
SS1 / SST = 0.58, λ = 1, objλ = 11.01
Pairs plot
X1
X2
0.90
0.54



Issue 1: re-scaling vs. not re-scaling the raw scores

In the way we construct the (*raw*) scores, we have that, in $d = 2$:

$$\begin{aligned}\text{Score}_1 &= \text{Distance\_along\_curve\_from\_projection\_to\_mean} \in [-l/2, l/2)\,, l \leq 4\pi,\\ \text{Score}_2 &= \text{Distance\_from\_point\_to\_curve\_projection} \in \left[-\frac{\sqrt{2}\pi}{2}, \frac{\sqrt{2}\pi}{2}\right).\end{aligned}$$

Therefore, $(\text{Score}_1, \text{Score}_2) \notin [-\pi, \pi)^2$. Using `distanceScaled = TRUE`, these are scores are multiplied by $\frac{\pi}{l/2}$ and $\frac{\pi}{\sqrt{2}\pi/2}$ in order to force them to be in $[-\pi, \pi)$.

Recall that for a random variable X with $\text{supp}(X) = (a, b)$, $\mathbb{V}\text{ar}[X] \leq \frac{(b-a)^2}{4}$. This means that:

- $\mathbb{V}\text{ar}[\text{Score}_1] \leq \frac{1}{4}l^2 \leq 4\pi^2 = 39.47 =: C_1$.
- $\mathbb{V}\text{ar}[\text{Score}_2] \leq \frac{1}{4}2\pi^2 = \frac{1}{2}\pi^2 = 4.93 =: C_2$.

For general d,

$$\text{Score}_d = \text{Distance\_from\_point\_to\_surface\_projection} \in [-\frac{\sqrt{d}\pi}{2}, \frac{\sqrt{d}\pi}{2})$$

and therefore $\mathbb{V}\text{ar}[\text{Score}_d] \leq \frac{1}{4}d\pi^2 = \frac{d}{4}\pi^2 =: C_d$. This means that

$$C_2 < C_3 < \ldots < C_{d-1} < C_d$$

and that

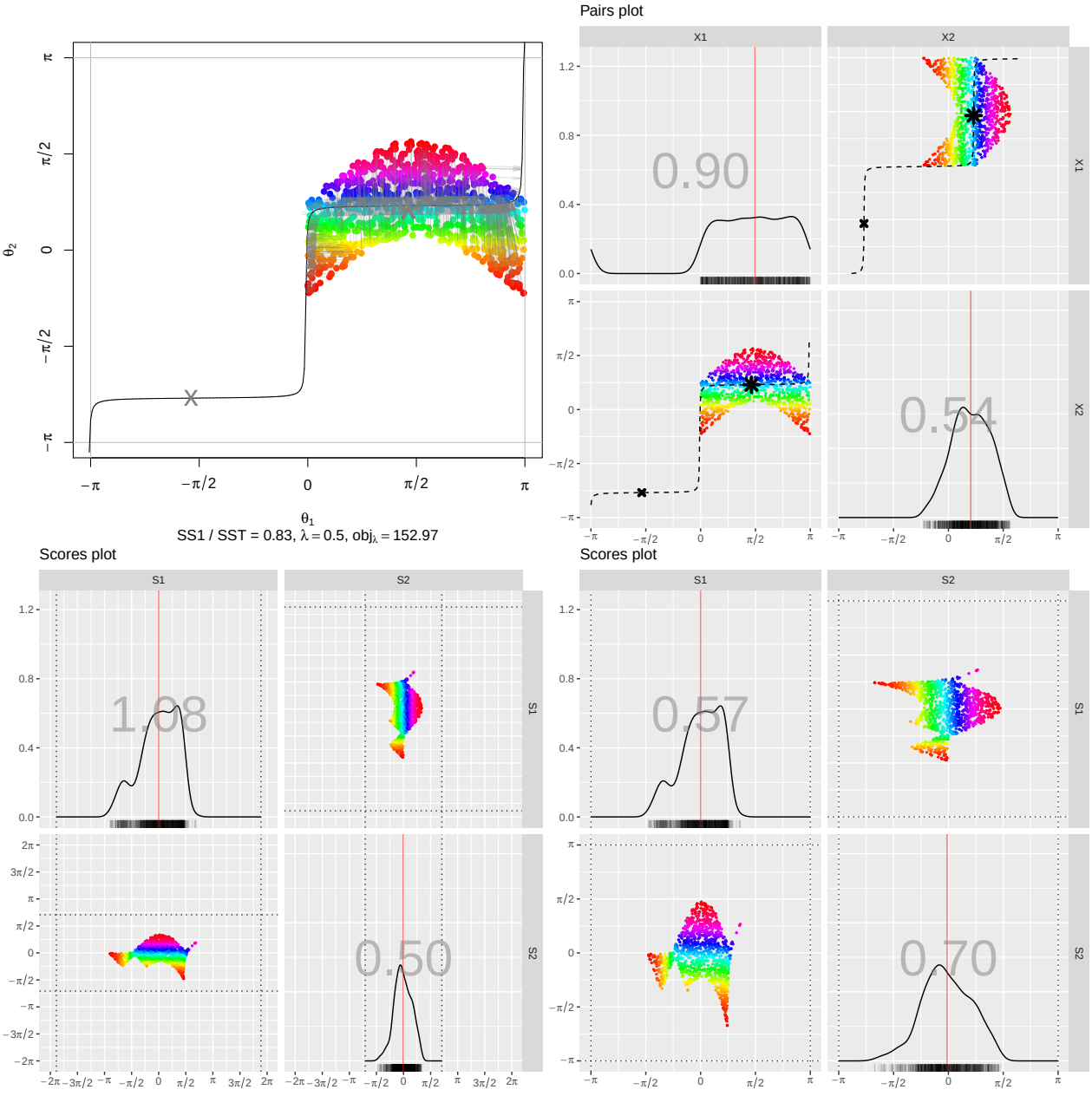
$$C_1 = C_{16} > C_{15} > \ldots > C_2, \quad C_1 < C_{17} < C_{18} < \ldots < C_d.$$

Setting `distanceScaled = FALSE` (raw scores) or `distanceScaled = TRUE` (applies rescaling) has consequences, as illustrated below.

Shorter-support $\sin(x)$

```
## Reduction to dimension d = 2. Time: 0.419 seconds.
## Reduction to dimension d = 1. Time: 1.209 seconds.
```

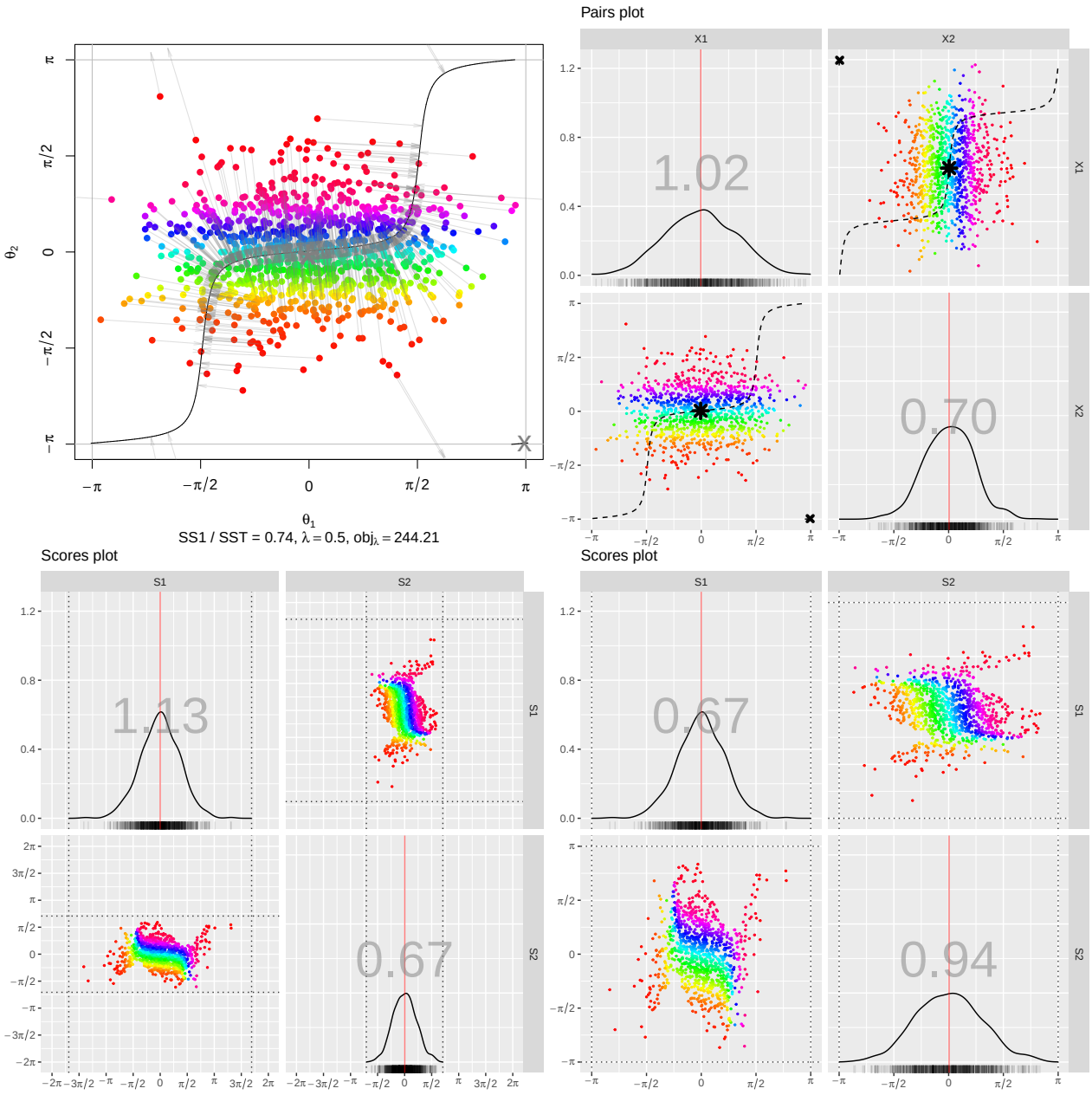
```
## Reduction to dimension d = 2. Time: 0.59 seconds.
## Reduction to dimension d = 1. Time: 1.254 seconds.
```



Non-isotropic Gaussian $\sin(x)$

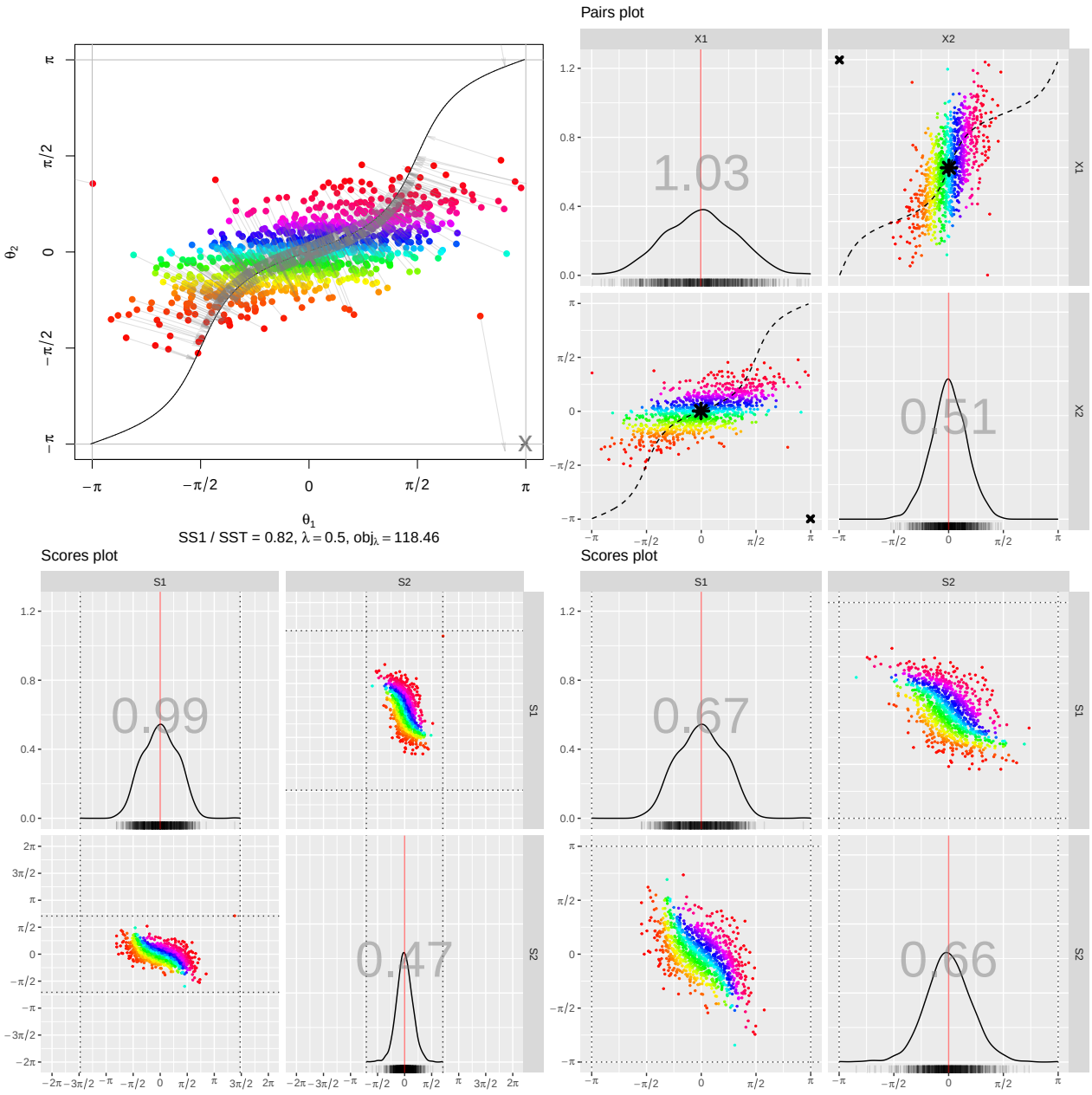
```
## Reduction to dimension d = 2. Time: 0.563 seconds.
## Reduction to dimension d = 1. Time: 1.321 seconds.
```

```
## Reduction to dimension d = 2. Time: 0.617 seconds.
## Reduction to dimension d = 1. Time: 1.202 seconds.
```



Rotated Gaussian

```
## Reduction to dimension d = 2. Time: 0.38 seconds.
## Reduction to dimension d = 1. Time: 1.024 seconds.

## Reduction to dimension d = 2. Time: 0.383 seconds.
## Reduction to dimension d = 1. Time: 0.965 seconds.
```

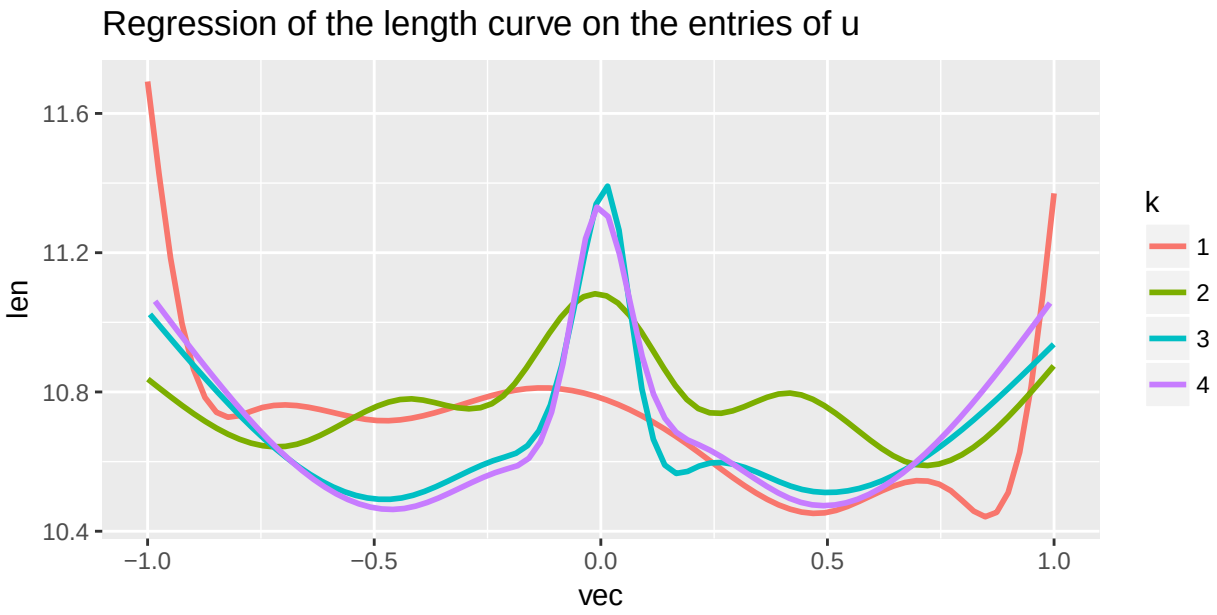


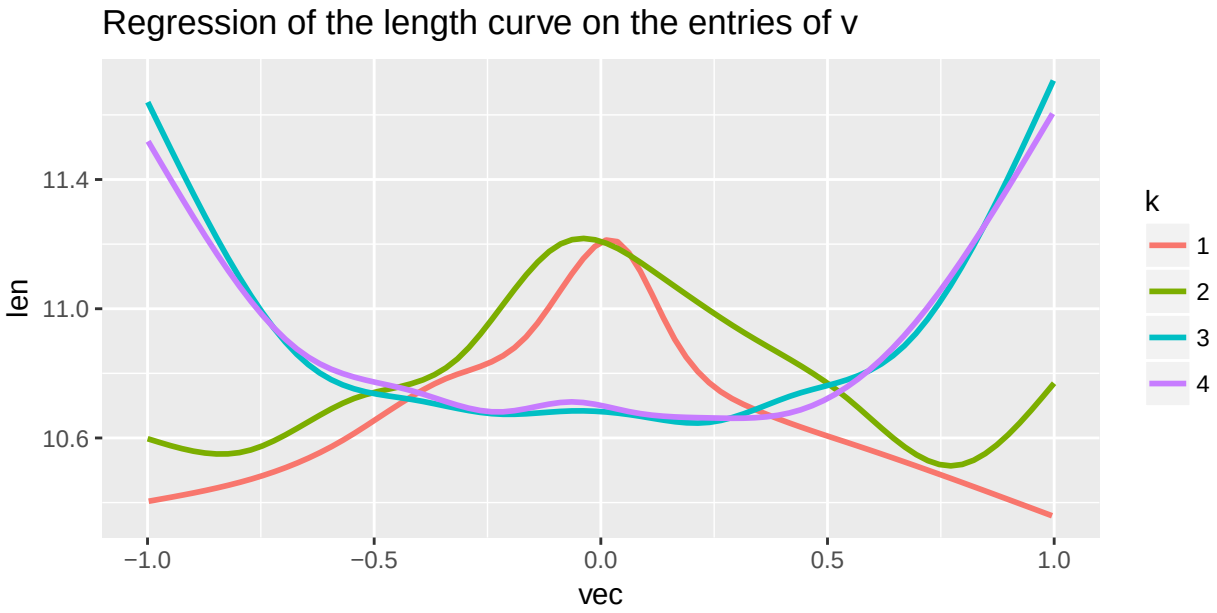
Issue 2: excentricity of the curves

Some choices of the vectors **u** and **v** provide more squarish curves than others. Squarish fits tend to yield degenerate projections to the corners. This is likely more problematic in higher dimensions, since the moment the projections collapse, they will remain degenerate for lower-dimensional fits, hence producing a sequence of degenerate scores.

I do not know what is the parametrization of **u** and **v** that provides squarish curves (*squarish* can be characterized by the length of the curves, which is close to 4π for the squarish ones). It is somehow related with the vectors **u** and **v** having entries close to 0 (respectively, close to 1), as the following empirical evidence suggests (regressions of lengths on entries of the vectors):

```r
# Sample random curves and evaluate their lengths
M <- 1e4
l <- numeric(M)
x <- matrix(nrow = M, ncol = 3)
y <- matrix(nrow = M, ncol = 2)
u <- v <- matrix(nrow = M, ncol = 4)
for (i in 1:M) {

  x[i, ] <- c(runif(2, min = 0, max = pi), runif(1, min = -pi, max = pi))
  y[i, ] <- c(runif(1, min = 0, max = pi), runif(1, min = -pi, max = pi))
  u[i, ] <- anglesToSphere(theta = x[i, ])
  v[i, ] <- c(anglesToSphere(theta = y[i, ]) %*% Hu(u = u[i, ])[-1, ])
  l[i] <- distPC1Curve(alpha = c(0, 2 * pi - 1e-4), u = u[i, ], v = v[i, ],
                       N = 1e3, shortest = FALSE, der = FALSE)

}
df <- data.frame("l" = l, "u" = u, "v" = v)
df <- reshape(df, direction = "long",
              varying = c("u.1", "u.2", "u.3", "u.4", "v.1", "v.2", "v.3", "v.4"),
              times = c("u", "v"), timevar = "k")
df <- reshape2::melt(df, measure.vars = c("u", "v"))
df$id <- NULL
df$k <- as.factor(df$k)
names(df) <- c("len", "k", "uv", "vec")
grid.arrange(
  ggplot(data = df[df$uv == "u", ], mapping = aes(x = vec, y = len, colour = k)) +
    geom_smooth(se = FALSE) + ggtitle("Regression of the length curve on the entries of u"),
  ggplot(data = df[df$uv == "v", ], mapping = aes(x = vec, y = len, colour = k)) +
    geom_smooth(se = FALSE) + ggtitle("Regression of the length curve on the entries of v")
)
```





If we could characterize when the curves are squarish and also the surface, we could add a small penalty in the obejective function to avoid degenerate solutions.