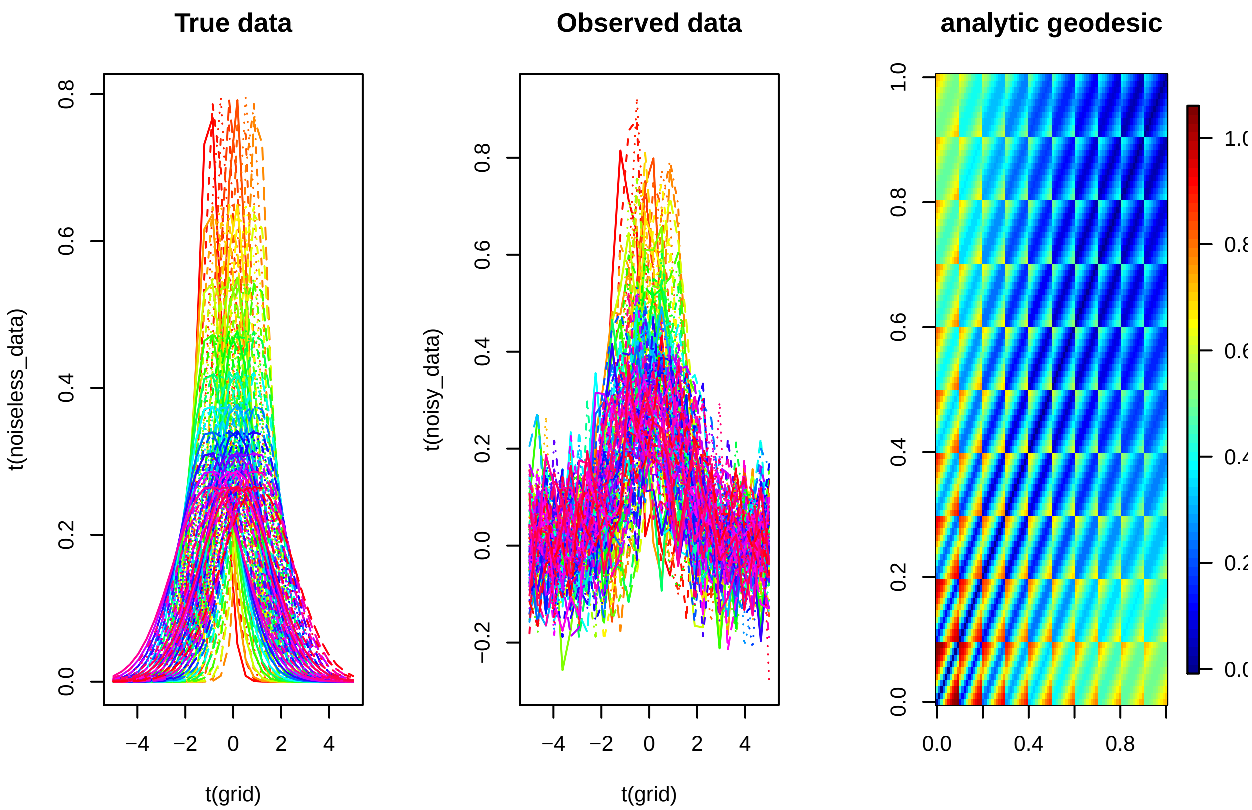
Test if analytic derivation of geodesic distance for curves from sim_functional_data.R is correct

```r
source('sim_functional_data.R')
```

```
## Loading required package: spam

## Warning: package 'spam' was built under R version 3.5.2

## Loading required package: dotCall64

## Loading required package: grid

## Spam version 2.2-2 (2019-03-07) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.

##
## Attaching package: 'spam'

## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve

## Loading required package: maps

## See www.image.ucar.edu/~nychka/Fields for
##  a vignette and other supplements.
```

The function sim_functional_data.R allows specification of several simulatin scenarios. The samplesize is purposefully set very high for reasons explained below.

```r
# Generate data
sim <- sim_functional_data(sce=3,samplesize=100)
```



The output actually contains more than just the simulated functional data. In particular, it contains the in-sample pairwise geodesic distance matrix (analytic_geo) via analytic derivation. The purpose of this notebook is to see if our analytic derivations match numerical results.

```
names(sim)
```

```
## [1] "noiseless_data" "noisy_data"     "analytic_geo"   "grid"
## [5] "reg_grid"
```

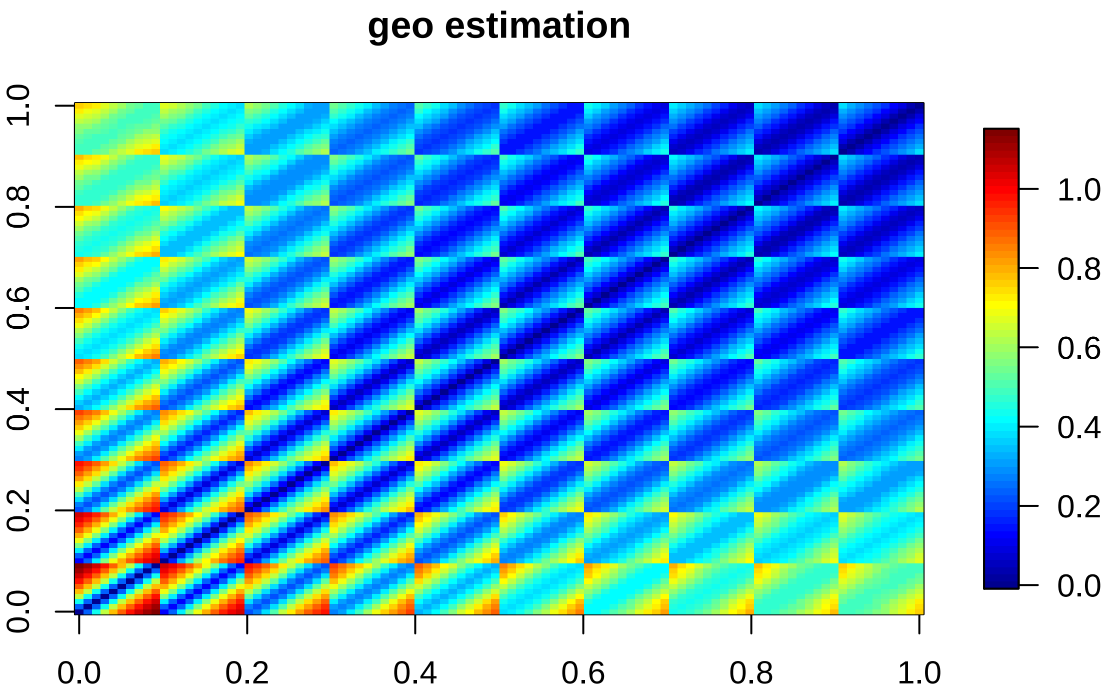
We would like to compare sim$analytic_geo to geodesic distance estimation via Floyd's algorithm for the noiseless (discretised) curve. This algorithm is guaranteed to do well if we sample the manifold densely enough? (That's why we set the samplesize to be quite high initially.)

```
library(reticulate)
```

```
## Warning: package 'reticulate' was built under R version 3.5.2
```

```
# sometimes reticulate requires specific instruction on where to find your python
# first call Sys.which("python") and then copy that path into

#use_python('/Users/suswei/anaconda3/bin/python',required=TRUE)
use_python('/Users/UQAM/anaconda3/bin/python',required=TRUE)


get_min_num_neighbors = import_from_path("get_min_num_neighbors",path='.')
getIsomapGdist = import_from_path("getIsomapGdist",path='.')

attach(sim)
a = reg_grid[1]
b = tail(reg_grid,1)
K = length(reg_grid)
```

```
samplesize = dim(noiseless_data)[1]
noiseless_data_tmp = (sqrt((b-a)/K))*noiseless_data

# Find a grid of possible values for the number of neigbors
num_neigh_min=get_min_num_neighbors$get_min_num_neighbors(noiseless_data_tmp)
num_neigh_true=seq(num_neigh_min,samplesize/2,by=2)

# Calculate the geo matrix for each number of neighbors and keep the one that gives the minimal error
norm_analytic_geo = sqrt(sum(analytic_geo^2))
Error_true_mani_K= rep(0,length(num_neigh_true))
for(j in 1:length(num_neigh_true)){
  IsomapGdist = getIsomapGdist$getIsomapGdist(noiseless_data_tmp,num_neigh_true[j])
  Error_true_mani_K[j]=sqrt(sum((IsomapGdist - analytic_geo )^2))/norm_analytic_geo
}
ind_op_true=min(which(Error_true_mani_K==min(Error_true_mani_K)))
estim_geo_noiseless_data = getIsomapGdist$getIsomapGdist(noiseless_data_tmp,num_neigh_true[ind_op_true]
image.plot(estim_geo_noiseless_data,main='geo estimation')
```



If the analytic derviation is correct, then estim_geo_noiseless_data should be extremely similar to analytic_geo. We can assess their closeness using assess_goodness_estimation.R. Running this notebook for sce=1 gives rmse on the order of e-16.

```
library(MESS)
```

```
## Warning: package 'MESS' was built under R version 3.5.2

## Loading required package: geepack

## Loading required package: geeM

## Loading required package: Matrix

## Warning: package 'Matrix' was built under R version 3.5.2

##
```

```
## Attaching package: 'Matrix'

## The following object is masked from 'package:spam':
##
##     det
```

```r
source('assess_goodness_estimation.R')
assess_goodness_estimation(estim_geo_noiseless_data,sim$analytic_geo)
```

```
## $rmse
## [1] 0.02128164
##
## $epsilon_isometry_auc
## [1] 0.984879
##
## $pearson_corr
##       cor
## 0.9991665
```