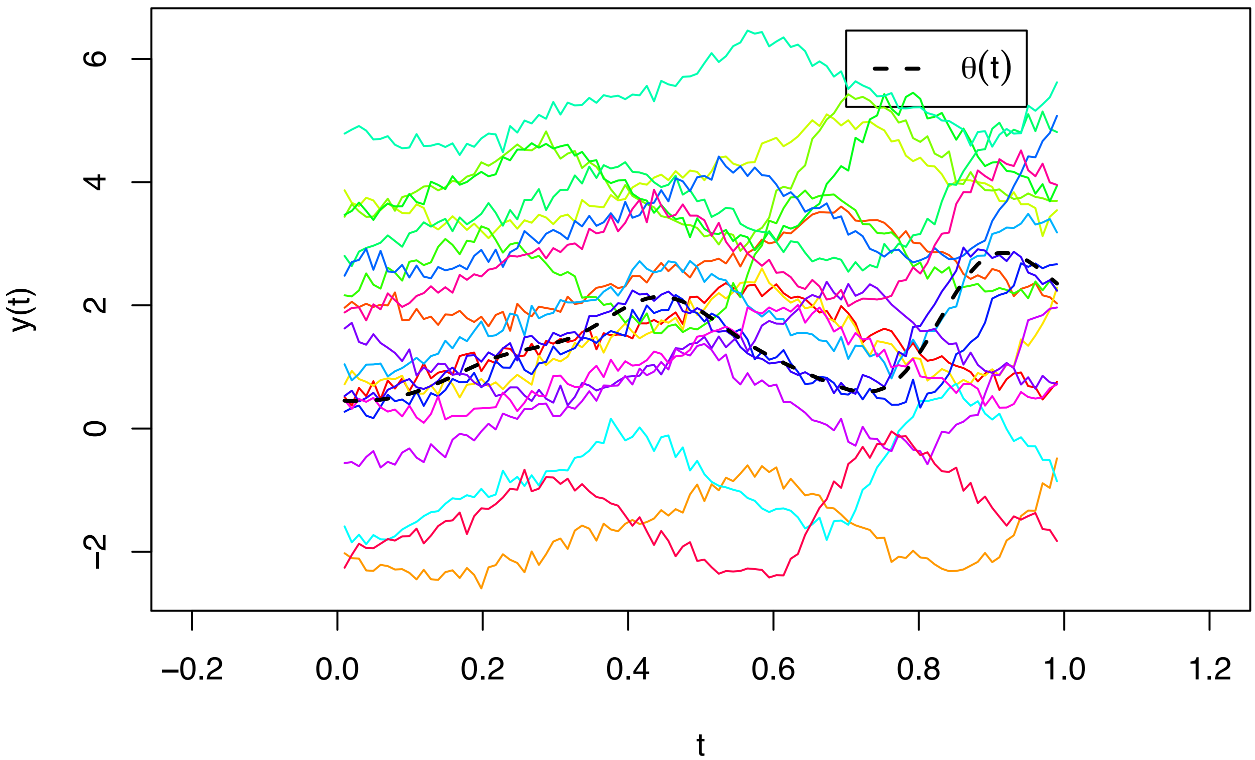
pavpop for sitar-type models

Lars Lau Raket

2016-01-08

Generate data

```r
# Number of samples
n <- 20
# Number of observation points
m <- 100

# Observation points
t <- seq(0, 1, length = m + 2)[2:(m + 1)]

# Common basis function (both mean and amplitude variation)
kts <- seq(0, 1, length = 10)
basis_fct <- make_basis_fct(kts = kts, intercept = TRUE,
                            control = list(boundary = c(-0.5, 1.5)))
amp_fct <- make_basis_fct(type = 'intercept')
df <- attr(basis_fct, 'df')

# Generate true mean weights
beta_t <- rexp(df, 0.5)

# Generate random intercepts
b_t <- rnorm(n, sd = 2)

# Generate warping function and random parameters
sigma <- 0.1
warp_fct <- make_warp_fct(type = 'linear')
warp_cov_t <- matrix(c(2, 0.5, 0.5, 1), 2, 2)
w_t <- replicate(n, (t(chol(warp_cov_t)) %*% rnorm(2, sd = sigma))[, 1])

# Generate data
y <- lapply(1:n, function(i) {as.numeric(basis_fct(warp_fct(w_t[, i], t)) %*% beta_t
                                         + amp_fct(t) %*% b_t[i]
                                         + rnorm(m, sd = sigma))})
t <- lapply(1:n, function(x) t)

# Plot observations
plot(0, 0, xlim = c(-0.2, 1.2), ylim = range(y), type = 'n',
     xlab = 't', ylab = 'y(t)')
legend(0.7, range(y)[2], legend = expression(theta(t)), lty = 2, lwd = 2)
for (i in 1:n) lines(t[[i]], y[[i]], col = rainbow(n)[i])
lines(t[[1]], basis_fct(t[[1]]) %*% beta_t , lwd = 2, lty = 2)
```



pavpop estimation

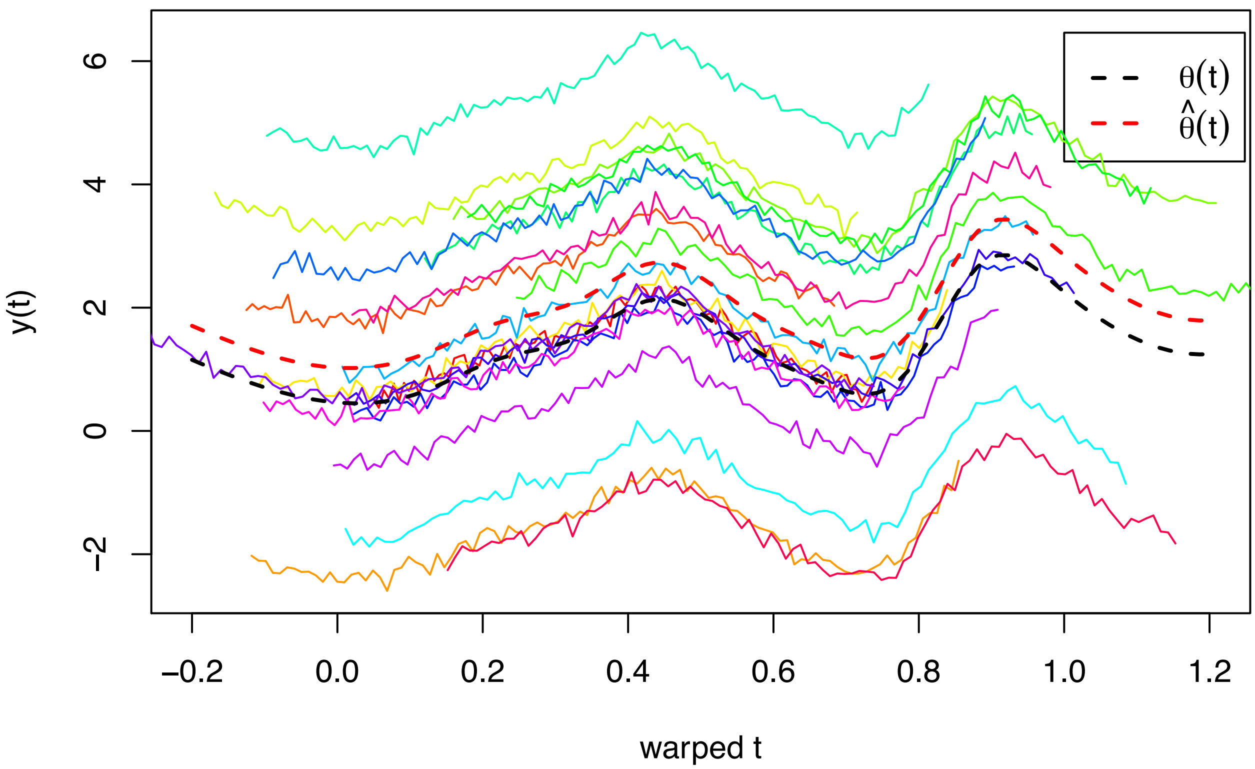
We now set up pavpop to estimate in the sitar model

```
amp_cov <- make_cov_fct(id_cov, noise = FALSE)
warp_cov <- make_cov_fct(unstr_cov, param = c(1, 1, 0), noise = FALSE)

res <- pavpop(y, t, basis_fct, warp_fct, amp_cov, warp_cov, amp_fct, iter = c(10, 10))
```

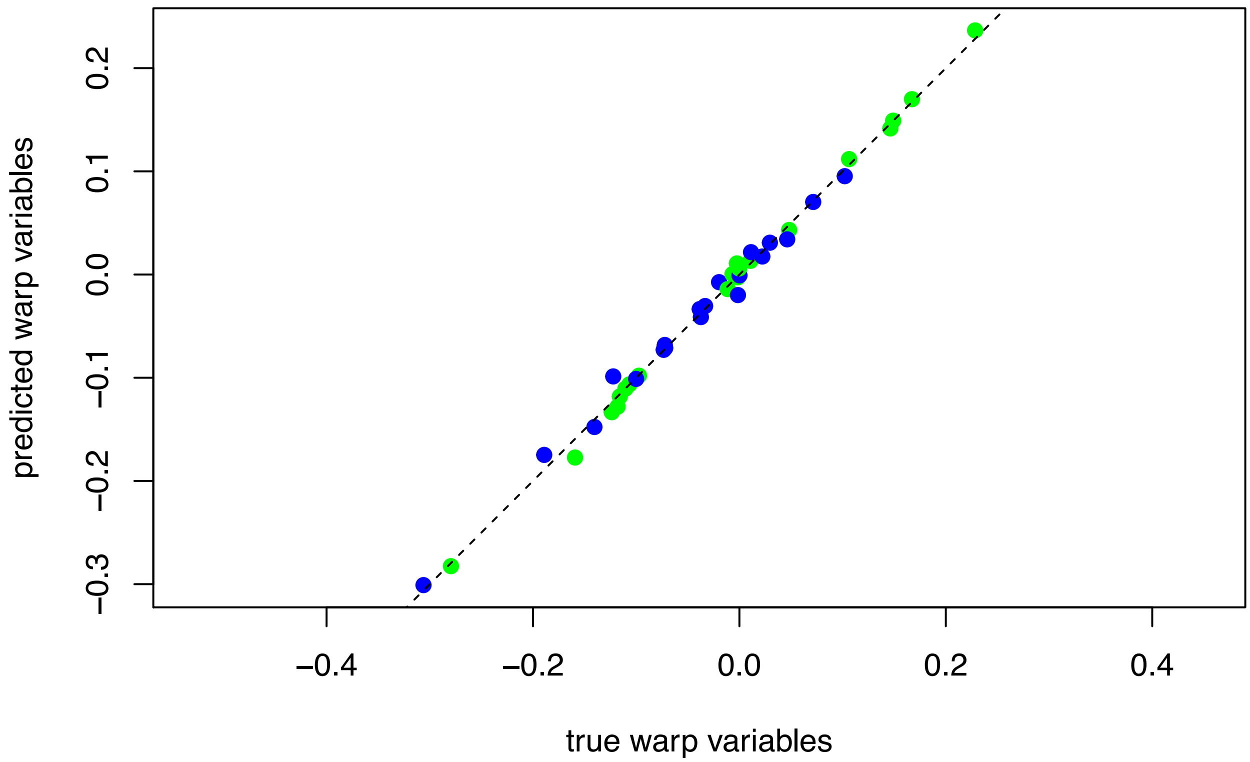
```
#> Outer    :   Inner   :   Estimates
#> 1    :    1    2    3    4    5    6    7    8    9    10   :    98.67771 0.8286171 0.6685879 0.001
#> Linearized likelihood:    -7498.142
#> 2    :    1    2    3    4    5    6    7    8    9    10   :    221.7119 1.032813 0.6209048 0.1720643
#> Linearized likelihood:    -7905.568
#> 3    :    1    :    221.7122 1.068347 0.6101104 0.1796776
#> Linearized likelihood:    -7953.722
#> 4    :    1    2    3    4    5    6    7    8    9    10   :    281.1653 1.434581 0.7486549 0.2011374
#> Linearized likelihood:    -8366.284
#> 5    :    1    :    281.1658 1.4262 0.7460634 0.1781196
#> Linearized likelihood:    -8398.815
#> 6    :    1    2    3    4    5    6    7    8    9    10   :    281.169 1.635282 0.8460421 0.2030552
#> Linearized likelihood:    -8592.755
#> 7    :    1    :    281.1749 1.594797 0.7838515 0.1455889
#> Linearized likelihood:    -8604.034
#> 8    :    1    2    3    4    5    6    7    8    9    10   :    281.1803 1.643059 0.9271104 0.1973406
#> Linearized likelihood:    -8645.912
#> 9    :    1    :    326.0857 1.613557 0.7838672 0.132576
#> Linearized likelihood:    -8651.483
```

```
#> 10    :    1    2    3    4    5    6    7    8    9    10   :     326.0858 1.621135 0.9840639 0.190829
#> Linearized likelihood:     -8655.955
```

```r
# Plot aligned samples
t_p <- seq(-0.2, 1.2, length = m + 2)
plot(0, 0, xlim = c(-0.2, 1.2), ylim = range(y), type = 'n',
     xlab = 'warped t', ylab = 'y(t)')
legend(1, range(y)[2], legend = c(expression(theta(t)), expression(hat(theta)(t))),
       lty = 2, lwd = 2, col = c('black', 'red'))
for (i in 1:n) lines(warp_fct(res$w[, i], t[[i]]), y[[i]], col = rainbow(n)[i])
lines(t_p, basis_fct(t_p) %*% beta_t , lwd = 2, lty = 2)
lines(t_p, basis_fct(t_p) %*% res$c, lwd = 2, lty = 2, col = 'red')
```




```r
plot(as.numeric(w_t), as.numeric(res$w), xlab = 'true warp variables',
     ylab = 'predicted warp variables', pch = 19, col = c('green', 'blue'), asp = 1)
abline(0, 1, lty = 2)
```



Are the parameter estimates okay?

```r
# Noise standard deviation: 0.1
res$sigma
```

```
#> [1] 0.1014726
```

```r
# Amplitude standard deviations: 2
res$sigma * sqrt(res$amp_cov_par)
```

```
#>    scale
#> 1.832377
```

```r
# Warp covariance

# True
sigma^2 * warp_cov_t
```

```
#>       [,1]  [,2]
#> [1,] 0.020 0.005
#> [2,] 0.005 0.010
```

```r
# Observed
cov(t(w_t))
```

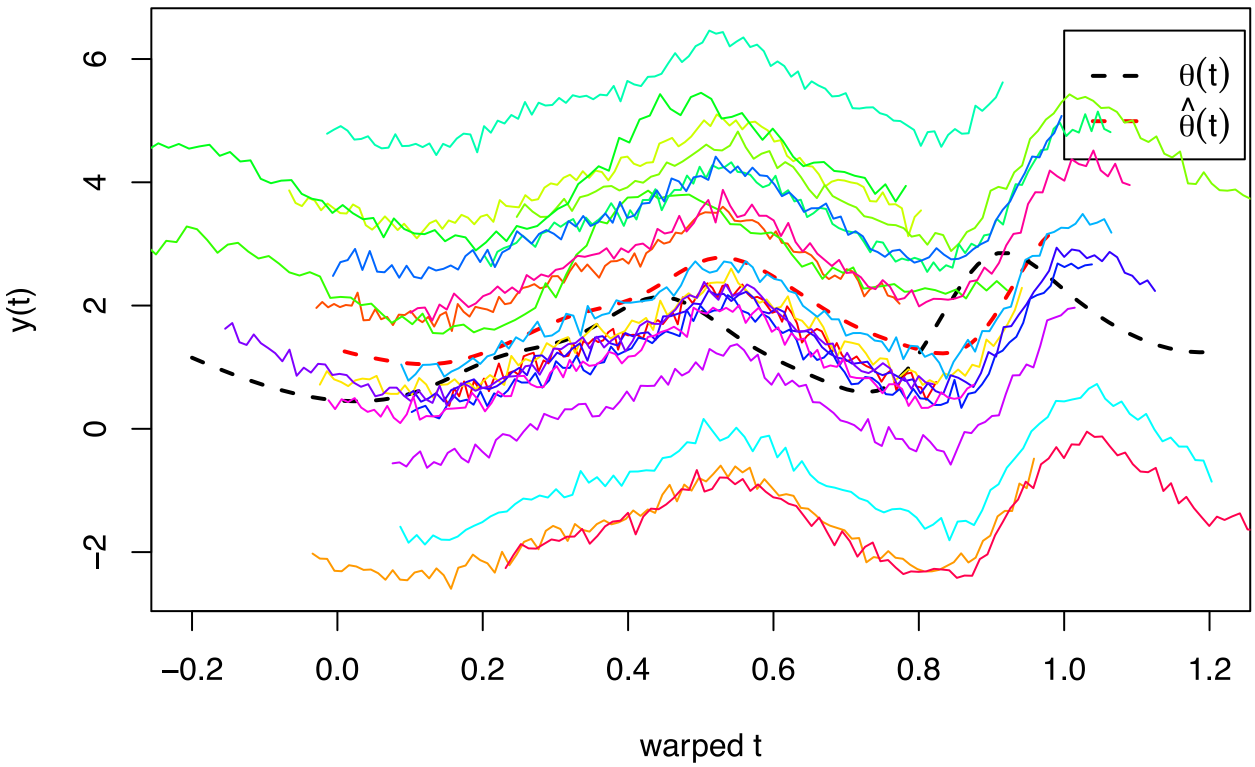
```
#>            [,1]        [,2]
#> [1,] 0.01647135 0.001772830
#> [2,] 0.00177283 0.008995296
```

```r
# estimated
res$sigma^2 * warp_cov(1:2, res$warp_cov_par)
```

```
#>             [,1]        [,2]
#> [1,] 0.016692332 0.001964908
#> [2,] 0.001964908 0.010132608
```

Traditional sitar estimation

```r
library(sitar)

dat <- data.frame(x = unlist(t), y = unlist(y), id = rep(1:n, each = m))

res_sitar <- sitar(x = x, y = y, id = id, data = dat, df = 14)


plot(res_sitar, opt = 'd', xlim = c(-0.2, 1.2), ylim = range(y), lwd = 2, lty = 2,
     col = 'red', xlab = 'warped t', ylab = 'y(t)')
lines(t_p, basis_fct(t_p) %*% beta_t , lwd = 2, lty = 2)
legend(1, range(y)[2], legend = c(expression(theta(t)), expression(hat(theta)(t))),
       lty = 2, lwd = 2, col = c('black', 'red'))


# Plot warped curves
abc <- random.effects(res_sitar)
abc$a <- 0
warped <- with(dat, xyadj(x, y, id, res_sitar, abc))
for (i in 1:n)
  with(warped, lines(x[1:m + (i - 1) * m], y[1:m + (i - 1) * m], col = rainbow(n)[i]))
```




```r
plot(as.numeric(w_t), rbind(-abc$b * exp(abc$c), exp(abc$c) - 1),
     xlab = 'true warp variables', ylab = 'predicted warp variables',
     pch = 19, col = c('green', 'blue'), asp = 1)
abline(0, 1, lty = 2)
```