Install the package

You should first download the source code available at https://github.com/guillemr/robust-fpop. Once this is done you can build and compile the package. In command line on linux you can do this using:

R CMD build robseg

and then

R CMD INSTALL robseg_2016.10.19.tar.gz

Load the package

You can then load the package as follow.

```
require(robseg)
```

```
##      robseg
```

Simple examples

In this section we will illustrate the robseg function for the L2, biweight, L1 and Huber loss. As an example we will consider the simulation made in [1] using a student noise rather than a Gaussian noise.

```
source("Simulation.R")
i <- 1 ## there are 6 scenarios we take the first one
dfree <- 6 ## degree of freedom of the Student noise
## we recover the info of the first scenario
Ktrue <- Simu[[i]]$Ktrue
bkptrue <- as.integer(Simu[[i]]$bkpPage29[-c(1, Ktrue+1)])
signaltrue <-  Simu[[i]]$signal
sigmatrue <- Simu[[i]]$sigma

## we simulate one profile
set.seed(1)
x.data <- signaltrue + rt(n=length(signaltrue), df=dfree)*sigmatrue
```

We estimate the variance using MAD.

```
est.sd <- varDiff(x.data)
```

Fpop (L2)

We then run fpop with the L2 loss [1] fixing the penalty (λ) to $\log(n)$. In this example, some outlier data points are detected as segments.

```
## run dynamic programming
res.l2 <- fpop_intern(x.data/est.sd,  test.stat="Normal", pen.value=2*log(length(x.data)))

## same thing with changepoint package
library(changepoint)
```

```
##      zoo
```

```
##
##    'zoo'
```

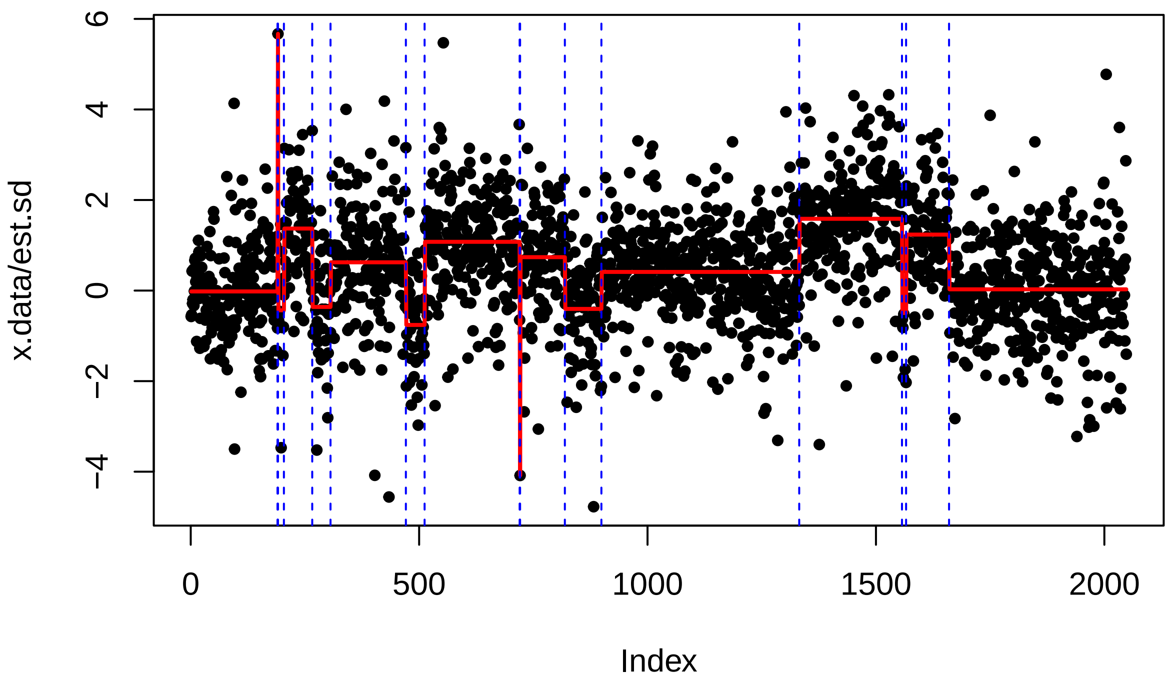
```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
## Successfully loaded changepoint package version 2.3
##  WARNING: From v.2.3 the default method in cpt.* functions has changed from AMOC to PELT.
##  See NEWS for details of all changes.
```

```
res.l2_ <- cpt.mean(x.data/est.sd, penalty="Manual", pen.value=2*log(length(x.data)), method="PELT")
identical(res.l2$cpts, res.l2_@cpts)
```

```
## [1] TRUE
```

```
## estimated changepoints
cpt <- res.l2$cpts[-length(res.l2$cpts)]

## simple ploting of changes and smoothed profile
plot(x.data/est.sd, pch=20, col="black")
lines(res.l2$smt.signal, col="red", lwd=2)
abline(v=cpt, lty=2, col="blue")
```



Robust Fpop (Huber)

We then run fpop with the Huber loss fixing the penalty to $1.4\log(n)$ and with a threshold parameter of 1.345.

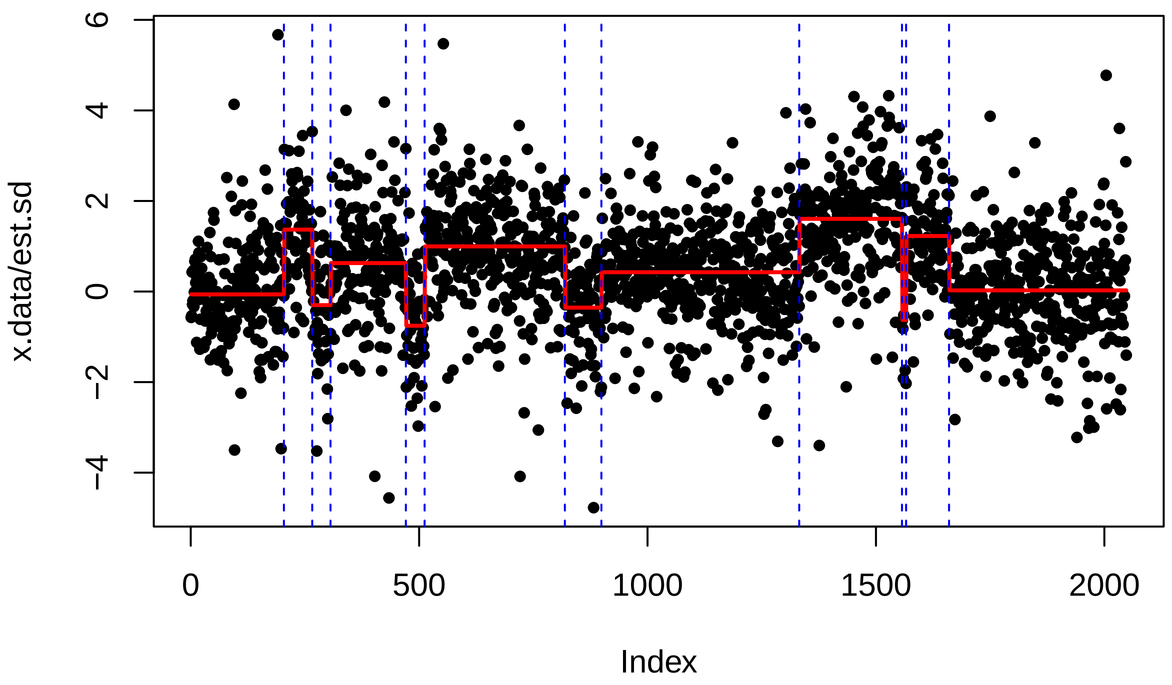
```
## run dynamic programming
res.hu <- fpop_intern(x.data/est.sd,  test.stat="Huber", pen.value=1.4*log(length(x.data)), lthreshold=

## check that it returns the same as Rob_seg.std
# res.hu_ <- Rob_seg.std(x.data/est.sd,  "Huber",  lambda=1.4*log(length(x.data)), lthreshold=1.345)
# identical(res.hu$cpts , res.hu_$t.est)
## end check

## estimated changepoints
cpt <- res.hu$cpts[-length(res.hu$cpts)]

## simple ploting of changes and smoothed profile
```

```r
plot(x.data/est.sd, pch=20, col="black")
lines(res.hu$smt.signal, col="red", lwd=2)
abline(v=cpt, lty=2, col="blue")
```



Robust Fpop (Biweight)

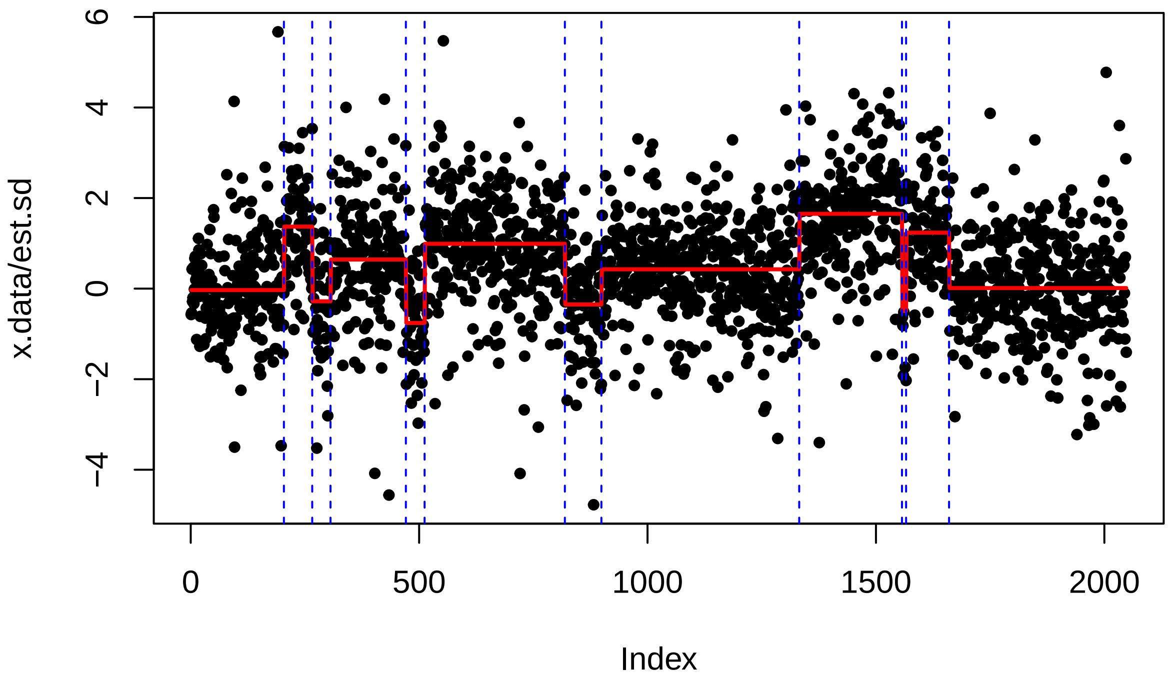
To run fpop with the biweight loss fixing the penalty to $2\log(n)$ and with a threshold parameter of 3.

```r
## run dynamic programming
res.ou <- fpop_intern(x.data/est.sd,  test.stat="Outlier", pen.value=2*log(length(x.data)), lthreshold=

## check that it returns the same as Rob_seg.std
# res.ou_ <- Rob_seg.std(x.data/est.sd,  "Outlier", lambda=2*log(length(x.data)), lthreshold=3)
# identical(res.ou$cpts , res.ou_$t.est)
## end check

## estimated changepoints
cpt <- res.ou$cpts[-length(res.ou$cpts)]

## simple ploting of changes and smoothed profile
plot(x.data/est.sd, pch=20, col="black")
lines(res.ou$smt.signal, col="red", lwd=2)
abline(v=cpt, lty=2, col="blue")
```



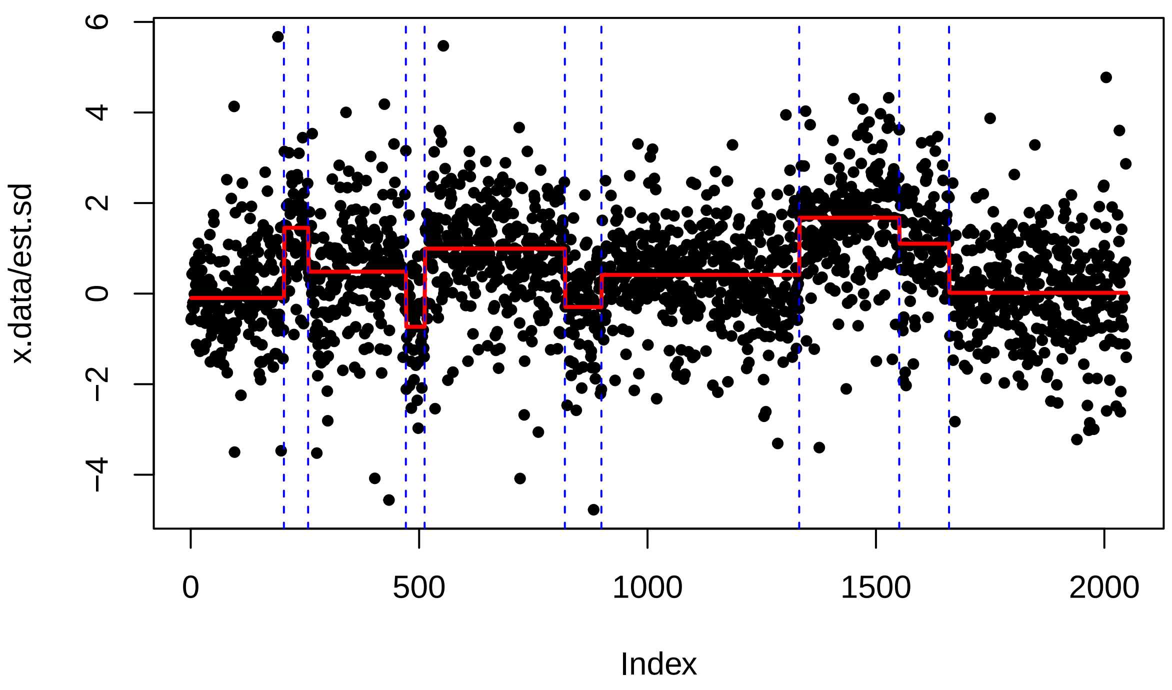
Robust Fpop (L1)

We then run fpop with L1 loss fixing the penalty to $\log(n)$. In this example on segment is not detected : [1556 - 1597].

```
## run dynamic programming
res.l1 <- fpop_intern(x.data/est.sd,  test.stat="Laplace", pen.value=1*log(length(x.data)))
## check that it returns the same as Rob_seg.std
# res.l1_ <- Rob_seg.std(x.data/est.sd,  "L1", lambda=1*log(length(x.data)))
# identical(res.l1$cpts , res.l1_$t.est)
## end check

## estimated changepoints
cpt <- res.l1$cpts[-length(res.l1$cpts)]

## simple ploting of changes and smoothed profile
plot(x.data/est.sd, pch=20, col="black")
lines(res.l1$smt.signal, col="red", lwd=2)
abline(v=cpt, lty=2, col="blue")
```



[1] Maidstone, Robert, et al. “On optimal multiple changepoint algorithms for large data.” Statistics and Computing (2014): 1-15.)