The *waveTiling* package

Kristof De Beuf

November 1, 2014

Contents

1 Introduction

In this *waveTiling* package vignette the package's main functionalities to conduct a tiling array trancriptome analysis are illustrated. The package contains an implementation of the basic wavelet-based functional model introduced in [1], and its extensions towards more complex designs described in [3]. The leaf development data set [2] contains genome-wide expression data measured for six developmental time points (day 8 to day 13) on the plant species *Arabidopsis thaliana*. The experiment was conducted with AGRONOMICS1 tiling arrays [4] and contains three biological replicates per time point.

2 Read in and prepare data for analysis

First we have to load the *waveTiling* package and the *waveTilingData* package. The latter contains an *TilingFeatureSet* (leafdev) from the *oligoClasses* package [5] with the expression values for the leaf development experiment. Make sure to also load the *pd.atdschip.tiling* package which contains the tiling array info to map the probe locations on the array to the exact genomic positions. The *pd.atdschip.tiling* package was created by using the *pdInfoBuilder* package [6], which should also be used to build similar packages for other array designs.

```
> library(waveTiling)
> library(waveTilingData)
> library(pd.atdschip.tiling)
> data(leafdev)
```

We first change the class to *WaveTilingFeatureSet*, which is used as input for the wavelet-based transcriptome analysis, and add the phenotypic data for this experiment.

```
> leafdev <- as(leafdev,"WaveTilingFeatureSet")
> leafdev <- addPheno(leafdev,noGroups=6,
+         groupNames=c("day8","day9","day10","day11","day12","day13"),
```

```
+         replics=rep(3,6))
> leafdev

WaveTilingFeatureSet (storageMode: lockedEnvironment)
assayData: 6553600 features, 18 samples
  element names: exprs
protocolData
  rowNames: caquinof_20091023_S100_v4.CEL
    caquinof_20091023_S101_v4.CEL ...
    caquinof_20091023_S117_v4.CEL (18 total)
  varLabels: exprs dates
  varMetadata: labelDescription channel
phenoData
  rowNames: day8.1 day8.2 ... day13.3 (18 total)
  varLabels: group replicate
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation: pd.atdschip.tiling

          used  (Mb) gc trigger   (Mb)  max used   (Mb)
Ncells 2483146 132.7    7626154  407.3   9532693  509.2
Vcells 4092660  31.3  165931772 1266.0 141946437 1083.0
```

Before starting the transcriptome analysis, the probes that map to several genomic locations (either PM or MM, or forward and reverse strand) are filtered using `filterOverlap`. This function can also be used if the probes have to be remapped to another version of the genome sequence as the version used for the array design. For instance, the probes on the AGRONOMICS1 array are build based on the TAIR 8 genome, and remapped onto the TAIR 9 sequence. The function needs an argument `BSgenomeObject` available from loading the appropriate *BSgenome* package [7]. The output is an object of class *mapFilterProbe*. After filtering and/or remapping, the expression data are background-corrected and quantile-normalized (`bgCorrQn`). The *mapFilterProbe* `leafdevMapAndFilterTAIR9` is used to make sure only the filtered probes are used in the background correction and normalization step.

```
> library(BSgenome.Athaliana.TAIR.TAIR9)
> # leafdevMapAndFilterTAIR9 <- filterOverlap(leafdev,remap=TRUE,
> #        BSgenomeObject=Athaliana,chrId=1:7,
> #        strand="both",MM=FALSE)
> data(leafdevMapAndFilterTAIR9)
>
> # leafdevBQ <- bgCorrQn(leafdev,useMapFilter=leafdevMapAndFilterTAIR9)
```

3 Wavelet-based transcriptome analysis

3.1 Standard analysis flow

The analysis has to be conducted in a chromosome- and strand-wise manner. First, the wavelet-based model is fitted to the expression data, leading to a *WfmFit*-class object leafdevFit.

```
> data(leafdevBQ)
> chromosome <- 1
> strand <- "forward"
```

```
> leafdevFit <- wfm.fit(leafdevBQ,filter.overlap=leafdevMapAndFilterTAIR9,
+         design="time",n.levels=10,
+         chromosome=chromosome,strand=strand,minPos=22000000,
+         maxPos=24000000,var.eps="marg",prior="improper",
+         skiplevels=1,save.obs="plot",trace=TRUE)
> leafdevFit

Fitted object from wavelet based functional model - Time Design
Wavelet filter used: haar
Number of wavelet decomposition levels: 10
Number of probes used for estimation: 52224

Genome Info :
        Chromosome: 1
        Strand: forward
        Minimum probe position: 22000000
        Maximum probe position: 23988867
```

If the redundant probes have been filtered using `filterOverlap` the resulting *mapFilterProbe* class object should be given as an argument `filter.overlap`, to ensure that the expression values are properly linked to the genomic information such as chromosome and strand. In this analysis we use a time-course design (`design`). The number of levels in the wavelet decomposition is 10 (`n.levels`). We use marginal maximum likelihood to estimate the residual variances (`var.eps`) and put an improper prior (`prior`) on the effect functions (see [1]).

Next, the *WfmFit*-class object `leafdevFit` is used as input for the inference function `wfm.inference`. This function outputs the *WfmInf*-class object `leafdevInf` from which transcriptionally active regions of interest, given a chosen threshold value, can be extracted.

```
> delta <- log(1.2,2)
> leafdevInfCompare <- wfm.inference(leafdevFit,
+         contrasts="compare",delta=c("median",delta))
> leafdevInfCompare

Inference object from wavelet based functional model - Pairwise Comparison
Genome Info :
        Chromosome: 1
        Strand: forward
        Minimum probe position: 22000000
        Maximum probe position: 23988867
```

The `contrasts` argument is used to indicate the type of inference analysis one wants to conduct, e.g. `compare` to detect differentially expressed regions between the different time points. By default, transcriptionally active regions based on the mean expression over all arrays are also given in the output. With the `delta` the threshold value to use in the statistical tests can be set. It is a *vector* with as first element the threshold for the overall mean trancript discovery. This is taken to be the median of the expression values over all arrays in this case. The second element is the threshold for the differential expression analysis. This threshold is equal for each pairwise comparison if the length of `delta` is 2. If one wants to use different thresholds the length of `delta` must be $r+1$ with r the number of pairwise comparisons, where each element is associated with an individual threshold value.

Much information is stored in the *WfmFit*-class and *WfmInf*-class objects. Primarily, we are interested in the genomic regions that are significantly transcriptionally affected according to the research question of interest.

```
> sigGenomeRegionsCompare <- getGenomicRegions(leafdevInfCompare)
> sigGenomeRegionsCompare[[2]]

IRanges of length 23
        start      end width
[1]  22448608 22448704    97
[2]  22700160 22700256    97
[3]  22804928 22805024    97
[4]  22824704 22824800    97
[5]  22825792 22825888    97
...       ...      ...   ...
[19] 23334464 23334560    97
[20] 23457312 23457408    97
[21] 23457824 23457920    97
[22] 23619042 23619138    97
[23] 23953826 23953986   161

> length(sigGenomeRegionsCompare)

[1] 16
```

The `getGenomicRegions` accessor outputs a `list` of `IRanges` objects [8] denoting the start and end position of each significant region. The first element in the `list` always gives the significant regions for the mean expression over all arrays (transcript discovery). Elements 2 to 16 in `sigGenomeRegions` give the differentially expressed regions between any pair of contrasts between different time points. The order is always 2-1, 3-1, 3-2, 4-1,... Hence, `sigGenomeRegions[[2]]` gives the differentially expressed regions between time point 2 and time point 1.

If information on the annotation of the studied organism is available, we can extract both significantly affected genes with `getSigGenes`, and the non-annotated regions with `getNonAnnotatedRegions`. Both functions output a *list* of *GRanges* objects [9]. The annotation info can be obtained from ab appropriate object of class TxDb representing an annotation database generated from BioMart. For the current data we make use of the *TxDb.Athaliana.BioMart.plantsmart22* package [10].

```
> library(TxDb.Athaliana.BioMart.plantsmart22)
> sigGenesCompare <- getSigGenes(fit=leafdevFit,inf=leafdevInfCompare,
+         biomartObj=TxDb.Athaliana.BioMart.plantsmart22)
> head(sigGenesCompare[[2]])

GRanges object with 6 ranges and 6 metadata columns:
      seqnames               ranges strand |     tx_id     tx_name
         <Rle>            <IRanges>  <Rle> | <integer> <character>
  [1]        1 [22447848, 22449526]      - |      9181 AT1G60970.1
  [2]        1 [22699715, 22701169]      + |      4010 AT1G61520.3
  [3]        1 [22700010, 22701383]      + |      4011 AT1G61520.1
  [4]        1 [22700073, 22701383]      + |      4012 AT1G61520.2
  [5]        1 [22804613, 22806318]      - |      9258 AT1G61750.1
  [6]        1 [22824440, 22826774]      + |      4027 AT1G61800.1
         regNo percOverGene percOverReg totPercOverGene
     <integer>    <numeric>   <numeric>       <numeric>
  [1]         1     5.777248         100        5.777248
  [2]         2     6.666667         100        6.666667
  [3]         2     7.059680         100        7.059680
```

```
  [4]         2     7.398932         100        7.398932
  [5]         3     5.685815         100        5.685815
  [6]         4     4.154176         100       12.462527
  -------
  seqinfo: 7 sequences (1 circular) from an unspecified genome
```

```
> nonAnnoCompare <- getNonAnnotatedRegions(fit=leafdevFit,inf=leafdevInfCompare,
+         biomartObj=TxDb.Athaliana.BioMart.plantsmart22)
> head(nonAnnoCompare[[2]])
```

```
GRanges object with 6 ranges and 0 metadata columns:
      seqnames              ranges strand
         <Rle>           <IRanges>  <Rle>
  [1]        1 [22001344, 22001440]      +
  [2]        1 [22004577, 22004801]      +
  [3]        1 [22007872, 22008192]      +
  [4]        1 [22009696, 22009920]      +
  [5]        1 [22010016, 22010432]      +
  [6]        1 [22012864, 22013216]      +
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

Using the same *WfmFit*-object `leafdevFit`, we can run the analysis to analyze transcriptional time effects (`leafDevInfTimeEffect`) and have a look at time-wise trancriptionally active regions (`leafdevInfMeans`).

```
> leafdevInfTimeEffect <- wfm.inference(leafdevFit,contrasts="effects",
+         delta=c("median",2,0.2,0.2,0.2,0.2))
> leafdevInfMeans <- wfm.inference(leafdevFit,contrasts="means",
+         delta=4,minRunPos=30,minRunProbe=-1)
```

Besides the available standard design analyses given by the `design` argument in the `wfm.fit` function and the `contrasts` argument in the `wfm.inference`, it is also possible to provide custom design and contrast matrices in the *waveTiling* package. This `custom` design is illustrated based on the polynomial contrast matrix used in a time-course analysis.

```
> custDes <- matrix(0,nrow=18,ncol=6)
> orderedFactor <- factor(1:6,ordered=TRUE)
> desPoly <- lm(rnorm(6)~orderedFactor,x=TRUE)$x
> custDes[,1] <- 1
> custDes[,2:6] <- apply(desPoly[,2:6],2,rep,getReplics(leafdevBQ))
> custDes
```

```
      [,1]       [,2]       [,3]       [,4]       [,5]        [,6]
 [1,]    1 -0.5976143  0.5455447 -0.3726780  0.1889822 -0.06299408
 [2,]    1 -0.5976143  0.5455447 -0.3726780  0.1889822 -0.06299408
 [3,]    1 -0.5976143  0.5455447 -0.3726780  0.1889822 -0.06299408
 [4,]    1 -0.3585686 -0.1091089  0.5217492 -0.5669467  0.31497039
 [5,]    1 -0.3585686 -0.1091089  0.5217492 -0.5669467  0.31497039
 [6,]    1 -0.3585686 -0.1091089  0.5217492 -0.5669467  0.31497039
 [7,]    1 -0.1195229 -0.4364358  0.2981424  0.3779645 -0.62994079
 [8,]    1 -0.1195229 -0.4364358  0.2981424  0.3779645 -0.62994079
 [9,]    1 -0.1195229 -0.4364358  0.2981424  0.3779645 -0.62994079
[10,]    1  0.1195229 -0.4364358 -0.2981424  0.3779645  0.62994079
```

```
[11,]    1  0.1195229 -0.4364358 -0.2981424  0.3779645  0.62994079
[12,]    1  0.1195229 -0.4364358 -0.2981424  0.3779645  0.62994079
[13,]    1  0.3585686 -0.1091089 -0.5217492 -0.5669467 -0.31497039
[14,]    1  0.3585686 -0.1091089 -0.5217492 -0.5669467 -0.31497039
[15,]    1  0.3585686 -0.1091089 -0.5217492 -0.5669467 -0.31497039
[16,]    1  0.5976143  0.5455447  0.3726780  0.1889822  0.06299408
[17,]    1  0.5976143  0.5455447  0.3726780  0.1889822  0.06299408
[18,]    1  0.5976143  0.5455447  0.3726780  0.1889822  0.06299408
```

```
> leafdevFitCustom <- wfm.fit(leafdevBQ,filter.overlap=leafdevMapAndFilterTAIR9,
+         design="custom",design.matrix=custDes,n.levels=10,
+         chromosome=chromosome,strand=strand,minPos=22000000,
+         maxPos=24000000,var.eps="marg",prior="improper",
+         skiplevels=1,save.obs="plot",trace=TRUE)
> noGroups <- getNoGroups(leafdevBQ)
> myContrastMat <- matrix(0,nrow=noGroups*(noGroups-1)/2,ncol=noGroups)
> hlp1 <- rep(2:noGroups,1:(noGroups-1))
> hlp2 <- unlist(sapply(1:(noGroups-1),function(x) seq(1:x)))
> for (i in 1:nrow(myContrastMat))
+ {
+         myContrastMat[i,hlp1[i]] <- 1
+         myContrastMat[i,hlp2[i]] <- -1
+ }
> myContrastMat
```

```
      [,1] [,2] [,3] [,4] [,5] [,6]
 [1,]   -1    1    0    0    0    0
 [2,]   -1    0    1    0    0    0
 [3,]    0   -1    1    0    0    0
 [4,]   -1    0    0    1    0    0
 [5,]    0   -1    0    1    0    0
 [6,]    0    0   -1    1    0    0
 [7,]   -1    0    0    0    1    0
 [8,]    0   -1    0    0    1    0
 [9,]    0    0   -1    0    1    0
[10,]    0    0    0   -1    1    0
[11,]   -1    0    0    0    0    1
[12,]    0   -1    0    0    0    1
[13,]    0    0   -1    0    0    1
[14,]    0    0    0   -1    0    1
[15,]    0    0    0    0   -1    1
```

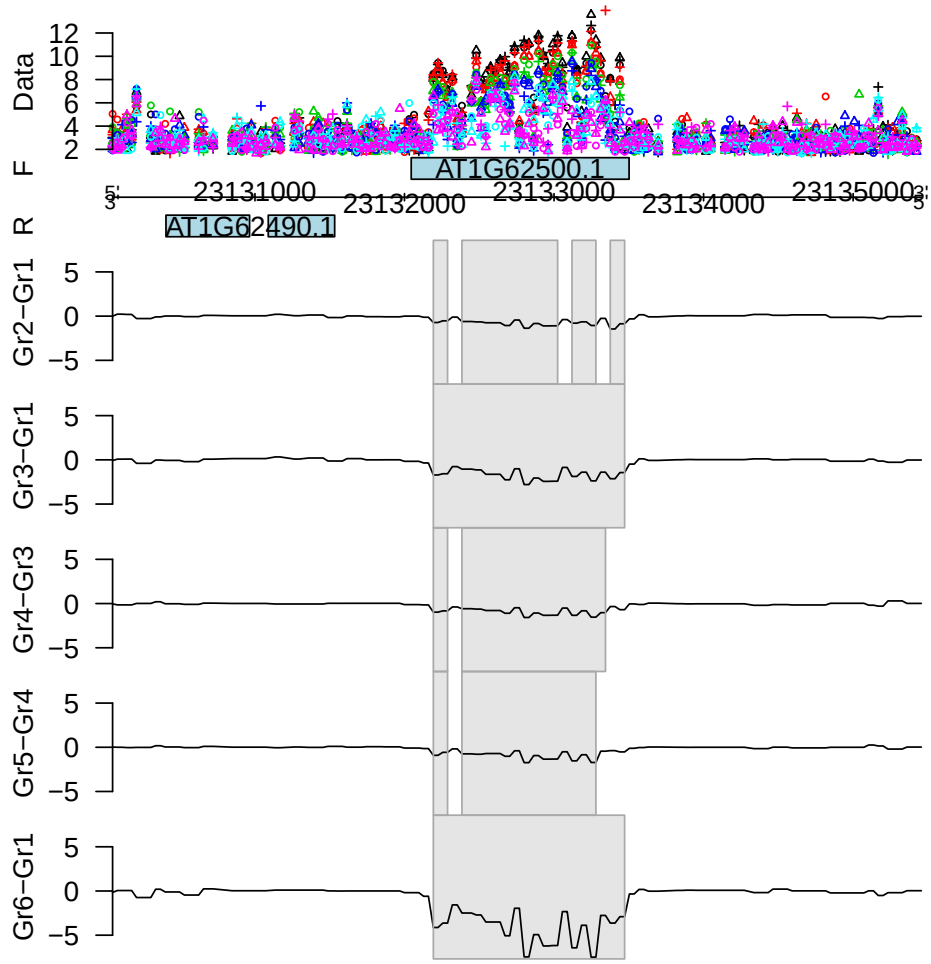
```
> leafdevInfCustom <- wfm.inference(leafdevFitCustom,contrast.matrix=myContrastMat,
+         delta=c("median",log(1.2,2)))
```

3.2 Plot function

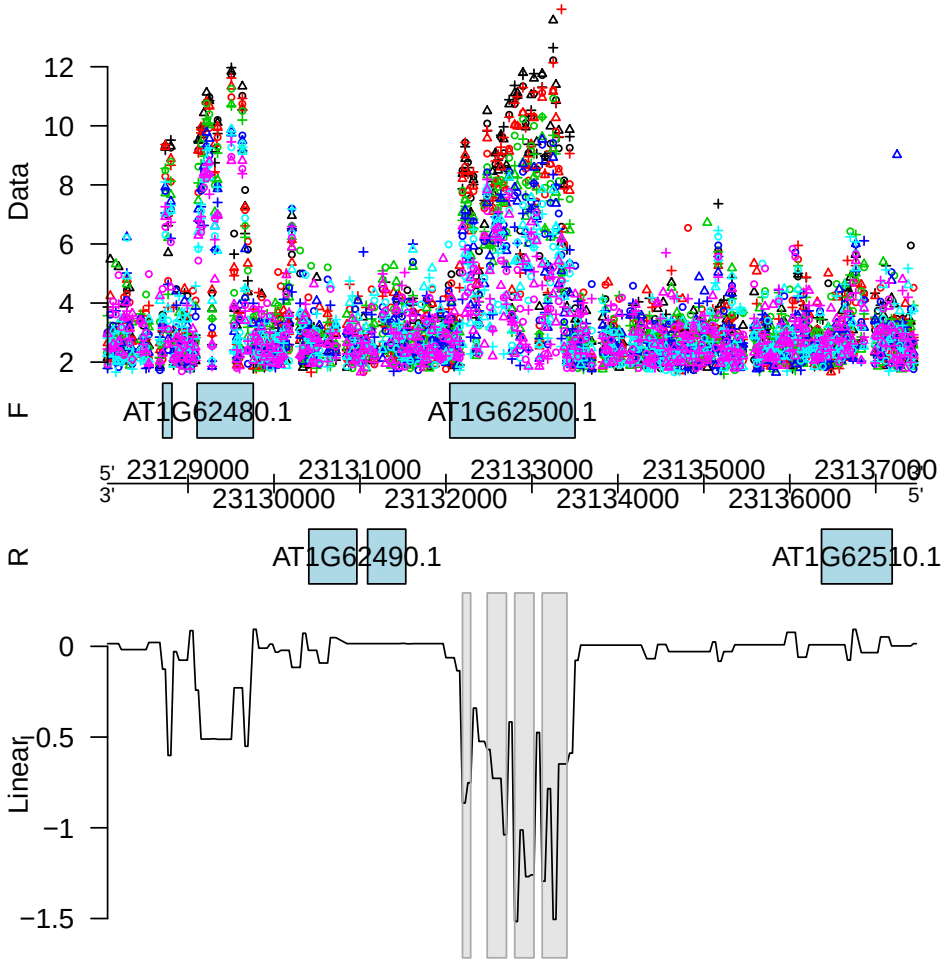
Plots can be made very easily using the `plotWfm` function which needs both the *WfmFit*- and *WfmInf*-class objects as input. It also needs an appropriate annotation file. The plot function makes use of the implementations in the *GenomeGraphs*-package [11].

```
> trs <- transcripts(TxDb.Athaliana.BioMart.plantsmart22)
> sel <- trs[elementMetadata(trs)$tx_name %in% "AT1G62500.1",]
```

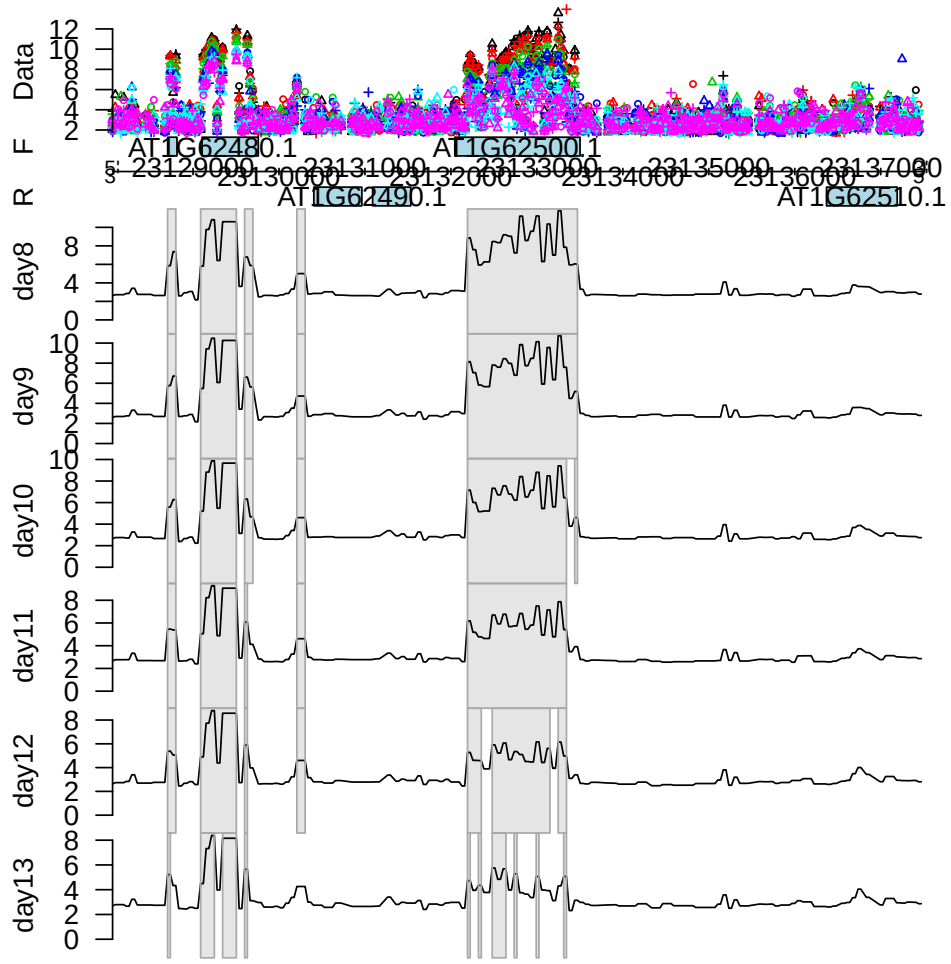
```
> start <- start(ranges(sel))-2000
> end <- end(ranges(sel))+2000
> plotWfm(fit=leafdevFit,inf=leafdevInfCompare,
+         biomartObj=TxDb.Athaliana.BioMart.plantsmart22,
+         minPos=start,maxPos=end,two.strand=TRUE,
+         plotData=TRUE,plotMean=FALSE,tracks=c(1,2,6,10,11))
```




```
> start <- start(ranges(sel))-4000
> end <- end(ranges(sel))+4000
> plotWfm(fit=leafdevFit,inf=leafdevInfTimeEffect,
+         biomartObj=TxDb.Athaliana.BioMart.plantsmart22,
+         minPos=start,maxPos=end,two.strand=TRUE,
+         plotData=TRUE,plotMean=FALSE,tracks=1)
```




```
> plotWfm(fit=leafdevFit,inf=leafdevInfMeans,
+         biomartObj=TxDb.Athaliana.BioMart.plantsmart22,
+         minPos=start,maxPos=end,two.strand=TRUE,
+         plotData=TRUE,plotMean=FALSE,tracks=1:6)
```



3.3 Accessor functions

There are a number of accessor functions available that are not necessarily needed to run a standard trancriptome analysis, but still can extract useful information from the *WfmFit*- and *WfmInf*-class objects. Some of the more interesting ones are illustrated below. For a complete overview, consult the package's help pages.

```
> getGenomeInfo(leafdevFit)

Genome Info :
        Chromosome: 1
        Strand: forward
        Minimum probe position: 22000000
        Maximum probe position: 23988867

> dataOrigSpace <- getDataOrigSpace(leafdevFit)
> dim(dataOrigSpace)

[1]    18 52224

> dataOrigSpace[1:8,1:8]
```

```
           31084     1221    37911    38008     4886    46714    36553
day8.1  3.609509 2.210917 2.035332 3.052743 2.687550 2.814693 2.441430
day8.2  2.690457 2.146153 2.630093 2.758048 3.306454 2.444077 2.561254
day8.3  2.486904 2.240285 1.980014 3.132112 3.926132 2.486904 1.782565
day9.1  1.848369 2.136446 2.320278 2.669716 4.710797 3.007356 1.966924
day9.2  2.015207 3.676953 2.536061 2.916489 2.194234 2.486904 2.486904
day9.3  3.916482 2.280221 1.960119 2.331042 4.875751 3.792536 2.954089
day10.1 2.702066 2.702066 2.600848 3.129100 3.070930 2.957153 2.165519
day10.2 3.504079 3.393904 2.564108 2.857383 2.295185 2.655298 3.393904
           50423
day8.1  3.609509
day8.2  2.758048
day8.3  2.430280
day9.1  2.773344
day9.2  2.644008
day9.3  4.570723
day10.1 3.849458
day10.2 3.122948
```

```
> dataWaveletSpace <- getDataWaveletSpace(leafdevFit)
> dim(dataWaveletSpace)

[1]    18 52224

> dataWaveletSpace[1:8,1:8]

               [,1]       [,2]        [,3]       [,4]        [,5]
day8.1  -0.98895385 0.71941870  0.08990375  0.8259564  0.77453314
day8.2  -0.38488166 0.09047845 -0.60979259  0.1391550 -0.04047101
day8.3  -0.17438581 0.81465634 -1.01768780  0.4580037  0.03196901
day9.1   0.20370133 0.24709019 -1.20451459  0.5702250  0.52457438
day9.2   1.17503218 0.26900316  0.20694882  0.1110895  0.78258895
day9.3  -1.15701060 0.26228237 -0.76594871  1.1431329 -0.23621015
day10.1  0.00000000 0.37353024 -0.08045243  1.1907244  1.10961772
day10.2 -0.07790568 0.20737669  0.25463851 -0.1915943 -0.37206866
              [,6]        [,7]       [,8]
day8.1   0.5585598 -0.03129421  0.2240813
day8.2   0.2919523  0.62778205  0.3844214
day8.3   0.2091449 -0.18352604 -0.3584254
day9.1  -0.3094854  0.51377883  0.6471413
day9.2   0.7149518 -0.07017150 -0.3334522
day9.3  -1.4328004 -0.64600315  0.5937056
day10.1  0.5393628 -0.48174894 -0.3953267
day10.2 -0.4680818  1.02225539  0.1828262

> getDesignMatrix(leafdevFit)

      [,1]       [,2]       [,3]       [,4]       [,5]        [,6]
 [1,]    1 -0.5976143  0.5455447 -0.3726780  0.1889822 -0.06299408
 [2,]    1 -0.5976143  0.5455447 -0.3726780  0.1889822 -0.06299408
 [3,]    1 -0.5976143  0.5455447 -0.3726780  0.1889822 -0.06299408
 [4,]    1 -0.3585686 -0.1091089  0.5217492 -0.5669467  0.31497039
```

```
 [5,]    1 -0.3585686 -0.1091089  0.5217492 -0.5669467  0.31497039
 [6,]    1 -0.3585686 -0.1091089  0.5217492 -0.5669467  0.31497039
 [7,]    1 -0.1195229 -0.4364358  0.2981424  0.3779645 -0.62994079
 [8,]    1 -0.1195229 -0.4364358  0.2981424  0.3779645 -0.62994079
 [9,]    1 -0.1195229 -0.4364358  0.2981424  0.3779645 -0.62994079
[10,]    1  0.1195229 -0.4364358 -0.2981424  0.3779645  0.62994079
[11,]    1  0.1195229 -0.4364358 -0.2981424  0.3779645  0.62994079
[12,]    1  0.1195229 -0.4364358 -0.2981424  0.3779645  0.62994079
[13,]    1  0.3585686 -0.1091089 -0.5217492 -0.5669467 -0.31497039
[14,]    1  0.3585686 -0.1091089 -0.5217492 -0.5669467 -0.31497039
[15,]    1  0.3585686 -0.1091089 -0.5217492 -0.5669467 -0.31497039
[16,]    1  0.5976143  0.5455447  0.3726780  0.1889822  0.06299408
[17,]    1  0.5976143  0.5455447  0.3726780  0.1889822  0.06299408
[18,]    1  0.5976143  0.5455447  0.3726780  0.1889822  0.06299408
```

```
> probepos <- getProbePosition(leafdevFit)
> length(probepos)
```

```
[1] 52224
```

```
> head(probepos)
```

```
[1] 22000000 22000032 22000065 22000096 22000128 22000160
```

```
> effects <- getEff(leafdevInfCompare)
> dim(effects)
```

```
[1]    16 52224
```

```
> effects[1:8,1:8]
```

```
              [,1]          [,2]          [,3]          [,4]
[1,]  2.7427798252  2.7427798252  2.7427798252  2.7427798252
[2,] -0.0098101730 -0.0098101730 -0.0098101730 -0.0098101730
[3,] -0.0073576298 -0.0073576298 -0.0073576298 -0.0073576298
[4,]  0.0024525433  0.0024525433  0.0024525433  0.0024525433
[5,] -0.0008175144 -0.0008175144 -0.0008175144 -0.0008175144
[6,]  0.0089926586  0.0089926586  0.0089926586  0.0089926586
[7,]  0.0065401153  0.0065401153  0.0065401153  0.0065401153
[8,]  0.0016350288  0.0016350288  0.0016350288  0.0016350288
              [,5]          [,6]          [,7]          [,8]
[1,]  2.7427798252  2.7427798252  2.7427798252  2.7427798252
[2,] -0.0098101730 -0.0098101730 -0.0098101730 -0.0098101730
[3,] -0.0073576298 -0.0073576298 -0.0073576298 -0.0073576298
[4,]  0.0024525433  0.0024525433  0.0024525433  0.0024525433
[5,] -0.0008175144 -0.0008175144 -0.0008175144 -0.0008175144
[6,]  0.0089926586  0.0089926586  0.0089926586  0.0089926586
[7,]  0.0065401153  0.0065401153  0.0065401153  0.0065401153
[8,]  0.0016350288  0.0016350288  0.0016350288  0.0016350288
```

```
> fdrs <- getFDR(leafdevInfCompare)
> dim(fdrs)
```

```
[1]    16 52224
```

```
> fdrs[1:8,1:8]
```

```
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
[1,] 0.5816449 0.5816449 0.5816449 0.5816449 0.5816449 0.5816449
[2,] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[3,] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[4,] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[5,] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[6,] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[7,] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[8,] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
          [,7]      [,8]
[1,] 0.5816449 0.5816449
[2,] 1.0000000 1.0000000
[3,] 1.0000000 1.0000000
[4,] 1.0000000 1.0000000
[5,] 1.0000000 1.0000000
[6,] 1.0000000 1.0000000
[7,] 1.0000000 1.0000000
[8,] 1.0000000 1.0000000
```

References

[1] Clement L, De Beuf K, Thas O, Vuylsteke M, Irizarry RA, and Crainiceanu CM (2012). Fast wavelet based functional models for transcriptome analysis with tiling arrays. *Statistical Applications in Genetics and Molecular Biology*, **11**, Iss. 1, Article 4.

[2] Andriankaja M, Dhondt S, De Bodt S, Vanhaeren H, Coppens F, et al. (2012). Exit from proliferation during leaf development in arabidopsis thaliana: A not-so-gradual process. *Developmental Cell*, **22**, 64–78.

[3] De Beuf K, Pipelers, P, Andriankaja M, Thas O., Inze D, Crainiceanu CM, and Clement L (2012). Model-based Analysis of Tiling Array Expression Studies with Flexible Designs (waveTiling). *Technical document*.

[4] Rehrauer H, Aquino C, Gruissem W, Henz S, Hilson P, Laubinger S, Naouar N, Patrignani A, Rombauts S, Shu H, et al. (2010). AGRONOMICS1: a new resource for Arabidopsis transcriptome profiling. *Plant Physiology*, **152**, 487–499.

[5] Carvalho B and Scharpf R. oligoClasses: Classes for high-throughput arrays supported by oligo and crlmm. [R package version 1.18.0]

[6] Falcon S and Carvalho B with contributions by Vince Carey and Matt Settles and Kristof de Beuf. pdInfoBuilder: Platform Design Information Package Builder. [R package version 1.20.0]

[7] Pages H. BSgenome: Infrastructure for Biostrings-based genome data packages. [R package version 1.24.0]

[8] Pages H, Aboyoun P, and Lawrence M. IRanges: Infrastructure for manipulating intervals on sequences. [R package version 1.14.2]

[9] Aboyoun P, Pages H, and Lawrence M. GenomicRanges: Representation and manipulation of genomic intervals. [R package version 1.8.3]

[10] Carlson M. TxDb.Athaliana.BioMart.plantsmart22: Annotation package for TxDb object(s). [R package version 2.7.1]

[11] Durinck S and Bullard J. GenomeGraphs: Plotting genomic information from Ensembl. [R package version 1.16.0]