mbts

SW

1/31/2018

Generating Simulation Data

To generate simulated data:

```r
set.seed(1)

tbl <- gen_table()
```

The parameters in this function and their default values are:

- fl_sig=0: floor of the arima signal normalization
- w_sig=6: roof of the arima signal normalization
- fl_bg=-6: floor of the background normalization
- w_bg=6: roof of the background normalization
- bg_disp_mu=0: background noise poisson distribution mean
- bg_disp_sigma=1: background noise poisson distribution sd
- sig_disp_mu1=0: arima shared noise poisson distribution mean
- sig_disp_sigma1=0: arima shared noise poisson distribution sd
- sig_disp_mu2=0: signal taxa specific noise poission distribution mean
- sig_disp_sigma2=1: signal taxa specific noise poission distribution sd
- n_sig=10: number of arima signals
- n_clust=10: number of taxa in an arima signal
- n_tax_sig=1: number of taxa with FINAL signal (in beta atm)
- n_bg=700: number of background taxa (this + signal columns in output data)
- len_arima=1000: length of arima signal, needs to be larger than window
- len_ts=500: length of the time series (row size of output table)
- len_signal=300) # length of the shared signal

The output of gen_table() is the final simulated abundance table:

```r
tbl[1:5,1:5]
```

```
##      cl1_sig1 cl1_sig2 cl1_sig3 cl1_sig4 cl1_sig5
## [1,]       18       24        5       23       21
## [2,]       16       26        3       20       25
## [3,]       20       23        7       23       20
## [4,]       28       26        6       43       20
## [5,]       15       25        2       32       35
```

The column names reflect whether the features are signal or background (the noise):

```r
head(colnames(tbl))
```

```
## [1] "cl1_sig1" "cl1_sig2" "cl1_sig3" "cl1_sig4" "cl1_sig5" "cl1_sig6"
```

```r
tail(colnames(tbl))
```

```
## [1] "bg695" "bg696" "bg697" "bg698" "bg699" "bg700"
```

The table belongs to the class mbts:

```
class(tbl)
```

```
## [1] "mbts"
```

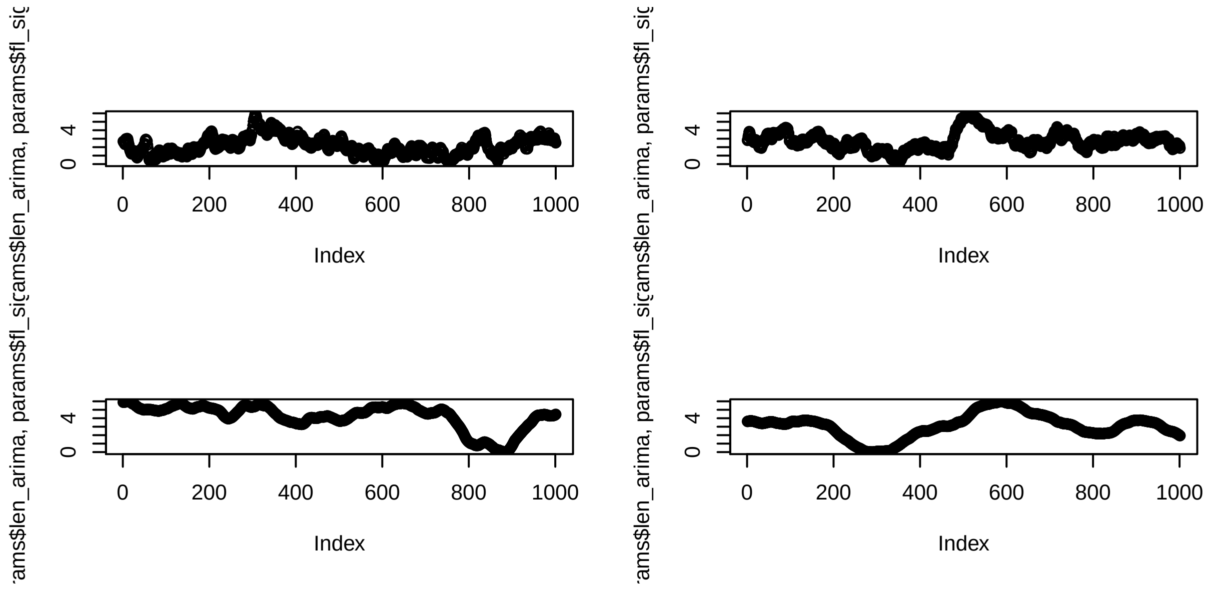
and it has a set of methods:

```
methods(class='mbts')
```

```
## [1] plot_sig  plot_sim  quantiles sig_cor   sparsity
## see '?methods' for accessing help and source code
```

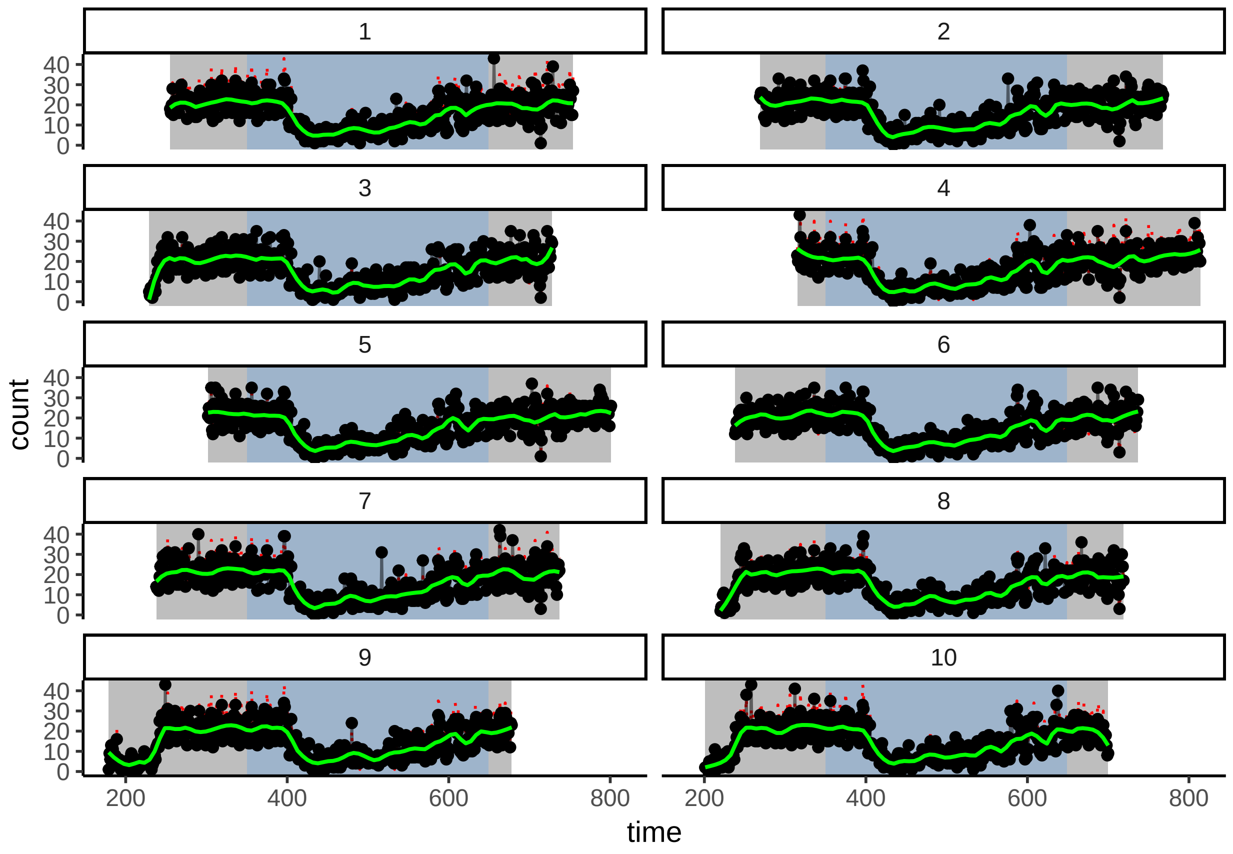
There are two plotting methods. plot_sig() plots **examples** of the arima signal. This isn't the same signal used to generate the simulated data; instead, it's a set of similar signals obtained using the same arima parameters.

```
plot_sig(tbl,n=4)
```



plot_sim plots the simulated data, where the blue shading shows the aligned, shared signal across all taxa (clusters):

```
plot_sim(tbl)
```



The class also has a set of attributes with some of the intermediate data:

```
names(attributes(tbl))
```

```
## [1] "dim"        "dimnames"   "signals"    "background" "params"
## [6] "class"
```

signals stores the underlying, untouched arima signals. For this example, there were 10 taxa (clusters), so this the signals slot will be a list of length 10:

```
signals <- attr(tbl,'signals')
length(signals)
```

```
## [1] 10
```

If we look at the first taxon, we can see that each slot in this list contains:

- the time series signal before noise (the pure signal)
- the time series signal after noise
- the final time series after timeshifting and adding the secound round of noise
- the indexes used to timeshift the signal

```
taxa_1 <- signals[[1]]
names(taxa_1)
```

```
## [1] "timeseries_pure" "timeseries"      "cluster"         "timesteps"
```

```
head(taxa_1$timeseries_pure)
```

```
## [1] 4.145562 3.890171 3.711378 3.912139 4.095823 4.003456
```

```
head(taxa_1$timeseries)
```

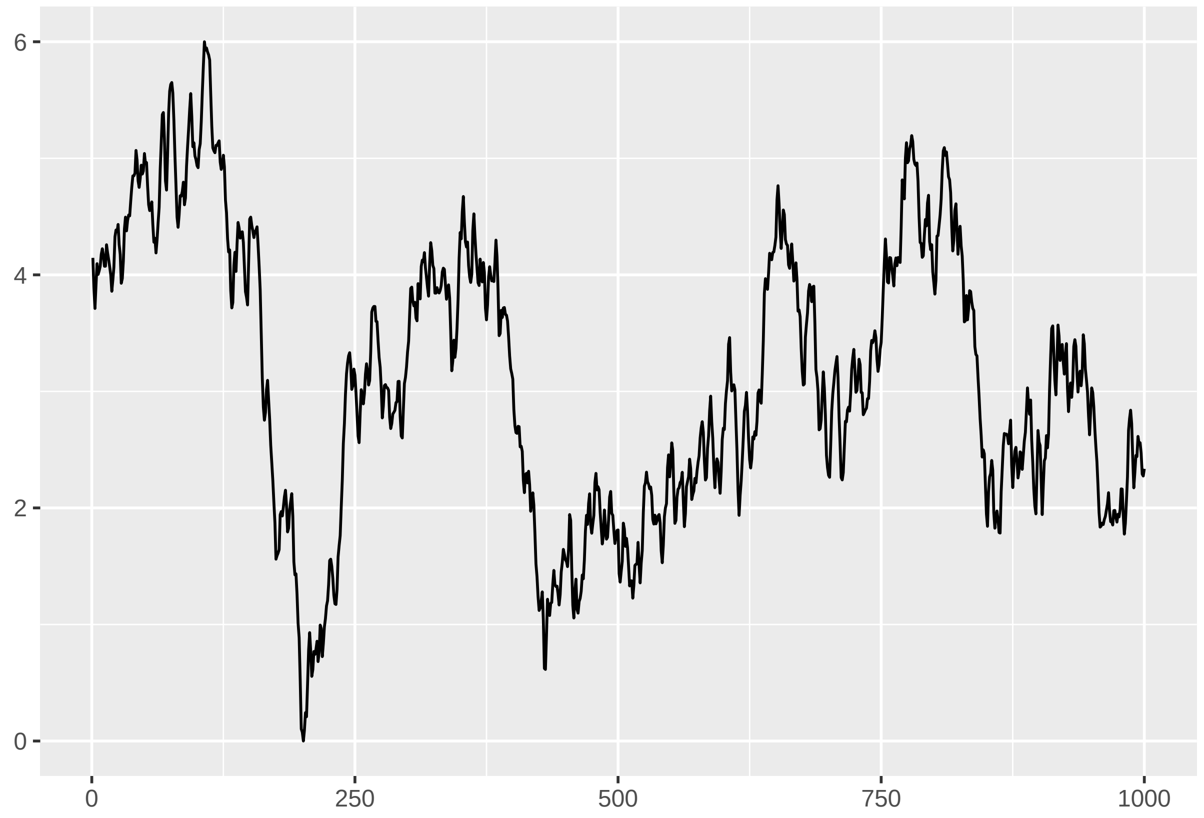
```
## [1] 23 25 18 23 23 10
```

```
taxa_1$cluster[1:5,1:5]
```

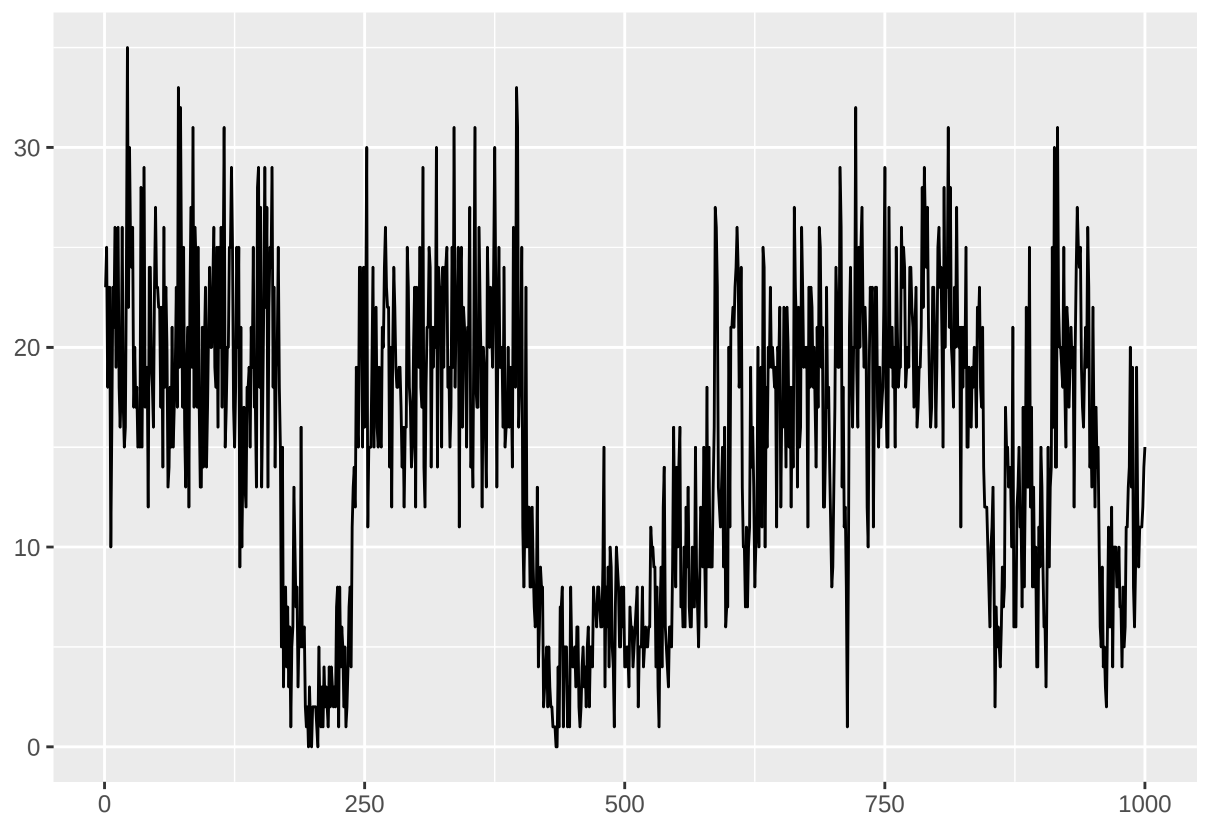
```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   18   24    5   23   21
## [2,]   16   26    3   20   25
## [3,]   20   23    7   23   20
## [4,]   28   26    6   43   20
## [5,]   15   25    2   32   35
```

And we can plot these:

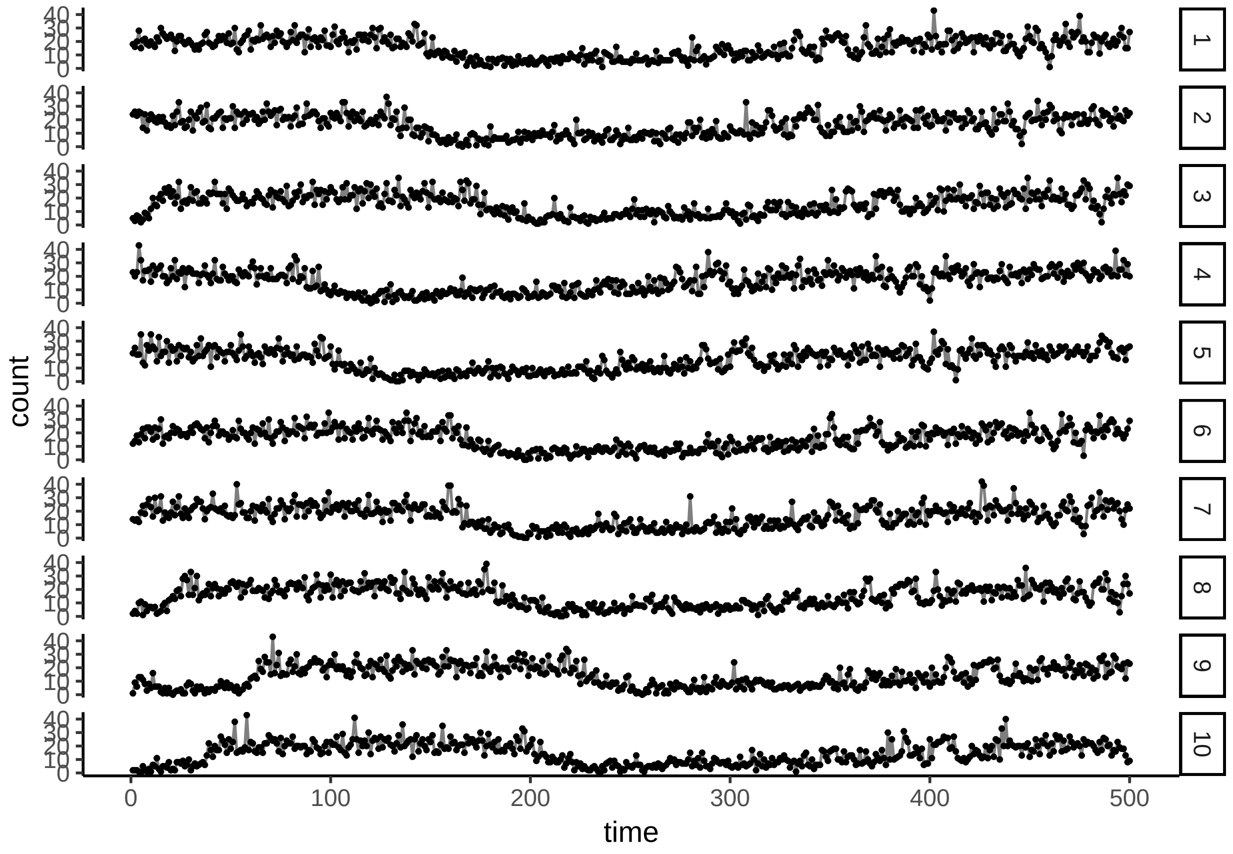
```
qplot(seq_along(taxa_1$timeseries_pure),
      taxa_1$timeseries_pure,
      geom='line',xlab='',ylab='')
```



```
qplot(seq_along(taxa_1$timeseries),
      taxa_1$timeseries,
      geom='line',xlab='',ylab='')
```



```r
data.frame(time=1:nrow(taxa_1$cluster),taxa_1$cluster) %>%
  gather(taxon,count,-time) %>%
  mutate(taxon=as.integer(gsub('X','',taxon))) %>%
  ggplot(aes(time,count)) +
  geom_point(size=.5) +
  geom_line(alpha=.5) +
  facet_grid(taxon~.) +
  theme_classic()
```



If we needed to access the background distribution, we can do the following:

```
background <- attr(tbl,'background')
background[1:5,1:5]
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    1   11    2    0
## [2,]    5    2    2    0    1
## [3,]    1    1    1    3   13
## [4,]    2    0    0    0    0
## [5,]    1    3    6    0    0
```

And if we needed to access the parameters we originally passed to generate the simulation:

```
attr(tbl,'params')
```

```
## $fl_sig
## [1] 0
##
## $w_sig
## [1] 6
##
## $fl_bg
## [1] -6
##
## $w_bg
## [1] 6
##
```

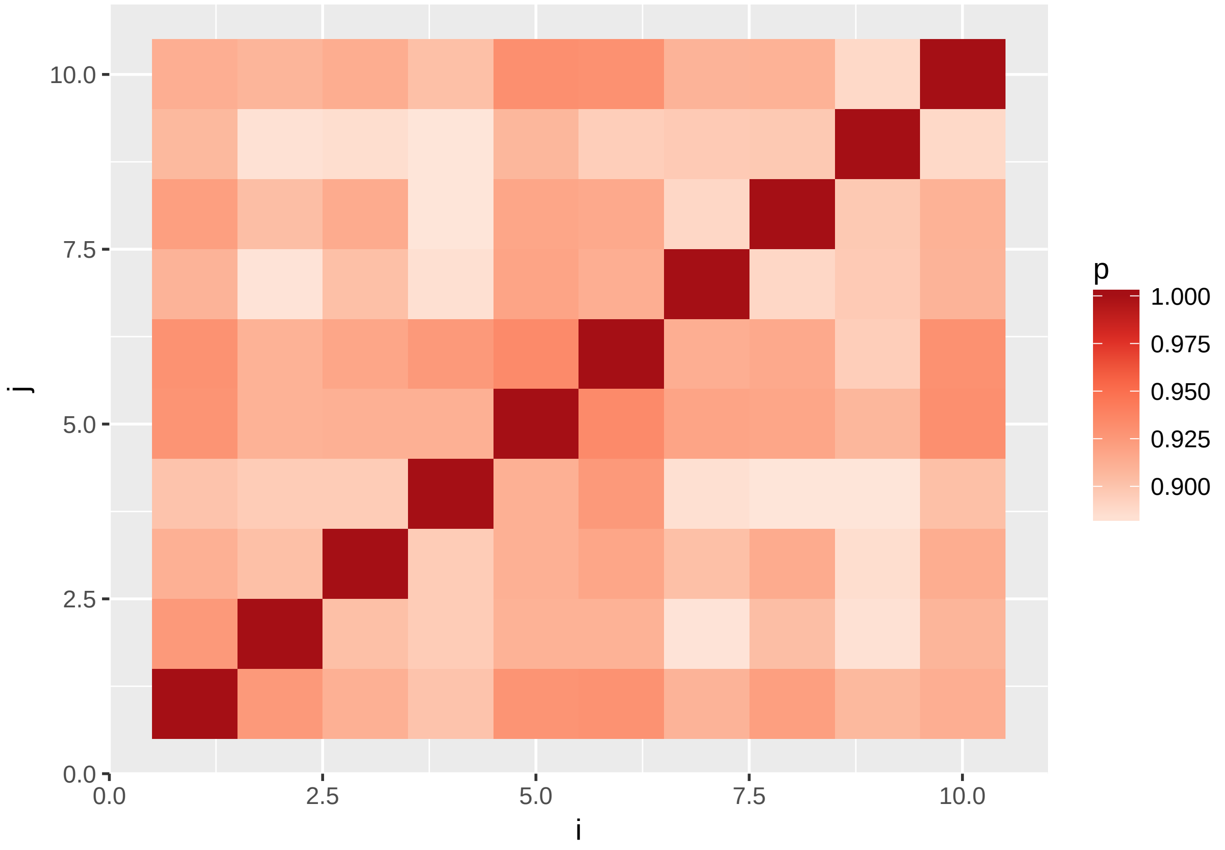
```
## $bg_disp_mu
## [1] 0
##
## $bg_disp_sigma
## [1] 1
##
## $sig_disp_mu1
## [1] 0
##
## $sig_disp_sigma1
## [1] 0
##
## $sig_disp_mu2
## [1] 0
##
## $sig_disp_sigma2
## [1] 1
##
## $n_clust
## [1] 10
##
## $n_sig
## [1] 10
##
## $n_tax_sig
## [1] 1
##
## $n_bg
## [1] 700
##
## $len_arima
## [1] 1000
##
## $len_ts
## [1] 500
##
## $len_signal
## [1] 300
```

The last set of methods help summarize the simulated signals. sig_cor() generates the spearman correlation matrix for each taxon for a given signal i:

```
rho <- sig_cor(tbl,i=1)
rho[1:5,1:5]
```

```
##       [,1]  [,2]  [,3]  [,4]  [,5]
## [1,] 1.000 0.925 0.912 0.901 0.928
## [2,] 0.925 1.000 0.903 0.896 0.911
## [3,] 0.912 0.903 1.000 0.896 0.912
## [4,] 0.901 0.896 0.896 1.000 0.912
## [5,] 0.928 0.911 0.912 0.912 1.000
```

```
data.frame(i=1:nrow(rho),rho) %>%
  gather(j,p,-i) %>%
  mutate(j=as.integer(gsub('X','',j))) %>%
  ggplot(aes(x=i,y=j,fill=p)) +
  geom_raster() +
  scale_fill_distiller(palette='Reds',direction=1)
```



We can also look at the quantiles of the signal and background:

```
quantiles(tbl)
```

```
##             signal background
## 75%             22          4
## 76.78571%       22          4
## 78.57143%       23          4
## 80.35714%       23          5
## 82.14286%       24          5
## 83.92857%       24          6
## 85.71429%       25          7
## 87.5%           25          8
## 89.28571%       26          9
## 91.07143%       26         10
## 92.85714%       27         13
## 94.64286%       28         16
## 96.42857%       29         21
## 98.21429%       32         34
## 100%            57       1158
```

And finally, we can look at how sparse each signal is:

```
sparsity(tbl)
```

```
##     signal background
##  0.0082600  0.3736771
```