Tutorial of InstaPrism

Mengying Hu

2023-02-08

Table of Contents

Introduction

InstaPrism is R package to deconvolute cellular proportion and gene expression in bulk RNA-Seq data. It is based on a Bayesian inference model from BayesPrism. **InstaPrism** provides fast implementation of the deconvolution module from BayesPrism: it replaces the time-consuming gibbs sampling steps in BayesPrism with a fixed-point algorithm that greatly accelerated the calculation speed while maintaining similar performance as BayesPrism.

In this tutorial, we provide three examples of running **InstaPrism** and compare the results with BayesPrism.

Getting started

load the **InstaPrism** package.
We recommend the readers to read through the tutorial of BayesPrism before running our examples.

```
library(InstaPrism)
#> Loading required package: dplyr
#>
#> Attaching package: 'dplyr'
#> The following objects are masked from 'package:stats':
#>
#>     filter, lag
#> The following objects are masked from 'package:base':
#>
#>     intersect, setdiff, setequal, union
#> Loading required package: tidyr
#> Loading required package: Matrix
#>
#> Attaching package: 'Matrix'
#> The following objects are masked from 'package:tidyr':
#>
#>     expand, pack, unpack
#> Loading required package: caret
#> Loading required package: ggplot2
#> Loading required package: lattice
#> Loading required package: NMF
#> Loading required package: pkgmaker
#> Loading required package: registry
#> Loading required package: rngtools
#> Loading required package: cluster
#> NMF - BioConductor layer [OK] | Shared memory capabilities [NO: windows] | Cores 15/16
```

Example 1: deconvolution on a small simulated dataset

Step 1. Create simulated single cell and bulk expression data

The following lines simulate bulk and single-cell expression, as well as marker genes and true proportions that can be used as an example of deconvolution using the simulation model the **BisqueRNA** package.

```
library(Biobase)
library(BisqueRNA)
cell.types <- c("Neurons", "Astrocytes", "Oligodendrocytes", "Microglia", "Endothelial Cells","others")
avg.props <- c(.5, .2, .15, .07, .03,.05)
sim.data <- SimulateData(n.ind=20, n.genes=1000, n.cells=500, cell.types=cell.types, avg.props=avg.props)
```

sim.data contains the following simulated objects:

- A 1000 × 10000 single-cell expression object for 20 individuals, with cells annotated with inidivual ID and cell type labels.
- A 1000 × 20 bulk expression object for 20 individuals.
- A 6 × 20 matrix indicating cellular proportions for 20 individuals.
- A dataframe indicating marker genes for each cell type

Alternatively, readers can load the example simulated data directly from the **InstaPrism** package.

```
data("sim.data")
```

Step 2. Prepare input for InstaPrism and BayesPrism

InstaPrism takes the same input format as BayesPrism:

- a single-cell expression data as prior information
- a bulk expression to run deconvolution
- a character vector indicating cell types of each cell from the scRNA data
- a character vector indicating cell states of each cell from the scRNA data

In real practice, cell.state.labels usually denote different cell states from a same given cell type. For example, malignant cells can be subclustered by different patients to denote different malignant states. With the *sim.data* we created above, we will generate some artificial cell state labels for *Neurons* given individual IDs.

```
library(Biobase)
sc.eset <- sim.data$sc.eset
bulk.eset <- sim.data$bulk.eset

sc_Expr = exprs(sc.eset)
sc_Expr = apply(sc_Expr,2,function(x)((x/sum(x))*1e+05))

bulk_Expr = exprs(bulk.eset)
bulk_Expr = apply(bulk_Expr,2,function(x)((x/sum(x))*1e+06))

cell_type_labels=sc.eset@phenoData@data[["cellType"]]

# create artifical cell-state labels for Neuron cells: Neuron_A, Neuron_B, Neuron_C
Neurons_states=names(table(sc.eset@phenoData@data$SubjectName))
names(Neurons_states)=c(rep("A",8),rep("B",6),rep("C",6))

cell_state_labels=ifelse(sc.eset@phenoData@data[["cellType"]]=='Neurons',paste0('Neurons_sub',names(Neurons_state
s)[match(sc.eset@phenoData@data$SubjectName,Neurons_states)]),sc.eset@phenoData@data[["cellType"]])

table(cell_state_labels)
#> cell_state_labels
#>        Astrocytes Endothelial Cells         Microglia      Neurons_subA
#>              1958               306               727              2014
#>      Neurons_subB      Neurons_subC  Oligodendrocytes            others
#>              1497              1477              1445               576
```

Step 3. Run deconvolution with InstaPrism and BayesPrism

We first consider a simple implementation where we use prior information from the scRNA Seq directly, without the need to update reference.

i. Run InstaPrism with raw input. In this mode, users don't need to have BayesPrism installed.

```
start.time = Sys.time()
InstaPrism.res = InstaPrism(input_type = 'raw',sc_Expr = sc_Expr,bulk_Expr = bulk_Expr,
                   cell.type.labels = cell_type_labels, cell.state.labels = cell_state_labels,
                   update=F,key='Neurons',return.Z = T)
end.time=Sys.time()
InstaPrism_mode1_running_time = difftime(end.time, start.time,units = 'secs') %>% as.numeric()
```

ii. Run InstaPrism with a Prism Object as input.

In this mode, users need first construct a prism object using the BayesPrism package.

```
library(BayesPrism)
#> Warning: replacing previous import 'gplots::lowess' by 'stats::lowess' when
#> loading 'BayesPrism'
#> Warning: replacing previous import 'BiocParallel::register' by 'NMF::register'
#> when loading 'BayesPrism'
start.time = Sys.time()
bp.obj = new.prism(reference = t(sc_Expr),input.type = 'count.matrix',
                   cell.type.labels = cell_type_labels, cell.state.labels = cell_state_labels,
                   key = 'Neurons',mixture = t(bulk_Expr))
InstaPrism.res2 = InstaPrism(input_type = 'prism',prismObj = bp.obj,update=F,return.Z = T)
end.time=Sys.time()
InstaPrism_mode2_running_time = difftime(end.time, start.time,units = 'secs') %>% as.numeric()
```

With update = F (default setting), InstaPrism function returns the following objects:

- Post.ini.cs: posterior information for cell.states
  - Z: a three dimension array indicating gene expression of each cell.state in different individual (returned when `return.Z=T`, default = F)
  - theta: cell.state fraction estimates for each individual
- Post.ini.ct: posterior information for cell.types
  - Z: a three dimension array indicating gene expression of each cell.types in different individual (returned when `return.Z=T`, default = F)
  - theta: cell.types fraction estimates for each individual

Note that InstaPrism with both input types produce the same results

```
all.equal(InstaPrism.res,InstaPrism.res2)
#> [1] TRUE
```

iii. Run BayesPrism

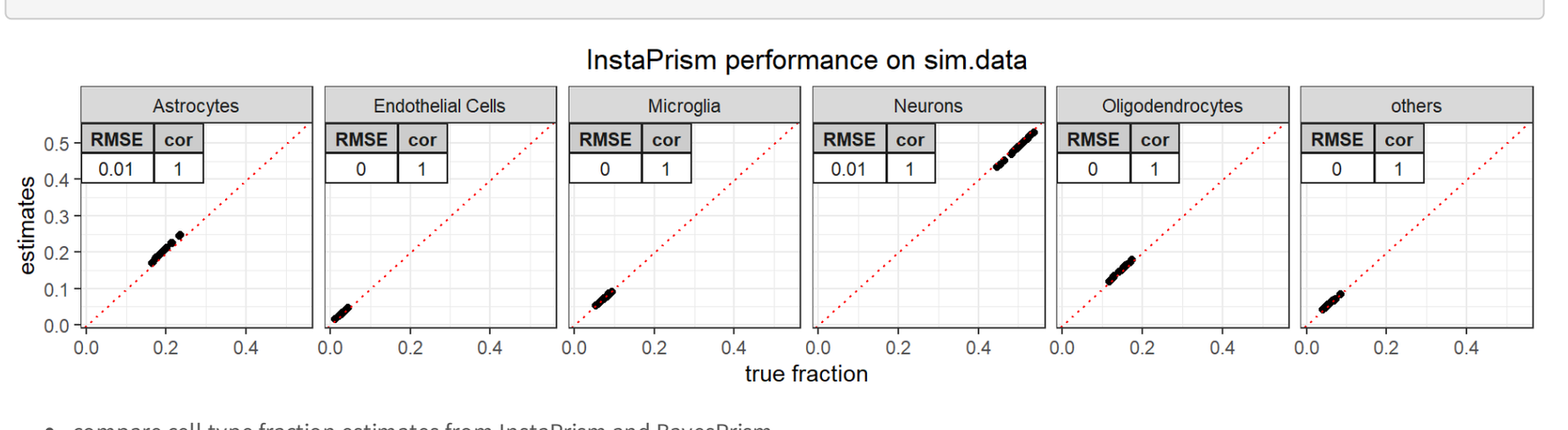
```
start.time = Sys.time()
bp.res = run.prism(bp.obj,update.gibbs = F)
end.time=Sys.time()
bp_running_time = difftime(end.time, start.time,units = 'secs') %>% as.numeric()
# save(bp.res,bp_running_time,file='extdata/tutorial_example1/bp.res.initial.RData')
```

Step 4. Evaluate deconvolution results of InstaPrism and BayesPrism

- deconvolution performance of InstaPrism

We provide a useful function `deconv_performance_plot()` that visualize correlations at per-cell.type level. From the plot, InstaPrism accurately predicts cellular proportions from sim.data

```
deconv_performance_plot(est = InstaPrism.res$Post.ini.ct$theta,true = sim.data$props,title = 'InstaPrism performa
nce on sim.data',nrow=1)
#> Loading required package: ggpubr
#> Loading required package: ggpp
#>
#> Attaching package: 'ggpp'
#> The following object is masked from 'package:ggplot2':
#>
#>     annotate
```

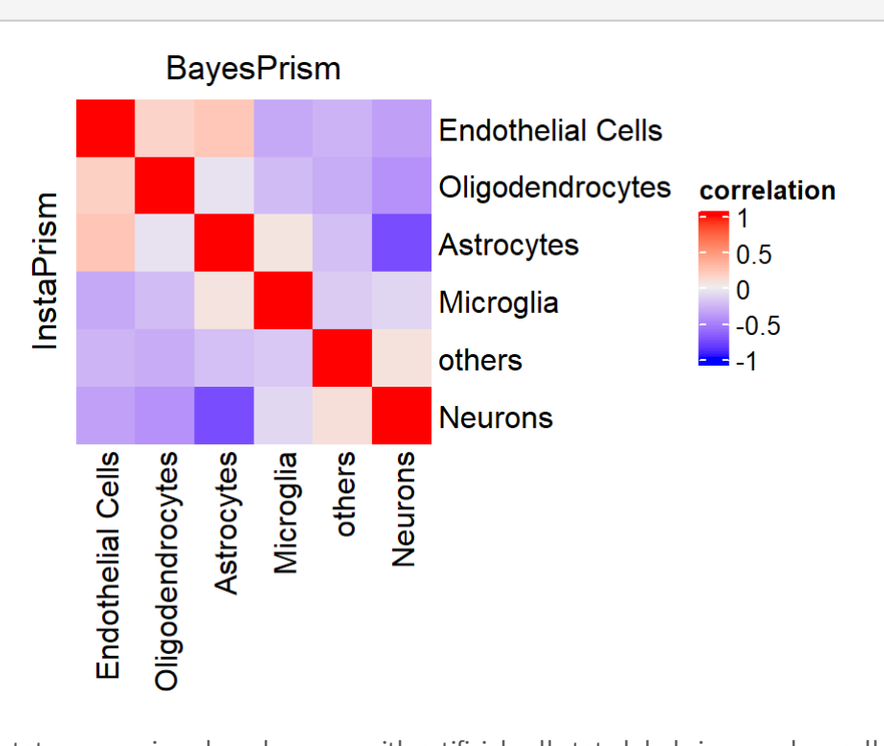


- compare cell type fraction estimates from InstaPrism and BayesPrism

We can visualize the correlation between two fraction estimates with a heatmap. As shown by the heatmap, cell.type estimates from both methods are highly correlated

```
corr=cor(t(InstaPrism.res$Post.ini.ct$theta),bp.res@posterior.initial.cellType@theta)
diag(corr)
#>           Neurons        Astrocytes  Oligodendrocytes         Microglia
#>         0.9999933         0.9999856         0.9999647         0.9998732
#> Endothelial Cells            others
#>         0.9993617         0.9995794
```

```
ComplexHeatmap::Heatmap(corr,show_row_dend = F,show_column_dend = F,column_title = 'BayesPrism',row_title = 'Inst
aPrism',name='correlation')
```

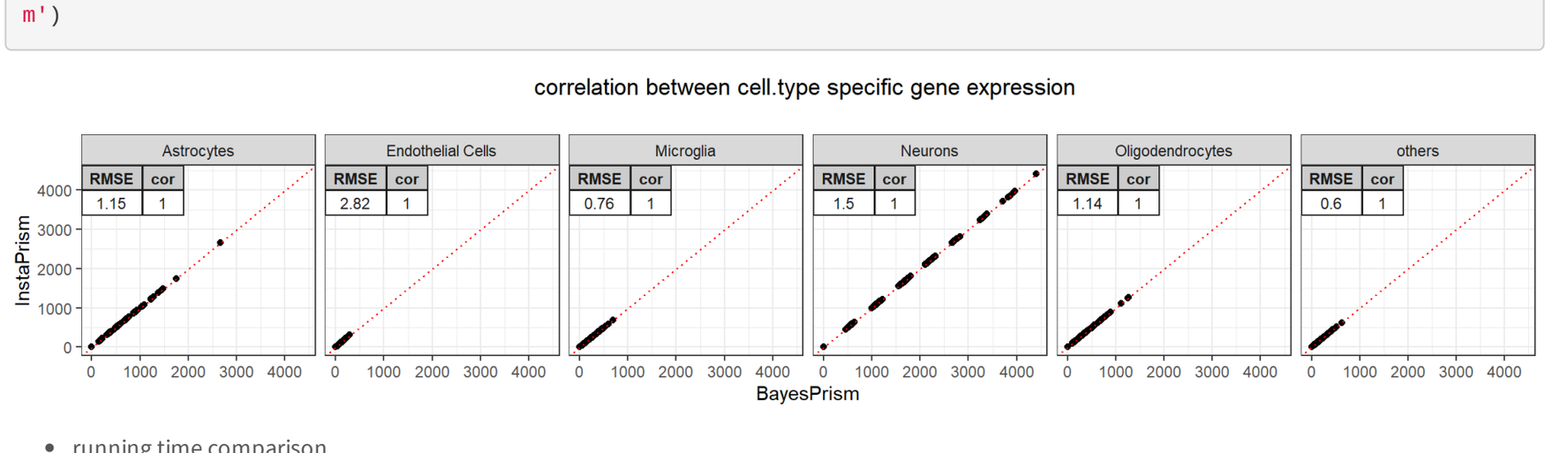


note that we did not include a cell.state comparison here because with artificial cell.state.labels imposed on cells that are not intrinsically distinguishable, it will generate bias on the cell.state estimates

- cell.type specific gene expression comparison

In the following plots, we showed the correlation between cell.type specific gene expression values (Z matrix) between BayesPrism and InstaPrim for one bulk sample. The plots suggest that the Z matrix estimates by both methods is highly correlated.

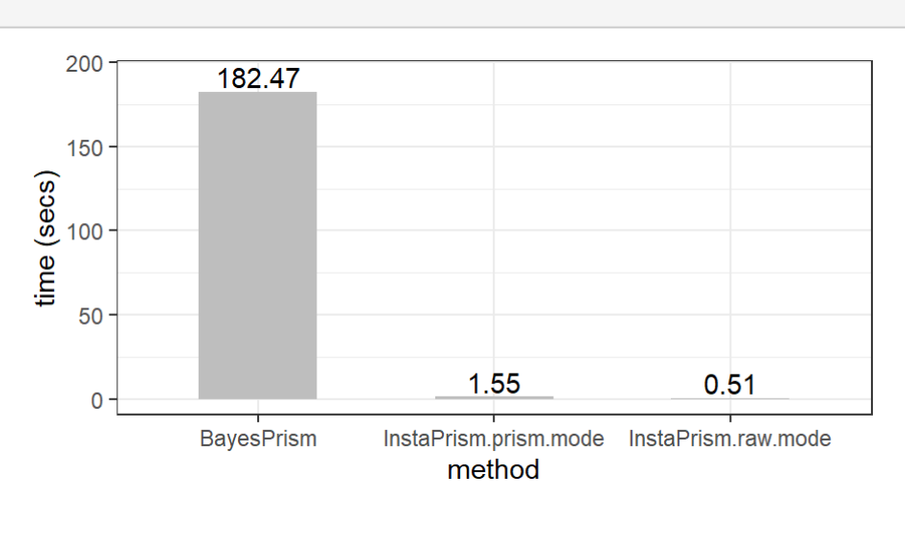
```
deconv_performance_plot(t(InstaPrism.res$Post.ini.ct$Z[1,,]),t(bp.res@posterior.initial.cellType@Z[1,,]),nrow =
1,title = 'correlation between cell.type specific gene expression \n',xlabel = 'BayesPrism',ylabel = 'InstaPris
m')
```



- running time comparison

InstaPrism significantly accelerates the deconvolution speed in either modes.

```
rt=data.frame(method=c('InstaPrism.raw.mode','InstaPrism.prism.mode','BayesPrism'),
              time=c(InstaPrism_mode1_running_time,InstaPrism_mode2_running_time,bp_running_time))
rt$time=round(rt$time,2)
ggplot(rt,aes(method,time))+
  geom_bar(stat="identity",fill='grey',width = 0.5)+
  theme_bw()+
  ylab('time (secs)')+
  geom_text(aes(label = time), vjust = -0.2)+
  ylim(0,max(rt$time)*1.05)
```



Step 5 (optional). Run deconvolution with InstaPrism and BayesPrism using the updated reference

We now consider a more advanced deconvolution problem where we want to leverage from the information shared by bulk samples and update the reference matrix accordingly.

This step is implemented in BayesPrism by setting `update.gibbs = TRUE` in `run.prism()` function, or by calling the `update.theta()` function on the initial deconvolution object. In **InstaPrism**, we concatenated the `update.theta()` module from BayesPrism and replaced the subsequent Gibbs Sampling step with our fixed-point aloigrithm.

Note that to run deconvolution with the updated reference, users need to have BayesPrism installed.

i. deconvolution with InstaPrism

```
start.time = Sys.time()
InstaPrism.res.updated = InstaPrism(input_type = 'raw',sc_Expr = sc_Expr,bulk_Expr = bulk_Expr,
                   cell.type.labels = cell_type_labels, cell.state.labels = cell_state_labels,
                   update=T,key='Neurons')
#> Number of outlier genes filtered from mixture = 0
#> Update the reference matrix ...
#> R Version:  R version 4.2.1 Patched (2022-07-22 r82614 ucrt)
#> snowfall 1.84-6.1 initialized (using snow 0.4-4): parallel execution on 1 CPUs.
#>
#> Stopping cluster
end.time=Sys.time()
InstaPrism_updated_running_time = difftime(end.time, start.time,units = 'secs') %>% as.numeric()
```

With update = T, InstaPrism function returns the following objects:

- Post.ini.cs: posterior information for cell.states
  - Z: a three dimension array indicating gene expression of each cell.state in different individual (returned when `return.Z=T`, default = F)
  - theta: cell.state fraction estimates for each individual
- Post.ini.ct: posterior information for cell.types
  - Z: a three dimension array indicating gene expression of each cell.types in different individual (returned when `return.Z=T`, default = F)
  - theta: cell.types fraction estimates for each individual
- Post.updated.ct: cell.type fraction estimates using updated reference

ii. deconvolution with BayesPrism

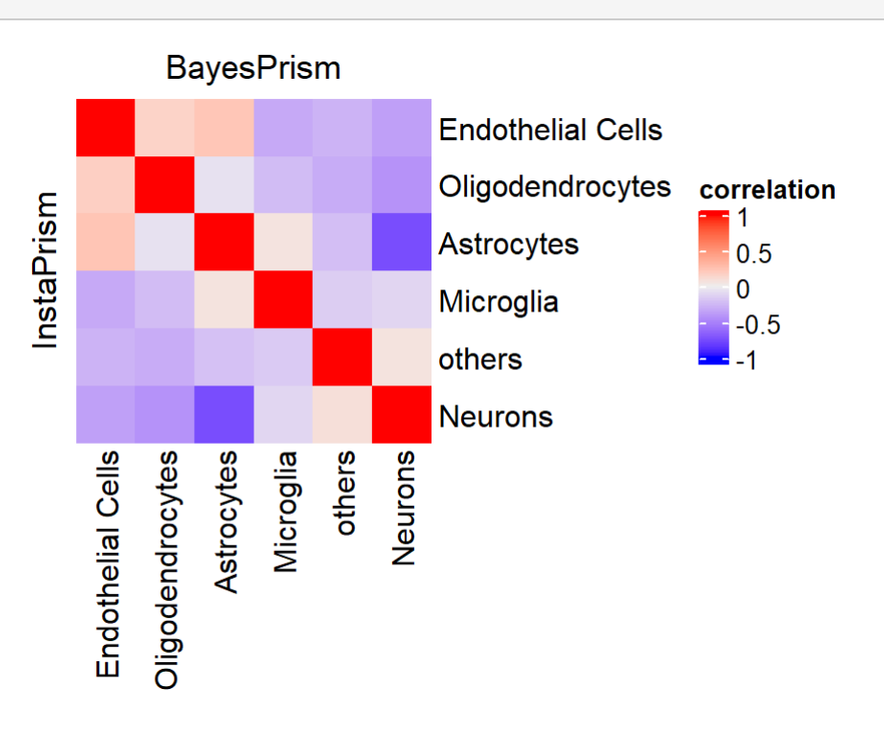
```
start.time = Sys.time()
bp.res.updated = run.prism(bp.obj,update.gibbs = T)
# alternatively, run initial theta estimation and updated theta estimation separately
# bp.res = run.prism(bp.obj,update.gibbs = F)
# bp.res.updated = update.theta(bp=bp.res)
end.time=Sys.time()
bp_updated_running_time = difftime(end.time, start.time,units = 'secs') %>% as.numeric()
save(bp.res.updated,bp_updated_running_time,file = 'extdata/tutorial_example1/bp.res.updated.RData')
```

iii. deconvolution results comparison

cell.type estimation from both methods are highly correlated

```
corr=cor(t(InstaPrism.res.updated$Post.updated.ct),bp.res.updated@posterior.theta_f@theta)
diag(corr)
#>           Neurons        Astrocytes  Oligodendrocytes         Microglia
#>         0.9999940         0.9999889         0.9999602         0.9998283
#> Endothelial Cells            others
#>         0.9990063         0.9994121
```

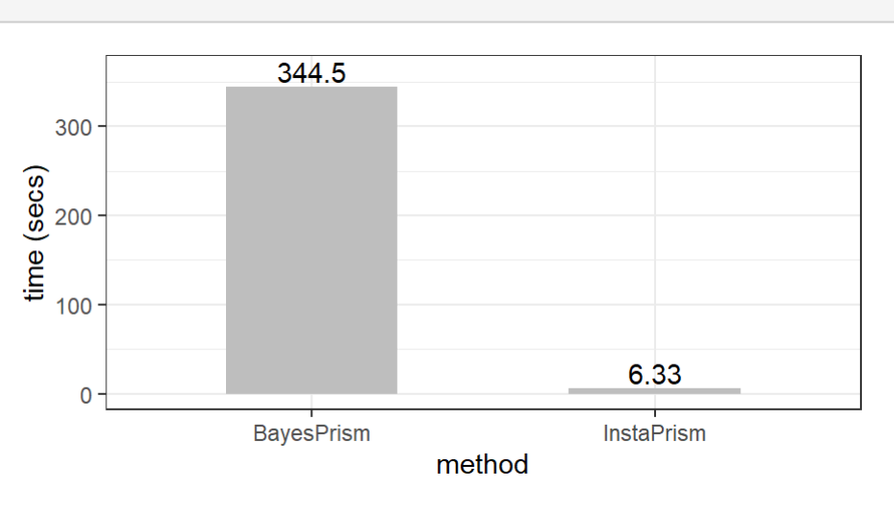
```
ComplexHeatmap::Heatmap(corr,show_row_dend = F,show_column_dend = F,column_title = 'BayesPrism',row_title = 'Inst
aPrism',name='correlation')
```



- running time comparison

InstaPrism significantly accelerates the deconvolution speed using the updated reference.

```
rt=data.frame(method=c('InstaPrism','BayesPrism'),
              time=c(InstaPrism_updated_running_time,bp_updated_running_time))
rt$time=round(rt$time,2)
ggplot(rt,aes(method,time))+
  geom_bar(stat="identity",fill='grey',width = 0.5)+
  theme_bw()+
  ylab('time (secs)')+
  geom_text(aes(label = time), vjust = -0.2)+
  ylim(0,max(rt$time)*1.05)
```



Example 2: deconvolution on the tutorial data from BayesPrism

In this example, we will use the tutorial data provided in BayesPrism and compare the deconvolution results between InstaPrism and BayesPrism.

Step 1. Run BayesPrism following the tutorial in BayesPrism.

Note that it takes more than 6 hours to run the following code from our end (using n.core=16), users can skip the BayesPrism running process by loading our processed results directly.

```
library(BayesPrism)
load('extdata/tutorial_example2/tutorial.gbm.rdata')

sc.stat <- plot.scRNA.outlier(
  input=sc.dat, #make sure the colnames are gene symbol or ENSMEBL ID
  cell.type.labels=cell.type.labels,
  species="hs", #currently only human(hs) and mouse(mm) annotations are supported
  return.raw=TRUE #return the data used for plotting.
  #pdf.prefix="gbm.sc.stat" specify pdf.prefix if need to output to pdf
)
bk.stat <- plot.bulk.outlier(
  bulk.input=bk.dat, #make sure the colnames are gene symbol or ENSMEBL ID
  sc.input=sc.dat, #make sure the colnames are gene symbol or ENSMEBL ID
  cell.type.labels=cell.type.labels,
  species="hs", #currently only human(hs) and mouse(mm) annotations are supported
  return.raw=TRUE
  #pdf.prefix="gbm.bk.stat" specify pdf.prefix if need to output to pdf
)

# filter outlier genes from scRNA-seq data
sc.dat.filtered <- cleanup.genes (input=sc.dat,
                                  input.type="count.matrix",
                                  species="hs",
                                  gene.group=c( "Rb","Mrp","other_Rb","chrM","MALAT1","chrX","chrY") ,
                                  exp.cells=5)

# Subset protein coding genes
sc.dat.filtered.pc <-  select.gene.type (sc.dat.filtered,
                                         gene.type = "protein_coding")

# construct a prism object
myPrism <- new.prism(
  reference=sc.dat.filtered.pc,
  mixture=bk.dat,
  input.type="count.matrix",
  cell.type.labels = cell.type.labels,
  cell.state.labels = cell.state.labels,
  key="tumor",
  outlier.cut=0.01,
    outlier.fraction=0.1,
)

# run BayesPrism
start.time = Sys.time()
bp.res <- run.prism(prism = myPrism, n.cores=16,update.gibbs=T) # set update.gibbs=T for full comparison between
two methods
end.time=Sys.time()
bp_running_time = difftime(end.time, start.time,units = 'mins') %>% as.numeric()
save(bp.res,bp_running_time,file = 'extdata/tutorial_example2/bp.res.RData')
```

Alternatively, users can download the processed results directly from zenodo repository (will update later)

```
load('extdata/tutorial_example2/bp.res.RData')
```

Step 2. Run InstaPrism

For full comparison between BayesPrism and InstaPrism, set `update=T` to get cell.type fraction estimates with updated reference.

Note that the following example takes about 15 mins to run from our end. To skip the process, users can download our processed data from zenodo repository (will update later) or accelerate the process by setting `update=F`.

```
library(BayesPrism)
start.time = Sys.time()
InstaPrism.res = InstaPrism(input_type = 'prism',prismObj = bp.res@prism,update=T,n.iter = 100,n.core=16)
# alternatively, users can use myPrism (constructed in the previous example) as a substitute of bp.res@prism; or
run InstaPrism under 'raw' mode by specifying corresponding arguments
end.time=Sys.time()
InstaPrism_running_time = difftime(end.time, start.time,units = 'mins') %>% as.numeric()
save(InstaPrism.res,InstaPrism_running_time,file = 'extdata/tutorial_example2/InstaPrism.res.RData')
```

For comparison of initial fraction estimates only, set `update=F` . (Note that by setting `update=F` , we no longer have the `Post.updated.ct` object.)

```
start.time = Sys.time()
InstaPrism.res2 = InstaPrism(input_type = 'prism',prismObj = bp.res@prism,update=F,n.iter = 100,n.core=16)
end.time=Sys.time()
InstaPrism_running_time2 = difftime(end.time, start.time,units = 'mins') %>% as.numeric()
save(InstaPrism.res,InstaPrism_running_time,InstaPrism_running_time2,file = 'extdata/tutorial_example2/InstaPris
m.res.RData')
```

Alternatively, download our processed results

```
load('extdata/tutorial_example2/InstaPrism.res.RData')
```

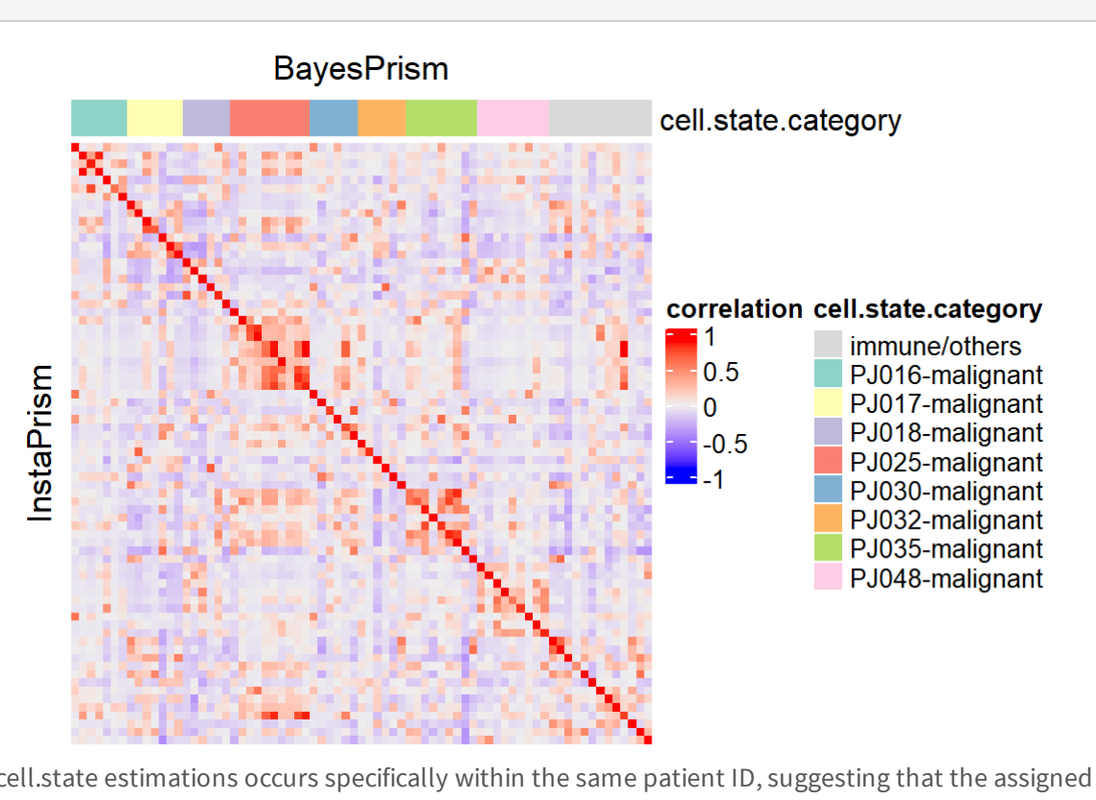
Step 3. Compare deconvolution results

- cell.state fraction comparison

The tutorial data provided in BayesPrism contains 73 different cell.state.labels. Here we provide an example of how to visualize correlation at per cell.state level, ordered by different categories of cell.states.

We can find that across 73 different cell.states, most of the fraction estimates are highly correlated.

```
# organize different cell.states in the tutorial data
PJ.cs=data.frame(cs=rownames(InstaPrism.res$Post.ini.cs$theta))
PJ.cs$group=ifelse(grepl('tumor',PJ.cs$cs),'malignant','immune/others')
PJ.cs$tumor=ifelse(PJ.cs$group=='malignant',sub("\\-.*", "", PJ.cs$cs),'immune/others')
PJ.cs$tumor=ifelse(PJ.cs$group=='malignant',paste0(PJ.cs$tumor,'-malignant'),'immune/others')
PJ.cs=PJ.cs[order(PJ.cs$group,decreasing = T),]

# get correlation between cell.state fraction estimates
cs.corr=cor(t(InstaPrism.res$Post.ini.cs$theta[PJ.cs$cs,]),bp.res@posterior.initial.cellState@theta[,PJ.cs$cs])

# visualize
library(RColorBrewer)
ha=ComplexHeatmap::HeatmapAnnotation(cell.state.category=PJ.cs$tumor,col =list(cell.state.category=setNames(brewe
r.pal(9,'Set3'),unique(PJ.cs$tumor))))
ComplexHeatmap::Heatmap(cs.corr,show_row_dend = F,show_column_dend = F,column_title = 'BayesPrism',row_title = 'I
nstaPrism',name='correlation',cluster_rows = F,cluster_columns = F,show_row_names = F,show_column_names = F,top_a
nnotation = ha)
```

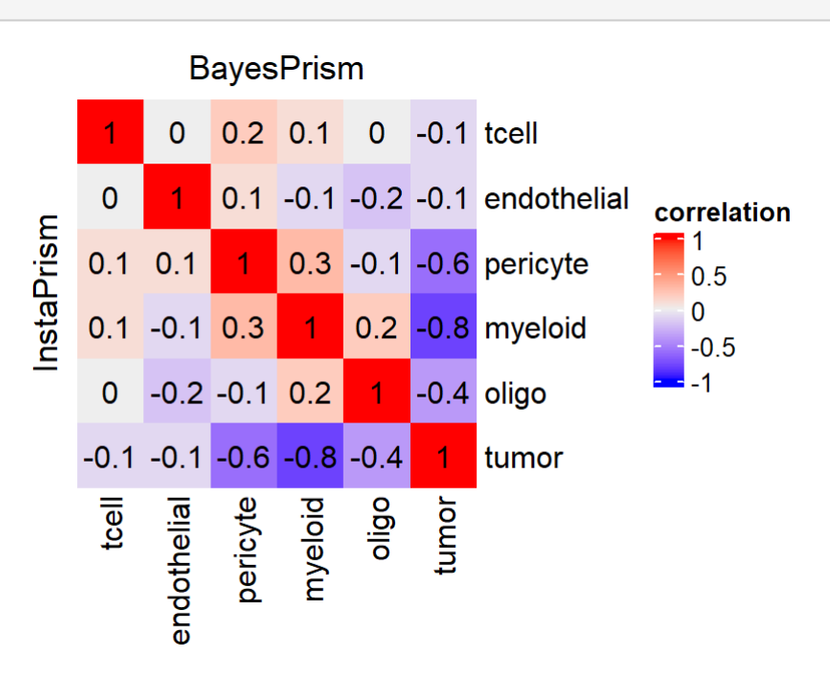


Note that some deviated cell.state estimations occurs specifically within the same patient ID, suggesting that the assigned cell.states for these cells may not be distinguishable enough, making the subsequent cell.state fraction estimation less separable.

- cell.type fraction comparison (with initial reference)

While there's inconsistency between two methods at per cell.state levels, by aggregating cell.state to cell.type levels, two methods provide exactly the same deconvolution results.

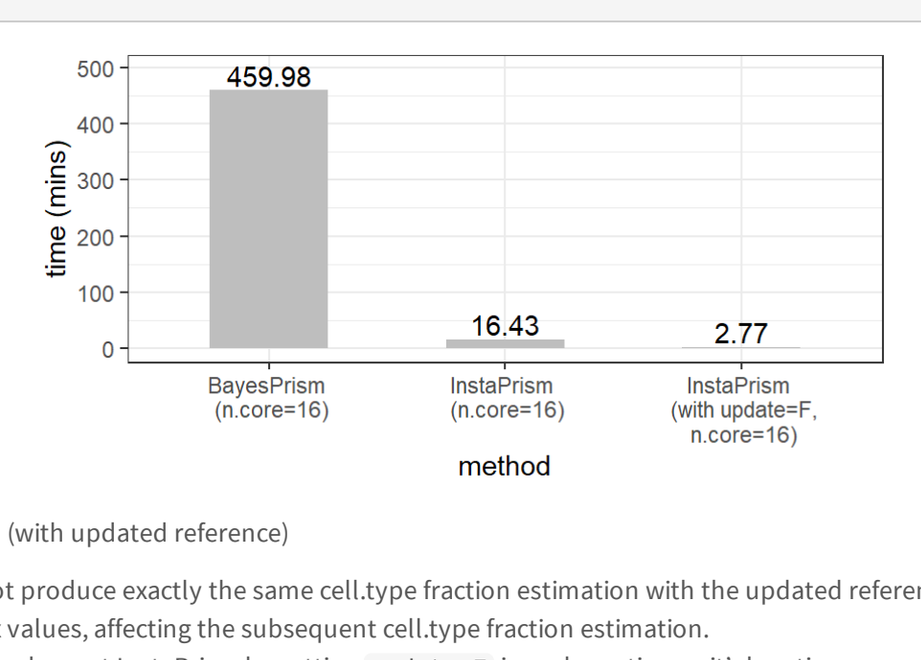
```
ct.corr=cor(t(InstaPrism.res$Post.ini.ct$theta),bp.res@posterior.initial.cellType@theta) %>% round(1)

ComplexHeatmap::Heatmap(ct.corr,show_row_dend = F,show_column_dend = F,column_title = 'BayesPrism',row_title = 'I
nstaPrism',name='correlation', cell_fun = function(j, i, x, y, w, h, col) {grid::grid.text(ct.corr[i, j], x, y)})
```



- running time comparison

For this tutorial data, InstaPrism shortened the running time from hours to only minutes. The process can be even accelerated by setting `udpate=F` in the `InstaPrism()` function.

```
rt=data.frame(method=c('InstaPrism \n (n.core=16)','InstaPrism \n (with update=F,\n n.core=16)','BayesPrism \n
(n.core=16)'),
              time=c(InstaPrism_running_time,InstaPrism_running_time2,bp_running_time))
rt$time=round(rt$time,2)
ggplot(rt,aes(method,time))+
  geom_bar(stat="identity",fill='grey',width = 0.5)+
  theme_bw()+
  ylab('time (mins)')+
  geom_text(aes(label = time), vjust = -0.2)+
  ylim(0,max(rt$time)*1.08)
```



- cell.type fraction comparison (with updated reference)

We note that InstaPrism does not produce exactly the same cell.type fraction estimation with the updated reference, this is because the updated reference is sensitive to the Z matrix values, affecting the subsequent cell.type fraction estimation.
We recommend the readers to implement InstaPrism by setting `update=F` in real practice, as it's less time-consuming while maintaining good performance.

```
ct.corr.updated=cor(t(InstaPrism.res$Post.updated.ct),bp.res@posterior.theta_f@theta)
diag(ct.corr.updated)
#>     tumor    myeloid   pericyte endothelial     tcell      oligo
#> 0.9386490  0.9873336  0.9749192   0.8749537  0.4937763  0.9480686
```

Example 3: deconvolution on heterogeneously simulated bulk data

We have recently proposed a heterogeneous bulk simulation pipeline that simulate bulk samples with realistic biological variance (for more details, check here). In the following example, we will run deconvolution on a heterogeneously simulated bulk dataset of HNSC tumors and evaluate the deconvolution performance by comparing the fraction estimates with real fractions.

Step 1. Load example data

For details about how the bulk samples are simulated, check here.

```
load('extdata/tutorial_example3/example3.RData')
```

The `example3.RData` contains the following objects:

- simulated_bulk: simulated bulk expression of HNSC tumors to run deconvolution
- simulated_frac: simulated cell type fractions for the bulk samples as ground truth
- scExpr_train: sc-RNA data as prior information
- cell.type.labels: a character vector indicating cell types of each cell from the scRNA data
- cell.state.labels: a character vector indicating cell states of each cell from the scRNA data

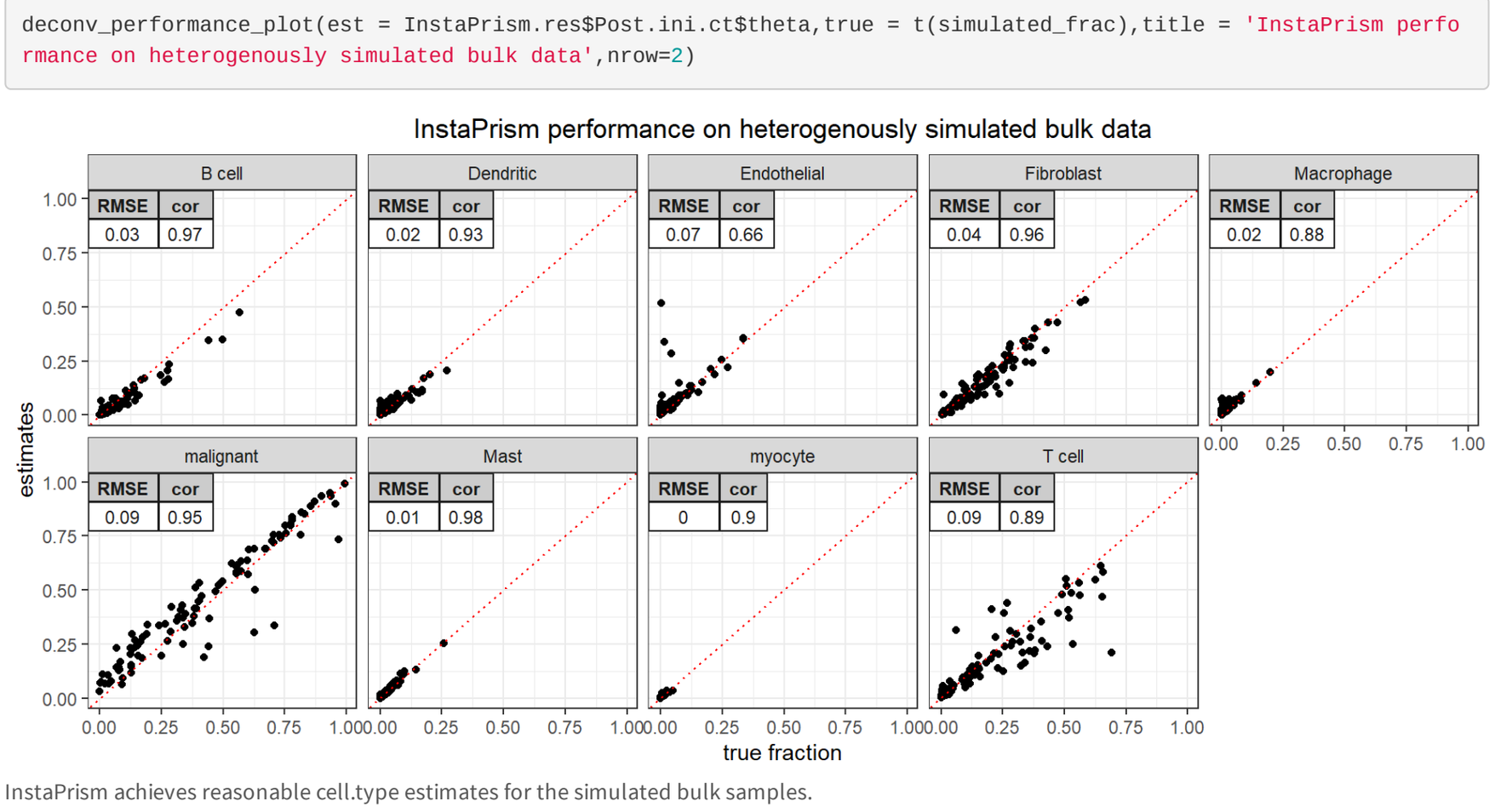
where we assign the malignant cells with different cell.states by their patient IDs.

Step 2. Run InstaPrism

```
InstaPrism.res = InstaPrism(input_type = 'raw',sc_Expr = scExpr_train,bulk_Expr = simulated_bulk,
                   cell.type.labels = cell.type.labels, cell.state.labels = cell.state.labels,
                   key='malignant',n.core = 16)
#> Number of outlier genes filtered from mixture = 19
```

Step 3. Compare InstaPrism results with real cell.type fractions

```
deconv_performance_plot(est = InstaPrism.res$Post.ini.ct$theta,true = t(simulated_frac),title = 'InstaPrism perfo
rmance on heterogeneously simulated bulk data',nrow=2)
```



InstaPrism achieves reasonable cell.type estimates for the simulated bulk samples.

Step 4 (optional). Compare with BayesPrism result

We provide an example below showing how to run BayesPrism using the heterogeneously simulated bulk data.

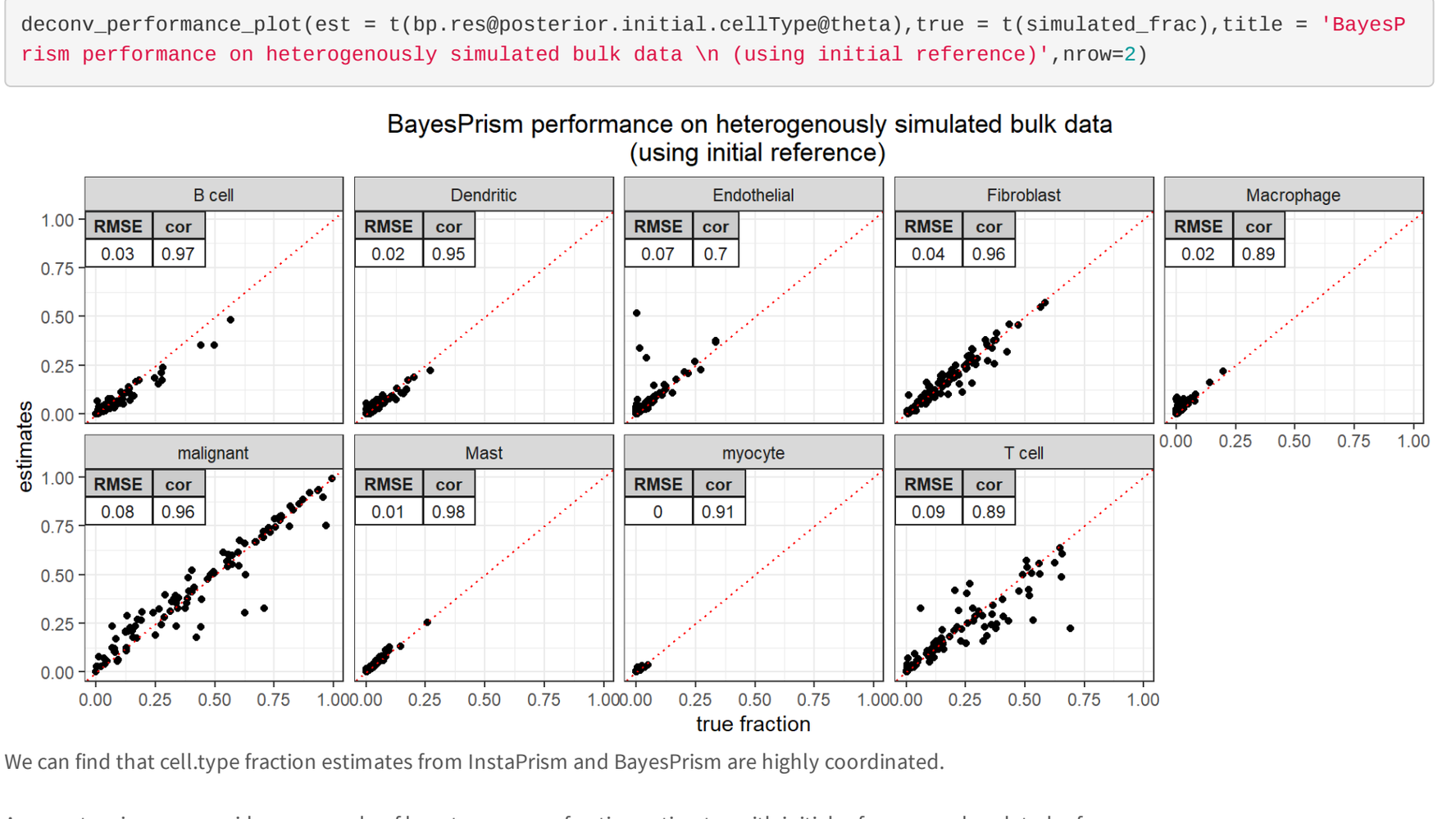
```
myPrism <- new.prism(
  reference=t(scExpr_train),
  mixture=t(simulated_bulk),
  input.type="count.matrix",
  cell.type.labels = cell.type.labels,
  cell.state.labels = cell.state.labels,
  key="malignant",
  outlier.cut=0.01,
  outlier.fraction=0.1,
)
bp.res <- run.prism(prism = myPrism, n.cores=16)
save(bp.res,file = 'extdata/tutorial_example3/bp.res.RData')
```

Alternatively, users can load our processed results directly.

```
load('extdata/tutorial_example3/bp.res.RData')
```

Evaluate BayesPrism performance (with initial reference)

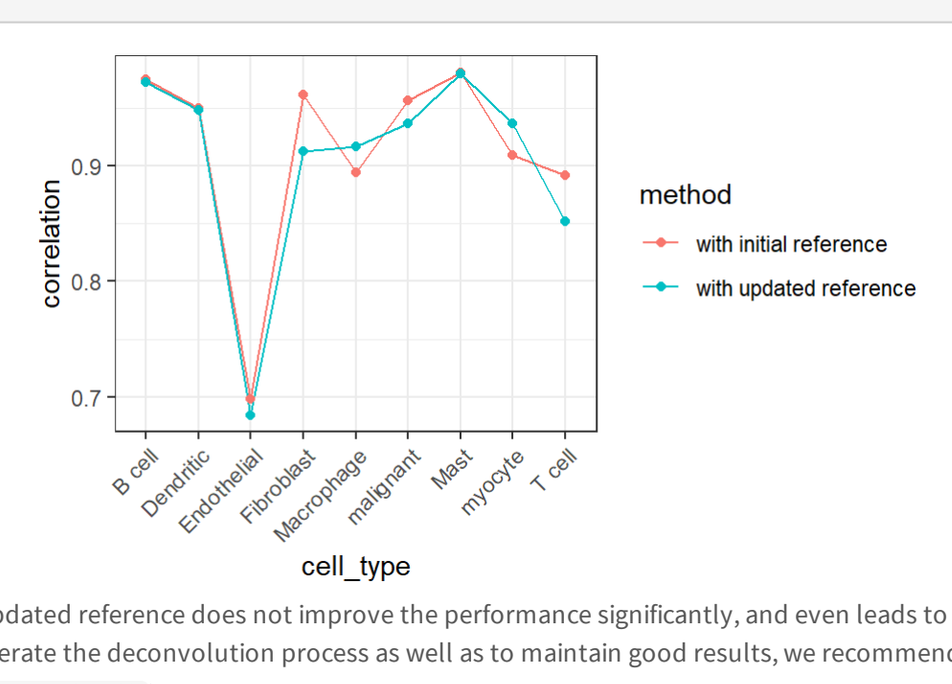
```
deconv_performance_plot(est = t(bp.res@posterior.initial.cellType@theta),true = t(simulated_frac),title = 'BayesP
rism performance on heterogeneously simulated bulk data \n (using initial reference)',nrow=2)
```



We can find that cell.type fraction estimates from InstaPrism and BayesPrism are highly coordinated.

As an extension, we provide an example of how to compare fraction estimates with initial reference and updated reference.

```
library(tibble)
initial.corr=diag(cor(bp.res@posterior.initial.cellType@theta[,colnames(simulated_frac)],simulated_frac)) %>% as.
data.frame() %>% rownames_to_column('cell_type')
updated.corr=diag(cor(bp.res@posterior.theta_f@theta[,colnames(simulated_frac)],simulated_frac)) %>% as.data.fram
e() %>% rownames_to_column('cell_type')

df=rbind(initial.corr,updated.corr)  %>% rename('correlation'='.') %>% mutate(method=c(rep('with initial referenc
e',9),rep('with updated reference',9)))

ggplot(df,aes(x = cell_type, y = correlation, group = method)) +
  geom_line(aes(color=method))+
  geom_point(aes(color=method))+
  theme_bw()+
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
```



As can be found here, using the updated reference does not improve the performance significantly, and even leads to performance drop in some cases. Therefore in real practice, to accelerate the deconvolution process as well as to maintain good results, we recommend the users to set `update=F` (which is the default setting in `InstaPrism()`).