Run a minimal deconvolution simulation

Sean Maden

2022-12-08

This vignette shows how to run minimal deconvolution simulations on some example synthetic data objects.

Run a minimal simulation

This section walks through how to set up, run, and analyze a deconvolution simulation series. The entire code to run the simulation is as follows:

```
num.sim <- 1e3
lgv <- random_lgv(gindexv = c(1,2), num.iter = num.sim)
size1 <- 1
size2 <- 100
lsv <- lapply(seq(num.sim), function(ii){c(size1, size2)})
lpv <- make_lpv()
lres <- decon_analysis(lgv = lgv, lpv = lpv, lsv = lsv)
```

Simulation setup

```
num.sim <- 1e3
```

We have set the number of simulations to 1000. We need to define the core deconvolution objects to run the simulation. These are the following 3 list variables:

- `lgv` : Marker signals. These are used to calculate Z. We can assign these as follows:

```
marker1 <- c(1, 0)
marker2 <- c(0, 1)
lgv <- lapply(seq(num.sim),function(ii){list(marker1, marker2)})
```

- `lsv` : Size factors. These are used to transform Z and also to make the Y pseudo-bulked sample. To assign these values, use:

```
size1 <- 1
size2 <- 100
lsv <- lapply(seq(num.sim), function(ii){c(size1, size2)})
```

- `lpv` : True prediction values. These are used to make the Y pseudo-bulked sample and to compare predictions returned from deconvolution. We can vary complementary proportions for 2 types using:

```
prop1 <- seq(1e-3, 1, 1e-3)
prop2 <- rev(prop1)
lpv <- lapply(seq(num.sim), function(ii){c(prop1[ii], prop2[ii])})
```

Run simulations

Now run simulations and store the results as `lres`.

```
lres <- decon_analysis(lgv, lpv, lsv)
```

```
## Loading required package: ggplot2

## Loading required package: dplyr

##
## Attaching package: 'dplyr'

## The following object is masked from 'package:Biobase':
##
##     combine

## The following objects are masked from 'package:GenomicRanges':
##
##     intersect, setdiff, union

## The following object is masked from 'package:GenomeInfoDb':
##
##     intersect

## The following objects are masked from 'package:IRanges':
##
##     collapse, desc, intersect, setdiff, slice, union

## The following objects are masked from 'package:S4Vectors':
##
##     first, intersect, rename, setdiff, setequal, union

## The following objects are masked from 'package:BiocGenerics':
##
##     combine, intersect, setdiff, union

## The following object is masked from 'package:matrixStats':
##
##     count

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

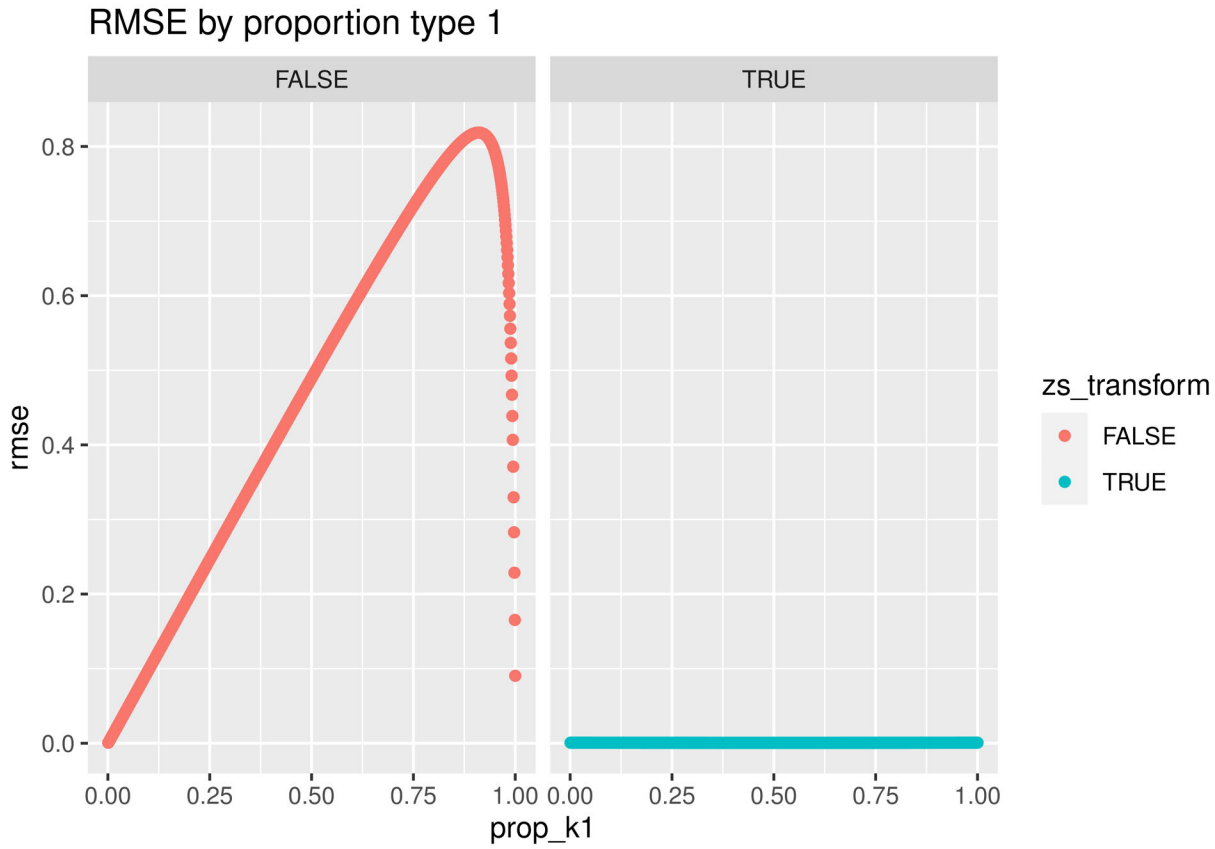
Analyze results

We access the plot objects from `lres`.

```
lgg <- lres$lgg
```

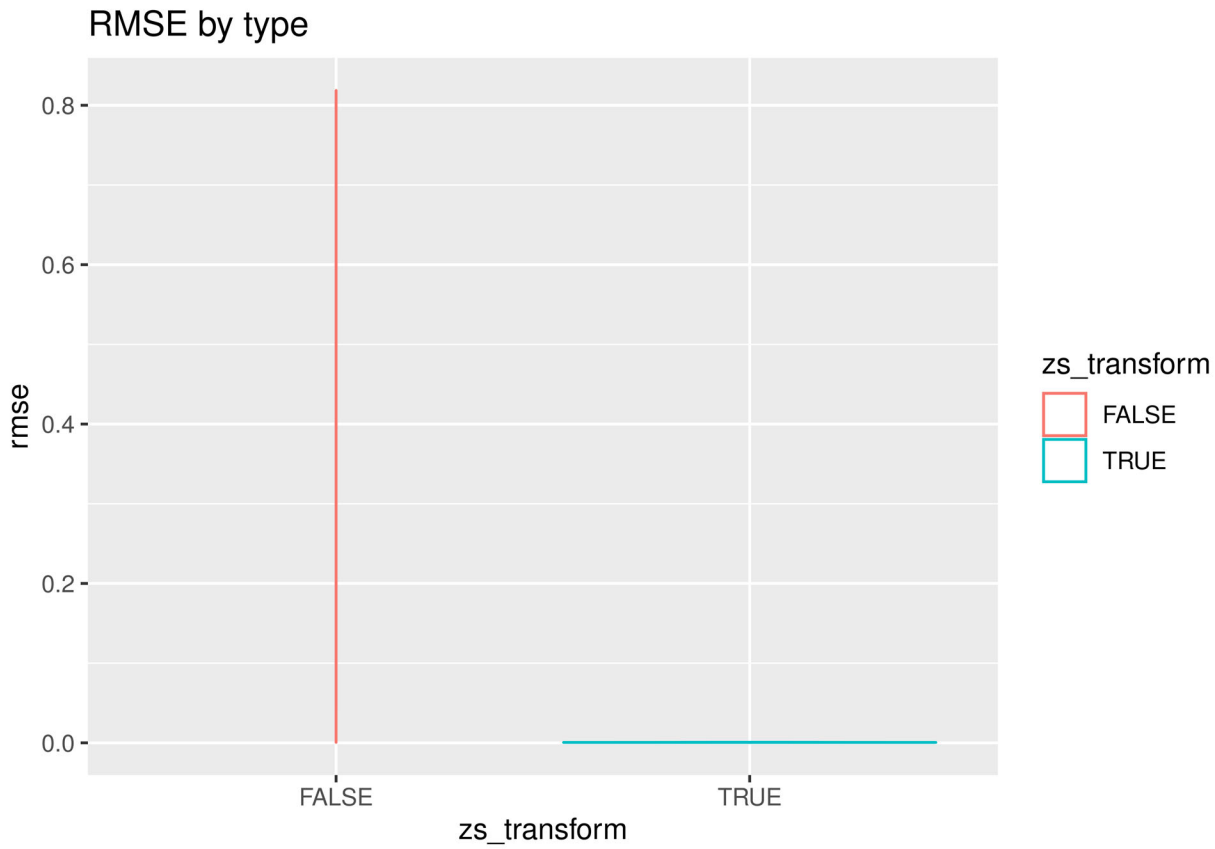
We can view the scatter plots of NNLS-predicted proportions for type 1 (x-axis) by the root mean squared error (RMSE), grouped on whether the S-transform was first applied to the Z reference, using:

```
lgg$ggpt1
```



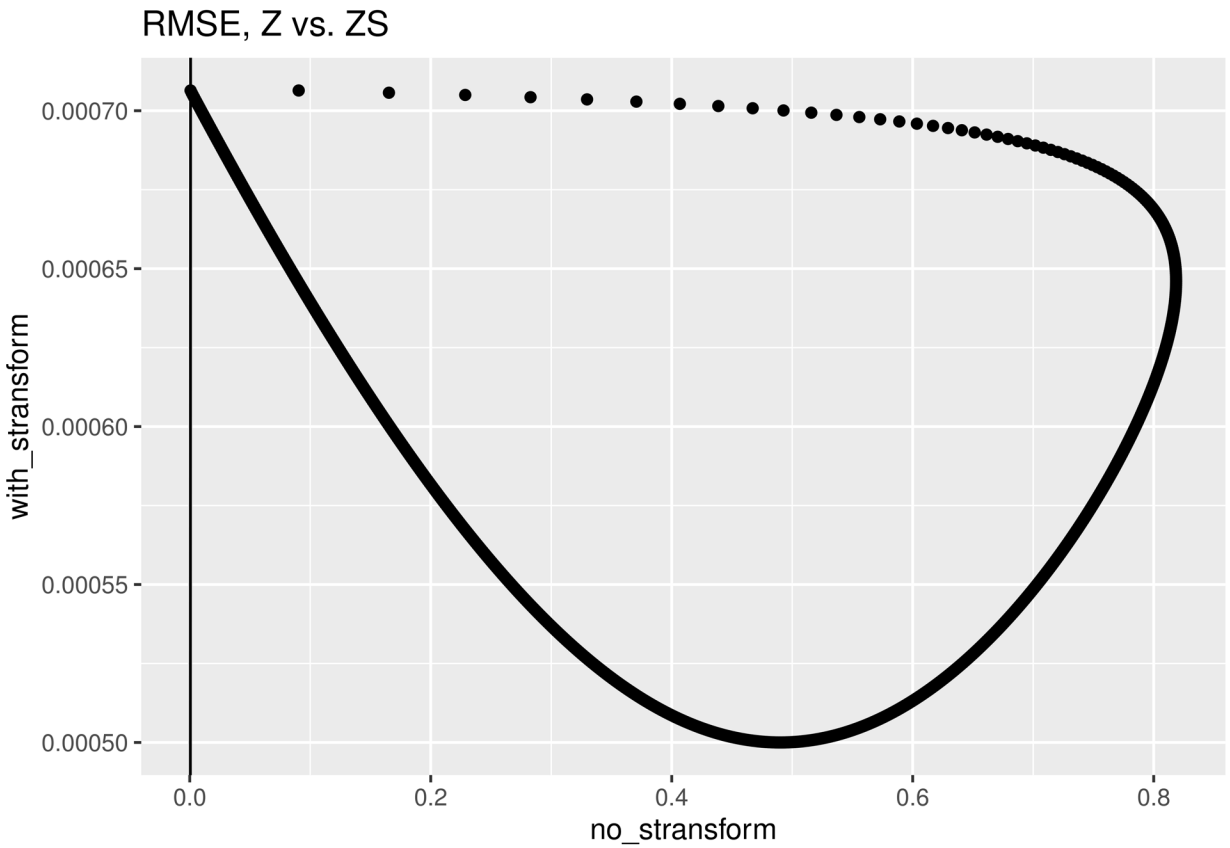
We can view the violin plots of RMSE grouped on S transformation status using:

```
lgg$ggvp
```



We can view the scatterplot of the RMSEs for the non-transformed data (x-axis) versus the S-tranformed data (y-axis), with a red reference line (y-intercept = 0, slope = 1), using:

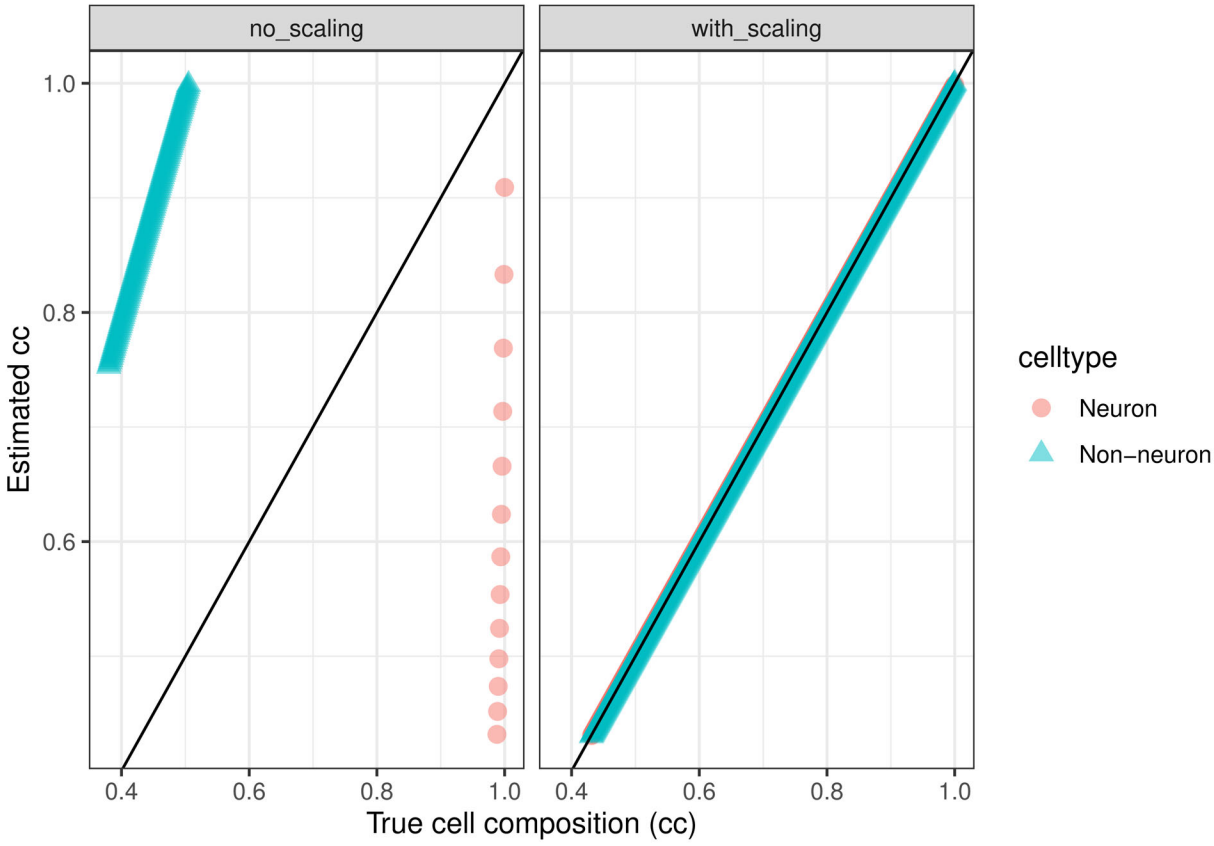
```
lgg$ggpt2
```



The final plot shows scatterplots of the bias, or proportions. The true cell type proportions are on the x-axis, the predicted proportions are on the y-axis, and the reference line with a slope of 1 and intercept of 0 is also shown.

```
lgg$ggpt.bias
```

```
## Warning: Removed 2721 rows containing missing values (geom_point).
```



Conclusions

This vignette showed how to perform a minimal simulation and analyze the results.

For general information about the `lute` R package, see the User's Guide.

For additional simulation examples, see `size_factor_experiments.Rmd` vignette.