bayesNMF Advanced Options

```
library(bayesNMF)
library(ggplot2)
```

```
Warning: package 'ggplot2' was built under R version 4.3.2
```

```
library(pryr)
```

```
data <- readRDS(system.file("extdata", "example_data.rds", package = "bayesNMF"))
dim(data$M)
```

```
[1] 96 64
```

Convergence Specifications

Unlike standard MCMC problems, we cannot use multiple chains to determine convergence because different chains can have different numbers of latent factors which we would be unable to align. We instead determine convergence through an approach rooted in machine learning. The `convergence_control` parameter determines the specifics of this approach. These parameters can be adjusted by the user, but the default values are noted below.

```
convergence_control = new_convergence_control(
    MAP_over = 1000,
    MAP_every = 100,
    tol = 0.001,
    Ninarow_nochange = 5,
    Ninarow_nobest = 10,
    miniters = 1000,
    maxiters = 5000,
    minA = 0,
    metric = "logposterior"
)
```

We pre-determine that the MAP estimate will be the average over `MAP_over` samples. Starting at iteration `MAP_over` and at every `MAP_every` iterations after, we perform a status report: we compute the MAP estimate *as if this is the last iteration* and record log likelihood, log posterior, RMSE, and KL Divergence. We say the MAP "hasn't changed" if the specified `metric` has changed by less than 100*`tol`% since the previous report. We say the MCMC has converged when the MAP hasn't changed in `Ninarow_nochange` reports (i.e., `Ninarow_nochange`*`MAP_every` iterations) OR if it hasn't improved in `Ninarow_nobest` reports (i.e., `Ninarow_nobest`*`MAP_every` iterations) OR if the iteration has reached `maxiters`. Convergence also requires that the iteration is at least `miniters` and that the MAP estimate for A appears at least `minA` times.

A specified convergence control can be passed to bayesNMF with the `convergence_control` parameter:

```
sampler_learn_rank <- bayesNMF(
    data$M, 1:10,
    convergence_control = convergence_control,
    output_dir = "../examples/learn_rank",
    save_all_samples = TRUE,
    overwrite = TRUE
)
```

```
sampler_learn_rank <- readRDS("../examples/learn_rank/sampler.rds")
```

Setting Hyperparameters and Initial Values

The `hyperprior_params`, `init_prior_params`, and `init_params` parameters can be used to set the hyperparameters, initial prior parameters, and initial parameters for the model.

The parameters that can be initialized are P and E. If the rank is learning, then A and R can also be initialized. If the likelihood is Normal, then σ^2 can also be initialized. If the likelihood is Poisson, then Z can also be initialized. If parameters are not provided, they will be initialized from their prior distributions.

```
N = 10

init_params = list(
    R = N,
    A = matrix(1, nrow = 1, ncol = N),
    P = matrix(1, nrow = nrow(data$M), ncol = N),
    E = matrix(1, nrow = N, ncol = ncol(data$M))
)
```

```
# this is a short convergence control for testing
# this is not recommended for actual analysis
short_convergence_control <-new_convergence_control(
    MAP_over = 100,
    MAP_every = 20,
    maxiters = 1000,
    tol = 0.01 # 1% change
)
```

```
Warning in new_convergence_control(MAP_over = 100, MAP_every = 20, maxiters =
1000, : miniters >= maxiters, setting miniters to 0.
```

The prior and hyperprior parameters that can be initialized depend on the prior.

Truncated Normal Prior

The priors are:

$$P_{kn} \sim \text{TruncatedNormal}(\mu^p_{kn}, (\sigma^p_{kn})^2, 0, \infty)$$
$$E_{ng} \sim \text{TruncatedNormal}(\mu^e_{ng}, (\sigma^e_{ng})^2, 0, \infty)$$

For Truncated Normal prior, the prior parameters are matrices μ^p and $(\sigma^p)^2$ of dimensions $K \times N$ and matrices μ^e and $(\sigma^e)^2$ of dimensions $N \times G$. If these matrices are not provided, they will be initialized from their hyperprior distributions.

The hyperpriors are:

$$\mu^p_{kn} \sim \text{Normal}(m_p, s_p)$$
$$(\sigma^p_{kn})^2 \sim \text{InvGamma}(a_p, b_p)$$
$$\mu^e_{ng} \sim \text{Normal}(m_e, s_e)$$
$$(\sigma^e_{ng})^2 \sim \text{InvGamma}(a_e, b_e)$$

For Truncated Normal prior, the hyperprior parameters are m_p, s_p, a_p, b_p, m_e, s_e, a_e, and b_e.

```
truncnorm_init_prior_params = list(
    Mu_p = matrix(0, nrow = nrow(data$M), ncol = N),
    Sigmasq_p = matrix(1, nrow = nrow(data$M), ncol = N),
    Mu_e = matrix(0, nrow = N, ncol = ncol(data$M)),
    Sigmasq_e = matrix(1, nrow = N, ncol = ncol(data$M))
)
```

```
# these are the default values
truncnorm_hyperprior_params = list(
    m_p = 0, s_p = sqrt(mean(data$M)/N),
    a_p = N + 1, b_p = sqrt(N),
    m_e = 0, s_e = sqrt(mean(data$M)/N),
    a_e = N + 1, b_e = sqrt(N)
)
```

```
sampler_learn_rank_PT_init <- bayesNMF(
    data$M, 1:10,
    hyperprior_params = truncnorm_hyperprior_params,
    init_prior_params = truncnorm_init_prior_params,
    init_params = init_params,
    output_dir = "../examples/learn_rank_PT_init",
    save_all_samples = TRUE,
    overwrite = TRUE,
    convergence_control = short_convergence_control # testing only, NOT RECOMMENDED
)
```

```
sampler_learn_rank_PT_init <- readRDS(
    "../examples/learn_rank_PT_init/sampler.rds"
)
```

Here we can check that initial values were set correctly.

```
all(sampler_learn_rank_PT_init$samples$P[[1]] == init_params$P)
```

```
[1] TRUE
```

```
all(sampler_learn_rank_PT_init$samples$Mu_p[[1]] == truncnorm_init_prior_params$Mu_p)
```

```
[1] TRUE
```

```
sampler_learn_rank_PT_init$hyperprior_params$m_p == truncnorm_hyperprior_params$m_p
```

```
[1] TRUE
```

Exponential Prior

The priors are:

$$P_{kn} \sim \text{Exponential}(\lambda^p_{kn})$$
$$E_{ng} \sim \text{Exponential}(\lambda^e_{ng})$$

For Exponential prior, the prior parameters are matrices λ^p and λ^e of dimensions $K \times N$ and $N \times G$, respectively. If these matrices are not provided, they will be initialized from their hyperprior distributions.

The hyperpriors are:

$$\lambda^p_{kn} \sim \text{Gamma}(a_p, b_p)$$
$$\lambda^e_{ng} \sim \text{Gamma}(a_e, b_e)$$

For Exponential prior, the hyperprior parameters are a_p, b_p, a_e, and b_e.

```
exp_init_prior_params = list(
    Lambda_p = matrix(1, nrow = nrow(data$M), ncol = N),
    Lambda_e = matrix(1, nrow = N, ncol = ncol(data$M))
)

# these are the default values
exp_hyperprior_params = list(
    a_p = 10*sqrt(N), b_p = 10*sqrt(mean(data$M)),
    a_e = 10*sqrt(N), b_e = 10*sqrt(mean(data$M))
)
```

```
sampler_learn_rank_PE_init <- bayesNMF(
    data$M, 1:10,
    prior = "exponential",
    hyperprior_params = exp_hyperprior_params,
    init_prior_params = exp_init_prior_params,
    init_params = init_params,
    output_dir = "../examples/learn_rank_PE_init",
    save_all_samples = TRUE,
    overwrite = TRUE,
    convergence_control = short_convergence_control # testing only, NOT RECOMMENDED
)
```

```
sampler_learn_rank_PE_init <- readRDS(
    "../examples/learn_rank_PE_init/sampler.rds"
)
```

Check that initial values were set correctly.

```
all(sampler_learn_rank_PE_init$samples$P[[1]] == init_params$P)
```

```
[1] TRUE
```

```
all(sampler_learn_rank_PE_init$samples$Mu_p[[1]] == exp_init_prior_params$Lambda_p)
```

```
[1] TRUE
```

```
sampler_learn_rank_PE_init$hyperprior_params$a_p == exp_hyperprior_params$a_p
```

```
[1] TRUE
```

Gamma Prior

The priors are:

$$P_{kn} \sim \text{Gamma}(\alpha^p_{kn}, \beta^p_{kn})$$
$$E_{ng} \sim \text{Gamma}(\alpha^e_{ng}, \beta^e_{ng})$$

For Gamma prior, the prior parameters are matrices α^p and β^p of dimensions $K \times N$ and matrices α^e and β^e of dimensions $N \times G$, respectively. If these matrices are not provided, they will be initialized from their hyperprior distributions.

The hyperpriors are:

$$\alpha^p_{kn} \sim \text{Gamma}(a_p, b_p)$$
$$\beta^p_{kn} \sim \text{Gamma}(c_p, d_p)$$
$$\alpha^e_{ng} \sim \text{Gamma}(a_e, b_e)$$
$$\beta^e_{ng} \sim \text{Gamma}(c_e, d_e)$$

For Gamma prior, the hyperprior parameters are a_p, b_p, c_p, d_p, a_e, b_e, c_e, and d_e.

```
gamma_init_prior_params = list(
    Alpha_p = matrix(1, nrow = nrow(data$M), ncol = N),
    Beta_p = matrix(1, nrow = nrow(data$M), ncol = N),
    Alpha_e = matrix(1, nrow = N, ncol = ncol(data$M)),
    Beta_e = matrix(1, nrow = N, ncol = ncol(data$M))
)
```

```
# these are the default values
gamma_hyperprior_params = list(
    a_p = 10*sqrt(N), b_p = 10,
    c_p = 10*sqrt(mean(data$M)), d_p = 10,
    a_e = 10*sqrt(N), b_e = 10,
    c_e = 10*sqrt(mean(data$M)), d_e = 10
)
```

```
# this is slow because MH = FALSE
sampler_learn_rank_PG_init <- bayesNMF(
    data$M, 1:10,
    prior = "gamma",
    hyperprior_params = gamma_hyperprior_params,
    init_prior_params = gamma_init_prior_params,
    init_params = init_params,
    output_dir = "../examples/learn_rank_PG_init",
    save_all_samples = TRUE,
    overwrite = TRUE,
    convergence_control = short_convergence_control # testing only, NOT RECOMMENDED
)
```

```
sampler_learn_rank_PG_init <- readRDS(
    "../examples/learn_rank_PG_init/sampler.rds"
)
```

Check that initial values were set correctly.

```
all(sampler_learn_rank_PE_init$samples$P[[1]] == init_params$P)
```

```
[1] TRUE
```

```
all(sampler_learn_rank_PE_init$samples$Mu_p[[1]] == gamma_init_prior_params$Alpha_p)
```

```
[1] TRUE
```

```
sampler_learn_rank_PE_init$hyperprior_params$a_p == gamma_hyperprior_params$a_p
```

```
[1] TRUE
```

Reduced compute options

For reference, here are the total runtime and memory usage of the sampler with the default settings.

```
length(sampler_learn_rank$samples$P)
```

```
[1] 4100
```

```
sampler_learn_rank$time$total
```

```
Time difference of 32.13316 mins
```

```
pryr::object_size(sampler_learn_rank)
```

```
217.40 MB
```

By default, the sampler is saved and trace plot png files are updated periodically. This I/O intensive operation can be reduced by specifying `periodic_save = FALSE` to only save the sampler object and trace plot png files at the very end.

```
sampler_learn_rank_faster <- bayesNMF(
    data$M, rank = 1:10,
    output_dir = "../examples/learn_rank_faster",
    periodic_save = FALSE,
    overwrite = TRUE
)
```

```
sampler_learn_rank_faster <- readRDS("../examples/learn_rank_faster/sampler.rds")
length(sampler_learn_rank_faster$samples$P)
```

```
[1] 4100
```

```
sampler_learn_rank_faster$time$total
```

```
Time difference of 17.24356 mins
```

```
pryr::object_size(sampler_learn_rank_faster)
```

```
217.40 MB
```

Also by default, all Gibbs samples are saved. However, for large datasets or long runs, this can be memory-intensive. The user can specify `save_all_samples = FALSE` to only save the last 1000 samples for MAP inference.

```
sampler_learn_rank_faster_smaller <- bayesNMF(
    data$M, rank = 1:10,
    output_dir = "../examples/learn_rank_faster_smaller",
    save_all_samples = FALSE,
    periodic_save = FALSE,
    overwrite = TRUE
)
```

```
sampler_learn_rank_faster_smaller <- readRDS("../examples/learn_rank_faster_smaller/sampler.r
length(sampler_learn_rank_faster_smaller$samples$P)
```

```
[1] 1000
```

```
sampler_learn_rank_faster_smaller$time$total
```

```
Time difference of 16.4734 mins
```

```
pryr::object_size(sampler_learn_rank_faster_smaller)
```

```
53.66 MB
```

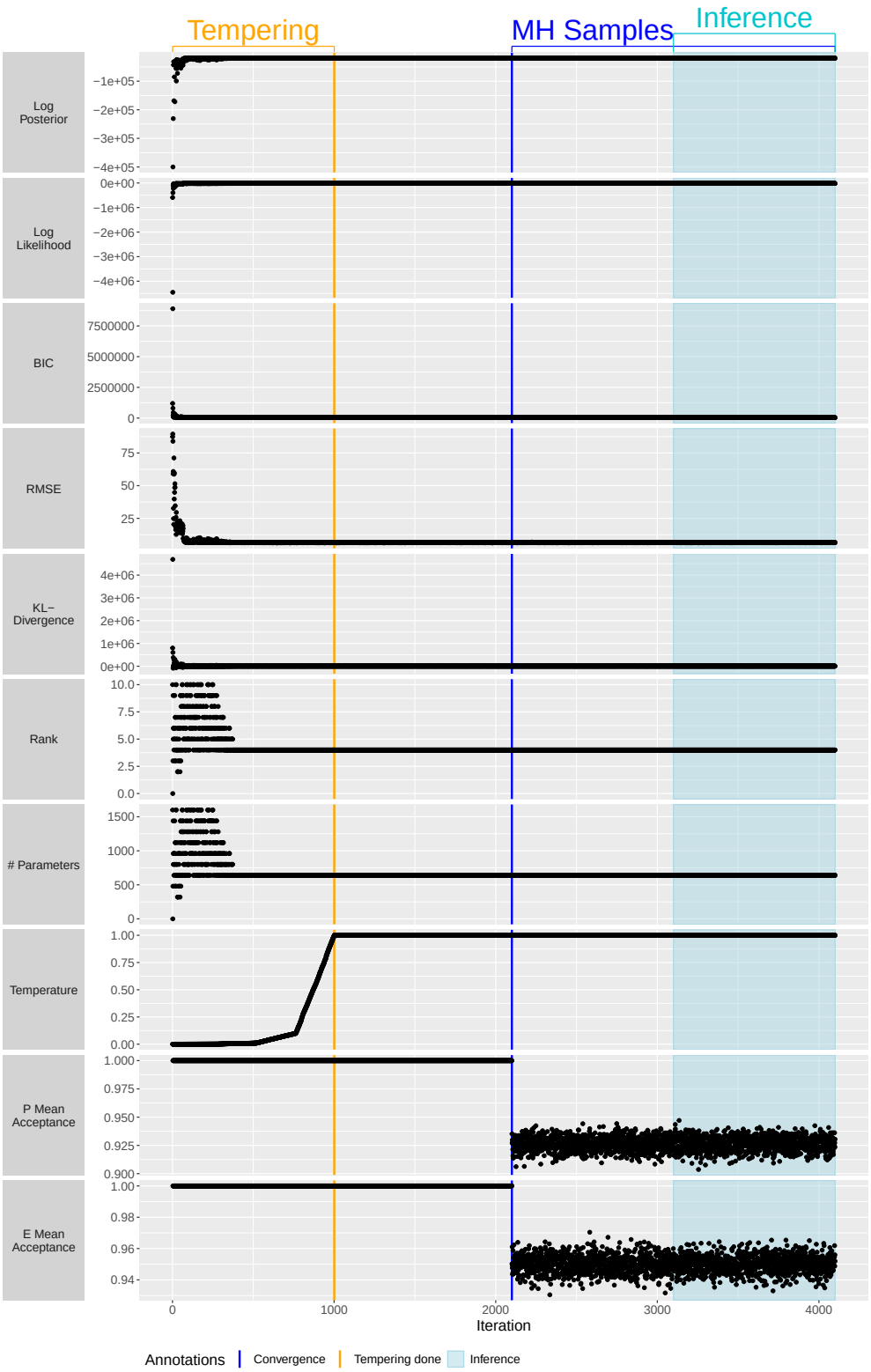
Even with these settings, the log file will still be updated and can be used to monitor progress.

We used both of these settings to run all analyses in the paper to minimize computational time and memory requirements.

Inference on Custom Indices

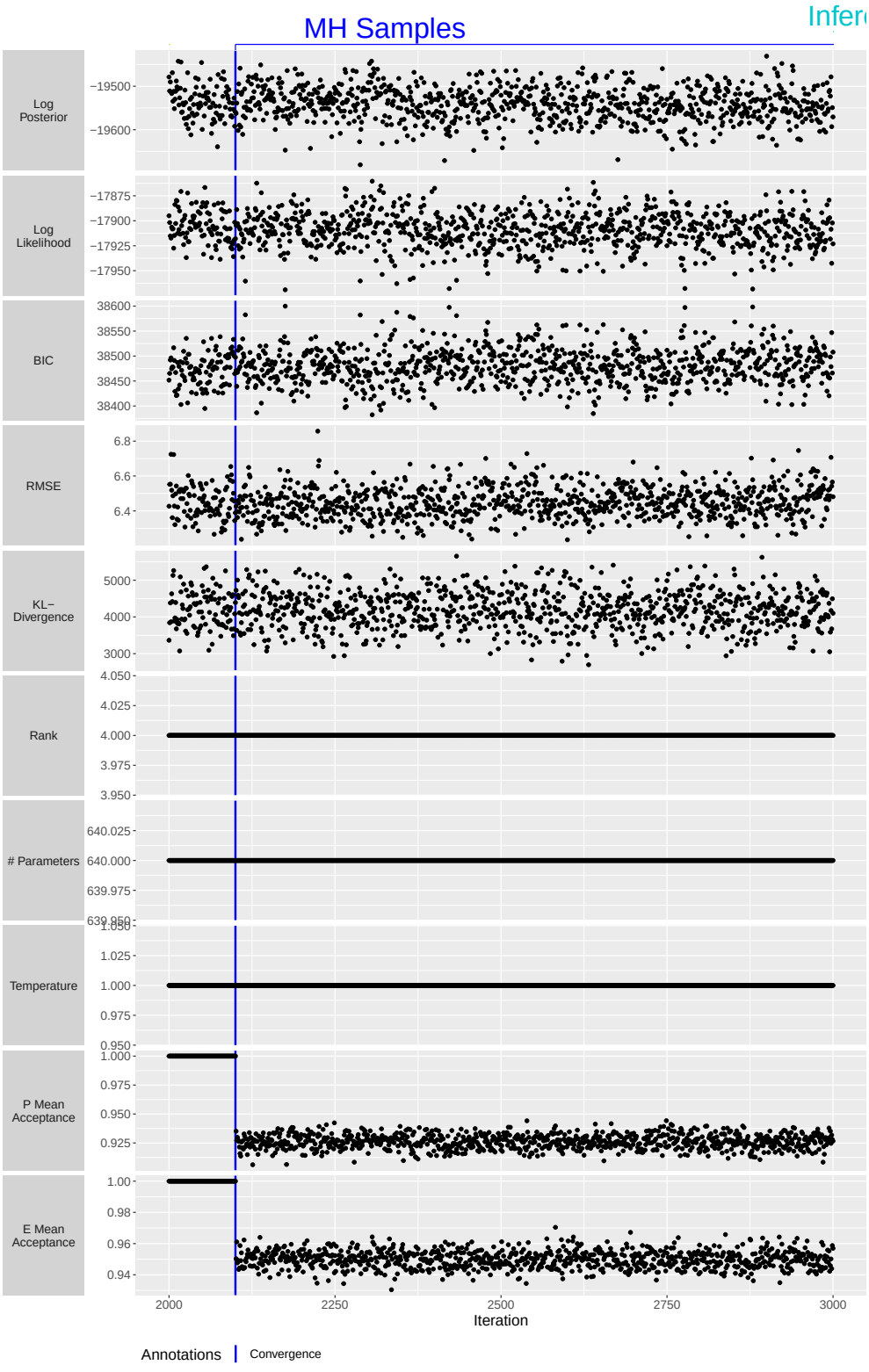
By default, bayesNMF performs inference over the last 1000 iterations of the sampler. However, the user can specify a custom range of indices to perform inference over. The `trace_plot` function can be used to plot the trace of the log posterior, log likelihood, RMSE, and KL Divergence, by default over all iterations.

```
trace_plot(sampler_learn_rank)
```


Tempering
MH Samples
Inference
Log Posterior
−1e+05
−2e+05
−3e+05
−4e+05
Log Likelihood
0e+00
−1e+06
−2e+06
−3e+06
−4e+06
BIC
7500000
5000000
2500000
0
RMSE
75
50
25
KL− Divergence
4e+06
3e+06
2e+06
1e+06
0e+00
Rank
10.0
7.5
5.0
2.5
0.0
Parameters
1500
1000
500
0
Temperature
1.00
0.75
0.50
0.25
0.00
P Mean Acceptance
1.000
0.975
0.950
0.925
0.900
E Mean Acceptance
1.00
0.98
0.96
0.94
0
1000
2000
3000
4000
Iteration
Annotations
Convergence
Tempering done
Inference

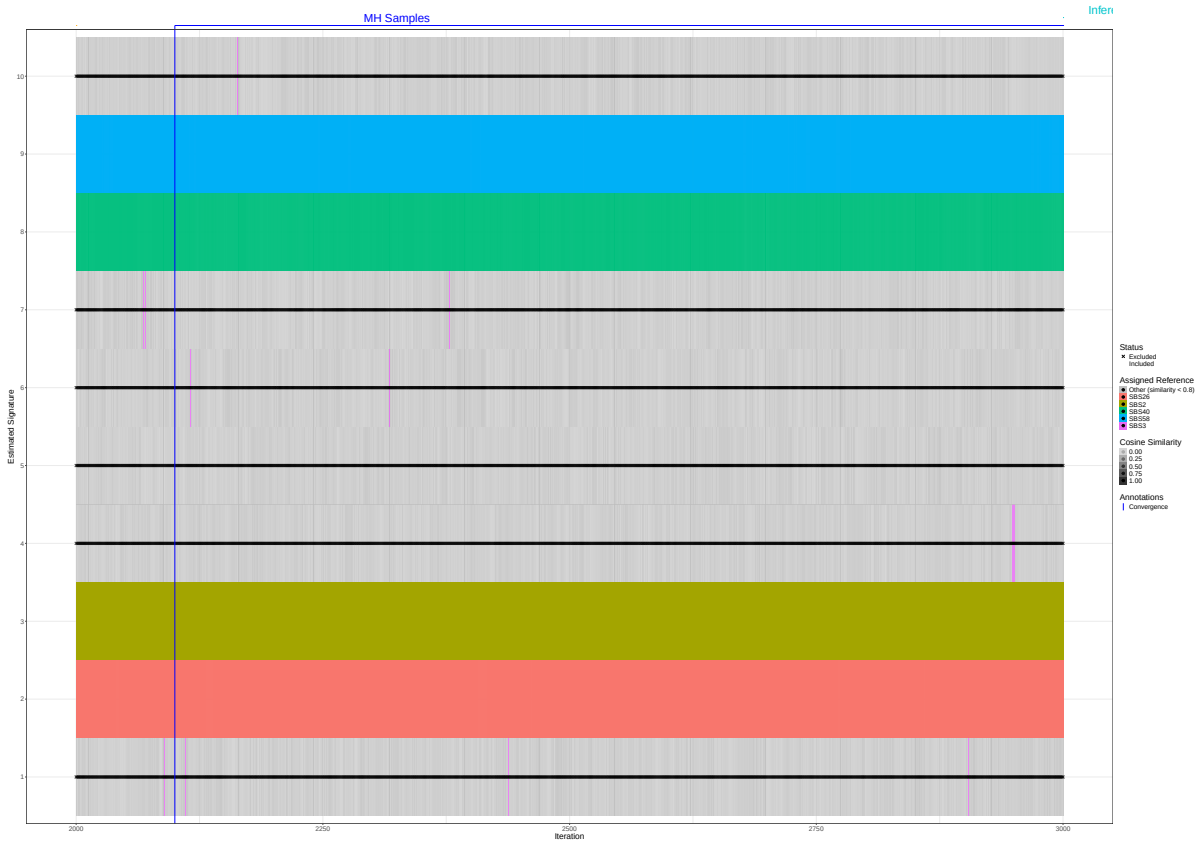
We can zoom in on a specific range of indices to get a better idea of the trace on a custom range of indices.

```
trace_plot(sampler_learn_rank, idx = 2000:3000)
```


MH Samples
Infer
Log Posterior
−19500
−19600
Log Likelihood
−17875
−17900
−17925
−17950
BIC
38600
38550
38500
38450
38400
RMSE
6.8
6.6
6.4
KL− Divergence
5000
4000
3000
Rank
4.050
4.025
4.000
3.975
3.950
Parameters
640.025
640.000
639.975
Temperature
1.050
1.025
1.000
0.975
0.950
P Mean Acceptance
1.000
0.975
0.950
0.925
E Mean Acceptance
1.00
0.98
0.96
0.94
2000
2250
2500
2750
3000
Iteration
Annotations
Convergence

We can also investigate any label switching in this subset of samples by plotting the label switching plot.

```
plot_label_switching(sampler_learn_rank, idx = 2000:3000)
```



We can also compute the MAP estimate for this custom range of indices by specifying the `end_iter` parameter to the `get_MAP` function. By default, the MAP estimate is still computed the nubmer of iterations specified in the convergence control (default 1000). This can also be changed, though, by specifying the `n_samples` parameter. Indicating `final = TRUE` will subset the results to only include the signatures in the final MAP estimate. For example, to compute the MAP estimate using iterations 2901-3000, we can do:

```
sampler_learn_rank$get_MAP(end_iter = 3000, n_samples = 100, final = TRUE)
dim(sampler_learn_rank$MAP$P)
```

```
[1] 96  4
```

```
sampler_learn_rank$MAP$idx
```

```
  [1] 2901 2902 2903 2904 2905 2906 2907 2908 2909 2910 2911 2912 2913 2914 2915
 [16] 2916 2917 2918 2919 2920 2921 2922 2923 2924 2925 2926 2927 2928 2929 2930
 [31] 2931 2932 2933 2934 2935 2936 2937 2938 2939 2940 2941 2942 2943 2944 2945
 [46] 2946 2947 2948 2949 2950 2951 2952 2953 2954 2955 2956 2957 2958 2959 2960
 [61] 2961 2962 2963 2964 2965 2966 2967 2968 2969 2970 2971 2972 2973 2974 2975
 [76] 2976 2977 2978 2979 2980 2981 2982 2983 2984 2985 2986 2987 2988 2989 2990
 [91] 2991 2992 2993 2994 2995 2996 2997 2998 2999 3000
```

After updating the sampler's MAP, any future visualizations or inference will be performed over the custom range of indices.

```
sampler_learn_rank$assign_signatures_ensemble()
```

```
$assignments
# A tibble: 4 x 5
  sig_est sig_ref MAP_cosine lower_cosine upper_cosine
    <dbl> <chr>        <dbl>        <dbl>        <dbl>
1       1 SBS26        0.999        0.998        0.999
2       2 SBS2         0.999        0.999        1.00
3       3 SBS40        0.957        0.942        0.965
4       4 SBS58        0.999        0.997        0.999

$votes
  sig_est sig_ref prop_votes
1       1   SBS26          1
2       2    SBS2          1
3       3   SBS40          1
4       4   SBS58          1
```

```
sampler_learn_rank$reference_comparison$idx
```

```
  [1] 2901 2902 2903 2904 2905 2906 2907 2908 2909 2910 2911 2912 2913 2914 2915
 [16] 2916 2917 2918 2919 2920 2921 2922 2923 2924 2925 2926 2927 2928 2929 2930
 [31] 2931 2932 2933 2934 2935 2936 2937 2938 2939 2940 2941 2942 2943 2944 2945
 [46] 2946 2947 2948 2949 2950 2951 2952 2953 2954 2955 2956 2957 2958 2959 2960
 [61] 2961 2962 2963 2964 2965 2966 2967 2968 2969 2970 2971 2972 2973 2974 2975
 [76] 2976 2977 2978 2979 2980 2981 2982 2983 2984 2985 2986 2987 2988 2989 2990
 [91] 2991 2992 2993 2994 2995 2996 2997 2998 2999 3000
```

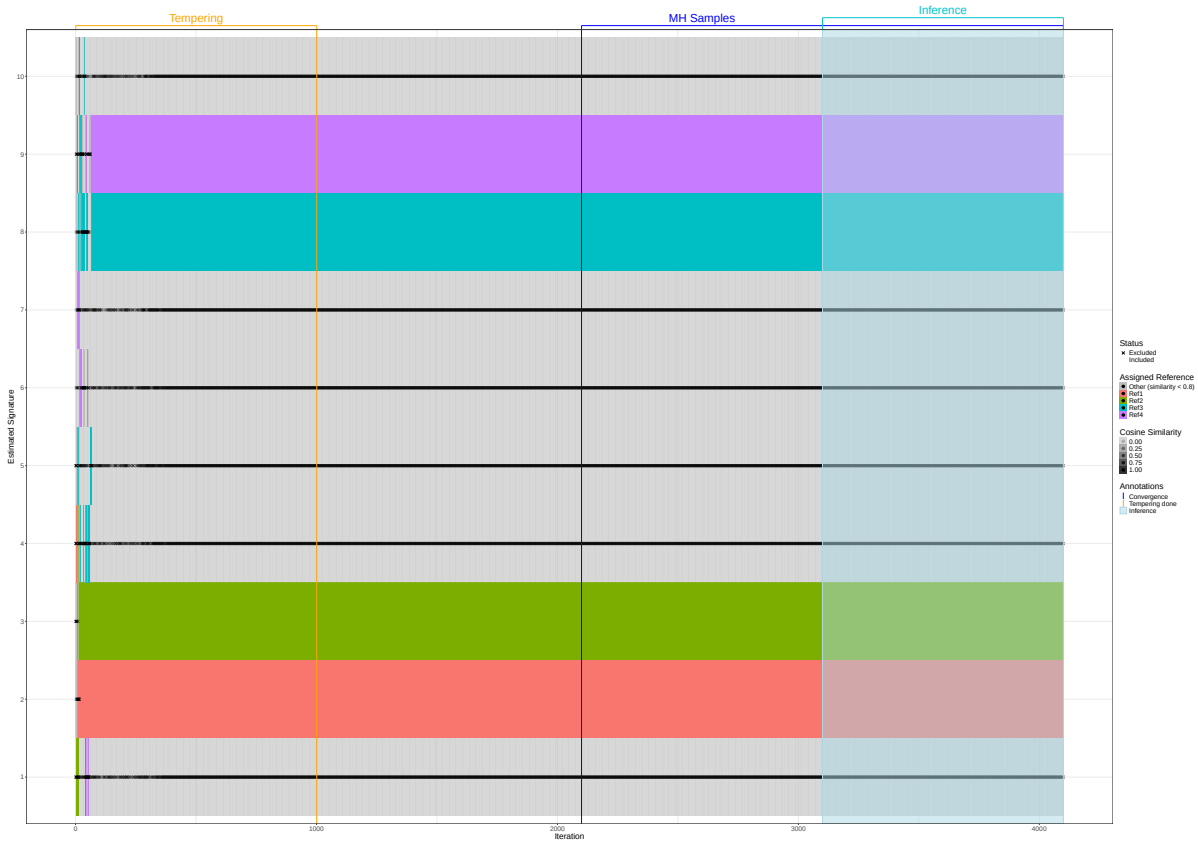
```
sampler_learn_rank$reference_comparison$assignments
```

```
# A tibble: 4 x 5
  sig_est sig_ref MAP_cosine lower_cosine upper_cosine
    <dbl> <chr>        <dbl>        <dbl>        <dbl>
1       1 SBS26        0.999        0.998        0.999
2       2 SBS2         0.999        0.999        1.00
3       3 SBS40        0.957        0.942        0.965
4       4 SBS58        0.999        0.997        0.999
```

Label Switching Diagnostic Without a Reference

If you don't have a reference to compare to, we recommend using the label switching diagnostic plot with respect to the final MAP estimate. This will still show you if the final estimated signatures were originally assigned to different latent indices.

```
plot_label_switching(
    sampler_learn_rank,
    reference_P = sampler_learn_rank$MAP$P
)
```



Additional Plots

Signature Plot Variations

The `plot_sig` function can be used to plot a reference mutational signature, an estimated mutational signature with posterior uncertainty as points with error bars, or both. The default reference is COSMIC, and unless otherwise specified, the reference signature chosen to plot is that with highest cosine similarity to the estimated signature.

Input

- `sampler`: bayesNMF_sampler object
- `sig`: selection of estimated signature. Integer, index of the estimated signature to plot. Can be NULL (default) to plot a reference signature on its own
- `ref`: selection of reference signature. Integer, column name string, or "best" (default). If "best", the reference signature with highest cosine similarity to the estimated signature will be used, and it's identification will be reported in the title. Can be NULL to plot an estimated signature on its own

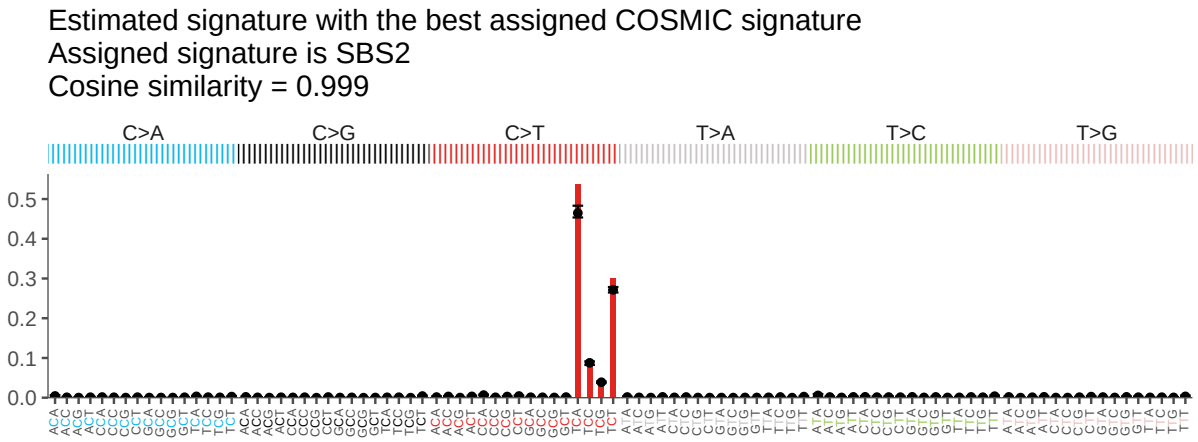
- `reference_P`: reference signatures to align to in the form of a reference P matrix or the string `"cosmic"` (default)
- `title`: optional first line of the title
- `cosine`: whether to report cosine similarity between estimate and reference in the title (ignored if `sig` or `ref` is NULL)

Output:

- `ggplot2` object

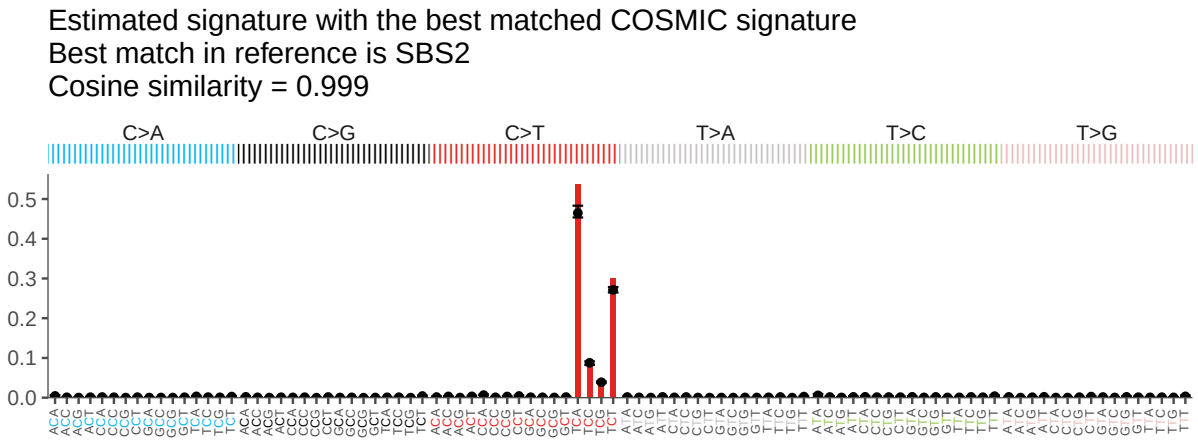
Plotting an estimated signature with the assigned COSMIC signature in the context of the full MAP estimate for P.

```
plot_sig(
    sampler = sampler_learn_rank, sig = 2,
    title = "Estimated signature with the best assigned COSMIC signature"
)
```



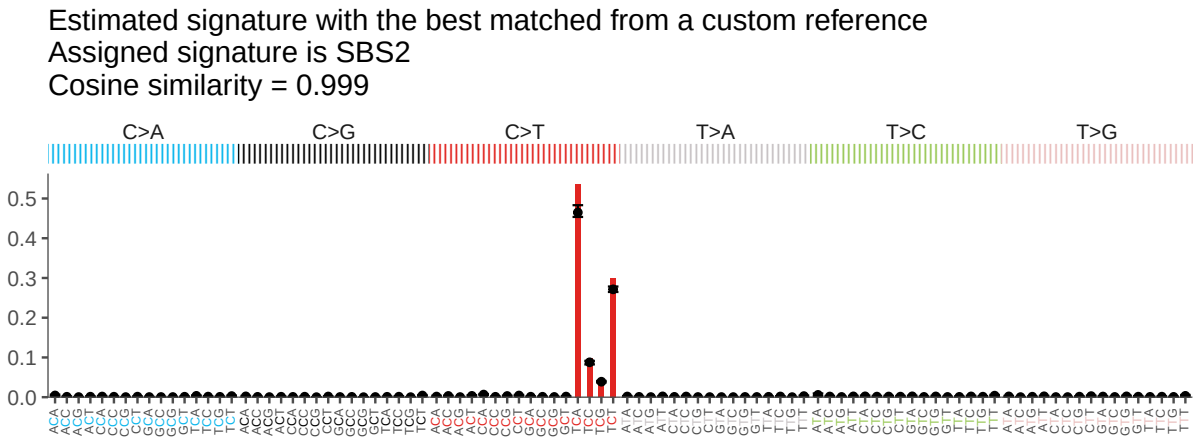
Plotting an estimated signature with the best matched COSMIC signature. This often aligns with the assigned COSMIC signature, but not necessarily, because assignment requires that each reference signature is assigned to exactly one estimated signature.

```
plot_sig(
    sampler = sampler_learn_rank, sig = 2, ref = "best",
    title = "Estimated signature with the best matched COSMIC signature"
)
```



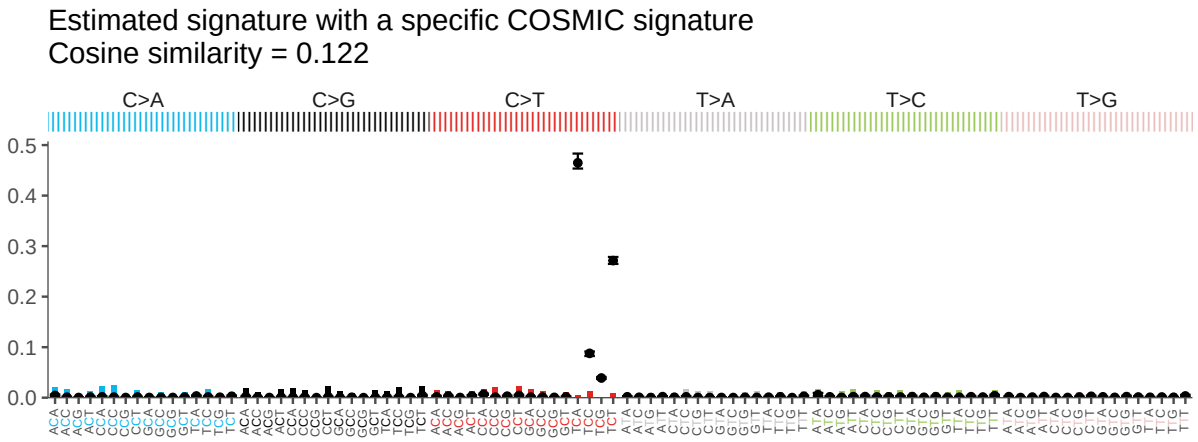
Plotting an estimated signature with the best matched signature from a custom reference.

```
plot_sig(
    sampler = sampler_learn_rank, sig = 2, reference_P = data$P,
    title = "Estimated signature with the best matched from a custom reference"
)
```



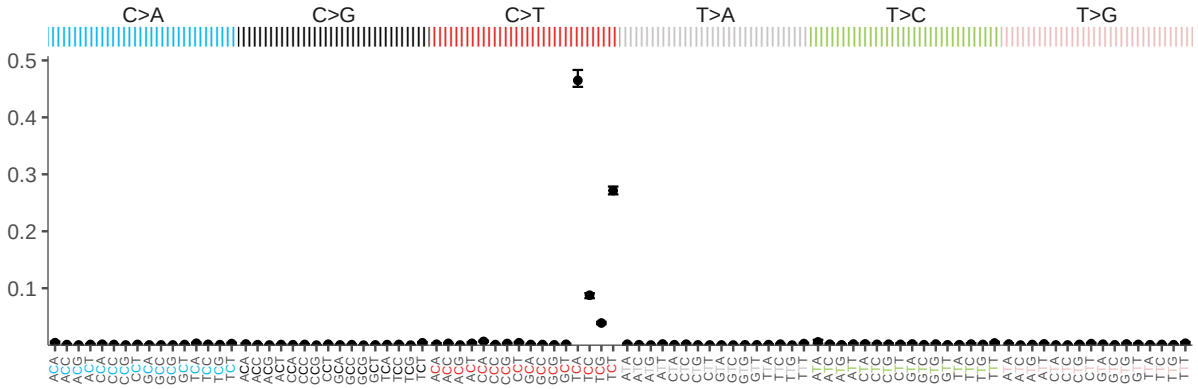
Plot an estimated signature with a specific COSMIC signature (in this case, one that does not align with the estimated signature).

```
plot_sig(
    sampler = sampler_learn_rank, sig = 2, ref = "SBS3",
    title = "Estimated signature with a specific COSMIC signature"
)
```



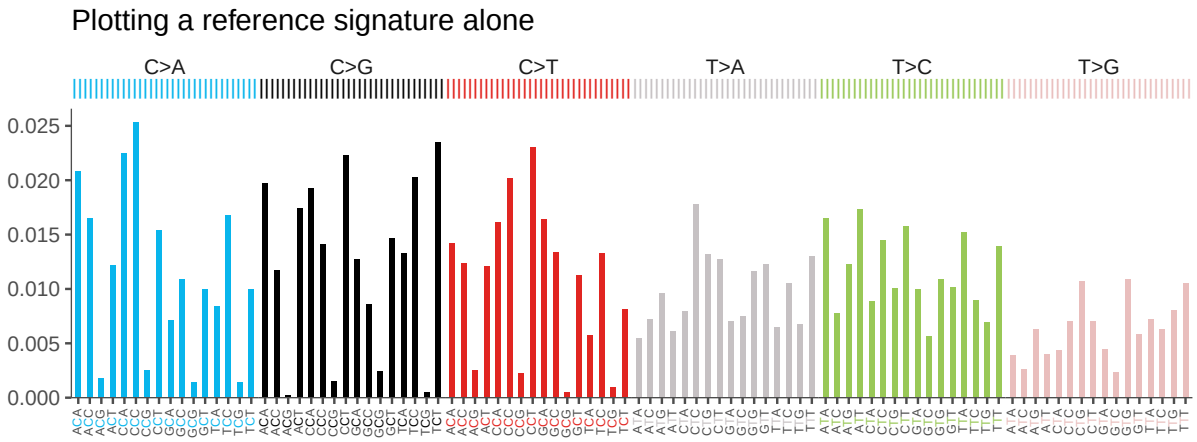
Plotting an estimated signature alone

```
plot_sig(
    sampler = sampler_learn_rank, sig = 2, reference_P = NULL,
    title = "Estimated signature alone"
)
```



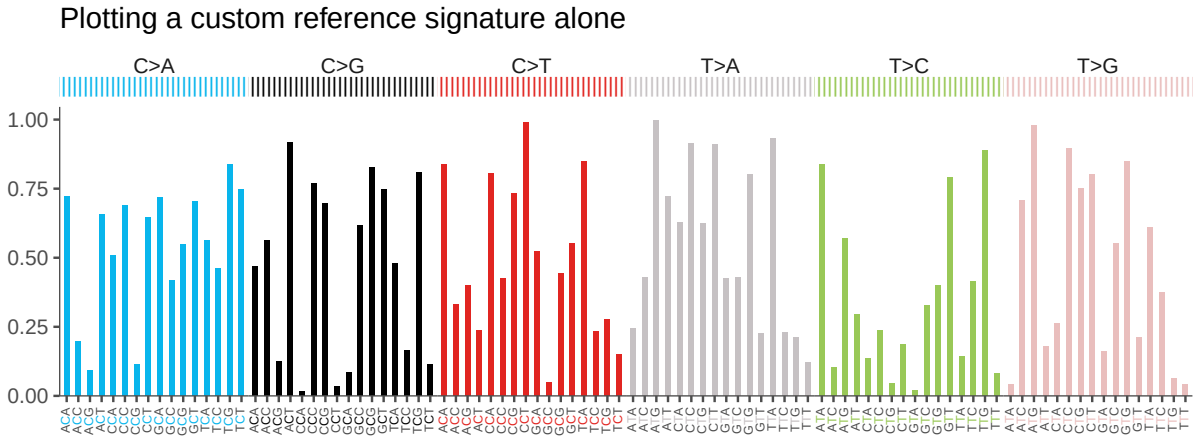
Plotting a reference signature alone

```
plot_sig(ref = "SBS3", title = "Plotting a reference signature alone")
```



Plotting a custom reference signature alone

```
plot_sig(ref = runif(96), title = "Plotting a custom reference signature alone")
```



Signature Distribution Plot Variations

The `plot_signature_dist` function plots the distribution of mutational counts across signatures for each mutation type across all, a subset, or one subject. This visualizes, for each trinucleotide mutation, how many are present and what signatures they are attributed to.

Input:

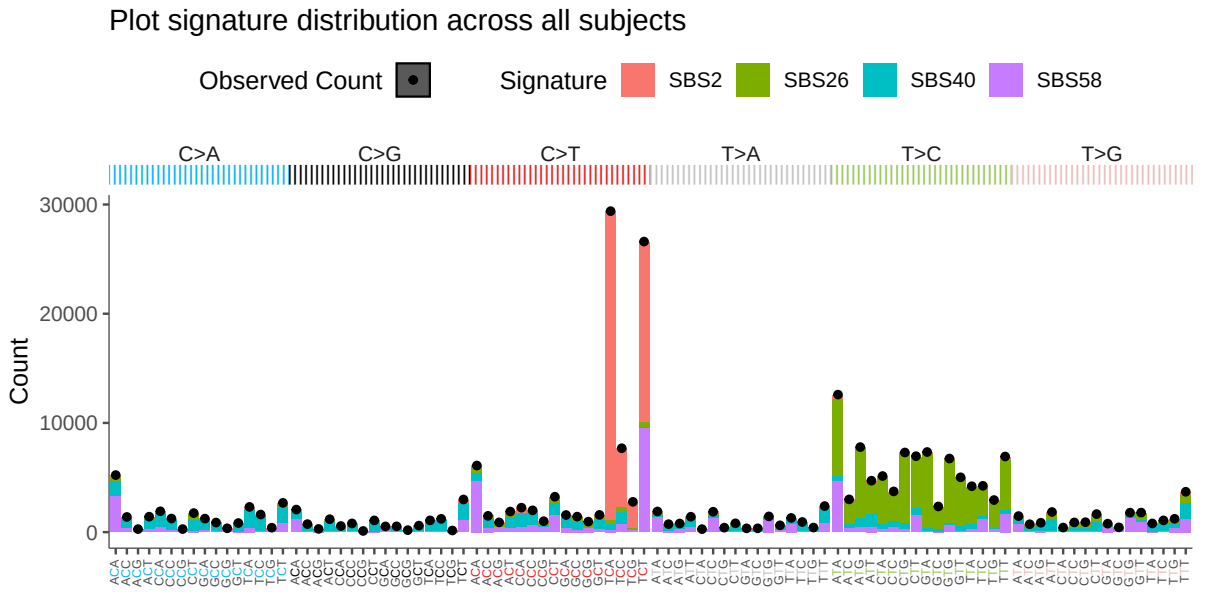
- `sampler`: bayesNMF_sampler object
- `subjects`: vector of subject indices to include in plot, defaults to all subjects
- `reference_P`: reference signatures to align to in the form of a reference P matrix or the string `"cosmic"` (default).
- `title`: plot title, default "Distribution of Signature Allocation"

Output:

- `ggplot2` object

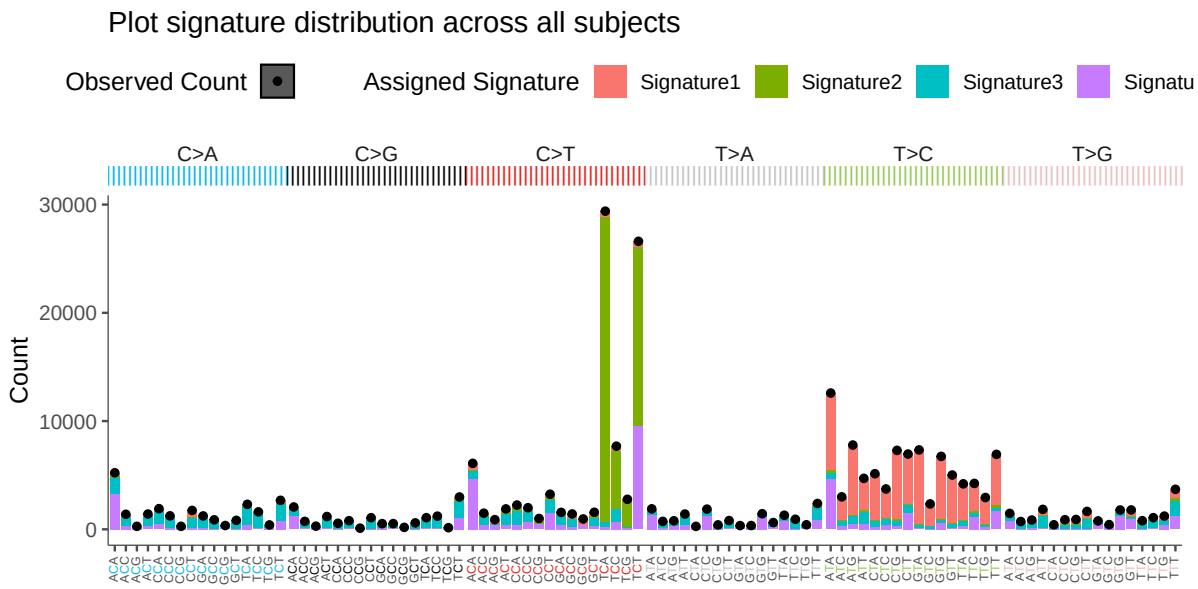
Plot signature distribution across all subjects (default).

```
plot_signature_dist(
    sampler = sampler_learn_rank,
    title = "Plot signature distribution across all subjects"
)
```



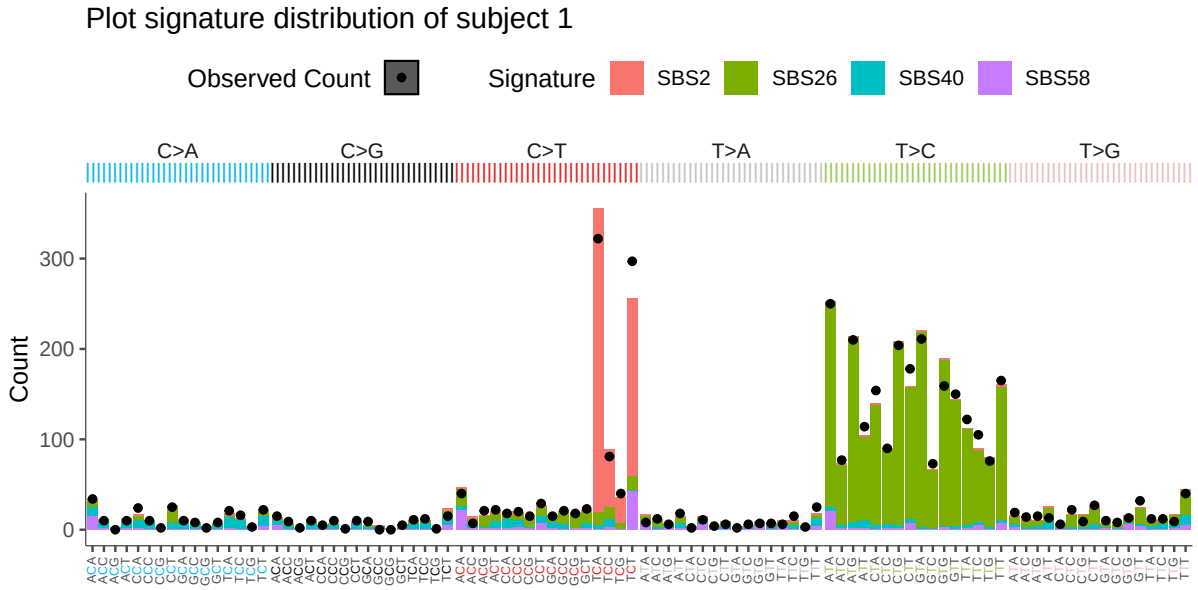
Plot signature distribution across all subjects without a reference.

```
plot_signature_dist(
    sampler = sampler_learn_rank,
    title = "Plot signature distribution across all subjects",
    reference_P = NULL
)
```



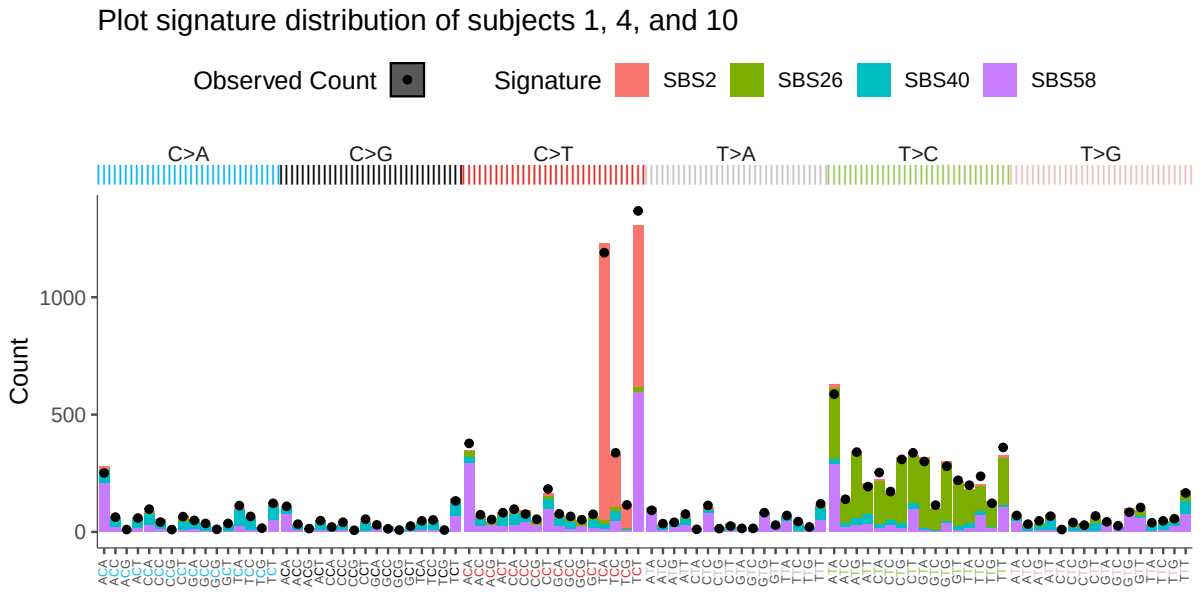
Plot signature distribution of a particular subject

```
plot_signature_dist(
    sampler = sampler_learn_rank,
    subject = 1,
    title = "Plot signature distribution of subject 1"
)
```



Plot signature distribution of a subset of subjects

```
plot_signature_dist(
    sampler = sampler_learn_rank,
    subjects = c(1,4,10),
    title = "Plot signature distribution of subjects 1, 4, and 10"
)
```



Summary Plot Variations

The `plot_summary` function plots a summary of the sampler's results, including the median contribution of each signature to the samples and the cosine similarity between the estimated and assigned reference signatures. Each sampler is plotted as its own x-axis tick, labeled by sampler name and sample size (G) of its dataset.

Input:

- `sampler_list`: named list of bayesNMF_sampler objects
- `reference_P`: matrix or "cosmic", reference signatures to align to
- `title`: string, title of the plot
- `fontsize`: integer, font size
- `keep_all_ref`: boolean, whether to keep all reference signatures in the plot, or only the ones assigned to at least one sampler

Output:

- `ggplot2` object

```
sampler_fixed_rank <- readRDS("../examples/fixed_rank/sampler.rds")
```

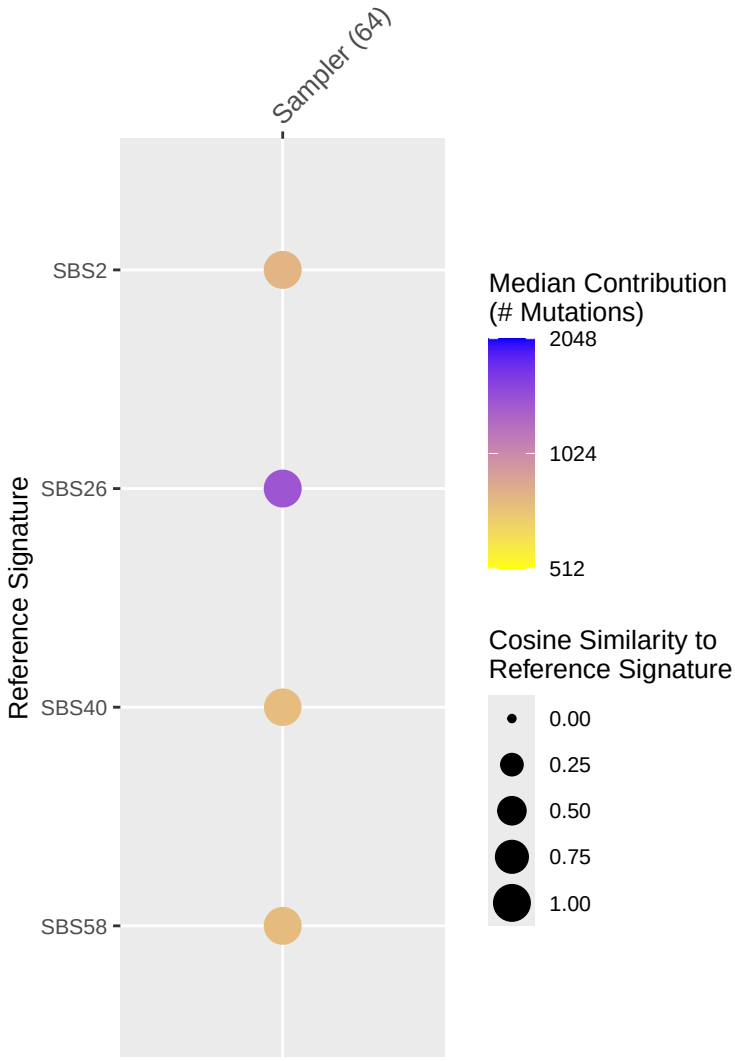
Get summary of a single sampler

```
summary(sampler_fixed_rank)
```

```
   G N  K Signature Med_Contribution Prop_atleast_1 Reference_Signature
1 64 4 96         1         792.8241              1                SBS2
2 64 4 96         2        1402.2311              1               SBS26
3 64 4 96         3         759.6027              1               SBS58
4 64 4 96         4         757.6485              1               SBS40
  Cosine_Similarity
1         0.9996804
2         0.9994094
3         0.9992593
4         0.9665231
```

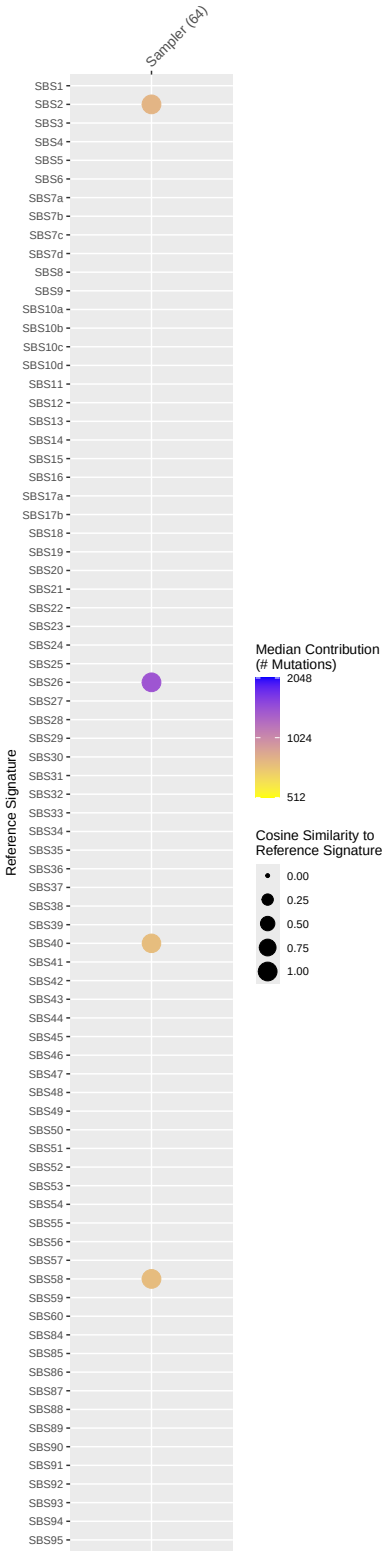
Plot summary of a single sampler

```
# make labels smaller for inline plots
small_text_theme <- ggplot2::theme(
    text = ggplot2::element_text(size = 10),
    title = ggplot2::element_text(size = 10),
    axis.title = ggplot2::element_text(size = 10),
    axis.text.x.top = ggplot2::element_text(size = 10),
    legend.title = ggplot2::element_text(size = 10)
)
plot_summary(sampler_fixed_rank) + small_text_theme
```



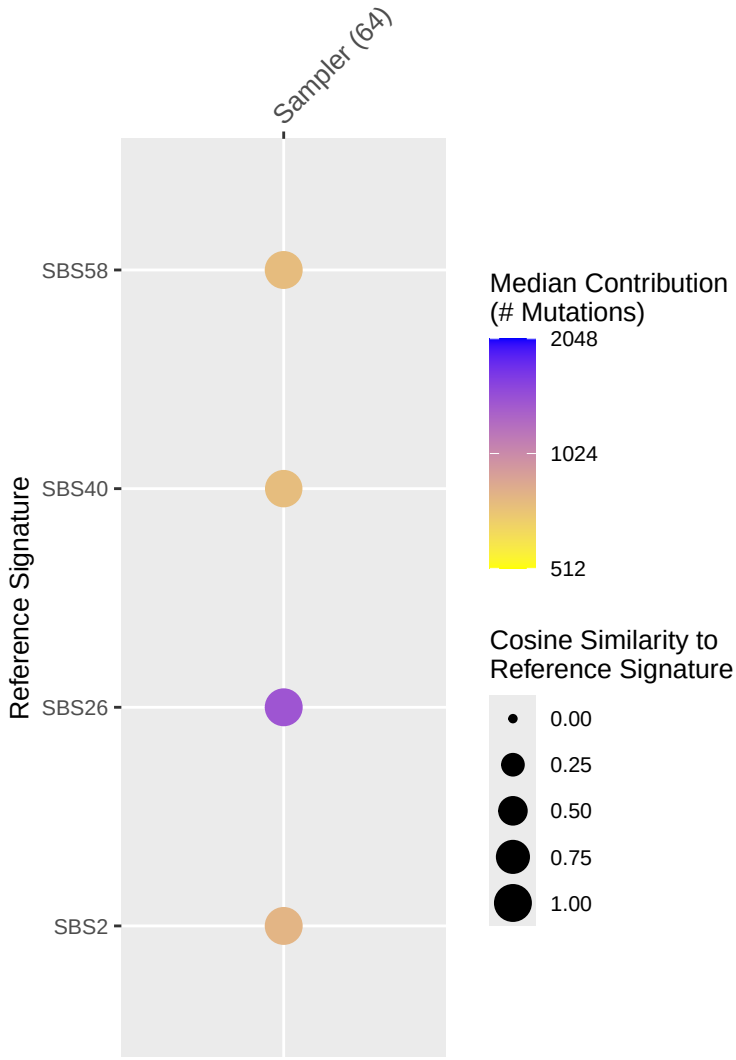
Plot summary of a single sampler with all reference signatures

```
plot_summary(sampler_fixed_rank, keep_all = TRUE) +
    small_text_theme
```


Sampler (64)
Reference Signature
SBS1
SBS2
SBS3
SBS4
SBS5
SBS6
SBS7a
SBS7b
SBS7c
SBS7d
SBS8
SBS9
SBS10a
SBS10b
SBS10c
SBS10d
SBS11
SBS12
SBS13
SBS14
SBS15
SBS16
SBS17a
SBS17b
SBS18
SBS19
SBS20
SBS21
SBS22
SBS23
SBS24
SBS25
SBS26
SBS27
SBS28
SBS29
SBS30
SBS31
SBS32
SBS33
SBS34
SBS35
SBS36
SBS37
SBS38
SBS39
SBS40
SBS41
SBS42
SBS43
SBS44
SBS45
SBS46
SBS47
SBS48
SBS49
SBS50
SBS51
SBS52
SBS53
SBS54
SBS55
SBS56
SBS57
SBS58
SBS59
SBS60
SBS84
SBS85
SBS86
SBS87
SBS88
SBS89
SBS90
SBS91
SBS92
SBS93
SBS94
SBS95
Median Contribution
(# Mutations)
2048
1024
512
Cosine Similarity to
Reference Signature
0.00
0.25
0.50
0.75
1.00

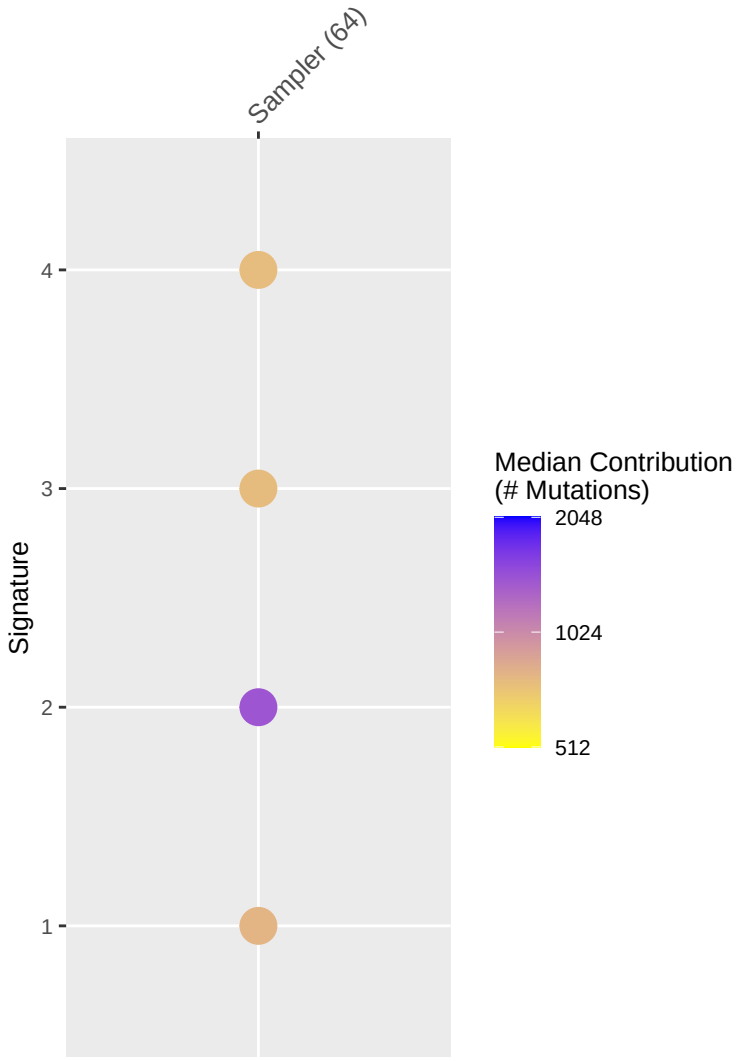
Plot summary of a single sampler with a custom reference

```
plot_summary(sampler_fixed_rank, reference_P = data$P) +
    small_text_theme
```



Plot summary of a single sampler with no reference

```
plot_summary(sampler_fixed_rank, reference_P = NULL) + small_text_theme
```



Get summary of a list of samplers

```
summarize_samplers(list(
    "Fixed Rank" = sampler_fixed_rank,
    "Learn Rank" = sampler_learn_rank
))
```

```
   G  N  K Signature Med_Contribution Prop_atleast_1 Reference_Signature
1 64  4 96         1         792.8241               1                SBS2
2 64  4 96         2        1402.2311               1               SBS26
3 64  4 96         3         759.6027               1               SBS58
4 64  4 96         4         757.6485               1               SBS40
```

```
5 64 10 96          1         1402.2075                1                SBS26
6 64 10 96          2          832.9475                1                 SBS2
7 64 10 96          3          720.2180                1                SBS40
8 64 10 96          4          755.4728                1                SBS58
  Cosine_Similarity            Name
1         0.9996804 Fixed Rank (64)
2         0.9994094 Fixed Rank (64)
3         0.9992593 Fixed Rank (64)
4         0.9665231 Fixed Rank (64)
5         0.9991500 Learn Rank (64)
6         0.9994219 Learn Rank (64)
7         0.9570266 Learn Rank (64)
8         0.9989187 Learn Rank (64)
```

Plot summary of a list of samplers

```
plot_summary(list(
    "Fixed Rank" = sampler_fixed_rank,
    "Learn Rank" = sampler_learn_rank
)) + small_text_theme
```