Recent Additions to tidymodels

Max Kuhn

tidymodels and Me

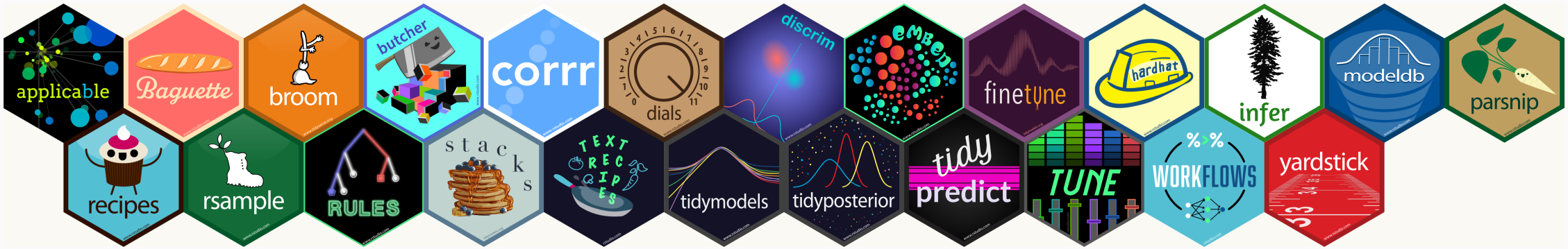
I'm a statistician and software Engineer for RStudio working on modeling.

Formerly 6y in infectious disease diagnostics and 12y in drug discovery.

max@rstudio.com, topepo, topepos

https://rstd.io/global2021/maxkuhn

tidymodels is a collection of modeling packages designed with the same principles as the tidyverse.



Recent Updates

Interfaces:

- `workflows` have a new interface for adding raw data columns (e.g. no formula or recipe)
- `recipes` now support all of the new `tidyselect` selectors
- `recipes` can now use `bake()` instead of `juice()` for the training set.
- The `autoplot()` method for `tune` objects works better for regular grids.

Today I'll be demonstrating a new package called **`finetune`**.

Efficiency:

- Parallel processing via PSOCK clusters on windows is less awful.
- `tune` can now use a larger number of parallel workers when tuning.
- For the 3 models that use them, sparse matrices can be passed from recipes to model functions.

Example Problem

For our example, the cell segmentation data are used to create a two-class classification model using a support vector machine (SVM).

- We'll tune this model over `cost` and `rbf_sigma`.

This talk is about new ways of **finding optimal values of these parameters**.

The data are resampled using 10-fold cross-validation.

The model definition and complete code set can be found at `https://rstd.io/global2021/maxkuhn` .

Climbing a Hill

We want to find good values of two tuning parameters (`cost` and `rbf_sigma`) that maximize the area under the ROC curve of the model. The true 2D surface:

Racing Methods

Racing methods can be used on pre-defined grids of tuning parameters.

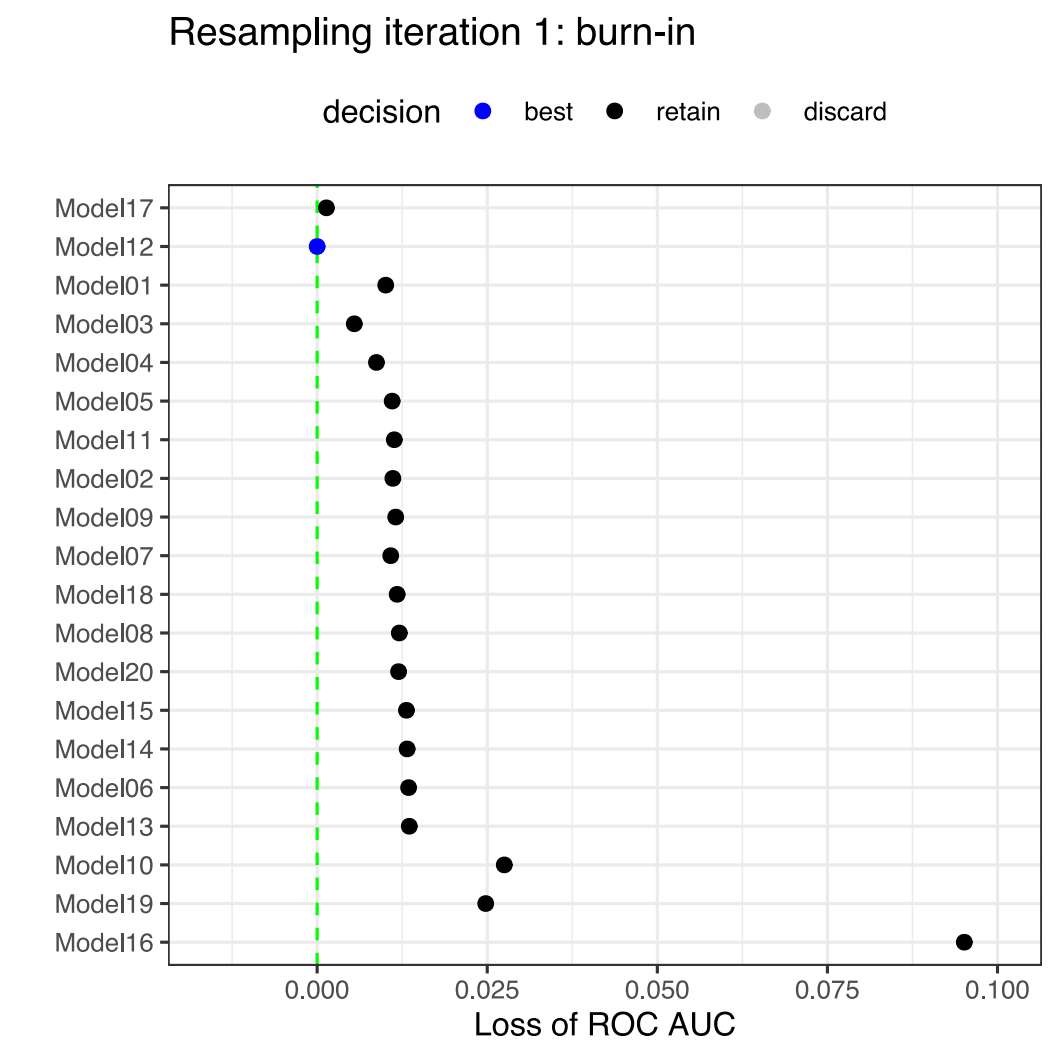
As the models are resampled (to compute performance), interim analyses are used to eliminate candidate parameters that have a low probability of of being the best.

The `finetune` package can use racing via ANOVA models that test for differences in parameters. A second method, based on win/loss statistics, can also be used.

- See *Futility Analysis in the Cross-Validation of Machine Learning Models* (Kuhn, 2014, *arXiv.org*)

Let's use a grid of 20 SVM tuning parameters generated using a space-filling design to demonstrate.

Illustrating the Race

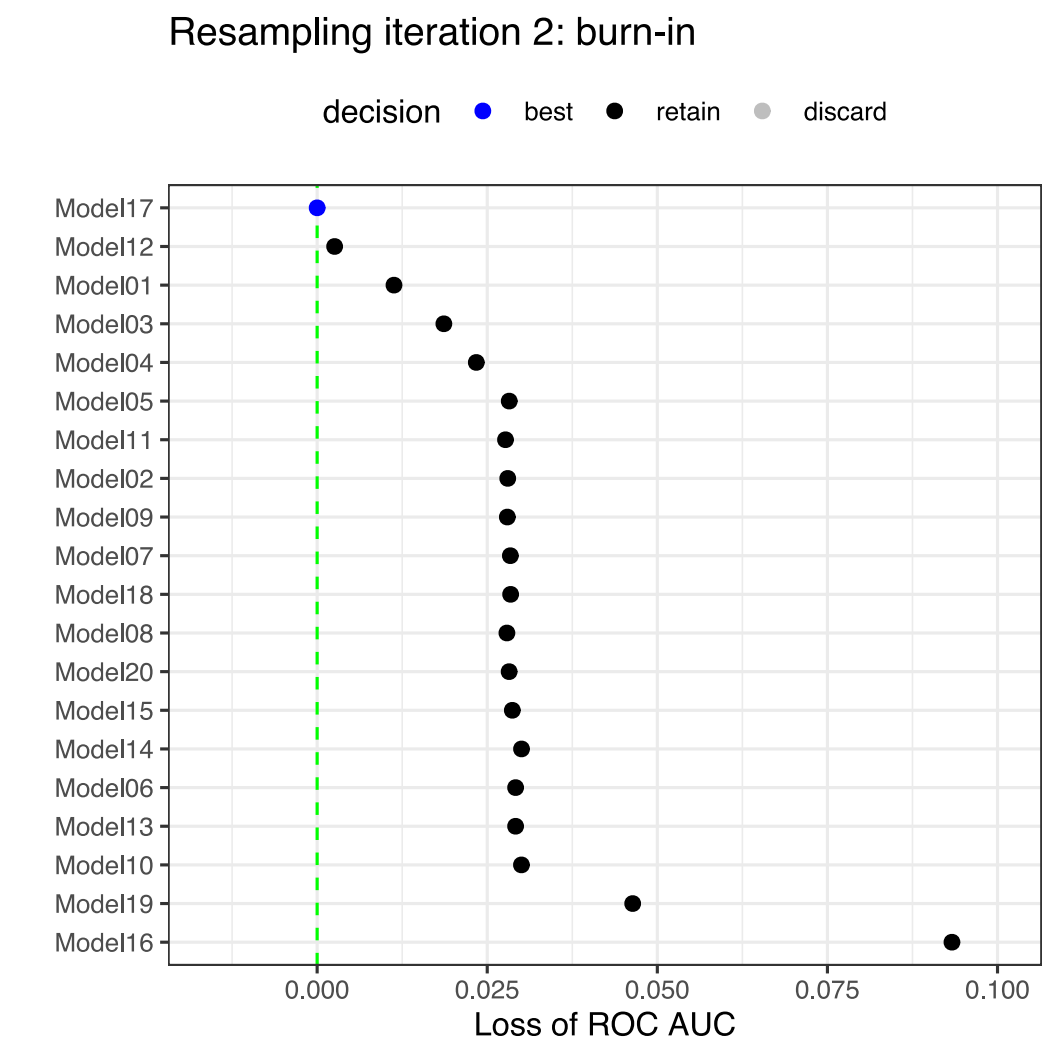


20 models go in... one comes out

The resampling method is simple 10-fold cross-validation.

Resampling iterations 1 and 2 proceed as normal.

Illustrating the Race

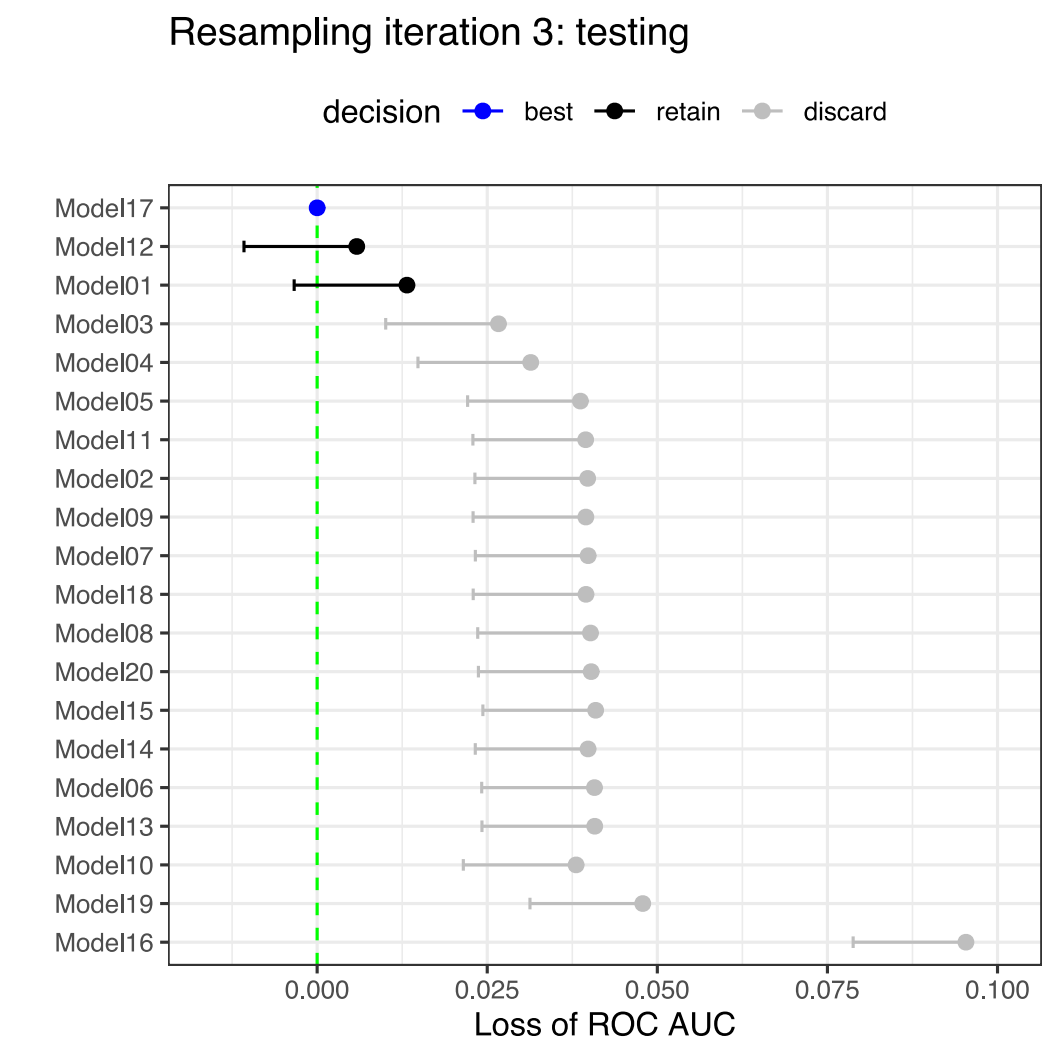


20 models go in... one comes out

The resampling method is simple 10-fold cross-validation.

Resampling iterations 1 and 2 proceed as normal.

Illustrating the Race

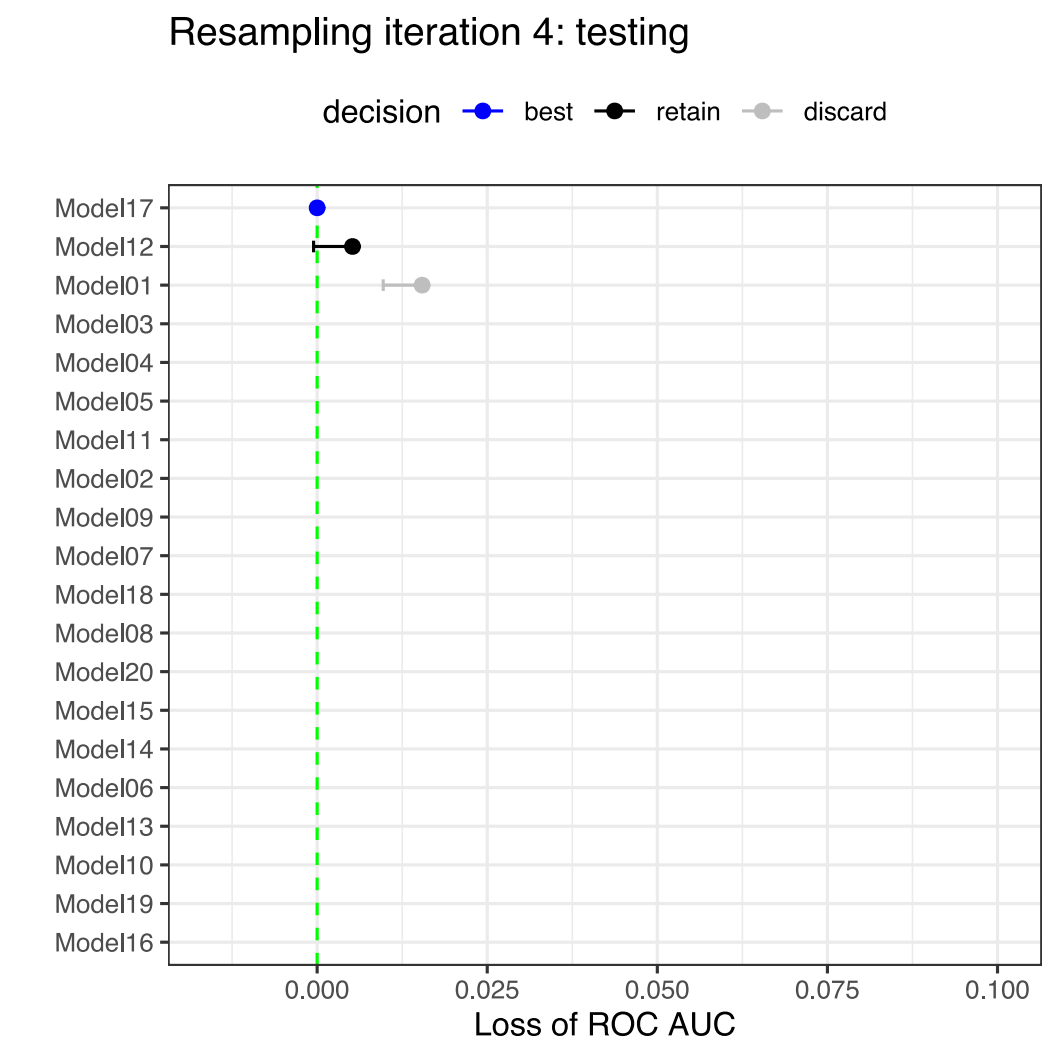


20 models go in... one comes out

The resampling method is simple 10-fold cross-validation.

At iteration 3, testing begins and many (poor) candidate models are removed.

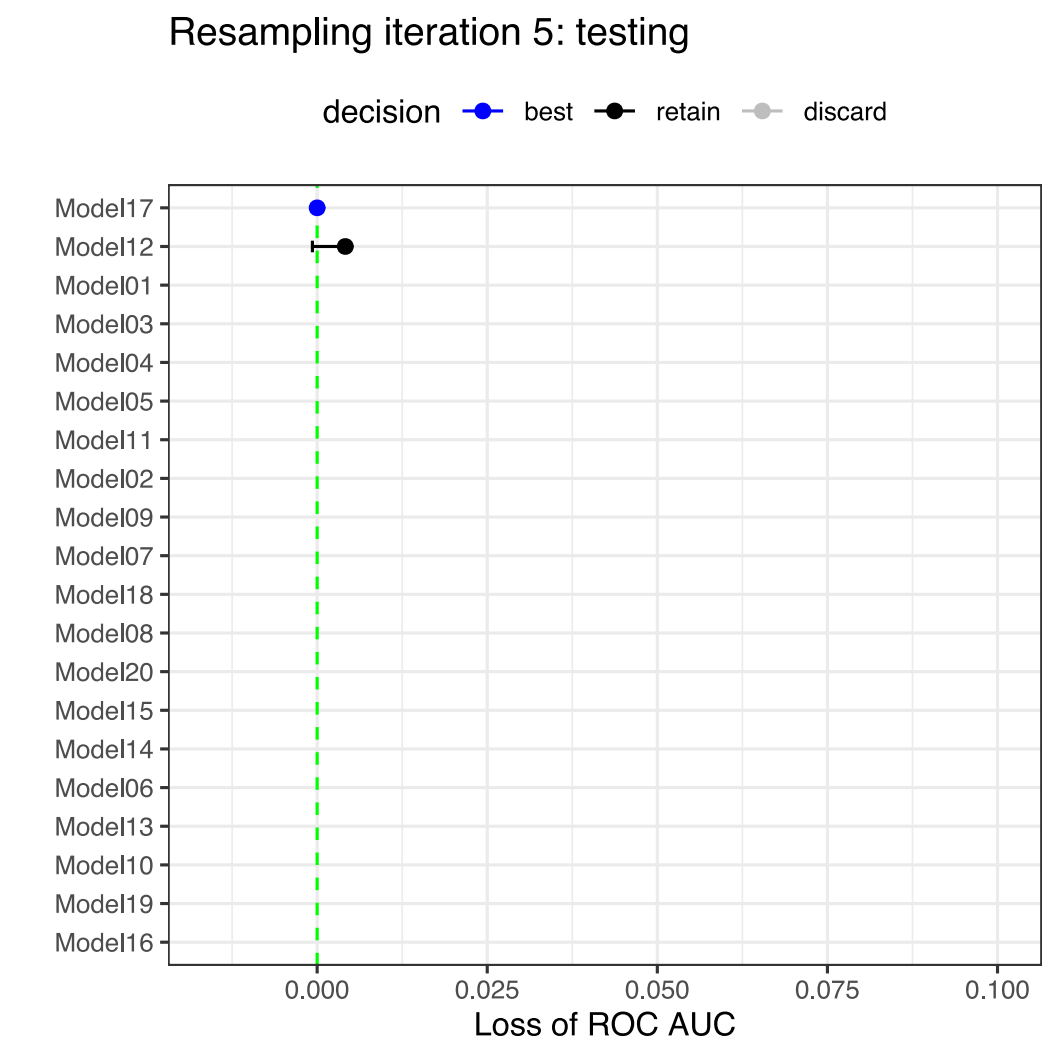
Illustrating the Race



20 models go in... one comes out

The resampling method is simple 10-fold cross-validation.

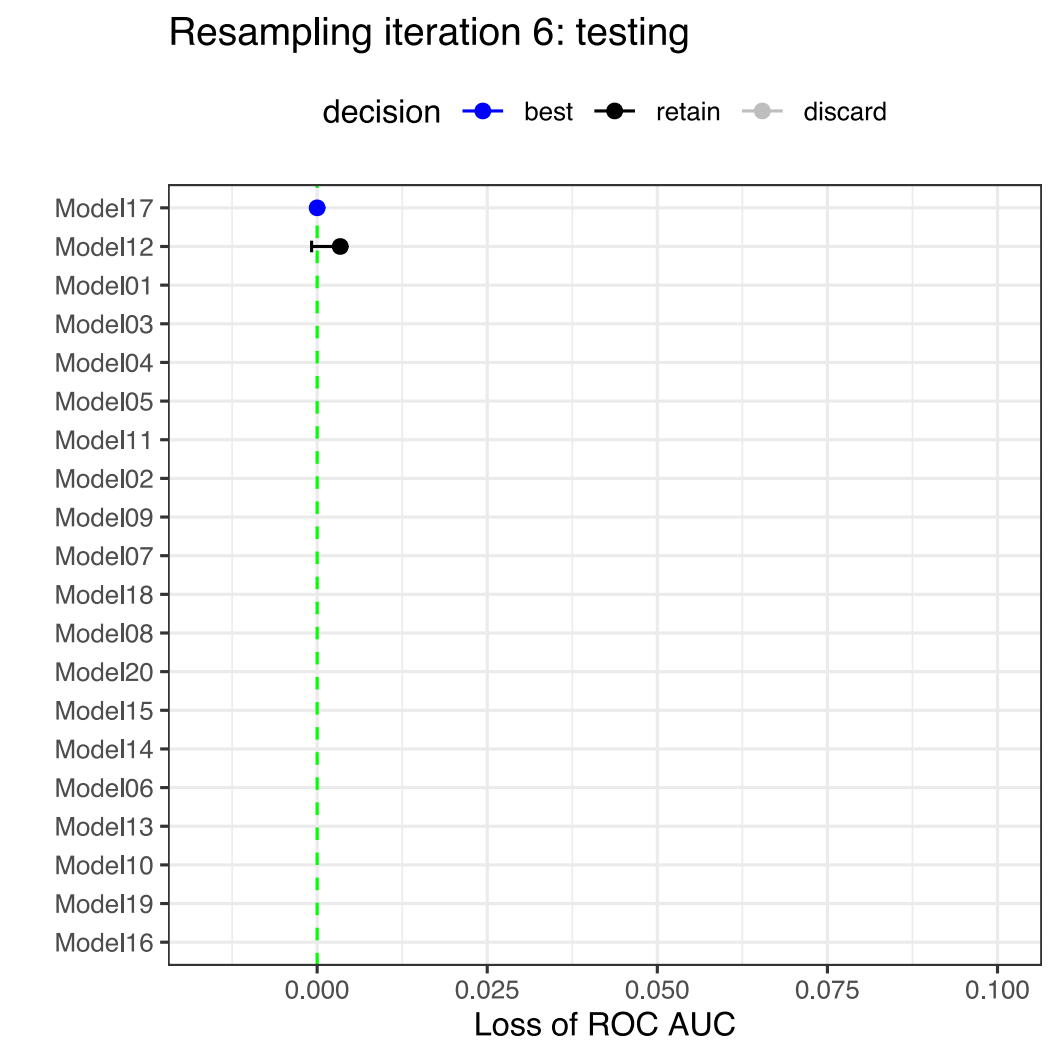
Illustrating the Race



20 models go in... one comes out

The resampling method is simple 10-fold cross-validation.

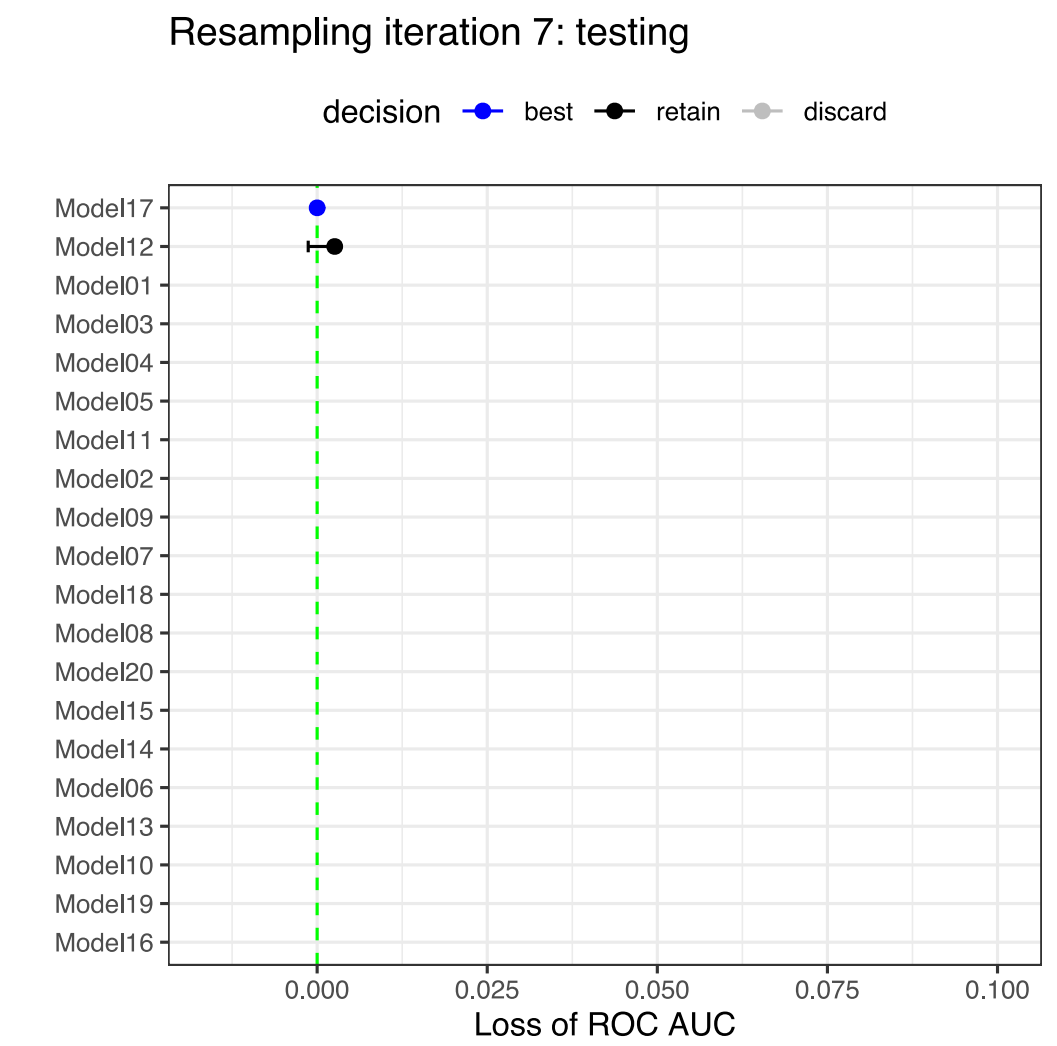
Illustrating the Race



20 models go in... one comes out

The resampling method is simple 10-fold cross-validation.

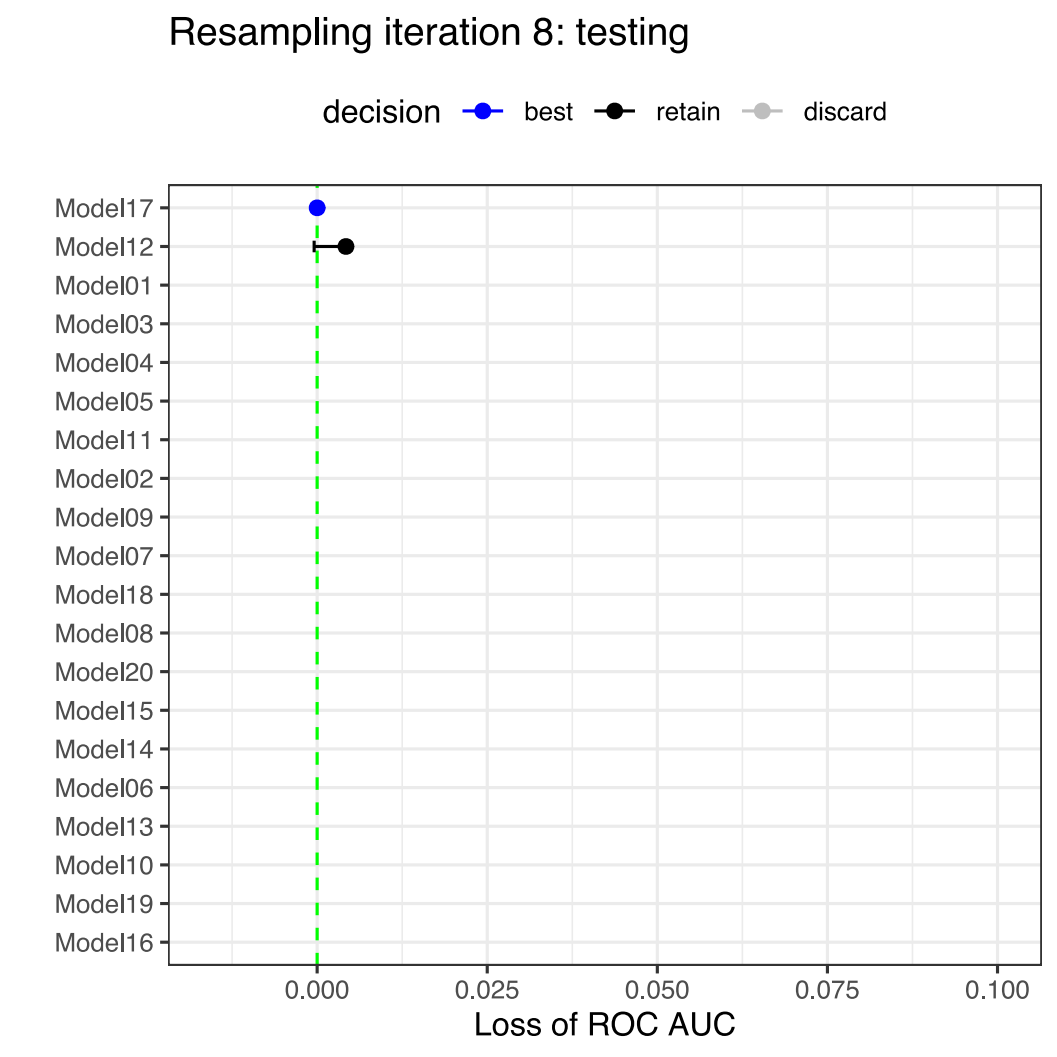
Illustrating the Race



20 models go in... one comes out

The resampling method is simple 10-fold cross-validation.

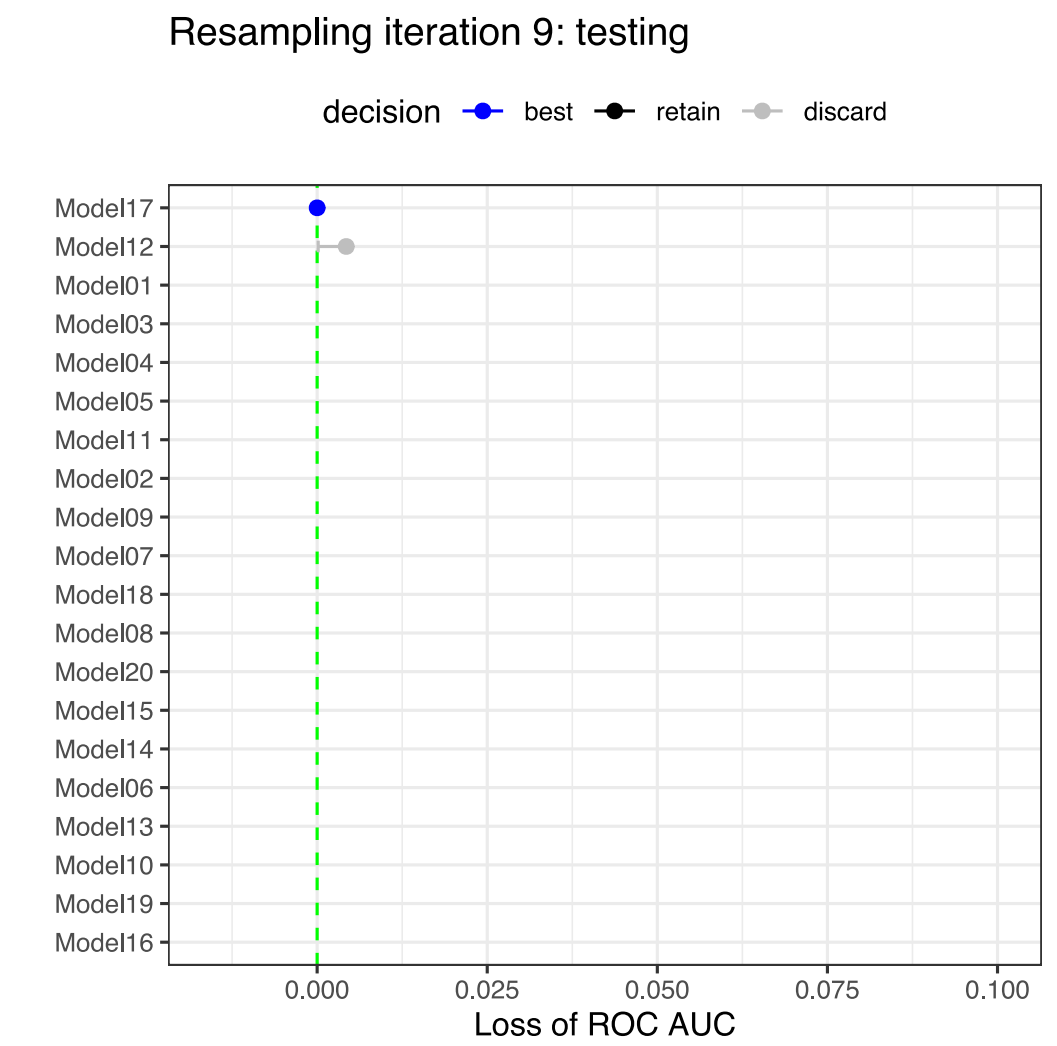
Illustrating the Race



20 models go in... one comes out

The resampling method is simple 10-fold cross-validation.

Illustrating the Race



20 models go in... one comes out

The resampling method is simple 10-fold cross-validation.

The 9th iteration removes the last competitor.

Iteration 10 just resamples the single model that remains.

A total of 74 models were fit instead of 200 (=37%).

Example Code for Racing

```
set.seed(99)
svm_race <-
  svm_wflow %>%
  tune_race_anova(resamples = cell_folds, grid = 20)
```

Using the `verbose_elim` option:

```
ℹ Racing will maximize the roc_auc metric.
ℹ Resamples are analyzed in a random order.
ℹ Fold10: 17 eliminated;  3 candidates remain.
ℹ Fold09:  1 eliminated;  2 candidates remain.
ℹ Fold05:  0 eliminated;  2 candidates remain.
ℹ Fold01:  0 eliminated;  2 candidates remain.
ℹ Fold07:  0 eliminated;  2 candidates remain.
ℹ Fold03:  0 eliminated;  2 candidates remain.
ℹ Fold06: All but one parameter combination were eliminated.
```

`tune_race_win_loss()` can also be used.

Simulated Annealing Search

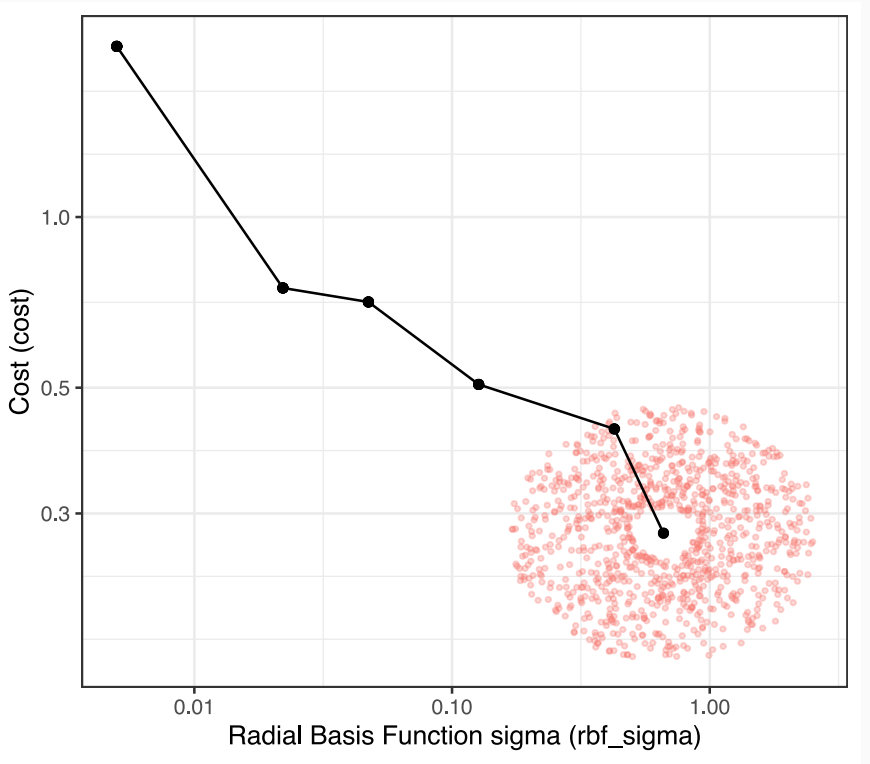
Simulated annealing (SA) is an old search routine that conducts a biased random walk around the tuning parameter space.

- The bias comes from the walk highly favoring steps that show better model results.

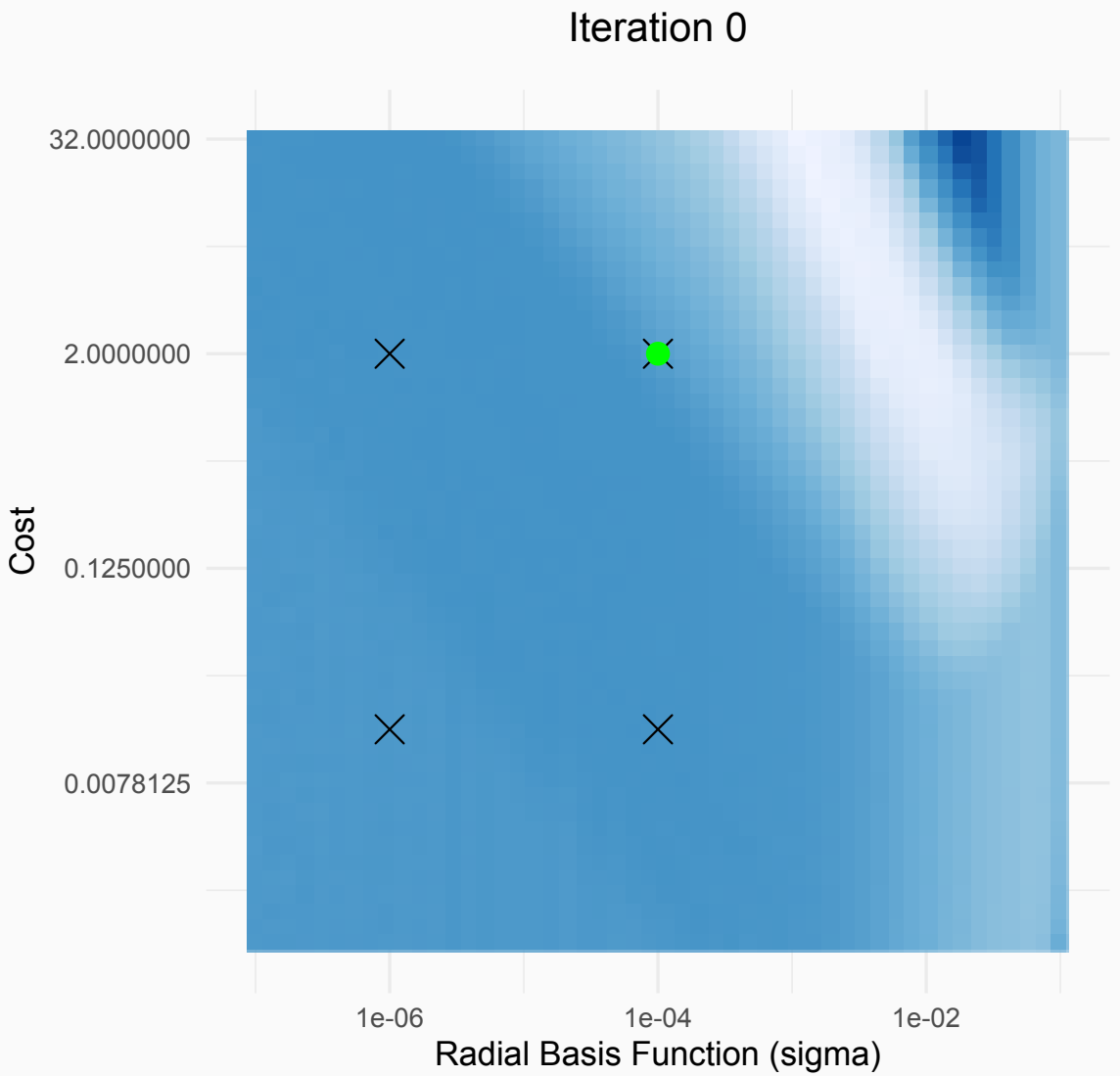
From an initial point, the next point is generated using a random value within a *neighborhood* of the current point.

- An example of candidate points in a neighborhood is shown 👉

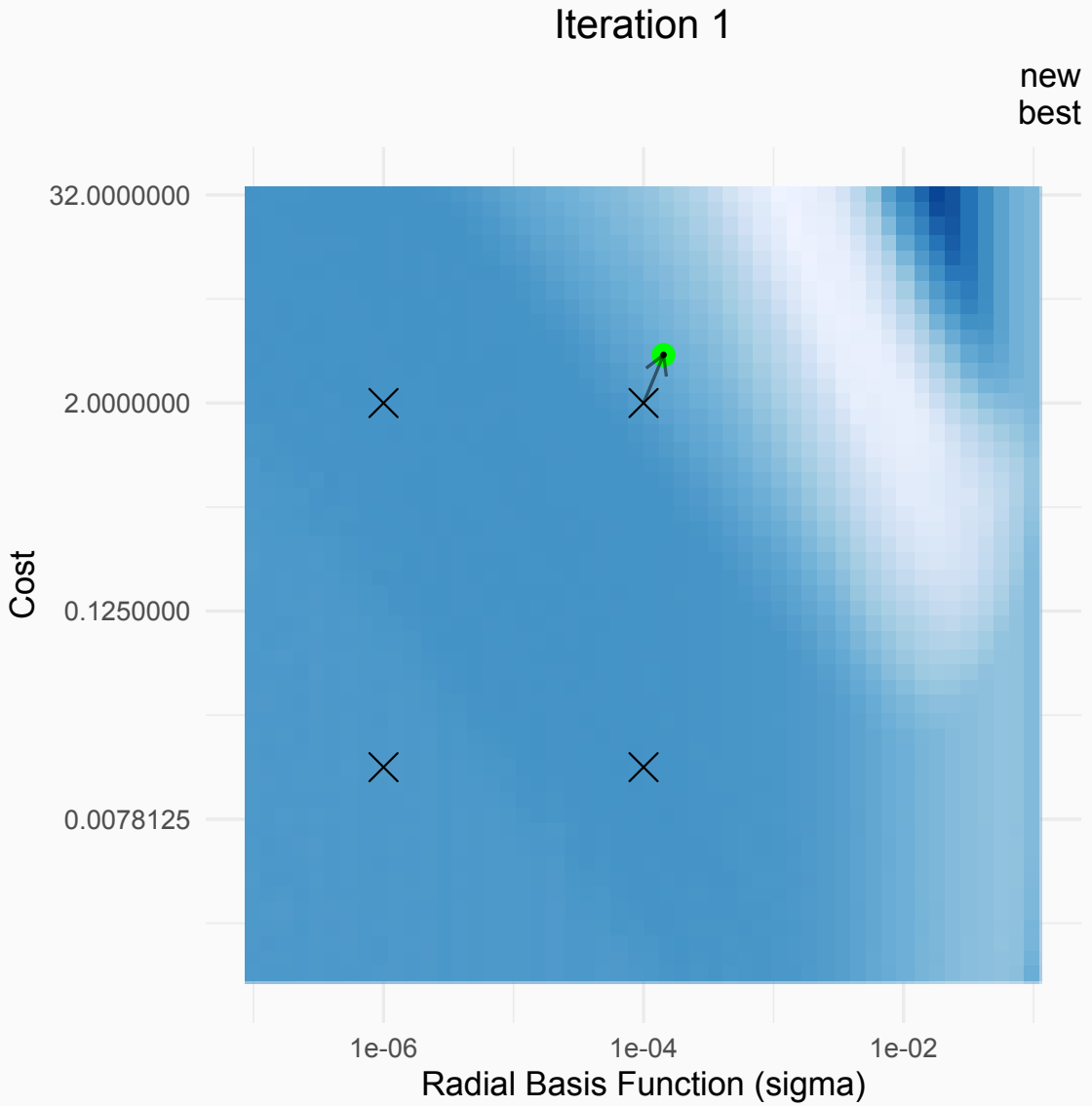
SA accepts suboptimal results early in the search process and less as time goes on.



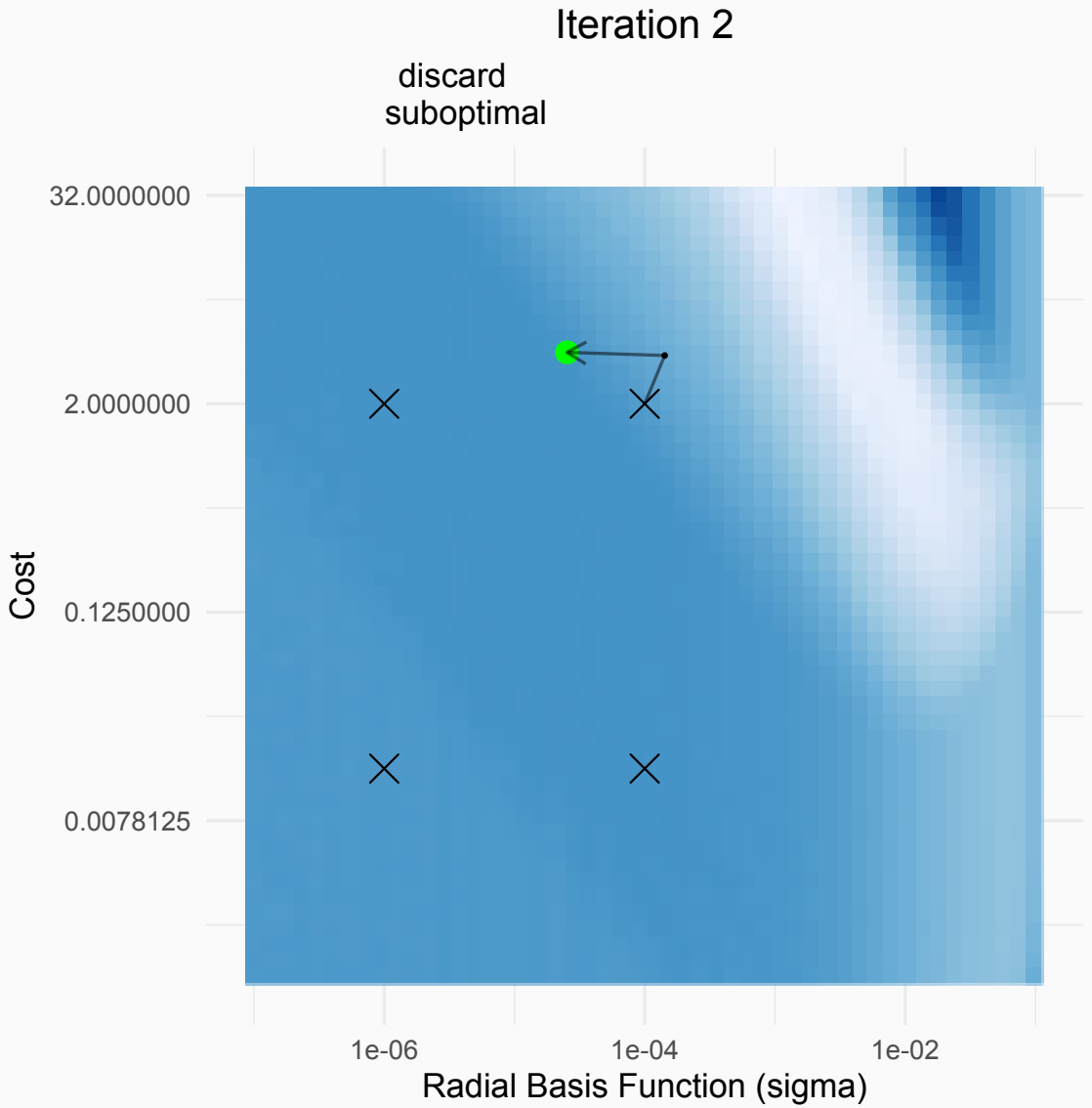
Simulated Annealing Search



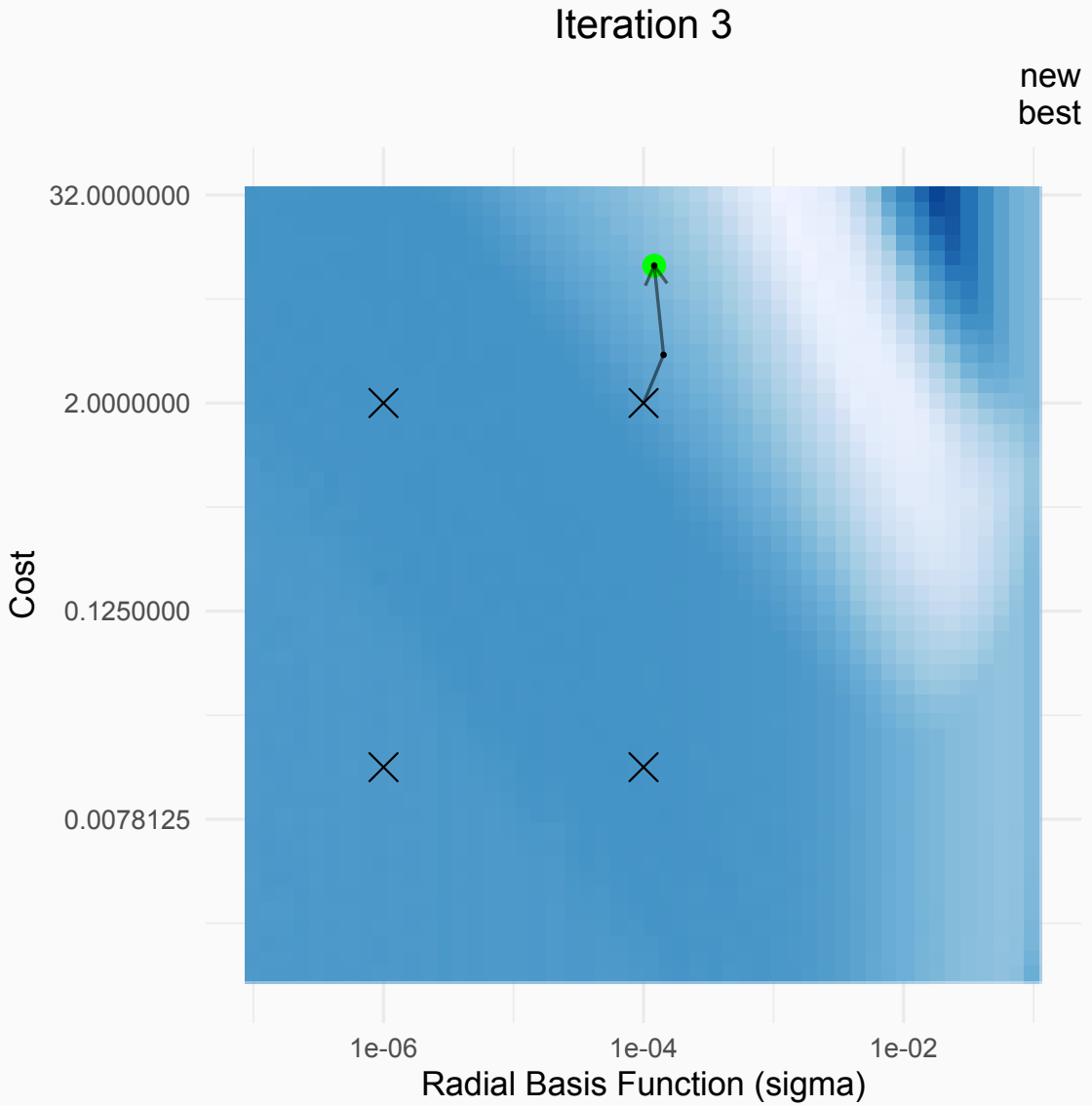
Simulated Annealing Search



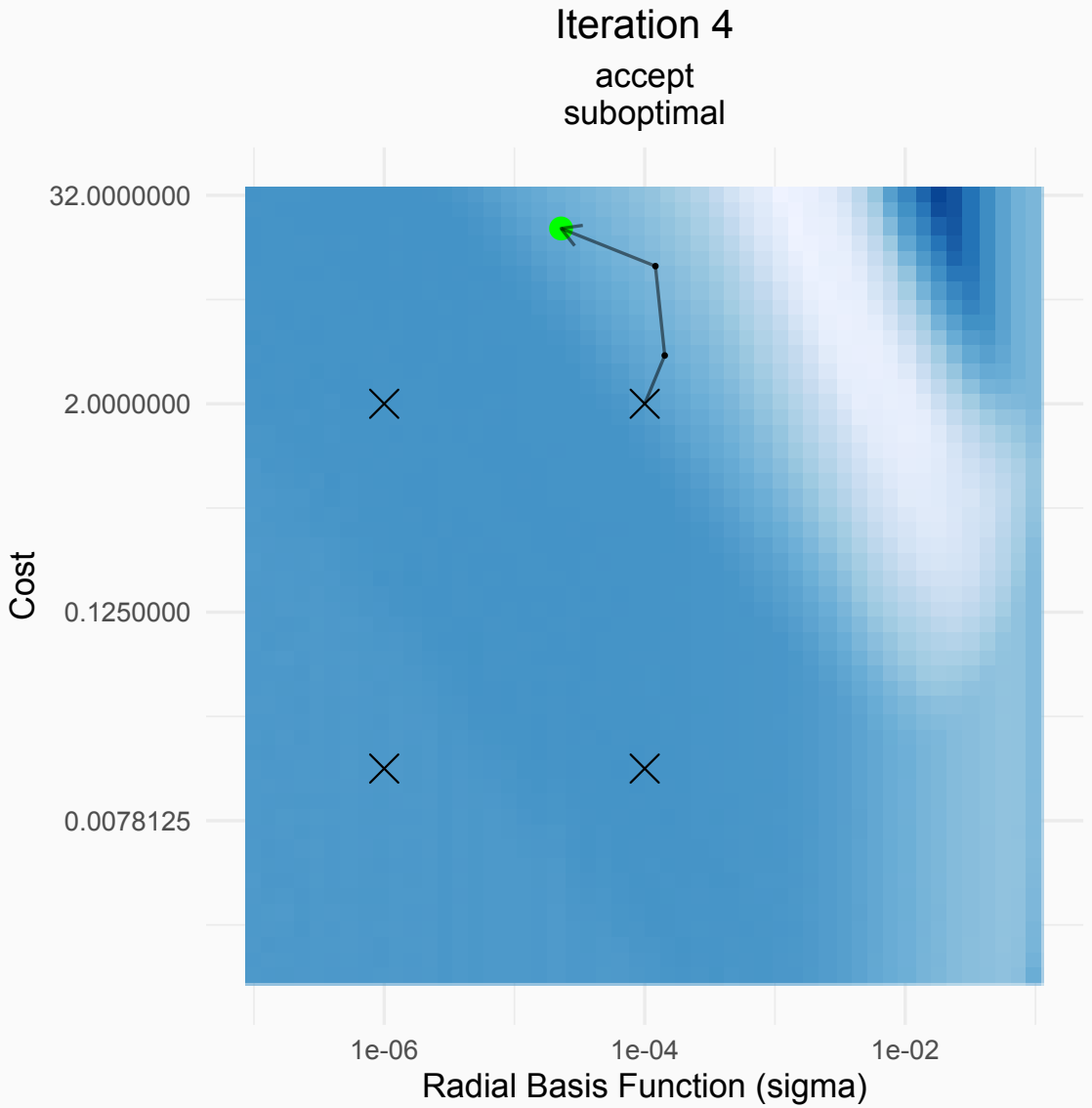
Simulated Annealing Search



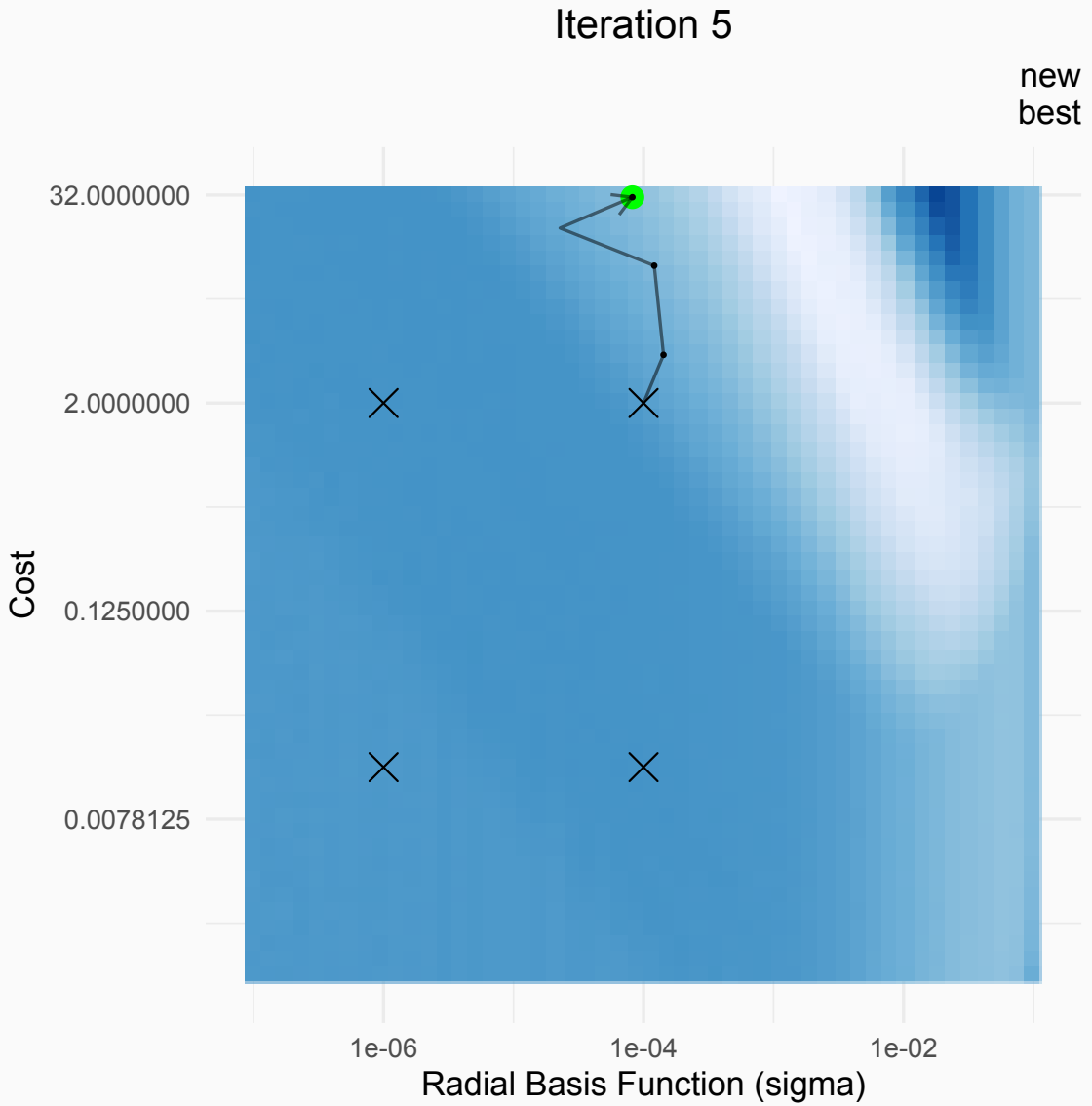
Simulated Annealing Search



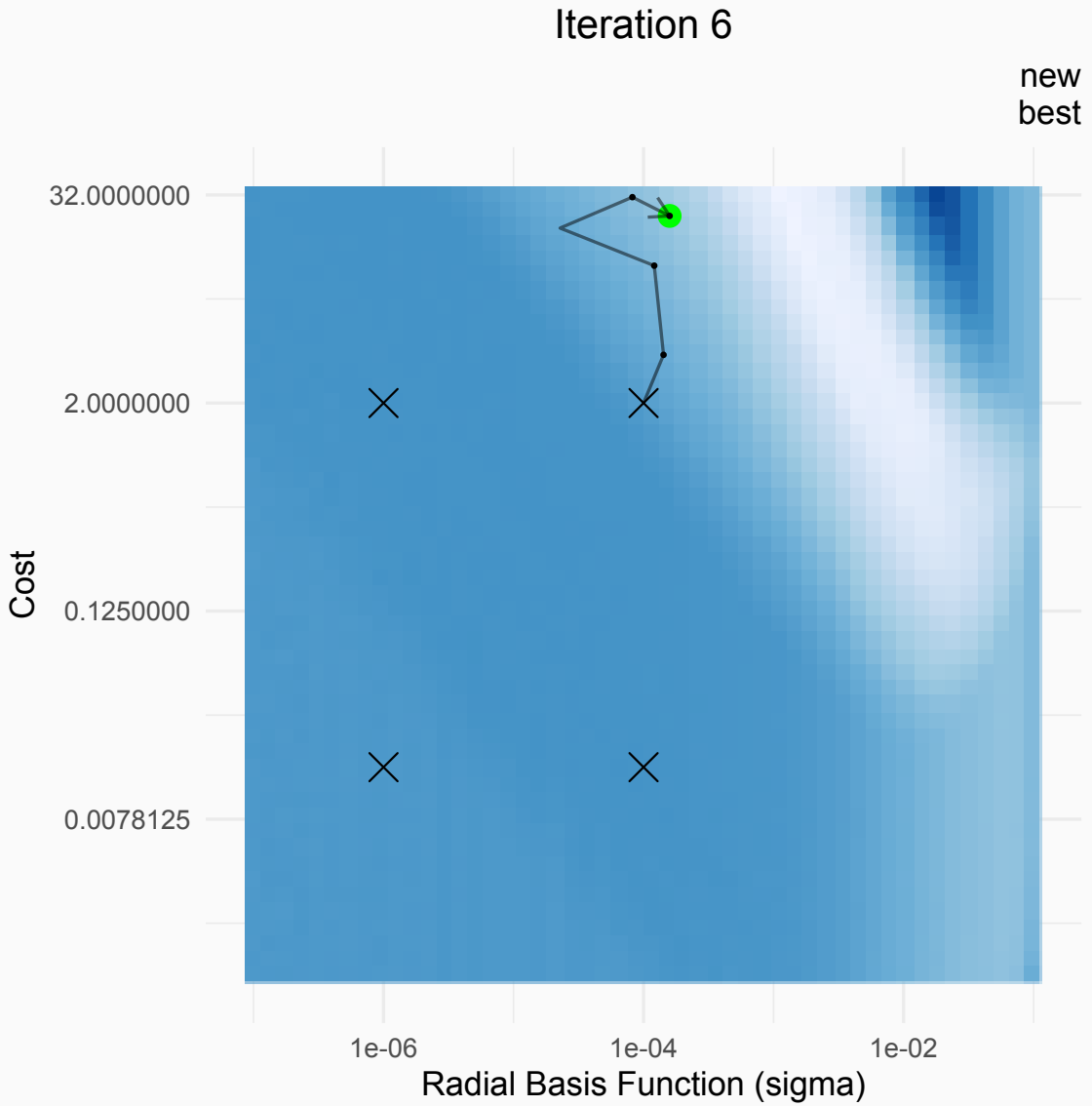
Simulated Annealing Search



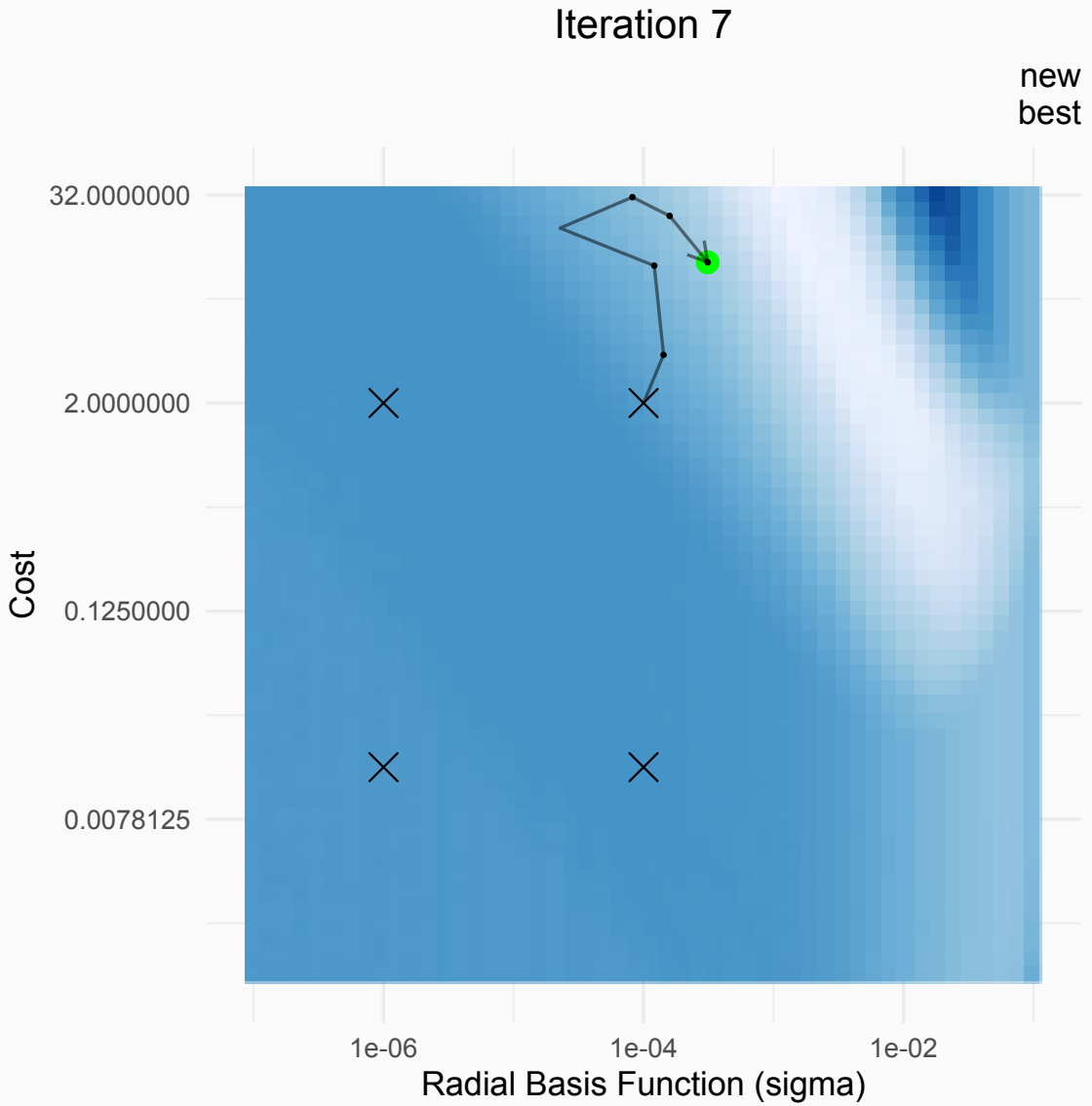
Simulated Annealing Search



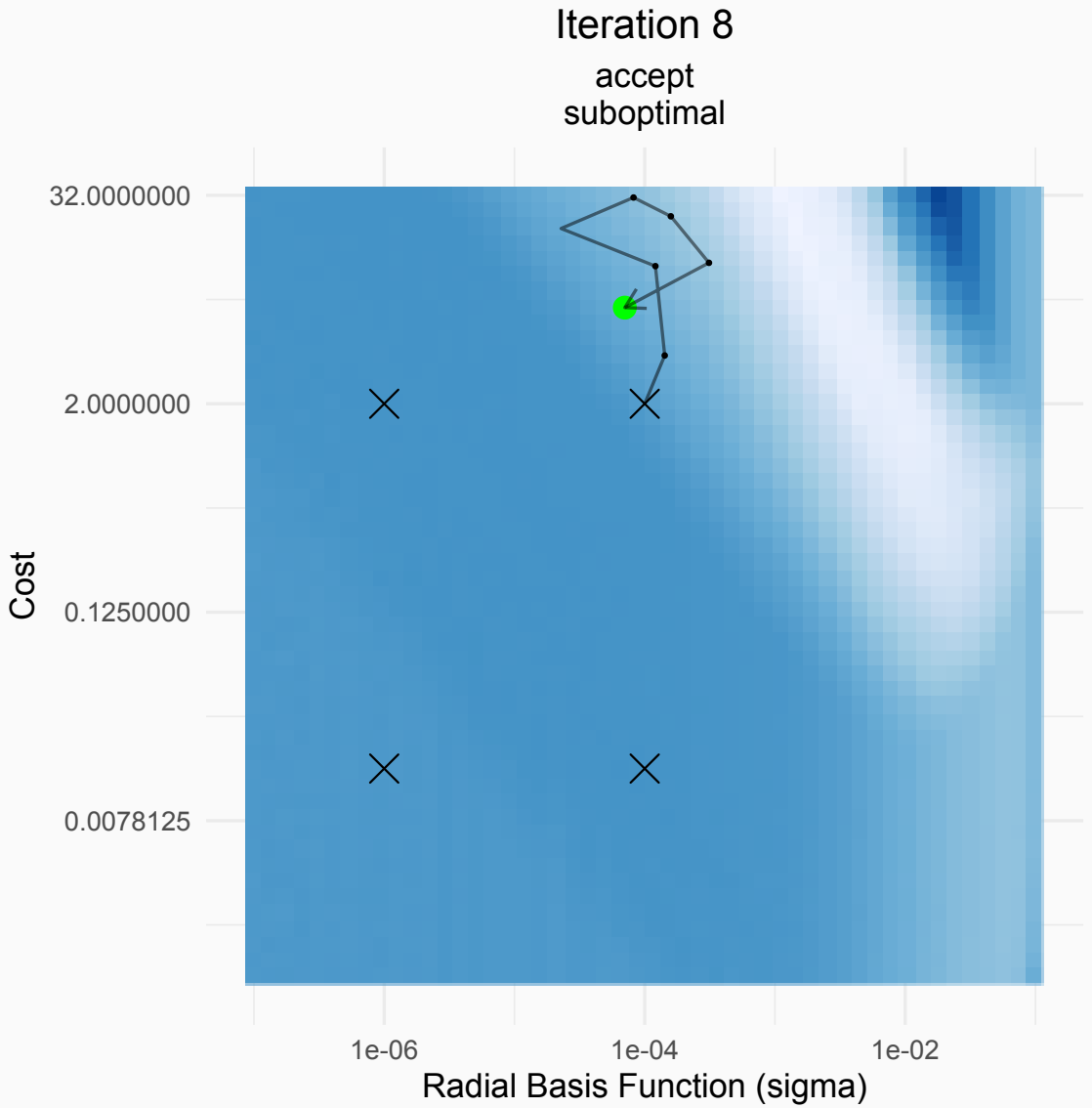
Simulated Annealing Search



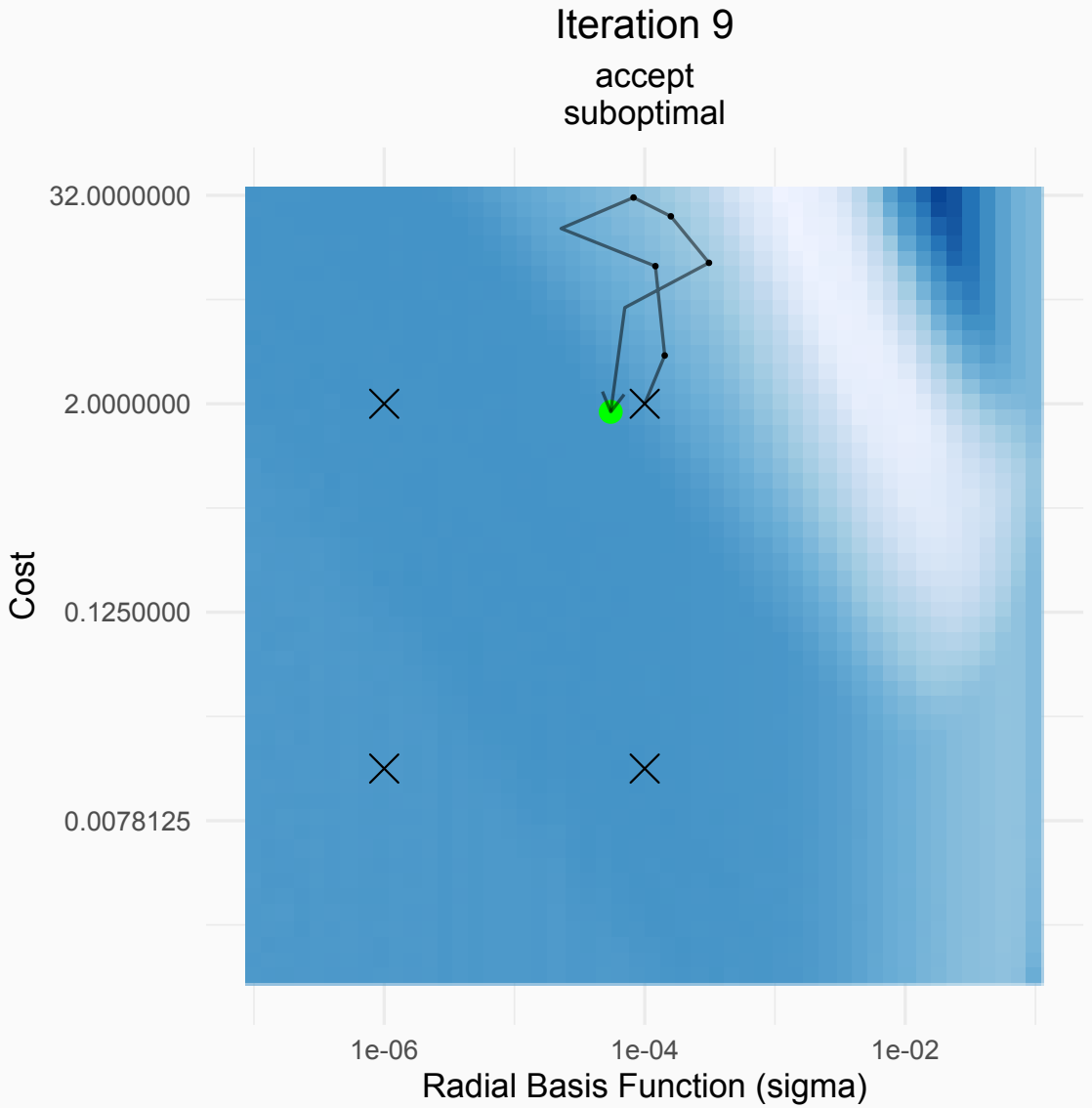
Simulated Annealing Search



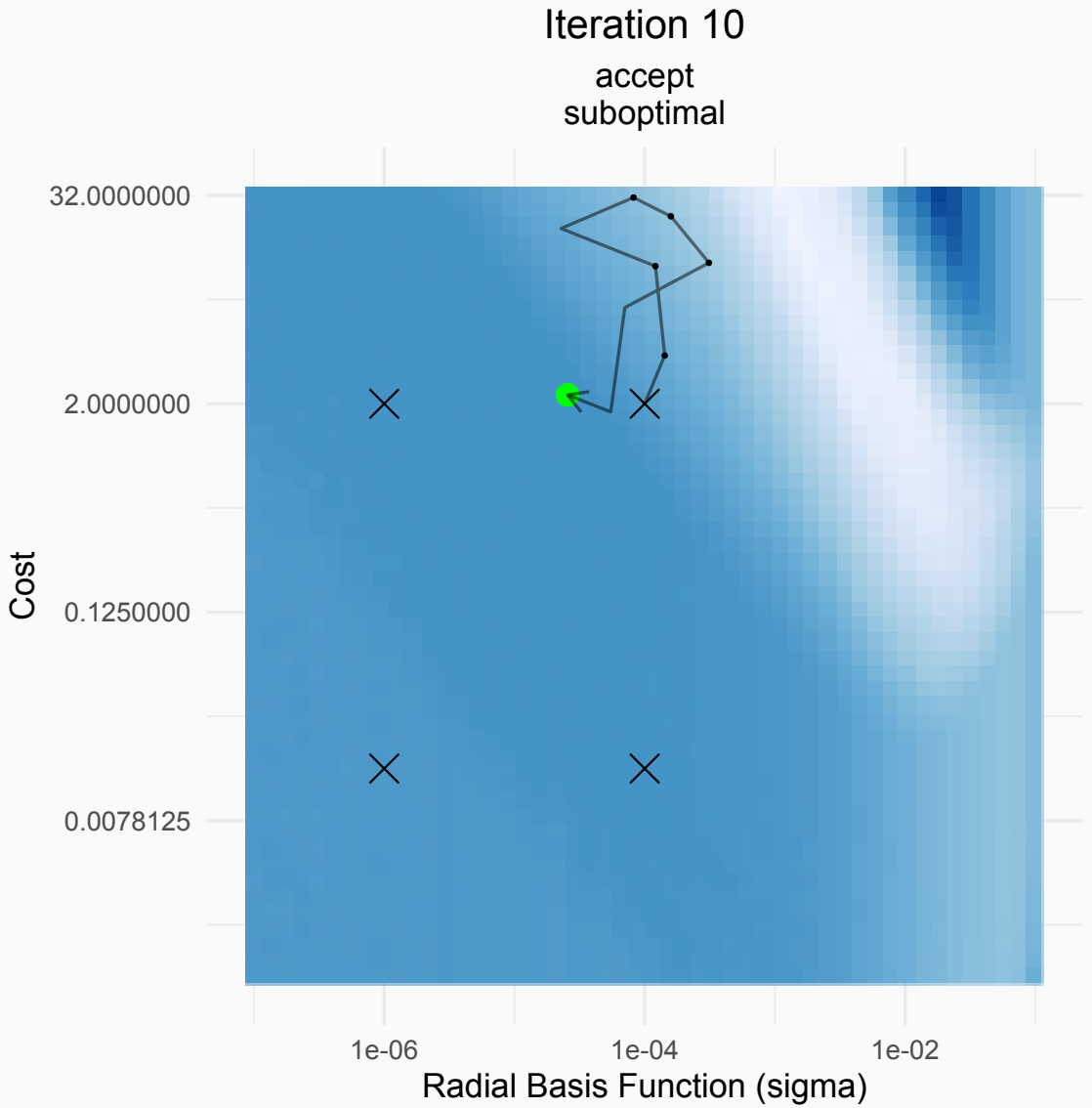
Simulated Annealing Search



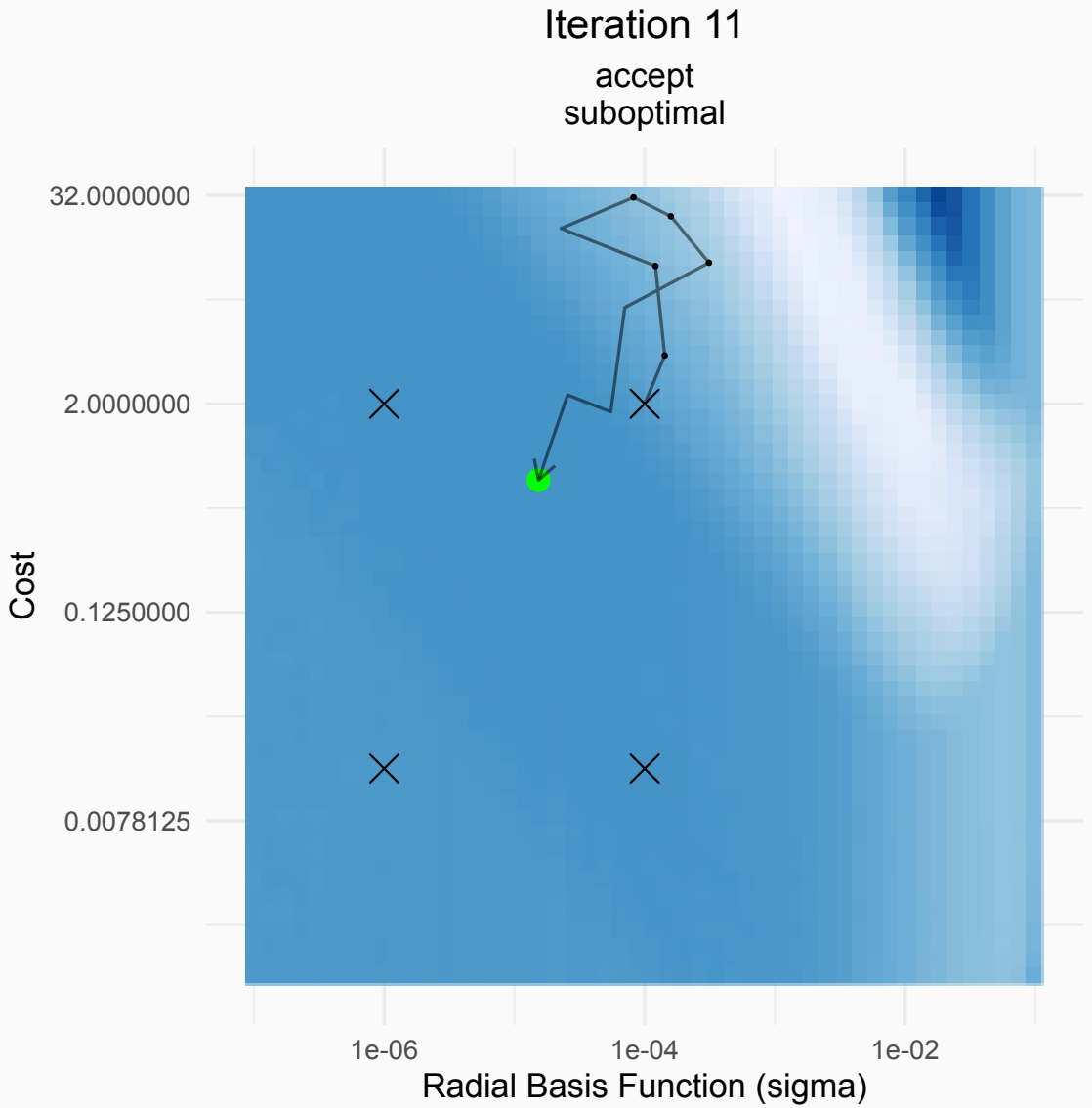
Simulated Annealing Search



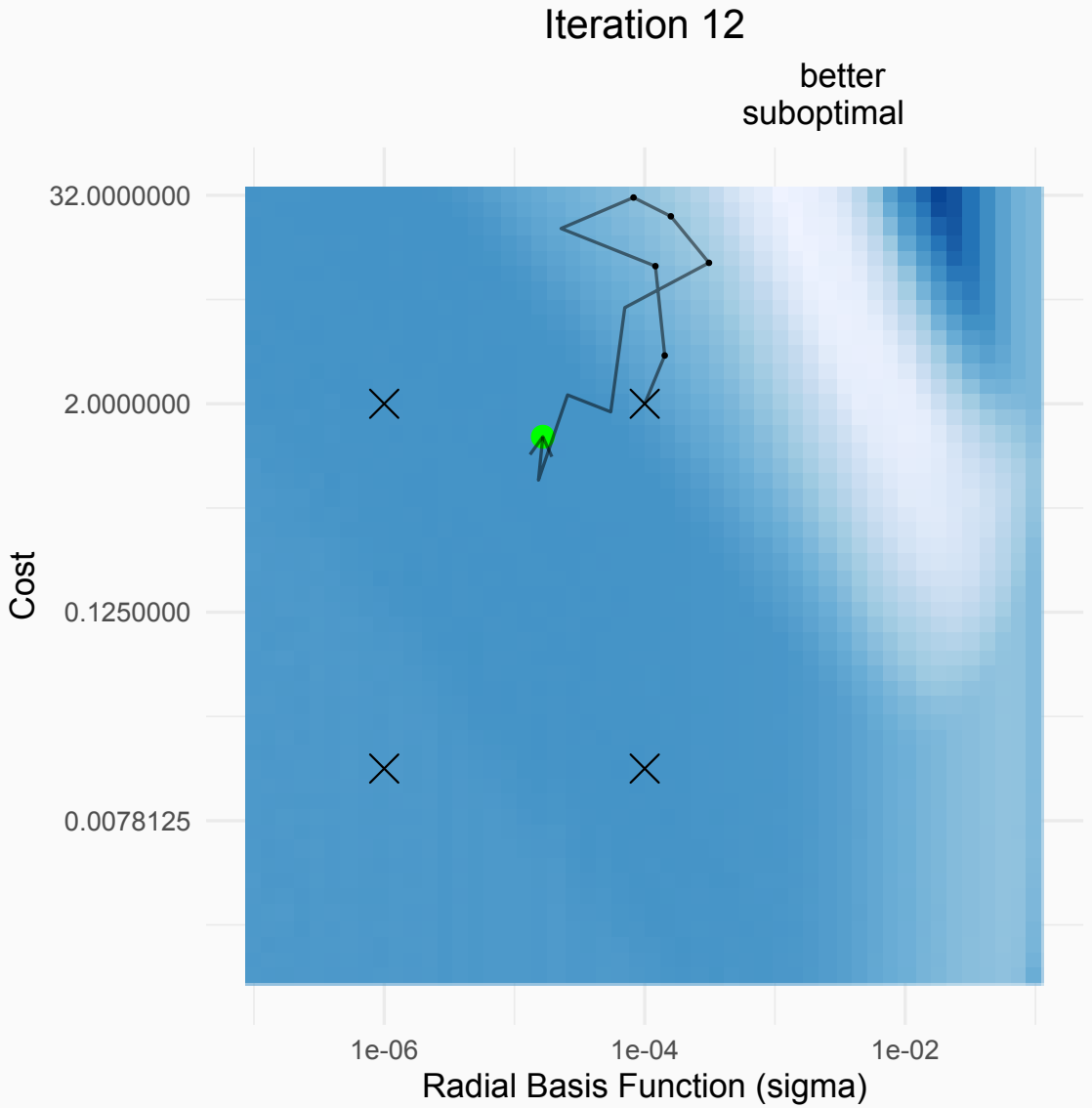
Simulated Annealing Search



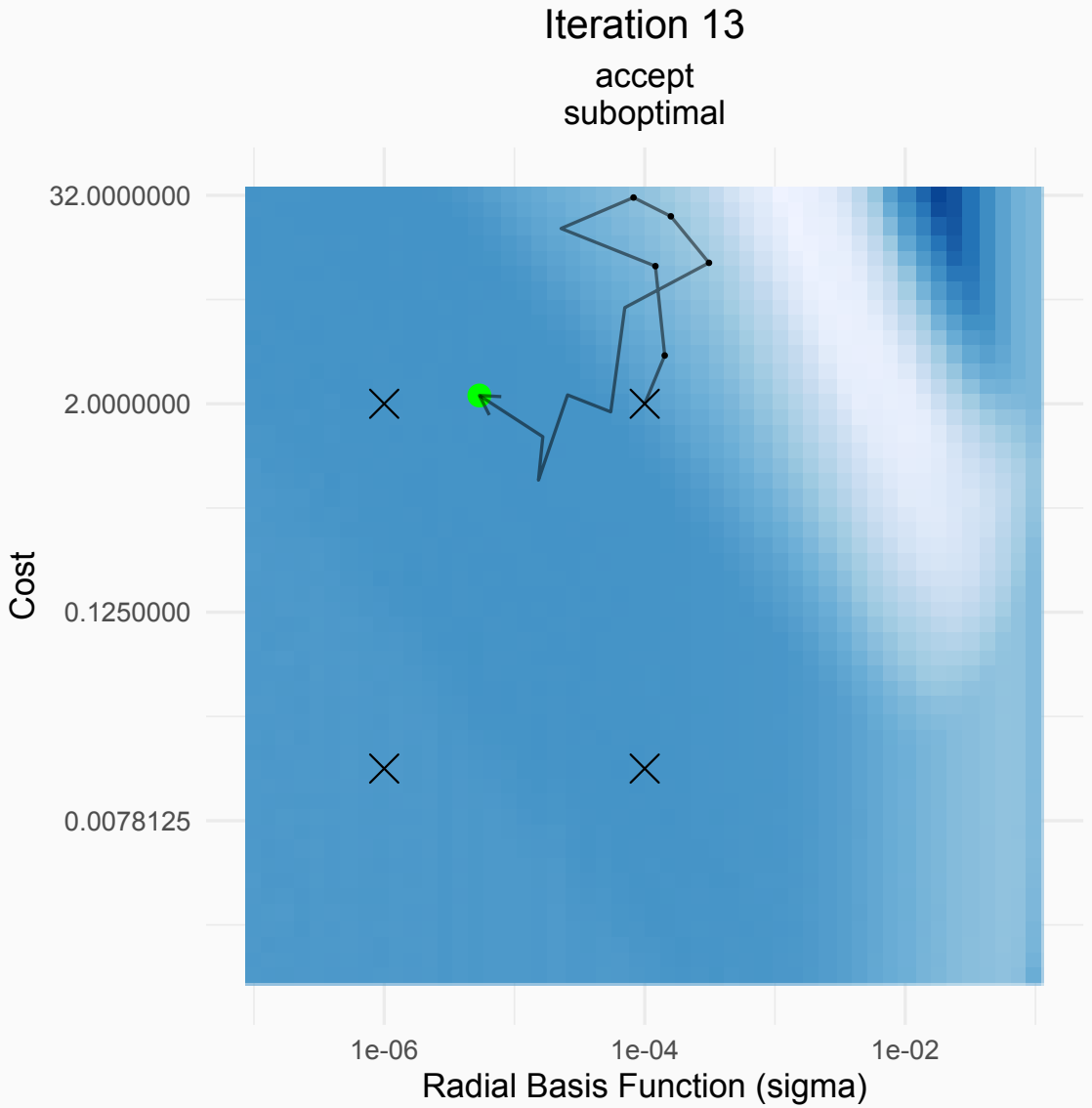
Simulated Annealing Search



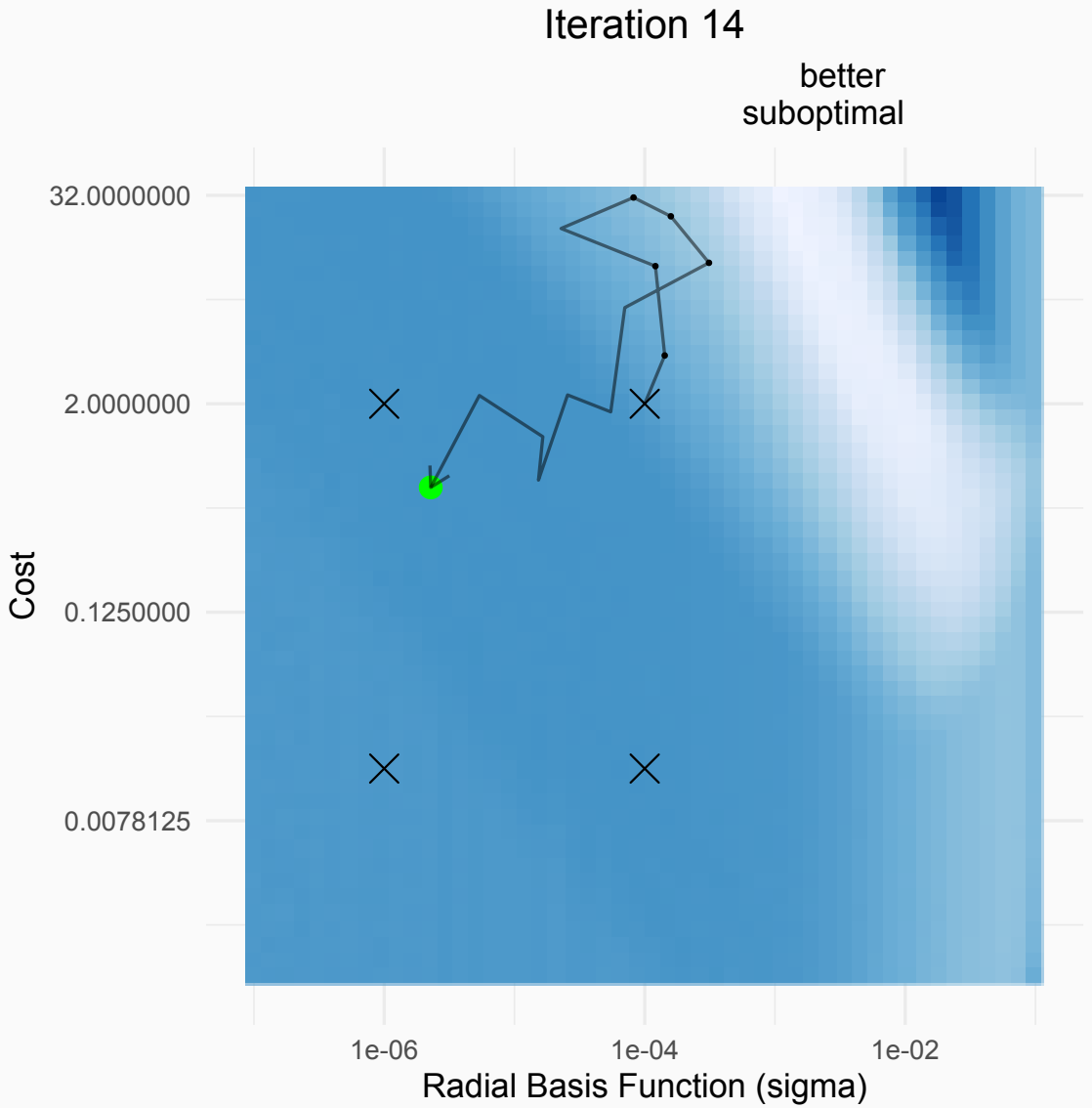
Simulated Annealing Search



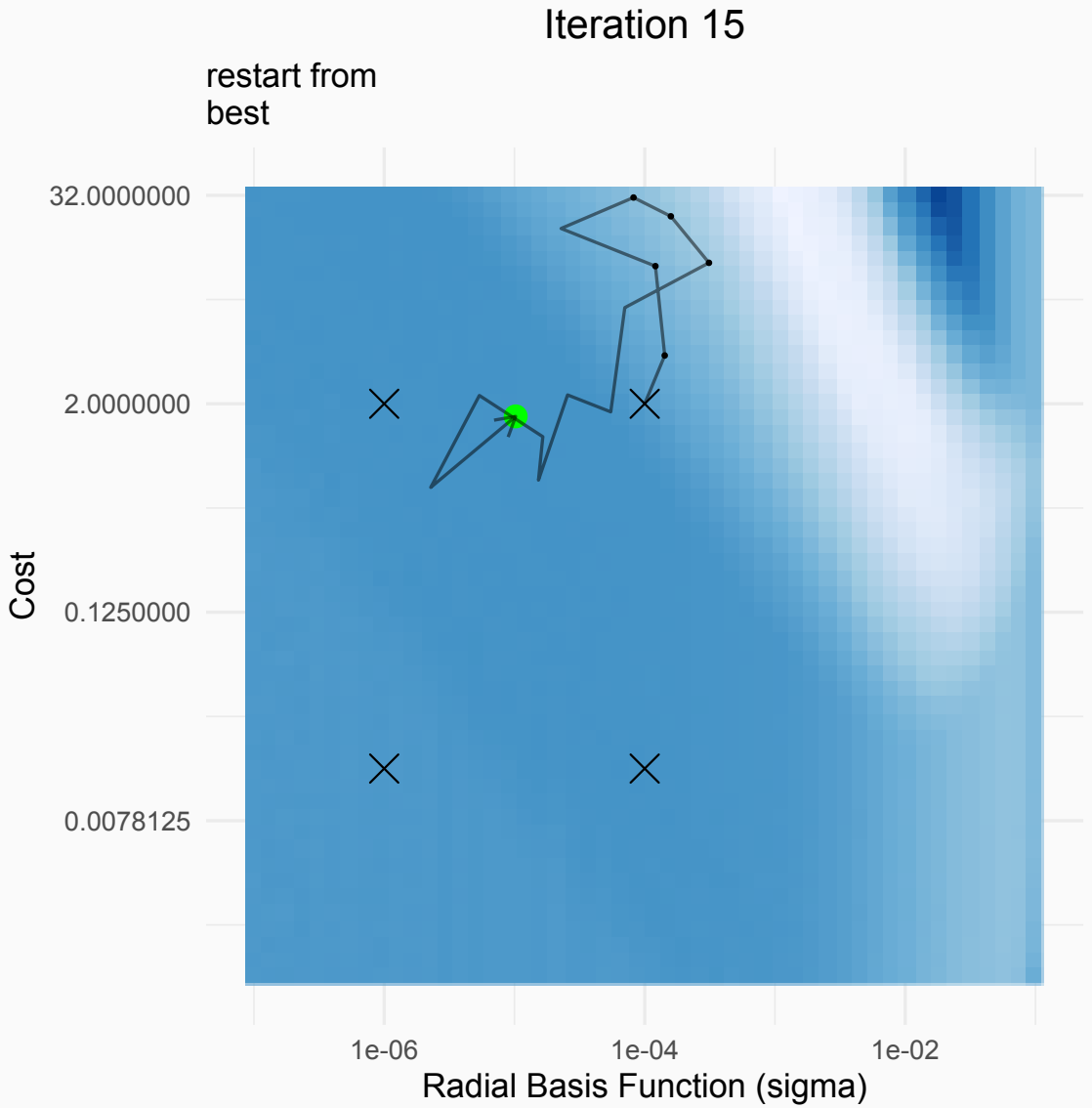
Simulated Annealing Search



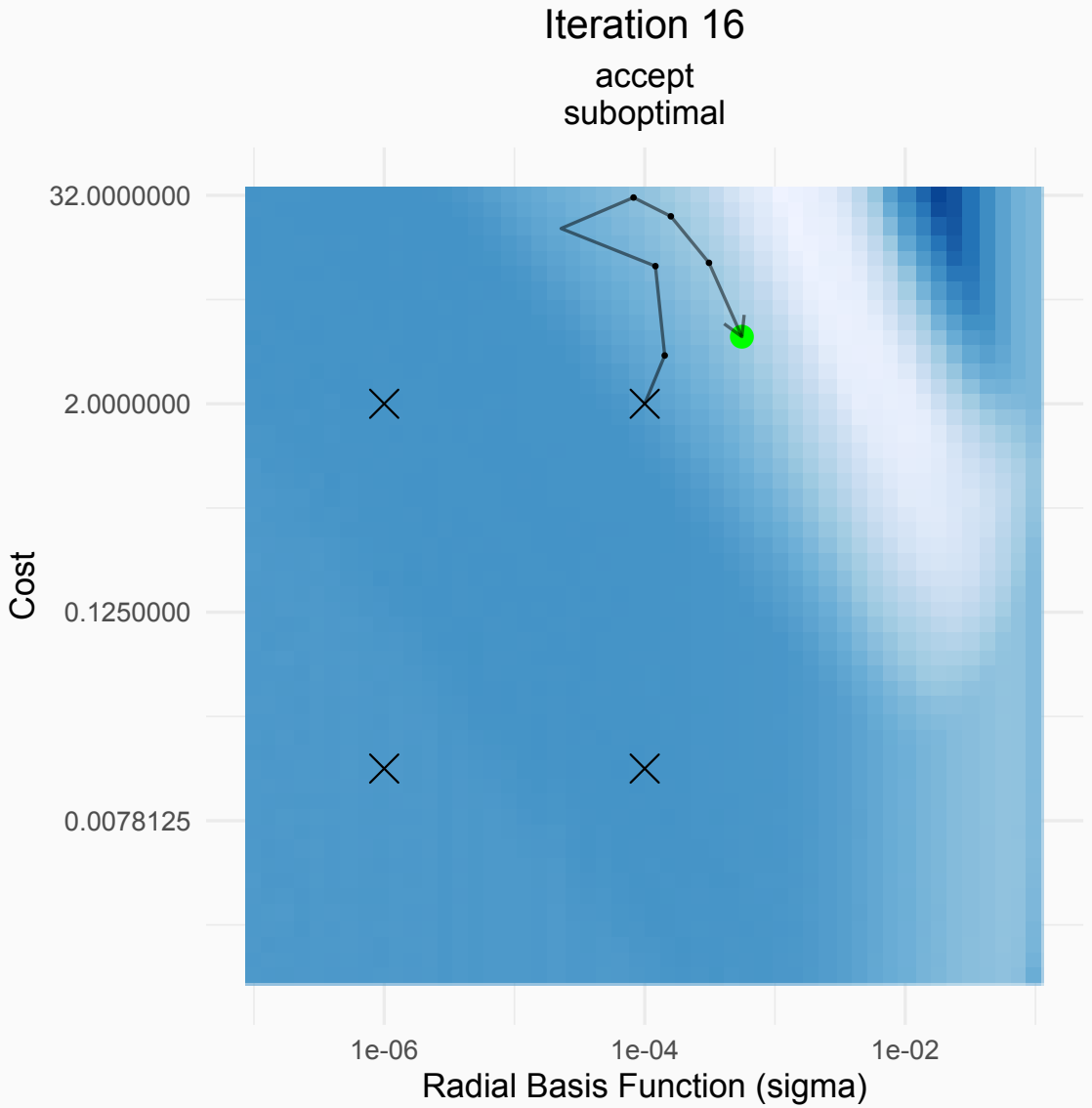
Simulated Annealing Search



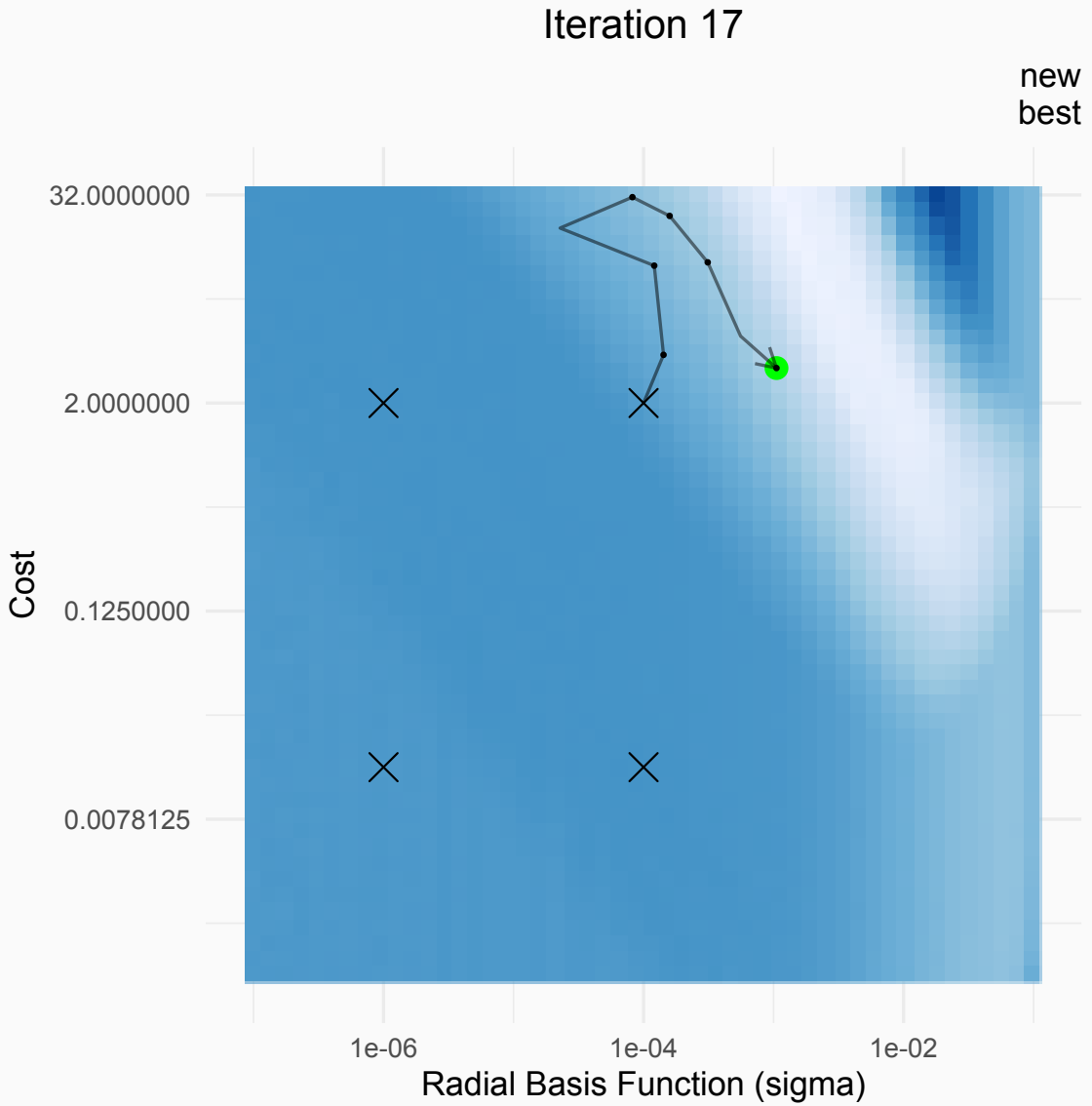
Simulated Annealing Search



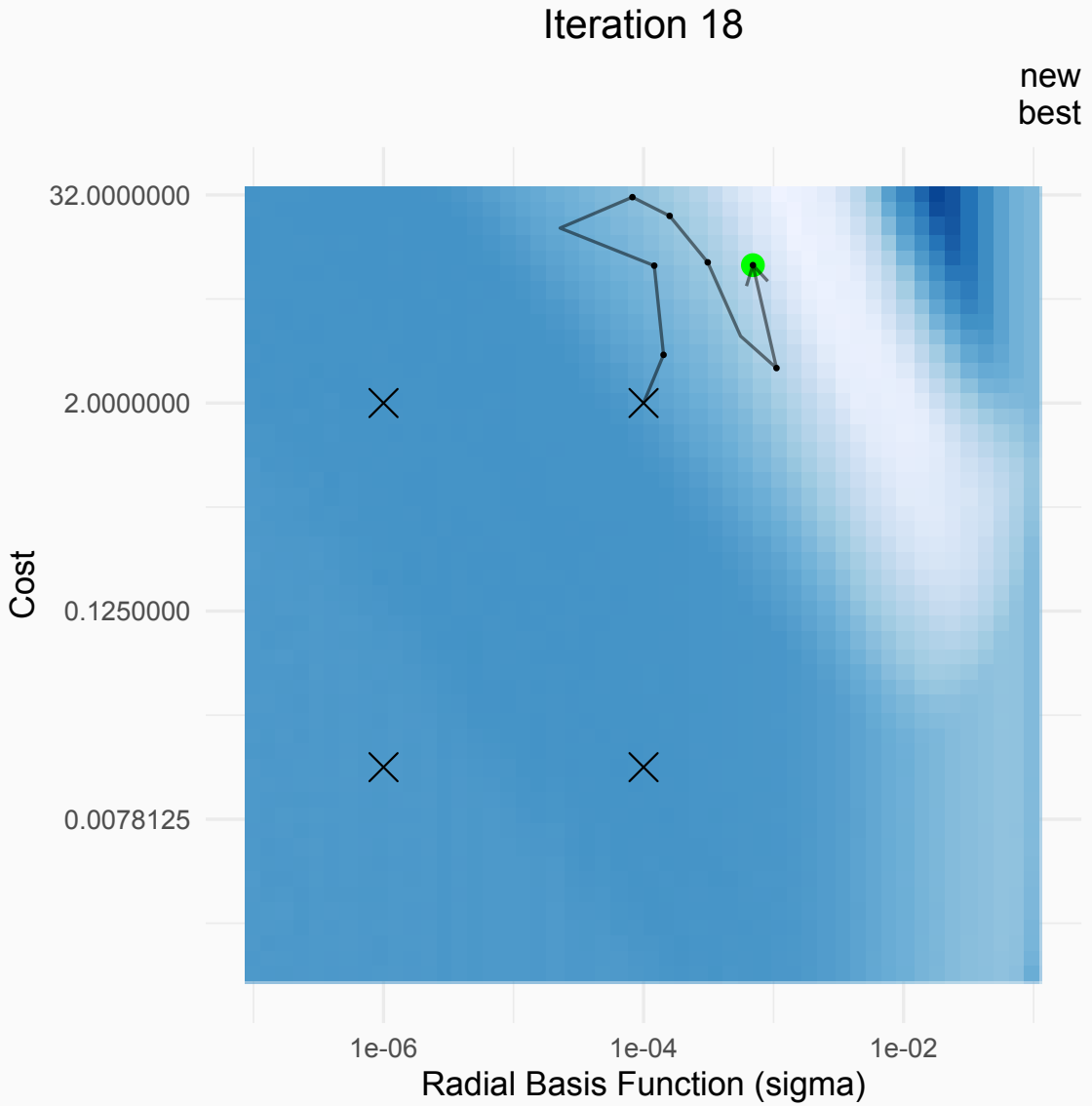
Simulated Annealing Search



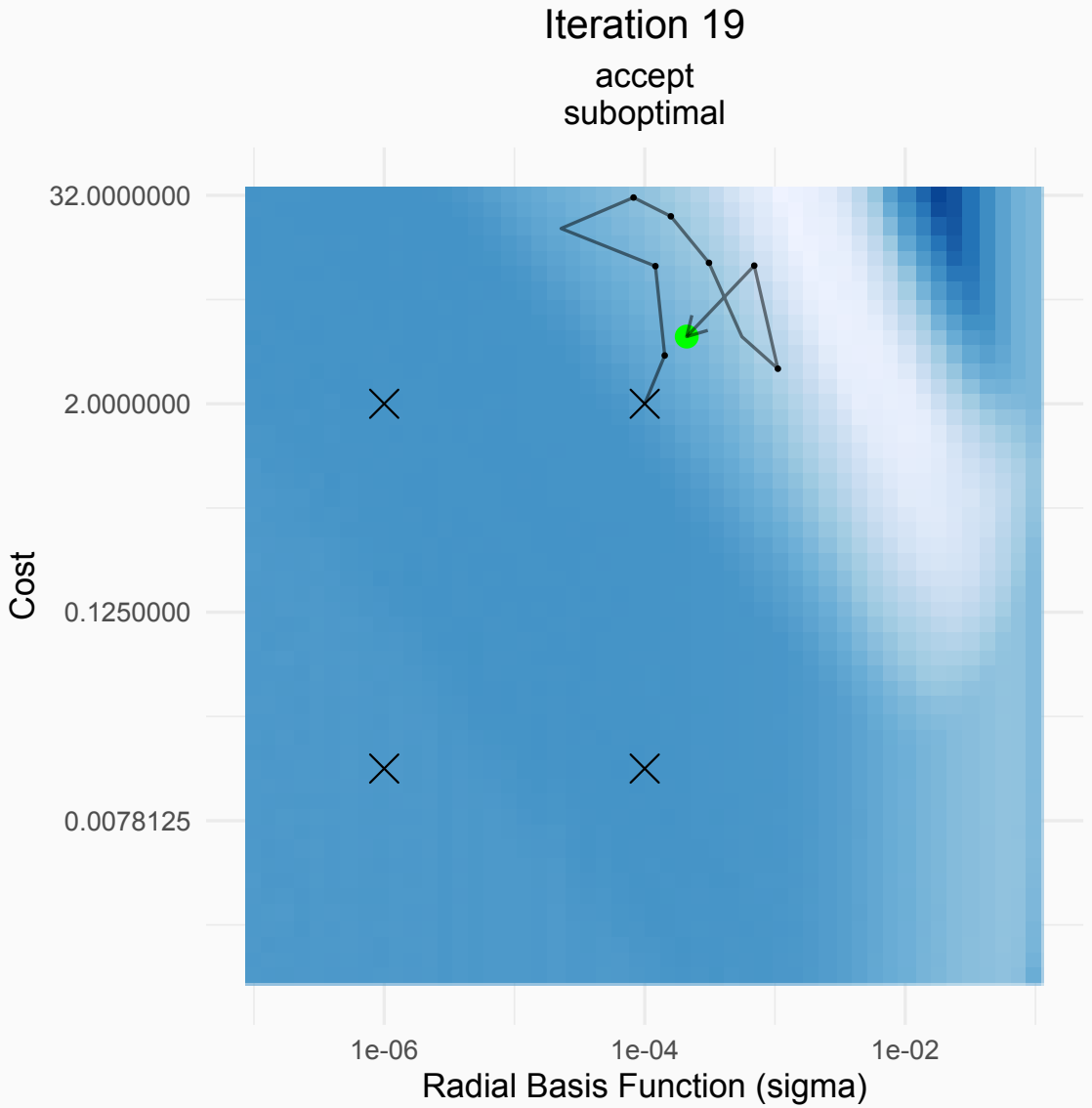
Simulated Annealing Search



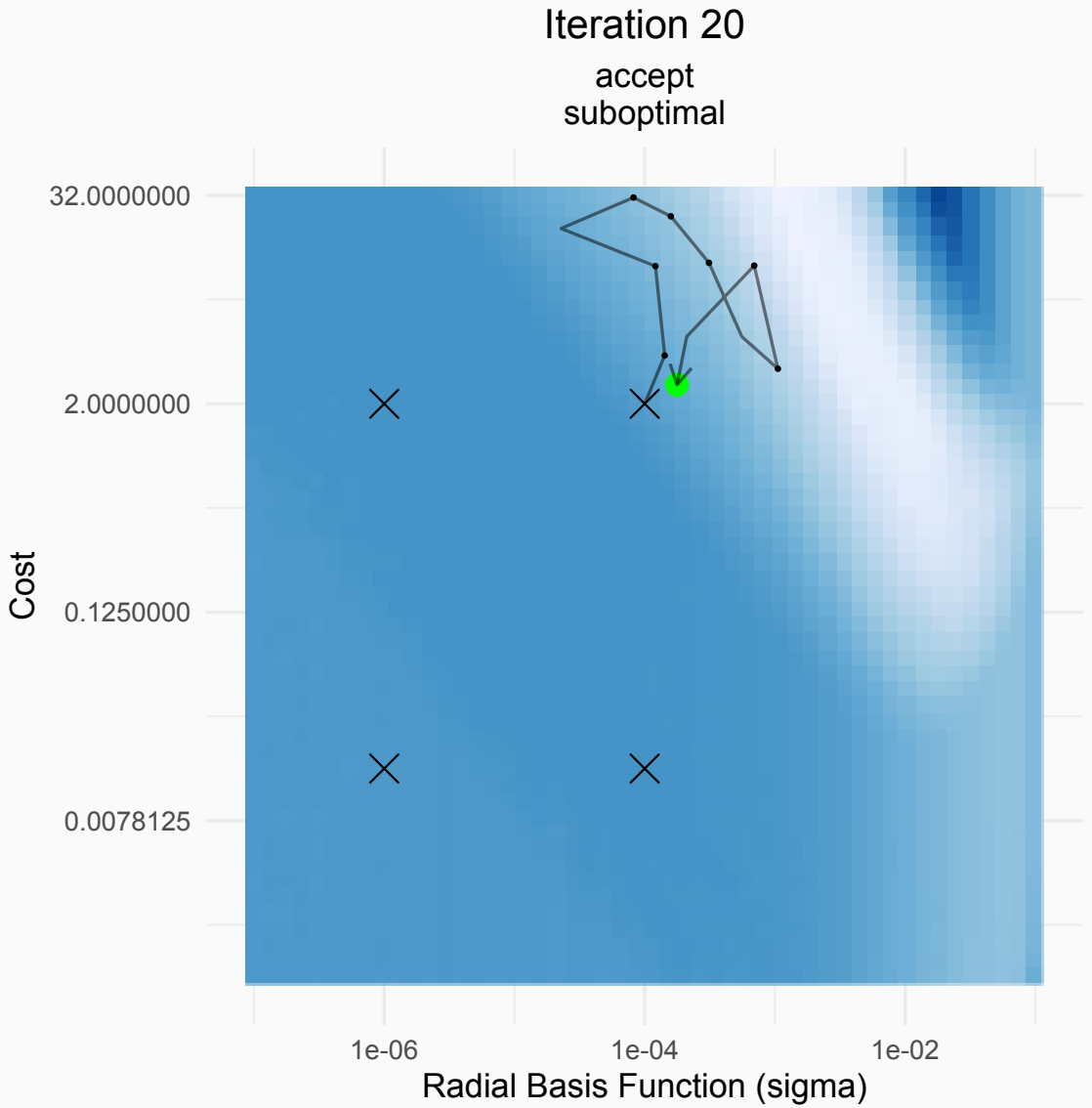
Simulated Annealing Search



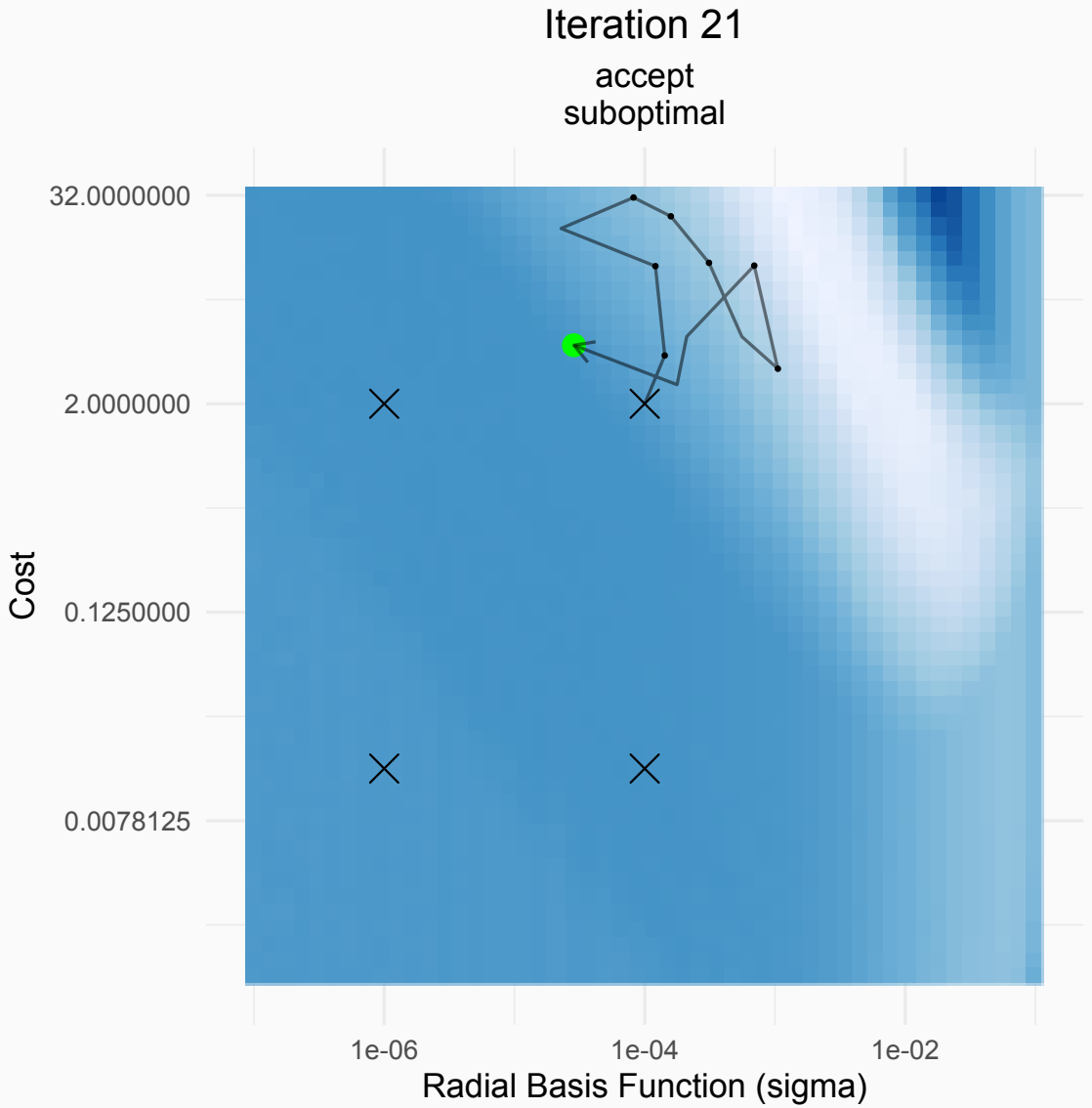
Simulated Annealing Search



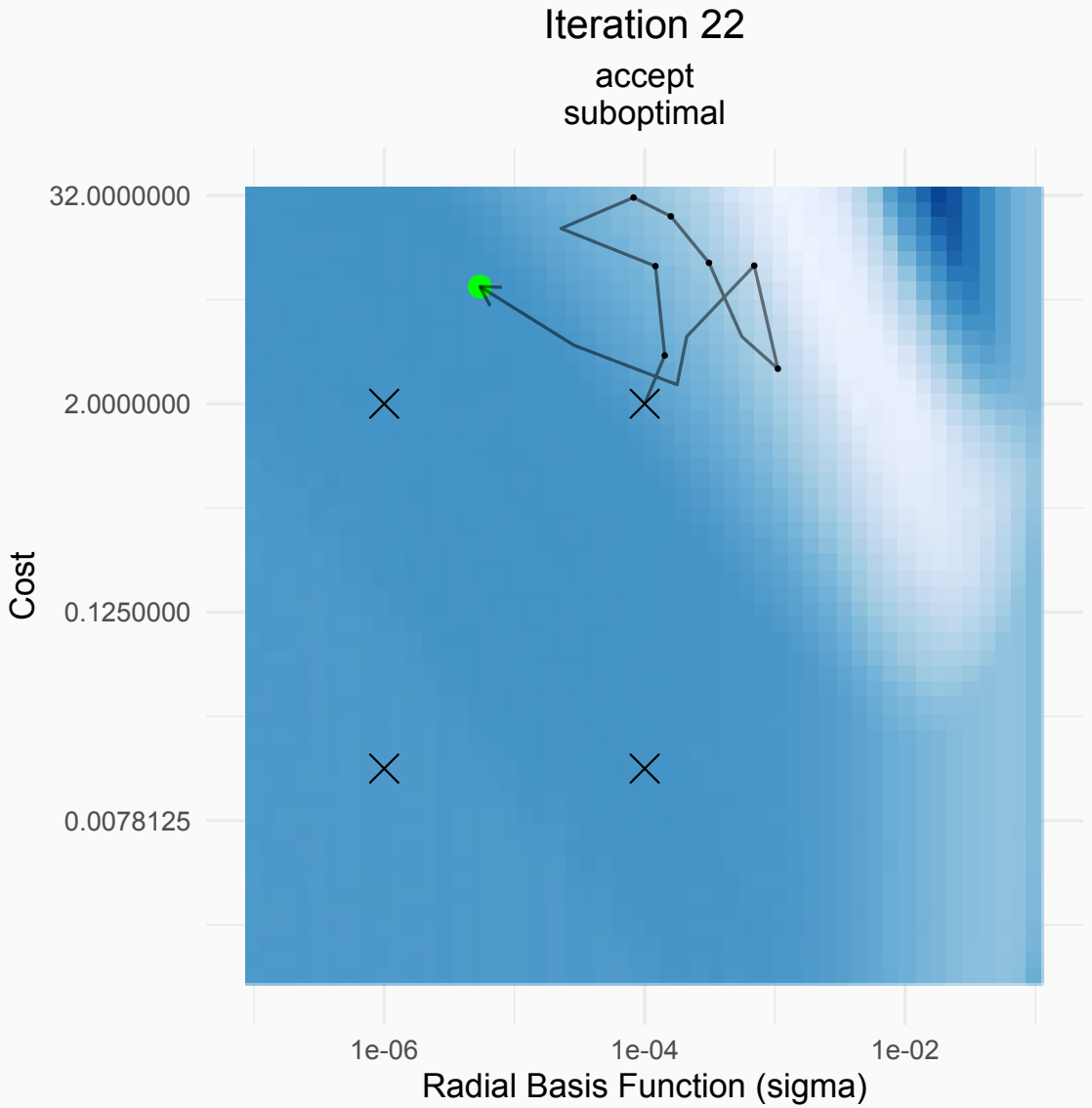
Simulated Annealing Search



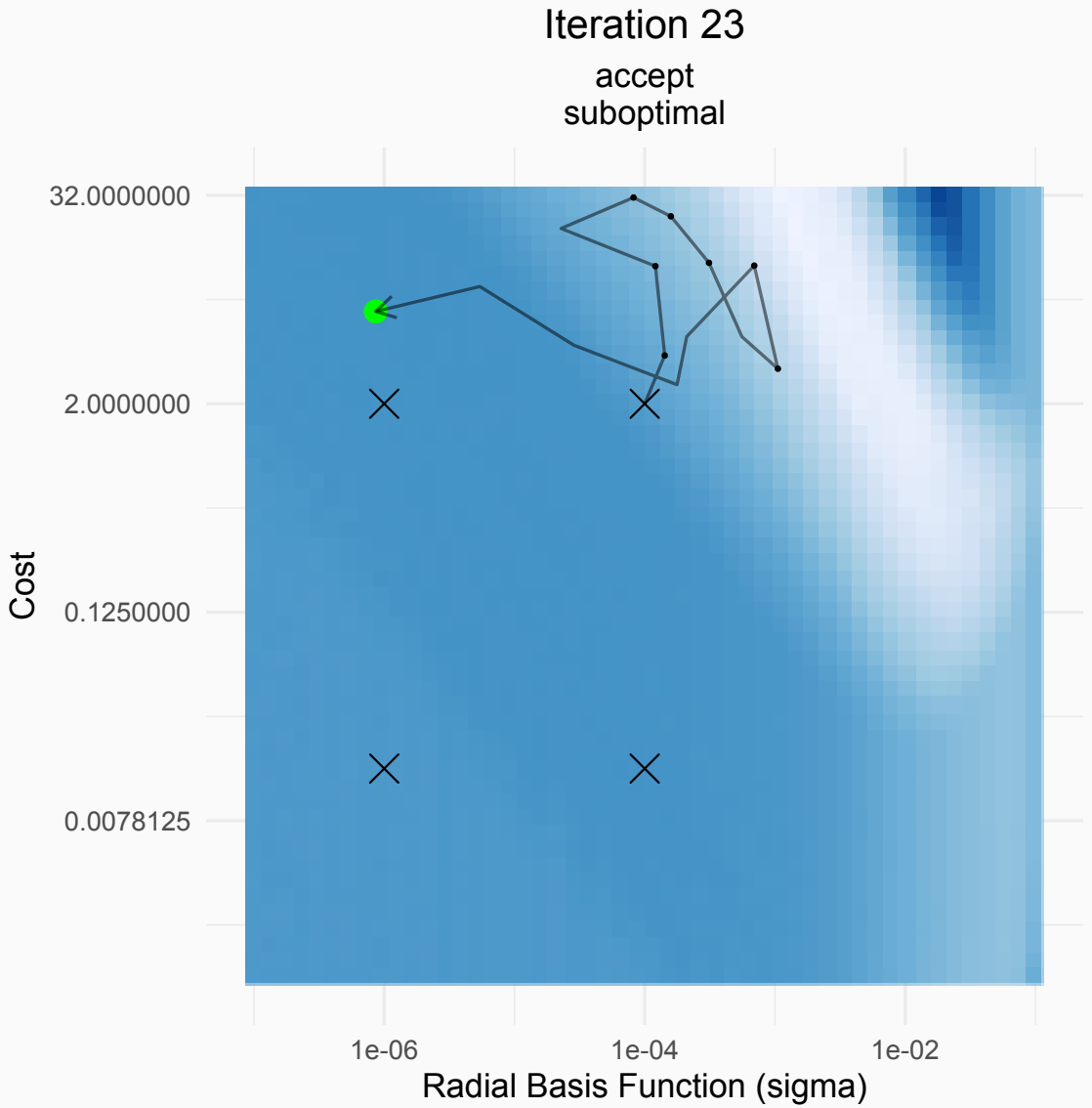
Simulated Annealing Search



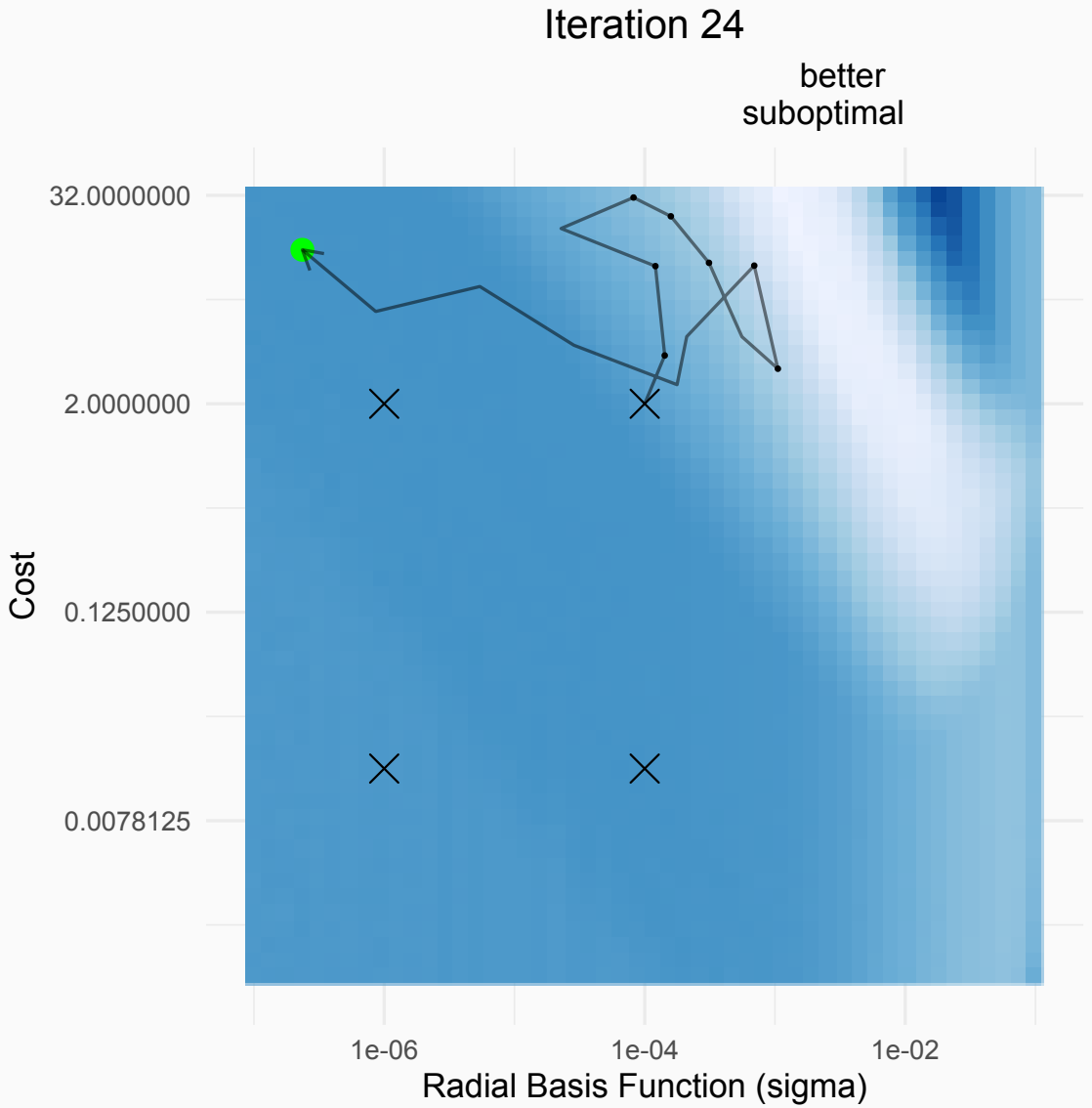
Simulated Annealing Search



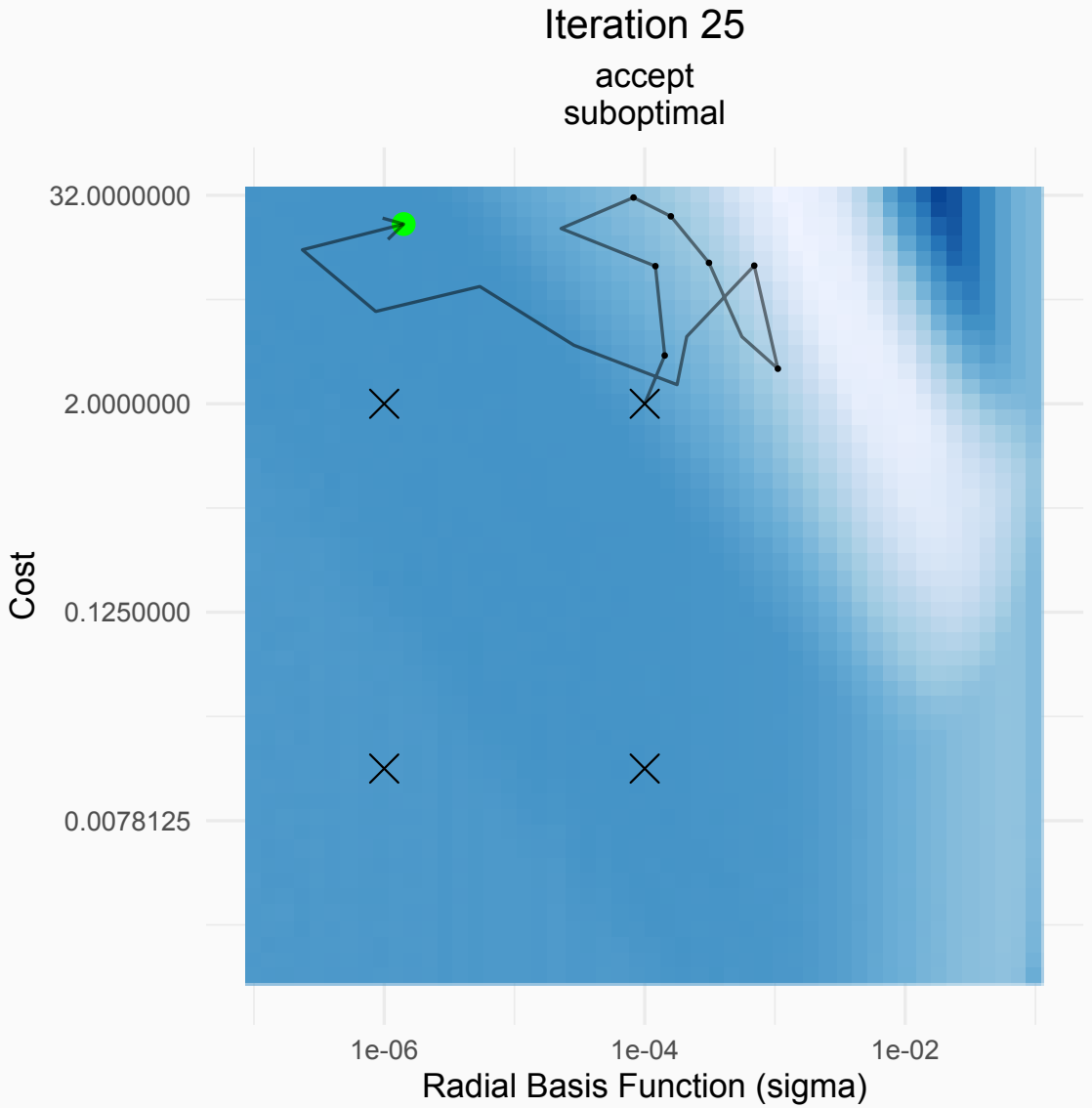
Simulated Annealing Search



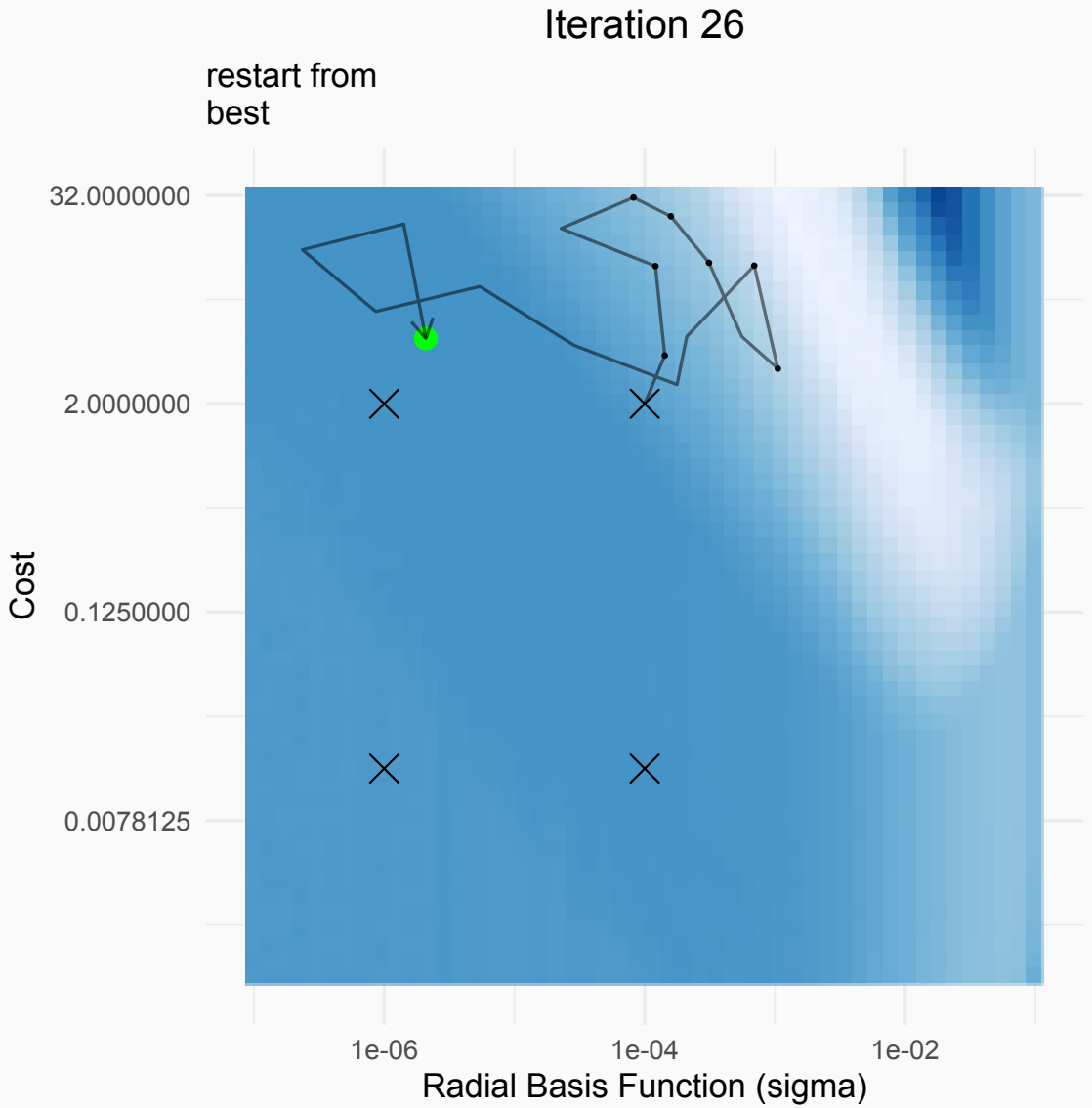
Simulated Annealing Search



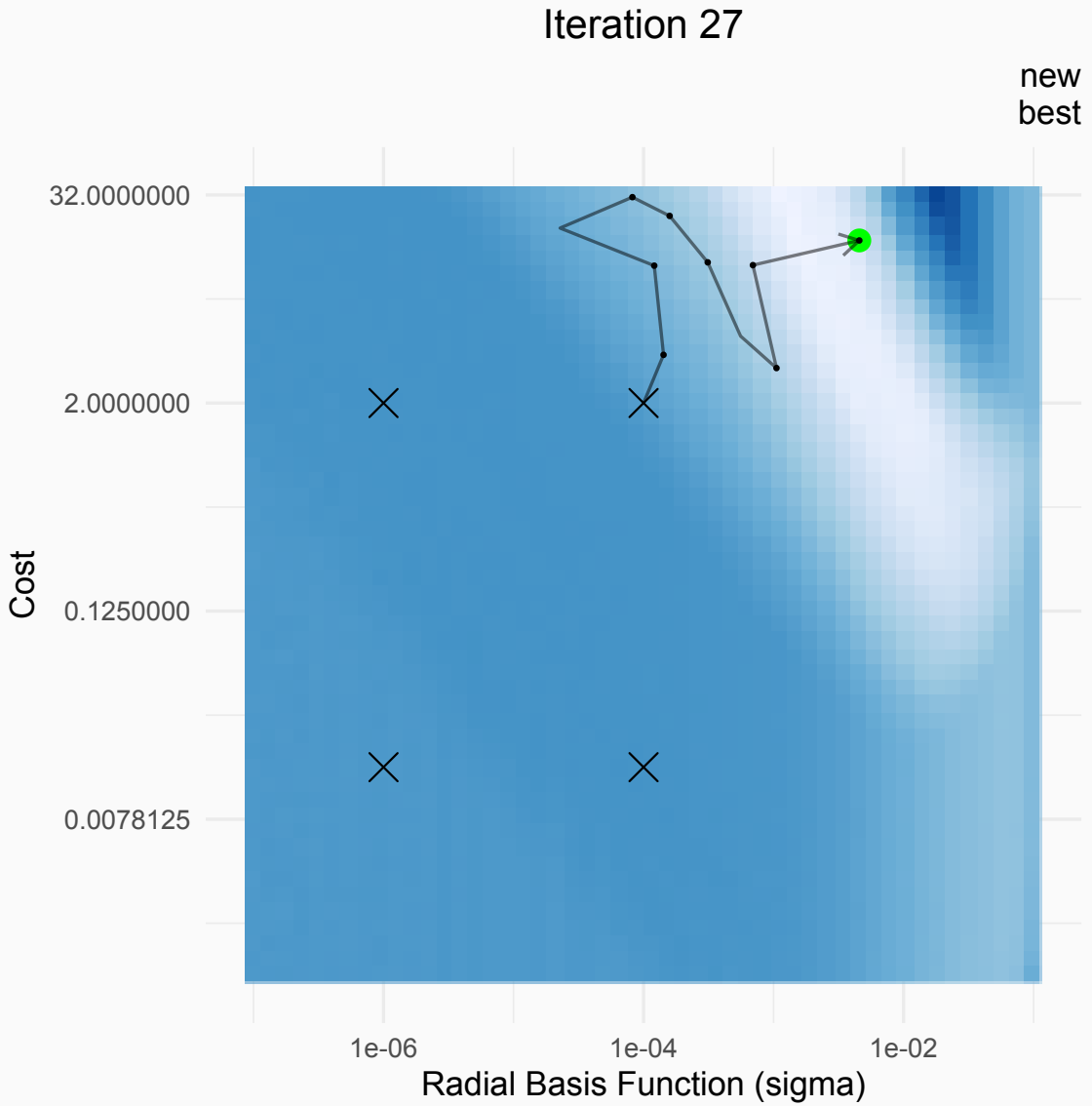
Simulated Annealing Search



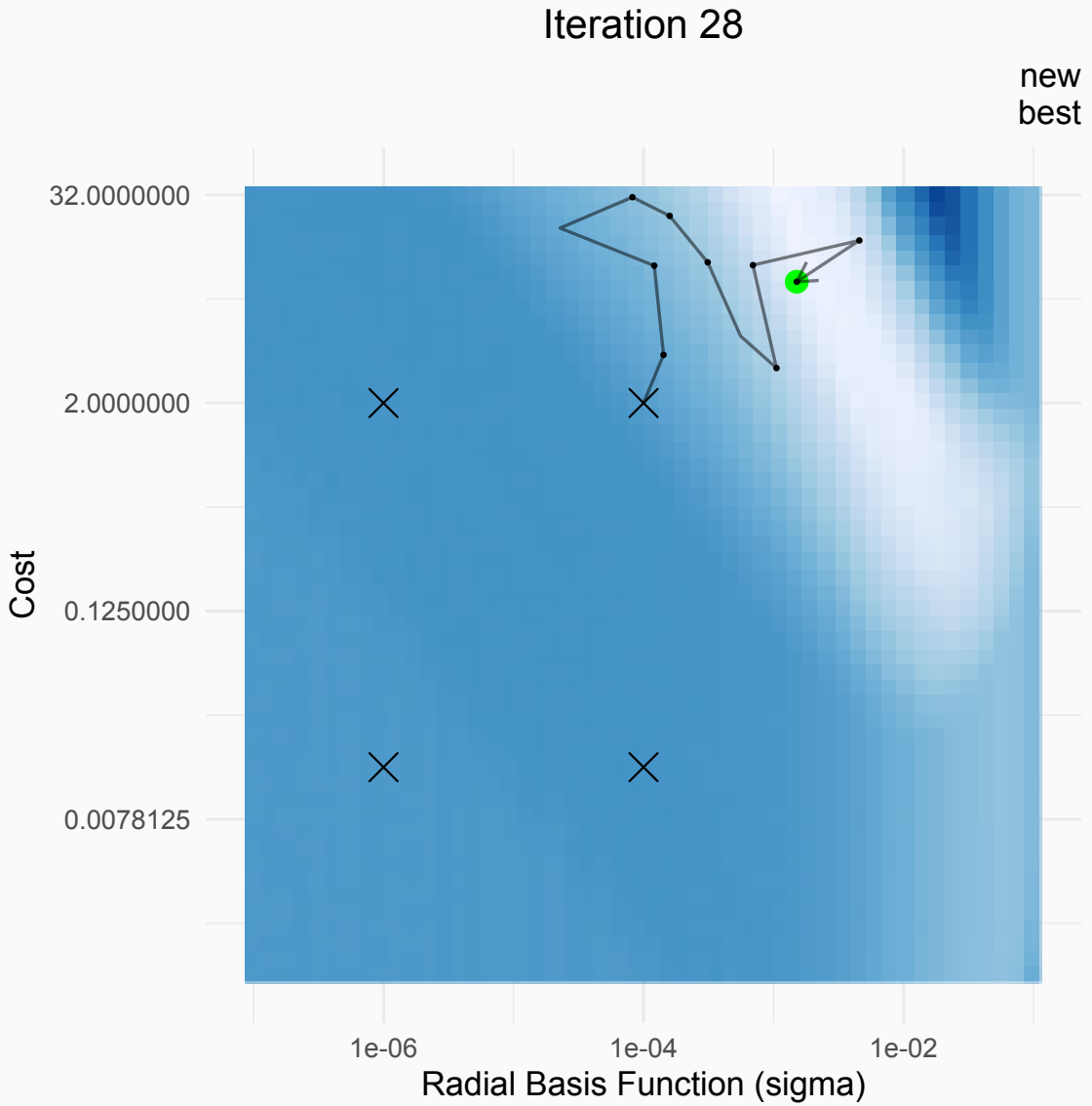
Simulated Annealing Search



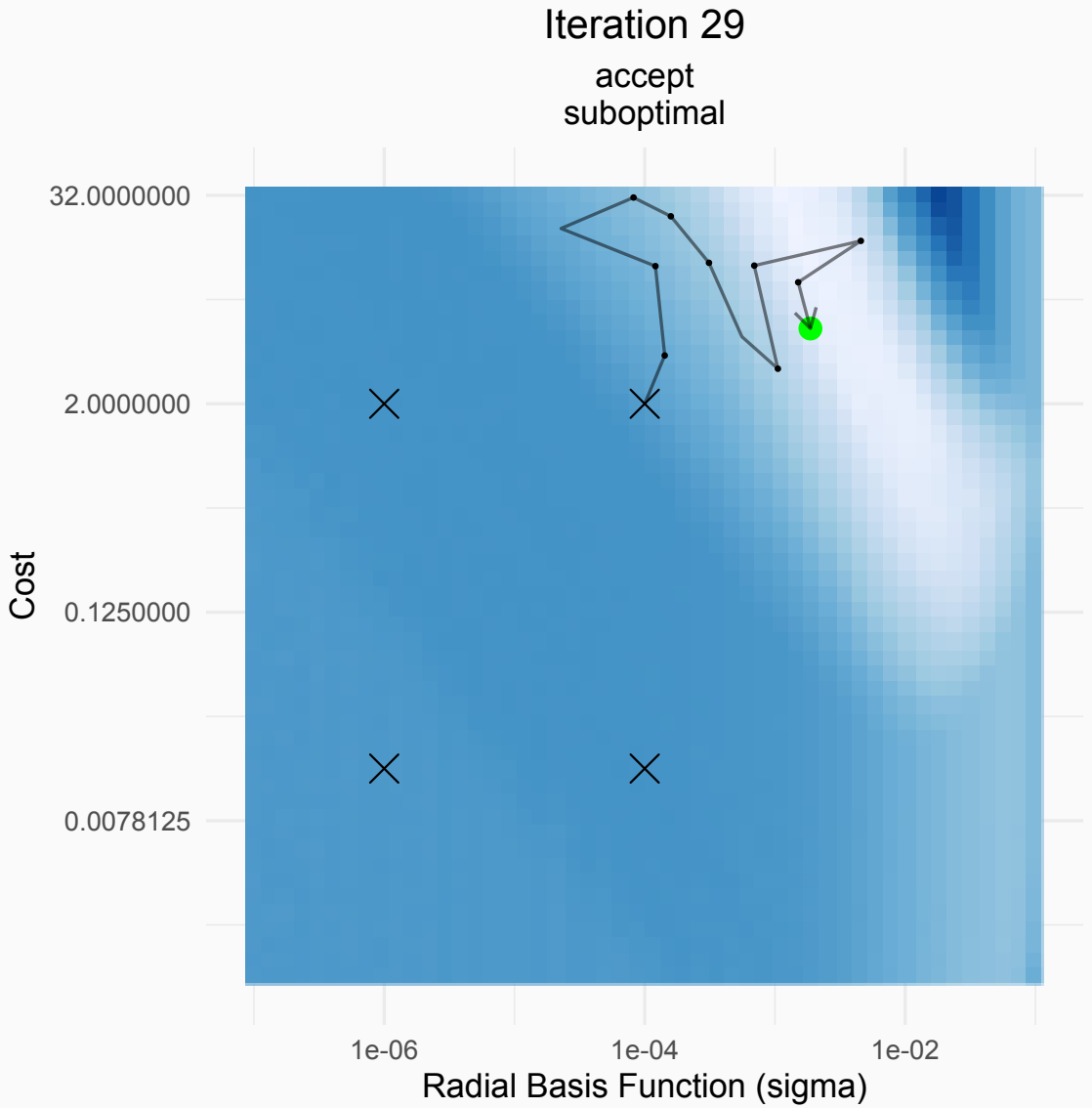
Simulated Annealing Search



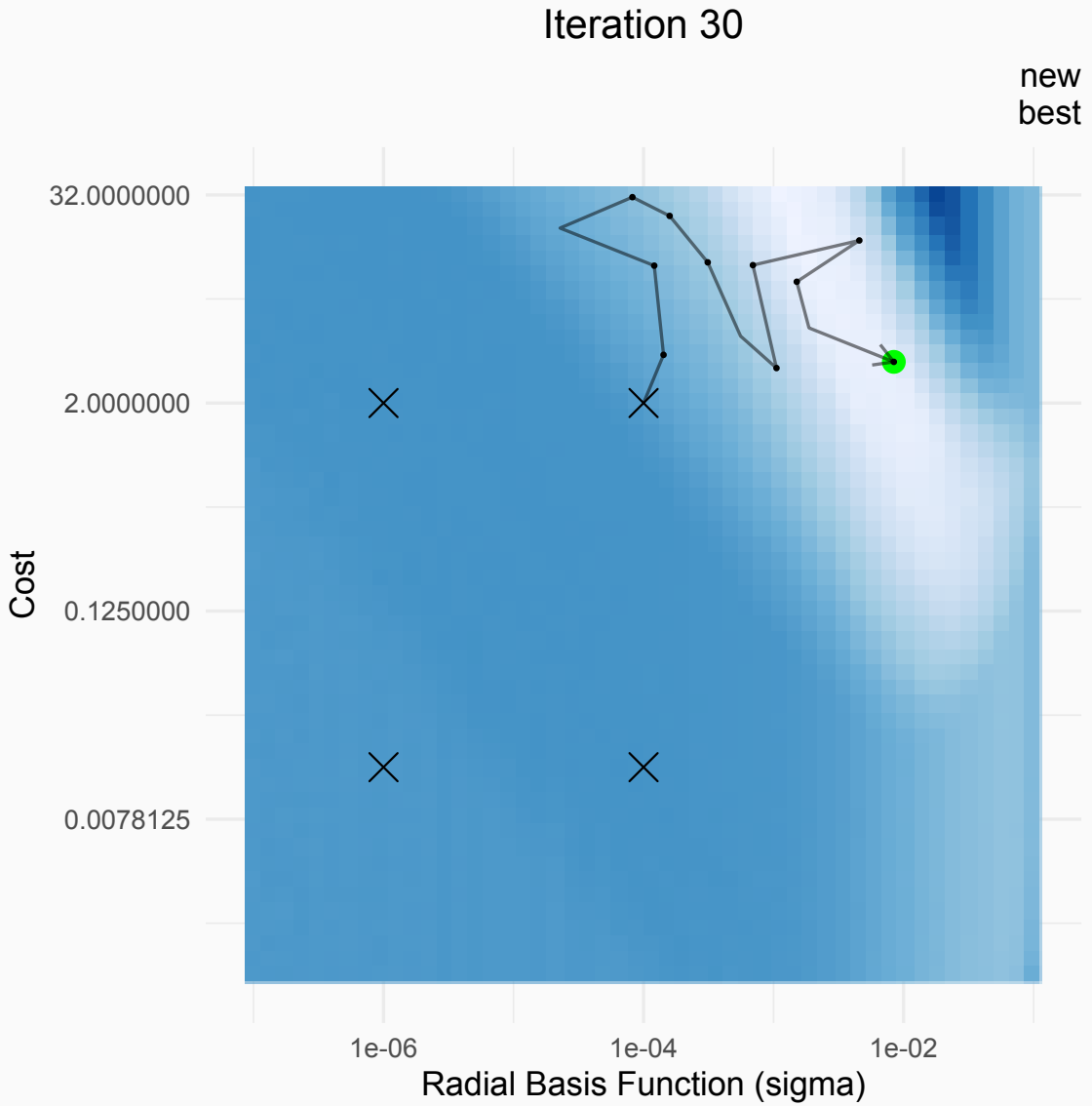
Simulated Annealing Search



Simulated Annealing Search



Simulated Annealing Search



Example Code for Simulated Annealing

```
set.seed(1234)
svm_sa <-
  svm_wflow %>%
  tune_sim_anneal(
    resamples = cell_folds,
    initial = svm_initial,
    iter = 30
  )
```

With the option `verbose = TRUE`:

```
Optimizing roc_auc
Initial best: 0.86627
 1 ♥ new best           roc_auc=0.87157    (+/-0.007672)
 2 − discard suboptimal roc_auc=0.86362    (+/-0.008314)
 3 ♥ new best           roc_auc=0.87554    (+/-0.00752)
 4 ○ accept suboptimal  roc_auc=0.87044    (+/-0.0077)
 5 ♥ new best           roc_auc=0.87764    (+/-0.007252)
 6 ♥ new best           roc_auc=0.87999    (+/-0.007074)
 7 ♥ new best           roc_auc=0.88168    (+/-0.00707)
 8 ○ accept suboptimal  roc_auc=0.871      (+/-0.007702)
 9 ○ accept suboptimal  roc_auc=0.86378    (+/-0.008341)
```

Thanks for Watching!

Resources for learning more:

- `tidymodels.org`
- *Tidy Modeling with R* book
  - Chapter 13 (*Grid Search*) has details on racing methods
  - Chapter 14 (*Iterative search*) describes simulated annealing
- `finetune` webpage and blog post