MACHINE LEARNING TOOLBOX

Reusing a `trainControl`

A real-world example

- The data: customer churn at telecom company
- Fit different models and choose the best
- Models must use the same training/test splits
- Create a shared `trainControl` object

Example: customer churn data

```
# Summarize the target variables
> library(caret)
> library(C50)
> data(churn)
> table(churnTrain$churn) / nrow(churnTrain)

      yes        no
0.1449145 0.8550855

# Create train/test indexes
> set.seed(42)
> myFolds <- createFolds(churnTrain$churn, k = 5)

# Compare class distribution
> i <- myFolds$Fold1
> table(churnTrain$churn[i]) / length(i)

      yes        no
0.1441441 0.8558559
```

Example: customer churn data

```
> myControl <- trainControl(
    summaryFunction = twoClassSummary,
    classProbs = TRUE,
    verboseIter = TRUE,
    savePredictions = TRUE,
    index = myFolds
  )
```

- Use folds to create a `trainControl` object
- Exact same cross-validation folds for each model

MACHINE LEARNING TOOLBOX

Let’s practice!

MACHINE LEARNING TOOLBOX

Reintroduce `glmnet`

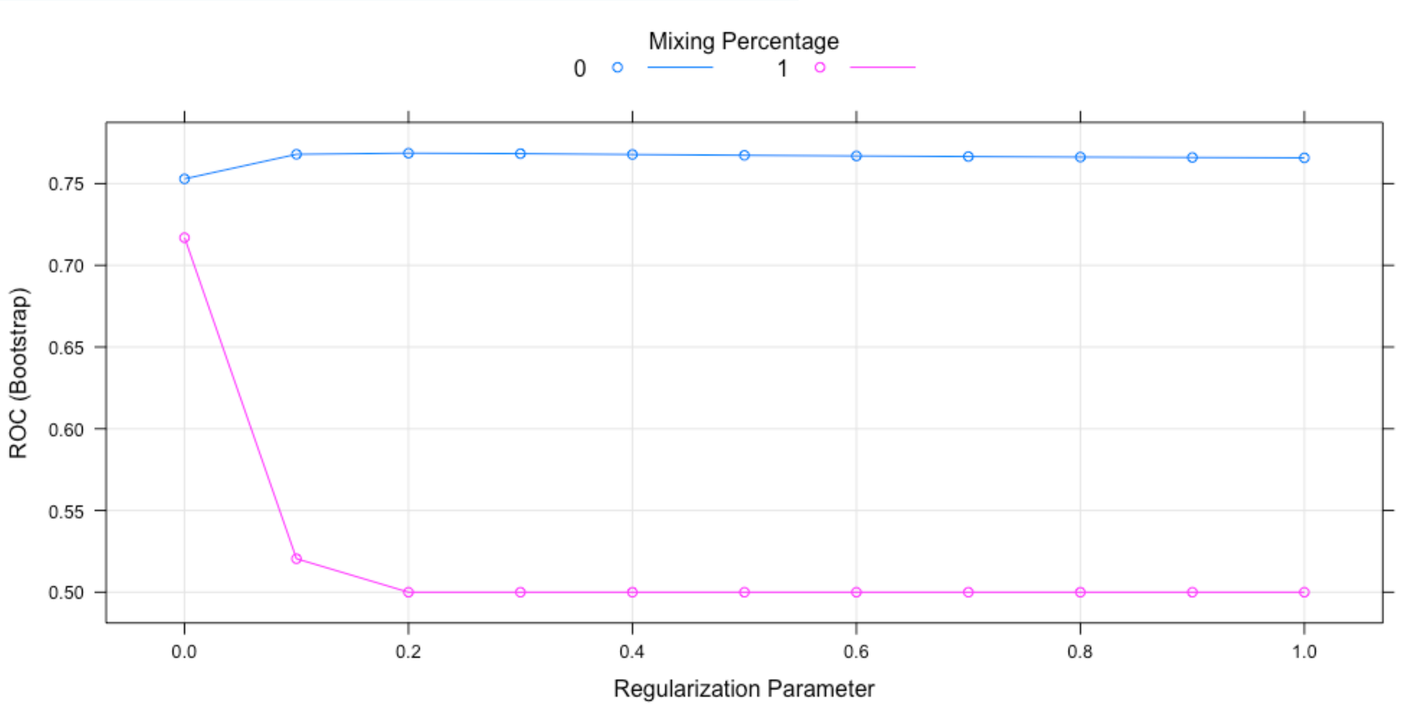
`glmnet` review

- Linear model with built-in variable selection
- Great baseline model
- Advantages
  - Fits quickly
  - Ignores noisy variables
  - Provides interpretable coefficients

Example: `glmnet` on churn data

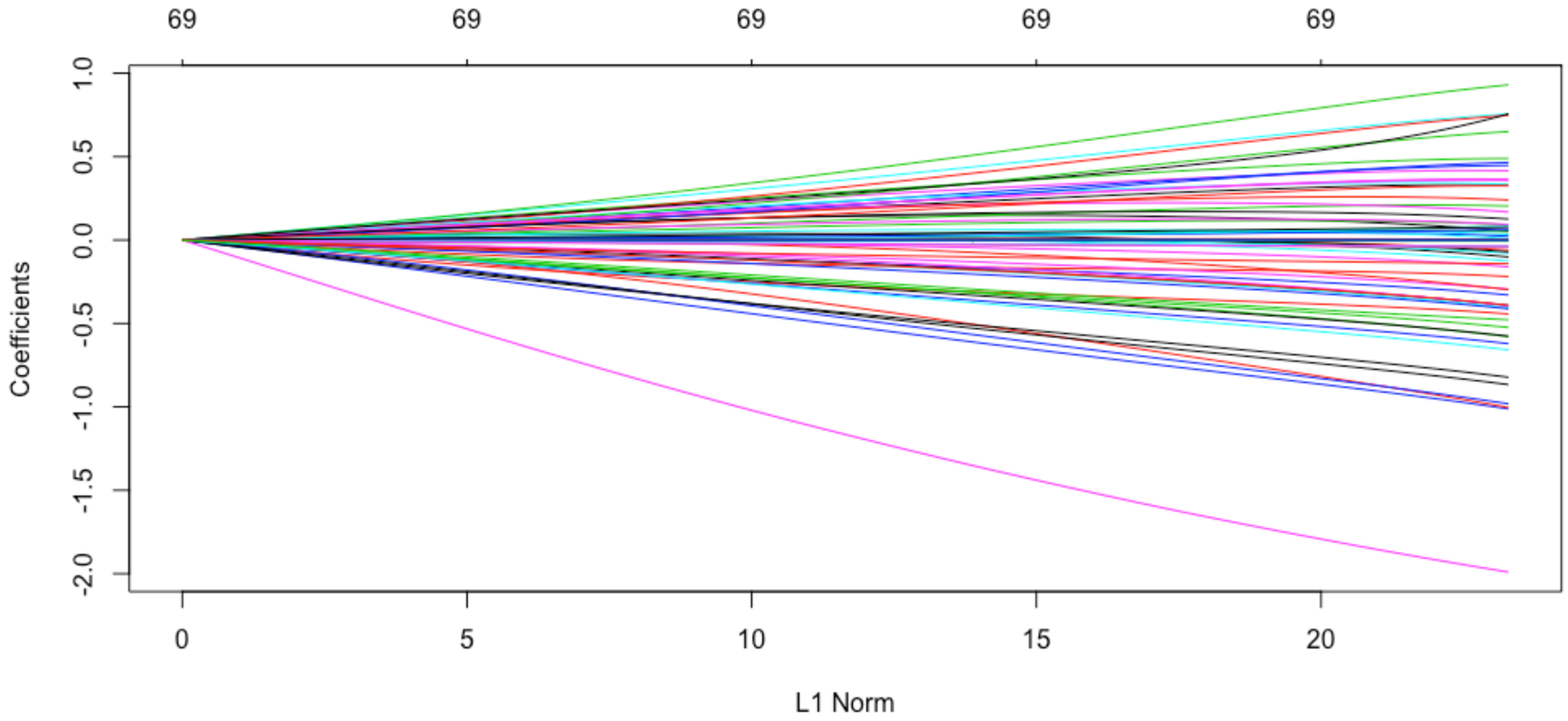
```
> # Fit the model
> set.seed(42)
> model_glmnet <- train(
    churn ~ ., churnTrain,
    metric = "ROC",
    method = "glmnet",
    tuneGrid = expand.grid(
      alpha = 0:1,
      lambda = 0:10/10
    ),
    trControl = myControl
  )

> # Plot the results
> plot(model_glmnet)
```



Plot the coefficients

```
> plot(model_glmnet$finalModel)
```



MACHINE LEARNING TOOLBOX

Let’s practice!

MACHINE LEARNING TOOLBOX

Reintroduce random forest

Random forest review

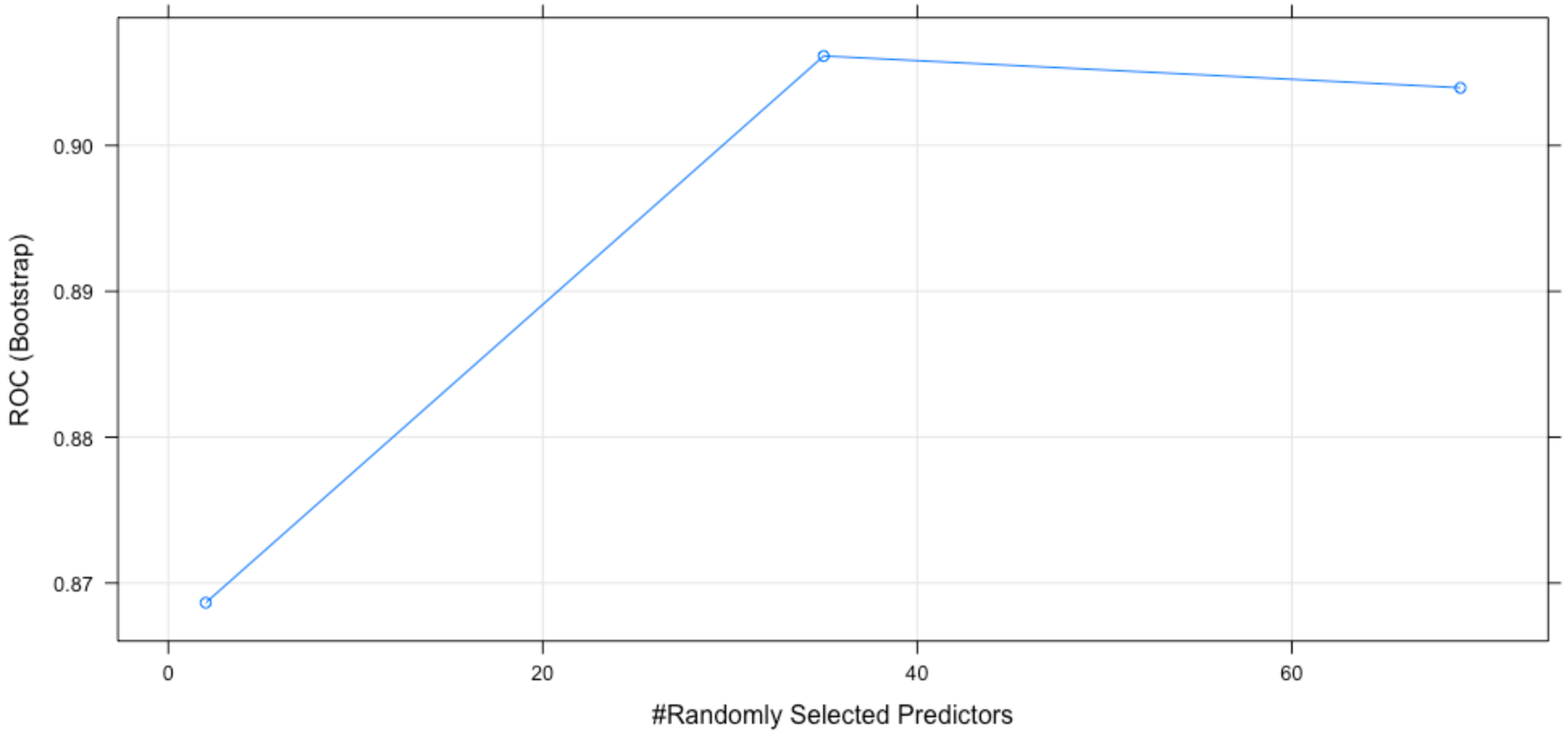
- Slower to fit than `glmnet`
- Less interpretable
- Often (but not always) more accurate than `glmnet`
- Easier to tune
- Require little preprocessing
- Capture threshold effects and variable interactions

Random forest on churn data

```
> set.seed(42)
> churnTrain$churn <- factor(churnTrain$churn, levels = c("no", "yes"))
> model_rf <- train(
    churn ~ ., churnTrain,
    metric = "ROC",
    method = "ranger",
    trControl = myControl
)
```

Random forest on churn data

```
> plot(model_rf)
```



MACHINE LEARNING TOOLBOX

Let’s practice!

MACHINE LEARNING TOOLBOX

Comparing models

Comparing models

- Make sure they were fit on the same data!
- Selection criteria
  - Highest average AUC
  - Lowest standard deviation in AUC
- The `resamples()` function is your friend

Example: `resamples()` on churn data

```
# Make a list
> model_list <- list(
    glmnet = model_glmnet,
    rf = model_rf
  )

# Collect resamples from the CV folds
> resamps <- resamples(model_list)
> resamps

Call:
resamples.default(x = model_list)

Models: glmnet, rf
Number of resamples: 5
Performance metrics: ROC, Sens, Spec
Time estimates for: everything, final model fit
```

Summarize the results

```
# Summarize the results
> summary(resamps)
Call:
summary.resamples(object = resamps)

Models: glmnet, rf
Number of resamples: 5

ROC
          Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
glmnet 0.7526  0.7624 0.7719 0.7686  0.7722 0.7840    0
rf     0.8984  0.9028 0.9077 0.9061  0.9093 0.9125    0
```

MACHINE LEARNING TOOLBOX

Let’s practice!

MACHINE LEARNING TOOLBOX

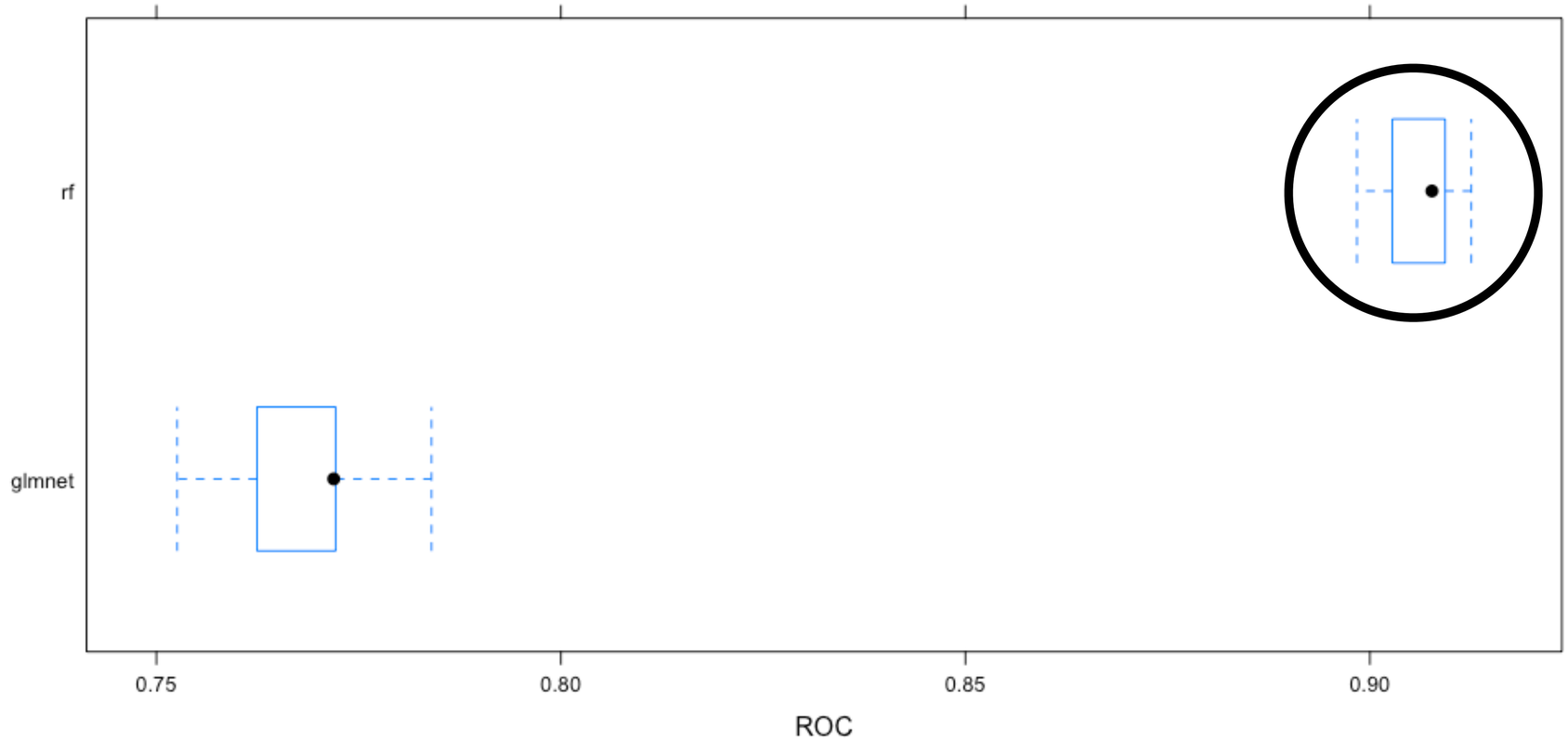
More on `resamples`

Comparing models

- Resamples has tons of cool methods
- One of my favorite functions (thanks Max!)
- Inspired the `caretEnsemble` package

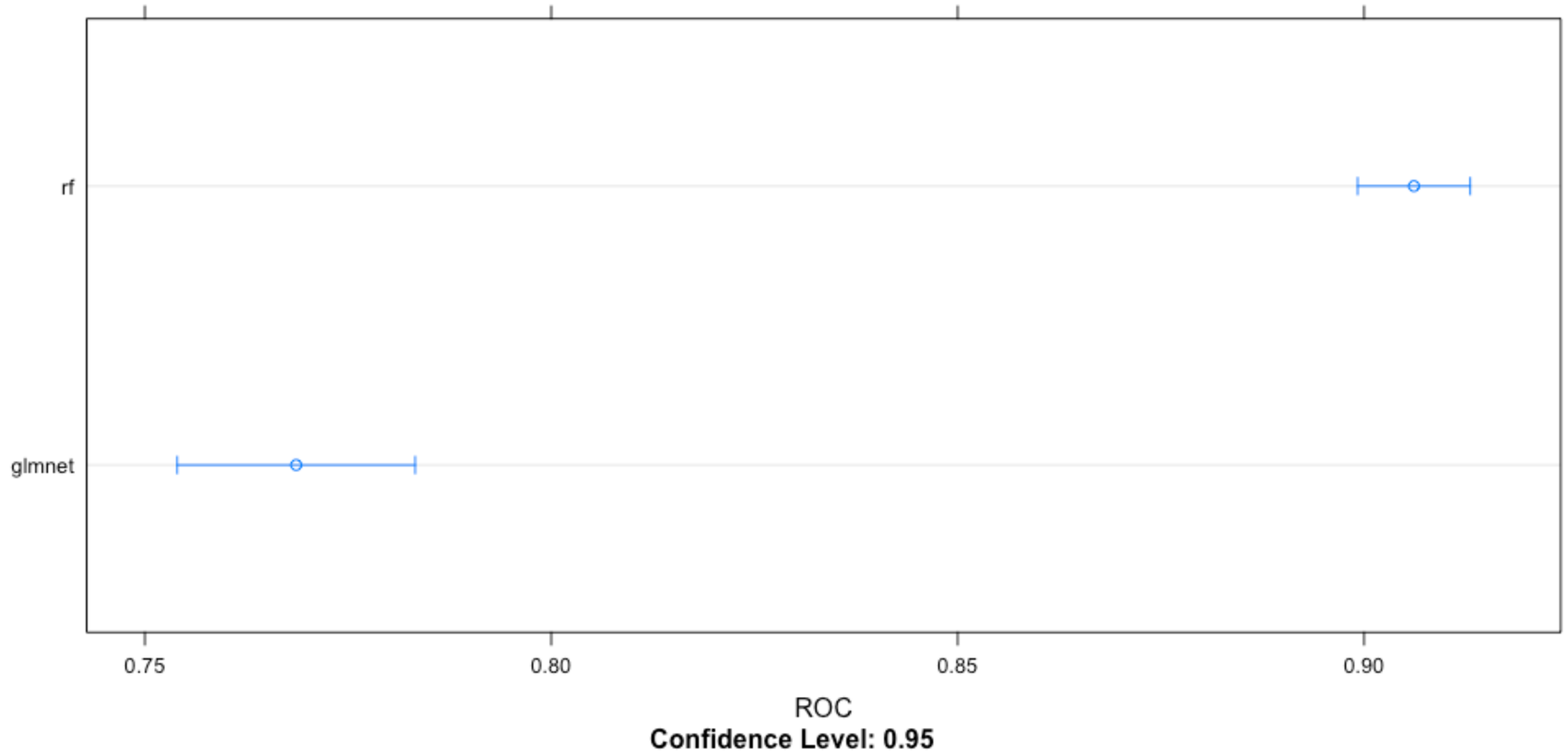
Box-and-whisker

```
> bwplot(resamps, metric = "ROC")
```



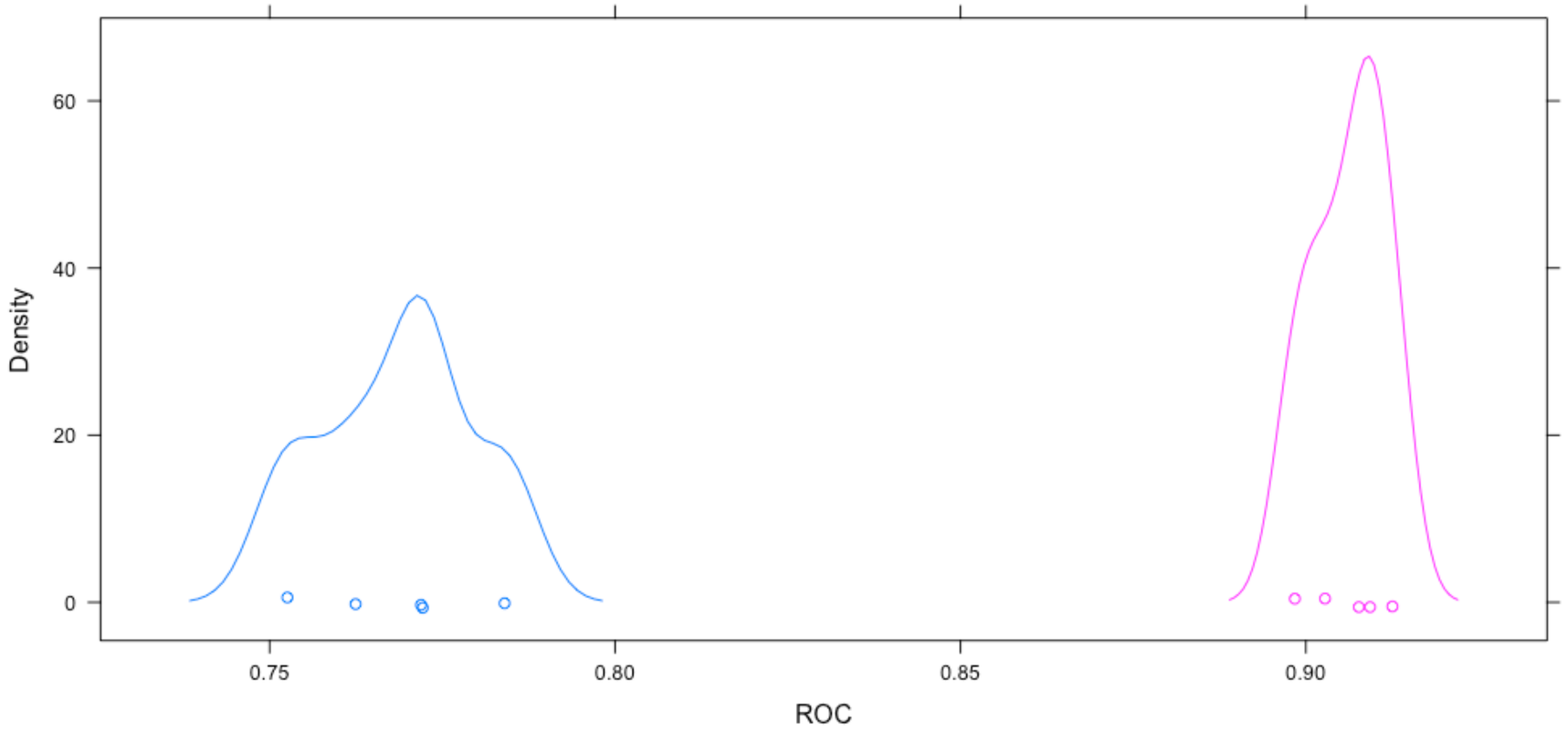
Dot plot

```
> dotplot(resamps, metric = "ROC")
```



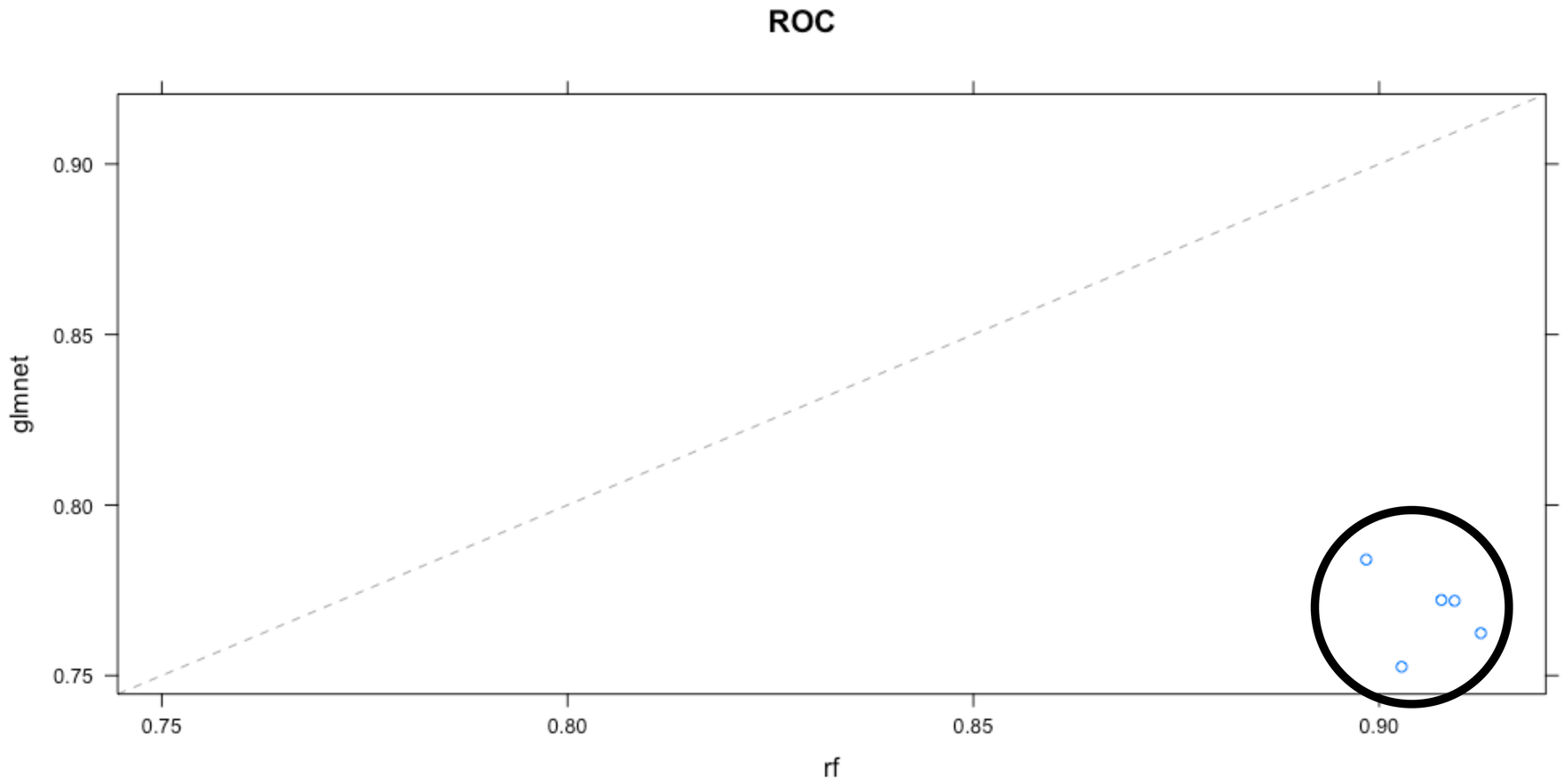
Density plot

```
> densityplot(resamps, metric = "ROC")
```



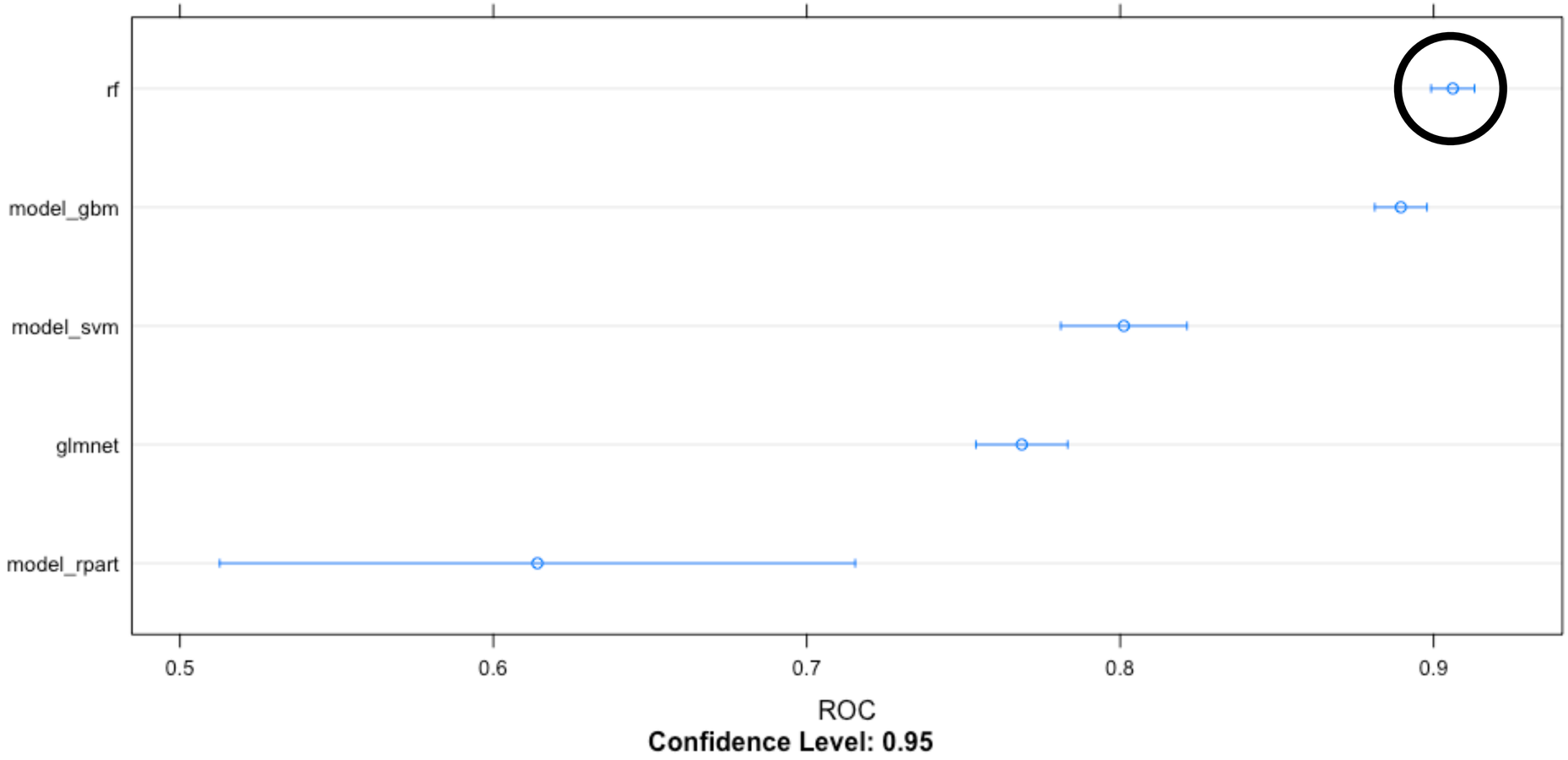
Scatter plot

```
> xyplot(resamps, metric = "ROC")
```



Another dot plot

```
> dotplot(lots_of_models, metric = "ROC")
```



MACHINE LEARNING TOOLBOX

Let’s practice!

MACHINE LEARNING TOOLBOX

Summary

What you've learned

- How to use the caret package
- Model fitting and evaluation
- Parameter tuning for better results
- Data preprocessing

Goals of the `caret` package

- Simplify the predictive modeling process
- Make it easy to try many models and techniques
- Provide common interface to many useful packages

MACHINE LEARNING TOOLBOX

Go build some models!