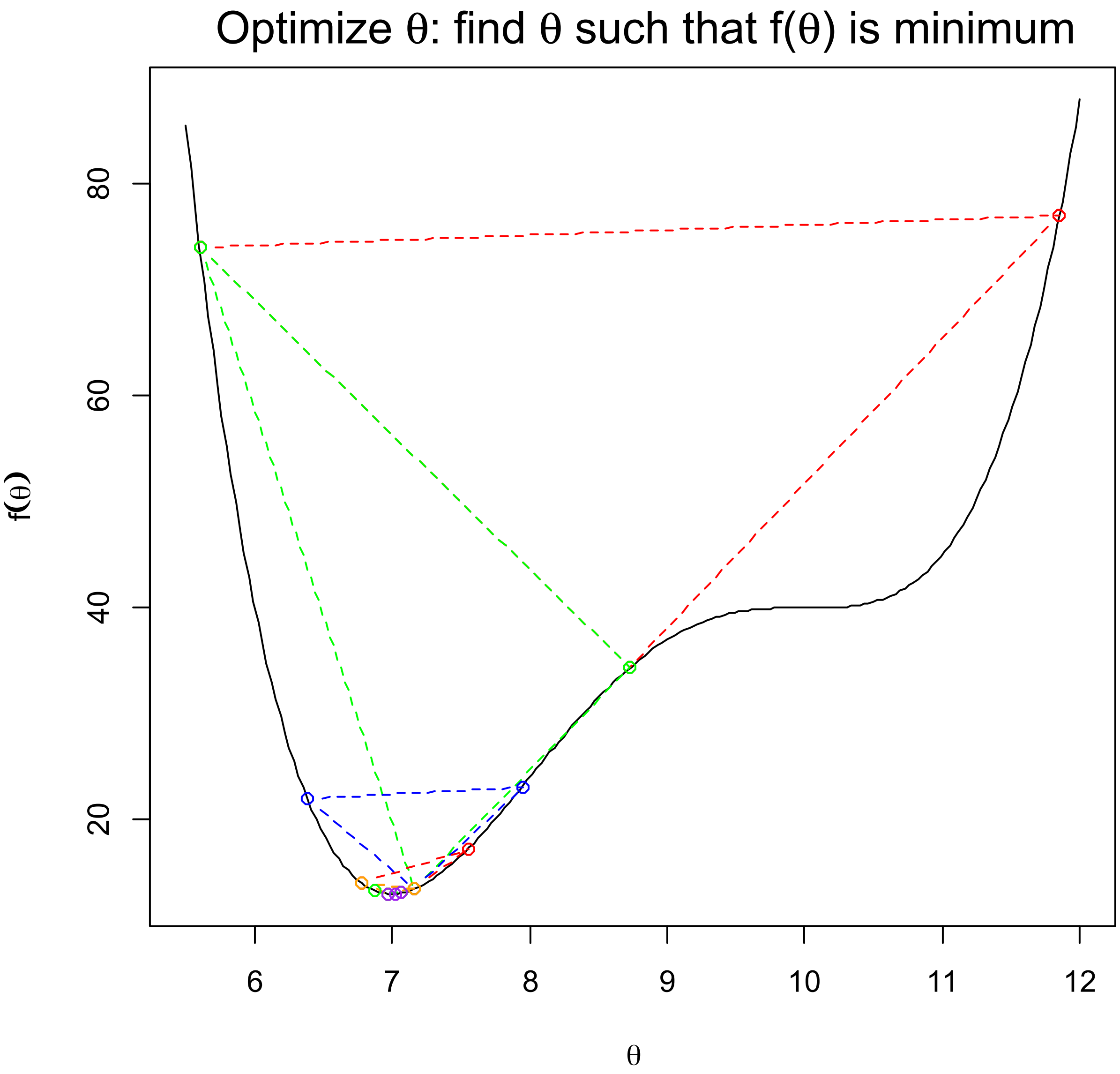
Today

- Share your linear training
- Grid search in R
- Coding a descent training algorithm
    - using R: optim()

R grid search

- See train_ssq_grid.R
- Think about general algorithm principles by relating this to your Python code
- Language agnostic thinking

Descent algorithms

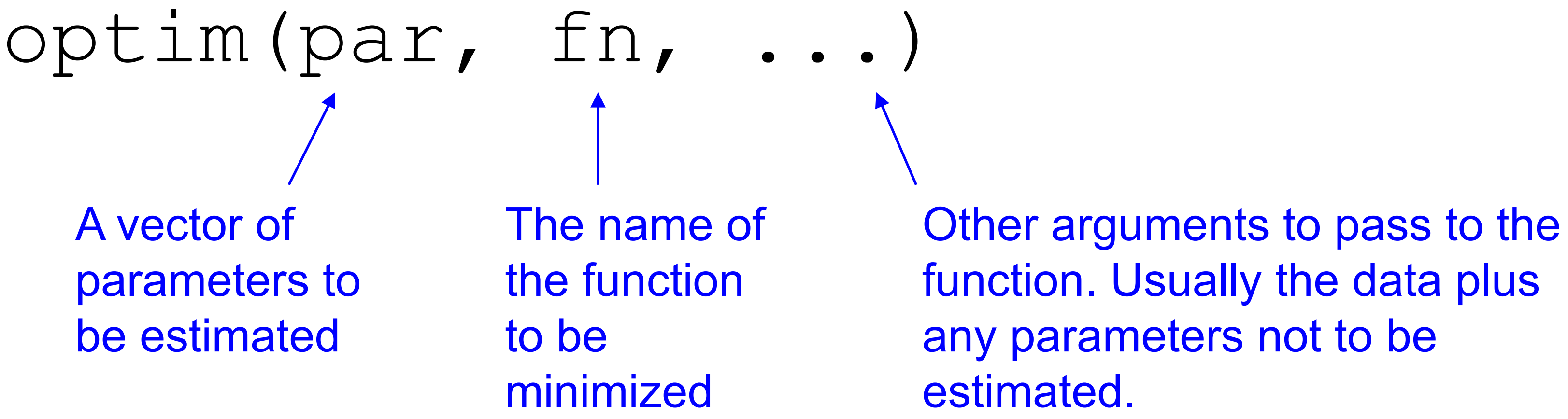


Narrowing in:

keep changing parameters in the direction that leads to lower SSQ

R: optim()

- Has various descent and Monte Carlo methods
- Nelder-Mead algorithm is default (`method="Nelder-Mead"`)



```
optim(par, fn, ...)
```

A vector of parameters to be estimated

The name of the function to be minimized

Other arguments to pass to the function. Usually the data plus any parameters not to be estimated.

Training models: general recipe

- 1) biology function
  - complex mechanistic to abstract pattern
- 2) error function
  - e.g. SSQ: distance of the model from the data

    ```
    sum((observed - predicted) ^ 2)
    ```
- 3) optimize
  - find biology parameters that minimize the error
- This recipe is the same no matter how complicated the process model or error function

Code (train_ssq_optim.R)

Biology function (e.g. linear)

```
lin_skel <- function(b_0, b_1, x) {
    y <- b_0 + b_1 * x
    return(y)
}
```

Parameters are first argument

Response data

Auxiliary data

Error function (SSQ)

```
ssq_lin_skel <- function(p, y, x) {
    y_pred <- lin_skel(b_0=p[1], b_1=p[2], x)
    e <- y - y_pred
    ssq <- sum(e^2)
    return(ssq)
}
```

Use the biology function to get predicted values

Compare predicted to the data

"Unpack" the parameters (self documenting)

Use optim to optimize error function

```
par <- c(b_0_start, b_1_start)
fit <- optim(par, ssq_lin_skel, y=data$y, x=data$x)
```

Starting values for parameters

Need “=” sign