# Simple Gibbs for RL

*Melody Jiang*

*2/20/2019*

```
library(dplyr)
library(igraph)
library(Rlab)
```

```
file_names <- c("proc_nltcs_82.txt", "proc_nltcs_89.txt", "proc_nltcs_94.txt")
k = length(file_names)
# load data
D <- lapply(file_names, function(files){
  read.table(files, header = FALSE)
})
```

Here, D is a large list containint 3 elements. Each element is a file. For the purpose of our simple Gibbs sampler, we take a portion (0.001) of data from each file so that we can work with fewer data.

```
# sample from original data
for (i in 1:3) {
  D[[i]] = sample_frac(D[[i]], size = 0.001)
}
```

```
# move id numbers to last column for future use
D <- lapply(D, function(x){
  x[, c(2:ncol(x), 1 ) ]
})
```

```
# obtain information from data
# vector of file lengths
nv <- sapply(D, function(x)dim(x)[1])
nt <- sum(nv)
num_field <- 6
```

```
# obtain information from data continued

# vector of number of levels for each cateogry
m_l_vec <- vector(mode = "numeric", length = 6)
# list of vectors of unique field values of a filed
levels_list_of_vec <- list()

for (iter_field in 1:6) {
  all_field_vals <- vector(mode = "numeric", length = 0)
  for (iter_file in 1:3) {
    all_field_vals <- c(all_field_vals, D[[iter_file]][, iter_field])
  }
  m_l_vec[iter_field] <- length(unique(all_field_vals))
  levels_list_of_vec[[iter_field]] <- sort(unique(all_field_vals))
  print("For field")
  print(iter_field)
  print(sort(unique(all_field_vals)))
}
```

```
## [1] "For field"
## [1] 1
## [1] 1 2
## [1] "For field"
## [1] 2
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12
## [1] "For field"
## [1] 3
##  [1]  1  2  3  4  7  8  9 10 11 13 14 16 17 18 19 20 21 23 24 25 26 27 29
## [24] 30 31
## [1] "For field"
## [1] 4
##  [1]  4  7 11 17 19 20 22 24 25 26 27 28 29 30 31 33 34 44 45 47 48 49 50
## [24] 51 55
## [1] "For field"
## [1] 5
##  [1]  1  2  3  4  5  6  7  9 10 11 12
## [1] "For field"
## [1] 6
##  [1]  1  2  3  4  5  8  9 11 12 14 15 18 19 20 22 23 25 26 27 28 33 36 37
## [24] 39 44
```

```r
# hyperparameters
a_vec <- rep(1, 6)
b_vec <- rep(99, 6)
```

```r
# hyperparameters continued
mu_list_of_vec <- list()
for (iter_field in 1:6) {
  mu_list_of_vec[[iter_field]] <- rep(x = 1/m_l_vec[iter_field], times = m_l_vec[iter_field])
}
```

```r
# initialize

# intialize beta vector
beta_vec <- vector(mode = "numeric", length = 6)
for (iter in 1:length(beta_vec)) {
  beta_vec[iter] <- rbeta(1, shape1 = a_vec[iter], shape2 = b_vec[iter])
}
```

```r
# intialize list of theta vectors
theta_list_of_vec <- list()
for (iter in 1:6) {
  theta_list_of_vec[[iter]] <- rep(x = 1/m_l_vec[iter_field], times = m_l_vec[iter_field])
}
```

```r
# initialize y matrix
y_matrix <- matrix(data = NA, nrow = nt, ncol = 6)
# initialization method 1
# for (iter in 1:6) {
#   y_matrix[, iter] <- sapply(y_matrix[, iter], function(x){
#   sample(levels_list_of_vec[[iter]], size = 1)
# })
# }

# initialization method 2
j_prime <- 1
```

```
for (i in 1:length(nv)){
  for (j in 1:nv[i]) {
    for (l in 1:num_field) {
      y_matrix[j_prime, l] <- D[[i]][j,l]
    }
    j_prime <- j_prime + 1
  }
}
```

```
# initialize z, list of matrices
z_list_of_mat <- list()
for (iter in 1:3) {
  z_list_of_mat[[iter]] <- matrix(data = NA, nrow = nv[iter], ncol = 6)
  z_list_of_mat[[iter]] <- apply(z_list_of_mat[[iter]], c(1, 2),function(x){
    sample(c(0,1), size = 1)
  })
}
```

```
# initialize Lambda, list of vectors
Lambda_list_of_vec <- list()
for (iter in 1:3) {
  Lambda_list_of_vec[[iter]] <- vector(mode="numeric", length = nv[iter])
  Lambda_list_of_vec[[iter]] <- sapply(Lambda_list_of_vec[[iter]], function(x){
    sample(seq(1:nt), size = 1)
  })
}
```

To do Gibbs sampling, we would need full condutionals. Hence let's code up full conditionals first.

```
# Full conditional of beta_l
rbeta_l_dist <- function(num_obs = 1, l) {
  # this function returns a random sample from the full conditional of beta_l

  # obtain values of a_l and b_l
  a_l <- a_vec[l]
  b_l <- b_vec[l]
  # obtain values of z_l
  z_l <- vector(mode = "numeric", length = 0)
  for (iter in 1:length(z_list_of_mat)) {
    z_l <- c(z_l, z_list_of_mat[[iter]][, l])
  }
  alpha <- a_l + sum(z_l)
  beta <- b_l + sum(1 - z_l)
  return( rbeta(n = num_obs,shape1 = alpha,shape2 = beta) )
}
```

```
# Full conditional of theta_l
rtheta_l_dist <- function(num_obs = 1, l){
  params <- vector(mode = "numeric", length = length(mu_list_of_vec[[l]]) )

  for (m in 1:length(mu_list_of_vec[[l]])) {
    mu_lm <- mu_list_of_vec[[l]][m]

    sum_indicator_y_l <- 0
    for (j_prime in 1:nt) {
      if (y_matrix[j_prime, l] == levels_list_of_vec[[l]][m]) {
```

```r
        sum_indicator_y_l <- sum_indicator_y_l + 1
      }
    }

    sum_z_l_times_indicator_x_l <- 0
    for (i in 1:length(nv)) {
      for (j in 1:nv[i]) {
        if (D[[i]][j, l] == levels_list_of_vec[[l]][m]) {
          sum_z_l_times_indicator_x_l <- sum_z_l_times_indicator_x_l + ( z_list_of_mat[[i]][j, l] * 1 )
        }
      }
    }

    big_sum <- sum_indicator_y_l + sum_z_l_times_indicator_x_l + 1

    params[m] <- mu_lm + big_sum

  }

  return(sample_dirichlet(num_obs, params))

}
```

```r
# Full conditional of y_j'l
ry_jprimel_dist <- function(num_samp = 1, j_prime, l){

  for (i in 1:3) {
    j_in_R <- FALSE
    z_ijl_is_zero <- FALSE
    j <- 0
    if(j_prime %in% Lambda_list_of_vec[[i]]){
      j_in_R <- TRUE
      j <- which(Lambda_list_of_vec[[i]] %in% j_prime)[1]
    }
    if(j_in_R == TRUE){
      if(z_list_of_mat[[i]][j, l] == 0){
        z_ijl_is_zero <- TRUE
      }
    }
    if(j_in_R && z_ijl_is_zero){
      return(D[[i]][j, l])
    }
  }

  multinorm_res <- rmultinom(n = num_samp, size = 1, prob = theta_list_of_vec[[l]])
  # transpose multinorm_res
  multinorm_res <- t(multinorm_res)
  position <- which(multinorm_res %in% 1)
  # get field value corresponding to the position
  multinorm_val <- levels_list_of_vec[[l]][position]
  return(multinorm_val)
}
```

```r
# Full conditional of z_ijl

rz_ijl_dist <- function(num_samp = 1, i, j, l){
  lambda_ij <- Lambda_list_of_vec[[i]][j]
  if (D[[i]][j, l] != y_matrix[lambda_ij, l]) {
    return(1)
  }
  else{
    big_prod <- 1
    for (m in 1:length(levels_list_of_vec[[l]])) {
      if(D[[i]][j, l] == levels_list_of_vec[[l]][m]){
        big_prod <- big_prod * theta_list_of_vec[[l]][m]
      }
    }
    prob <- beta_vec[l] * big_prod / (beta_vec[l] * big_prod + (1 - beta_vec[l]))

    return( rbern(n = num_samp, p = prob) )
  }
}
```

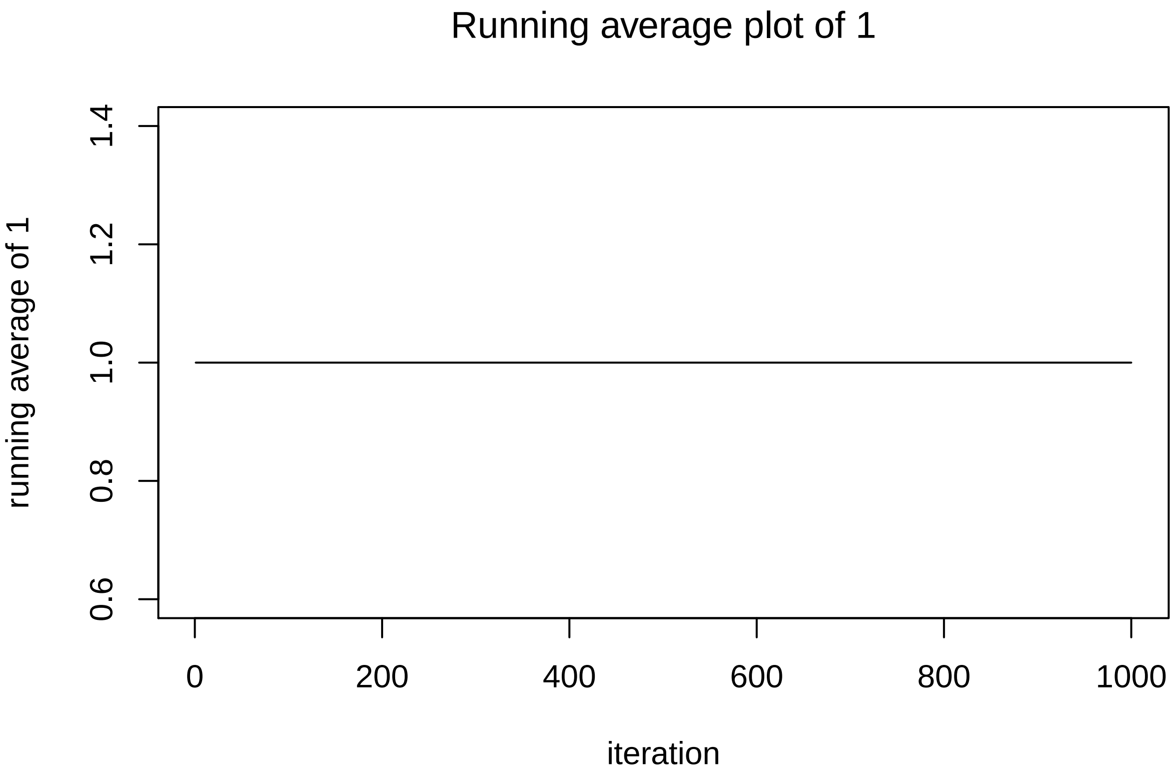
```r
# Full conditional of lambda_i

rlambda_i_dist <- function(i){
  while (TRUE) {
    lambda_i_vals <- sample(seq(1:nt), size = nv[i])
    z_ijl_is_zero <- FALSE
    x_ijl_not_equal_y_cjl <- FALSE

    for (j in 1:length(nv[i])) {
      for (l in 1:num_field) {
        if (z_list_of_mat[[i]][j,l] == 0
            && D[[i]][j, l] != y_matrix[lambda_i_vals[j], l]) {
          z_ijl_is_zero <- TRUE
          x_ijl_not_equal_y_cjl <- TRUE
        }
      }
    }

    if (!z_ijl_is_zero && !x_ijl_not_equal_y_cjl) {
      return(lambda_i_vals)
    }
  }
}
```

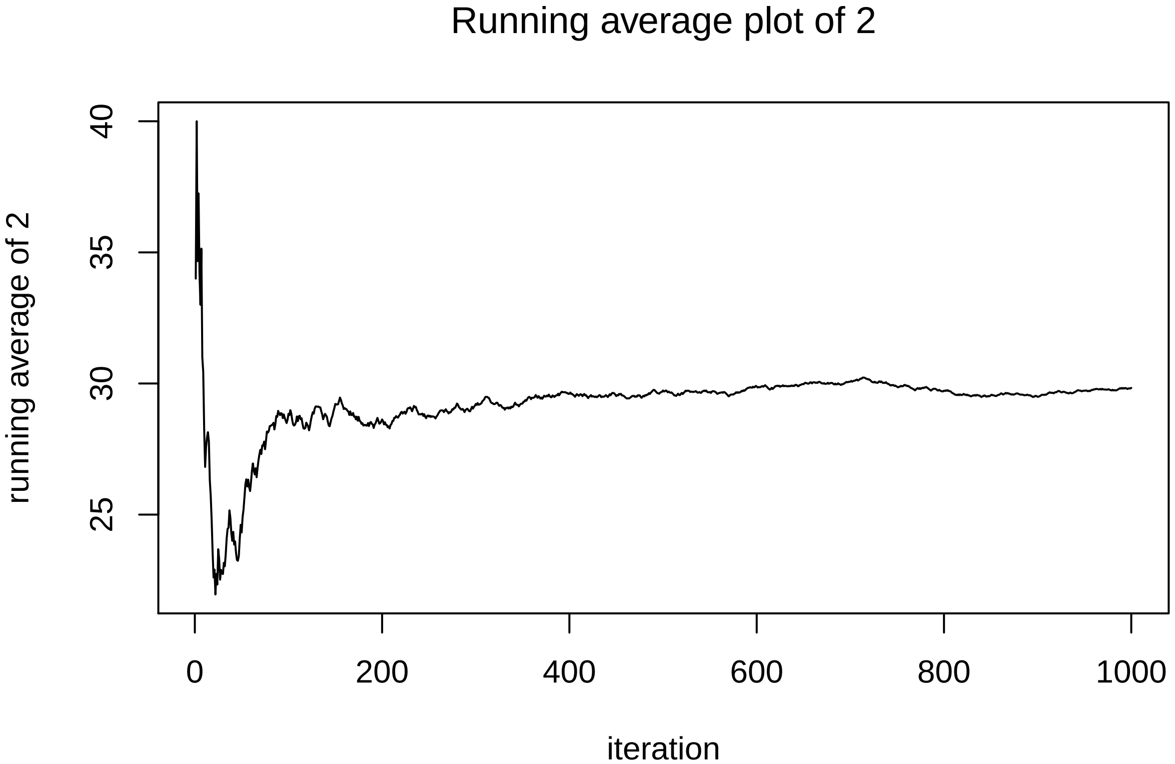
```r
sampleGibbs <- function (D, n.iter, a_vec, b_vec, mu_list_of_vec, beta_vec, theta_list_of_vec, y_matrix
  # some initializaiton
  # res <- matrix(data = NA, nrow = n.iter, ncol = 5)
  # colnames(res) <- c("beta", "theta", "y", "z", "Lambda")
  res_Lambda <- matrix(data = NA, nrow = n.iter, ncol = nt)
  for (overall_iter in 1:n.iter) {
    # update beta
    for (l in 1:num_field) {
      beta_vec[l] <- rbeta_l_dist(num_obs = 1, l)
    }
    # update theta
    # exist question about the sampler, coded up already, but correction needed
    for (l in 1:num_field) {
      theta_list_of_vec[[l]] <- rtheta_l_dist(num_obs = 1, l)
    }

    # update y
    for (j_prime in 1:nt) {
      for (l in 1:num_field) {
        y_matrix[j_prime, l] <- ry_jprimel_dist(num_samp = 1, j_prime, l)
      }
    }

    # update z
    for (i in 1:length(nv)) {
      for (j in 1:nv[i]) {
        for (l in num_field) {
          z_list_of_mat[[i]][j, l] <- rz_ijl_dist(num_samp = 1, i, j, l)
        }
      }
    }

    # update Lambda
    # haven't coded up sampler of lambda. Completion needed
    for (i in 1:length(nv)) {
      Lambda_list_of_vec[[i]] <- rlambda_i_dist(i)
    }

    # res[overall_iter, 1] <- beta_vec
    # res[overall_iter, 2] <- theta_list_of_vec
    # res[overall_iter, 3] <- y_matrix
    # res[overall_iter, 4] <- z_list_of_mat
    # res[overall_iter, 5] <- Lambda_list_of_vec

    res_Lambda[overall_iter,] <- c(Lambda_list_of_vec[[1]], Lambda_list_of_vec[[2]], Lambda_list_of_vec

  }
  return(res_Lambda)
}
```

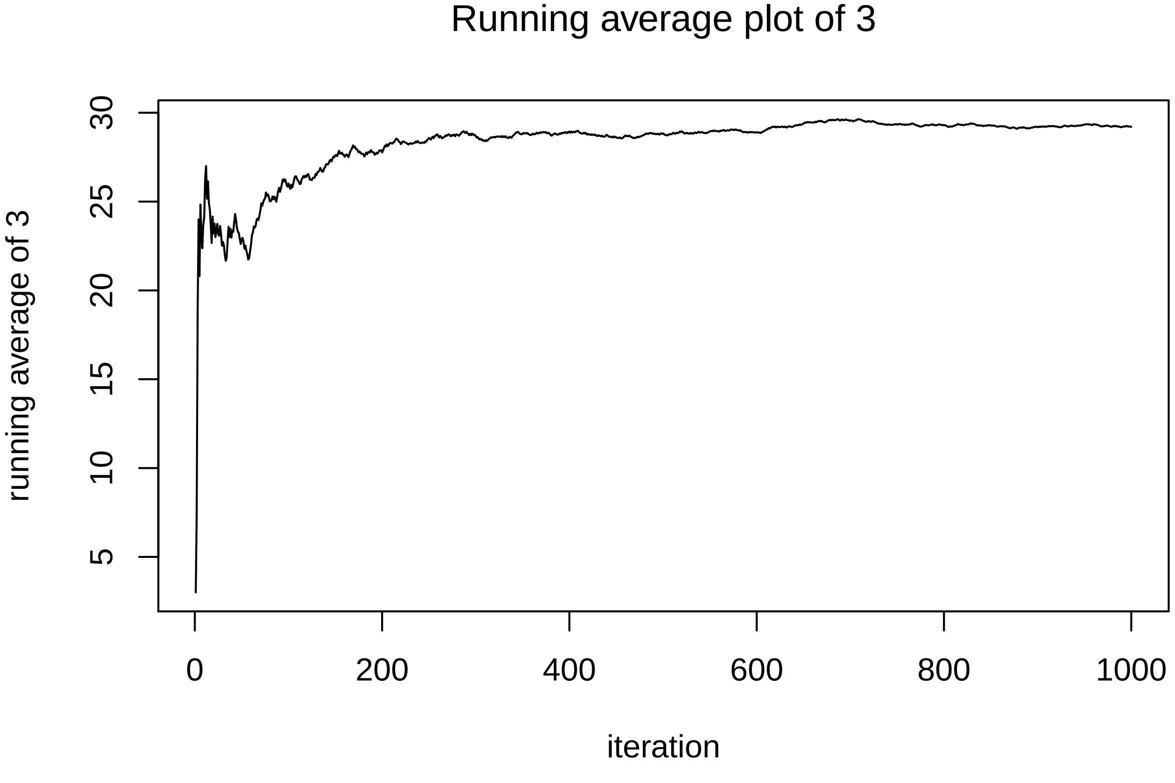
```
# Run Gibbs sampler
n.iter <- 1000
res_Lambda <- sampleGibbs(D, n.iter, a_vec, b_vec, mu_list_of_vec, beta_vec, theta_list_of_vec, y_matri
```

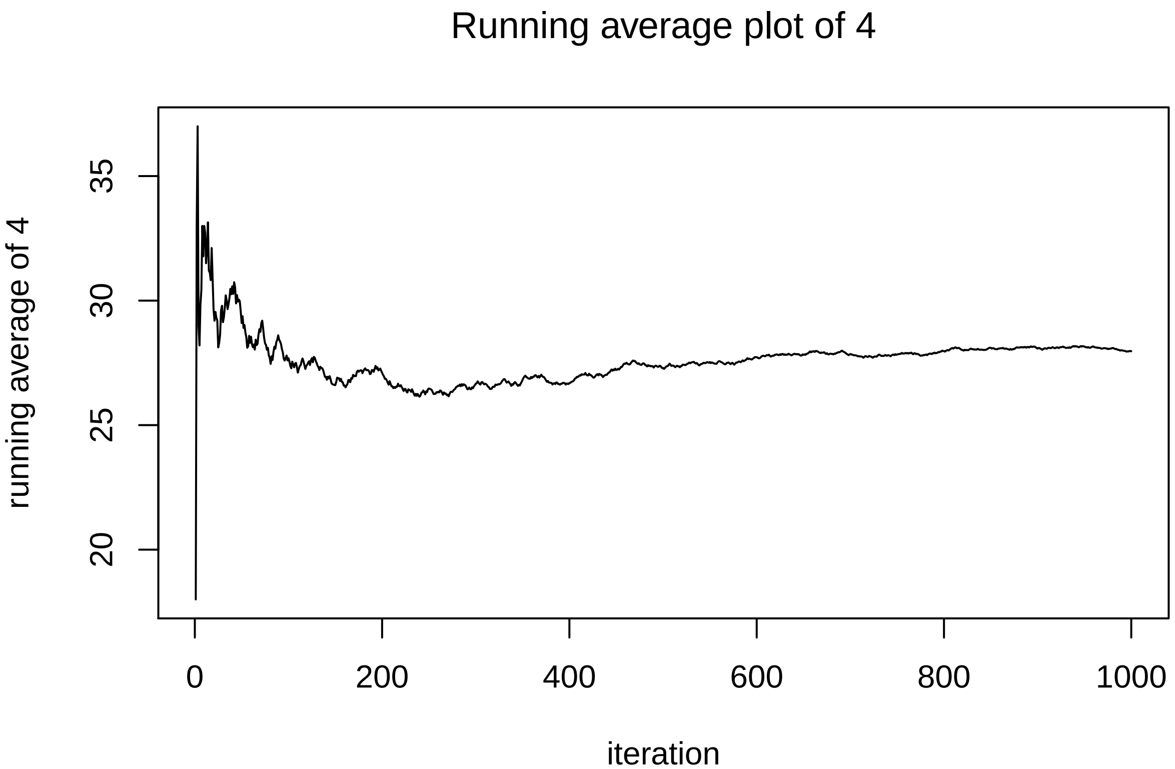
```
# Traceplot of Lambda_1,5
# plot(1:n.iter, res_Lambda[,5], type = "l", cex = 0.35,
#      xlab = "iteration", ylab = expression(lambda[15]),
#      main = expression(paste("Traceplot of ", lambda[15])))
```

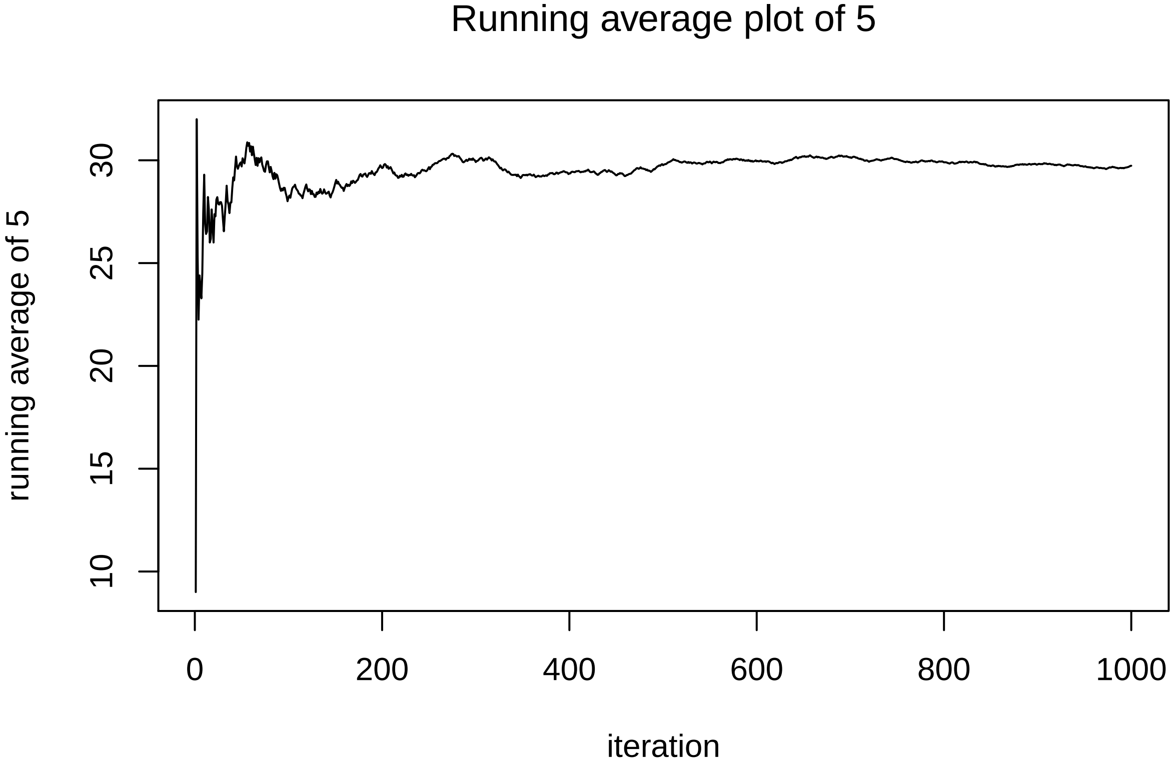
```
# Running average plot of Lambda_1,5

# get running averages of Lambdas
run.avg <- apply(res_Lambda, MARGIN = 2,FUN = function(x){
  cumsum(x)
}) / (1:n.iter)

# par(mfrow = c(dim(run.avg)[2]/2, 2))

for (iter in 1:dim(run.avg)[2]) {
  x.lab <- bquote(.(iter))
  plot(1:n.iter, run.avg[, iter], type = "l", cex = 0.5,
     xlab = "iteration", ylab = bquote(paste("running average of ", .(x.lab))),
     main = bquote(paste("Running average plot of ", .(x.lab))))
}
```

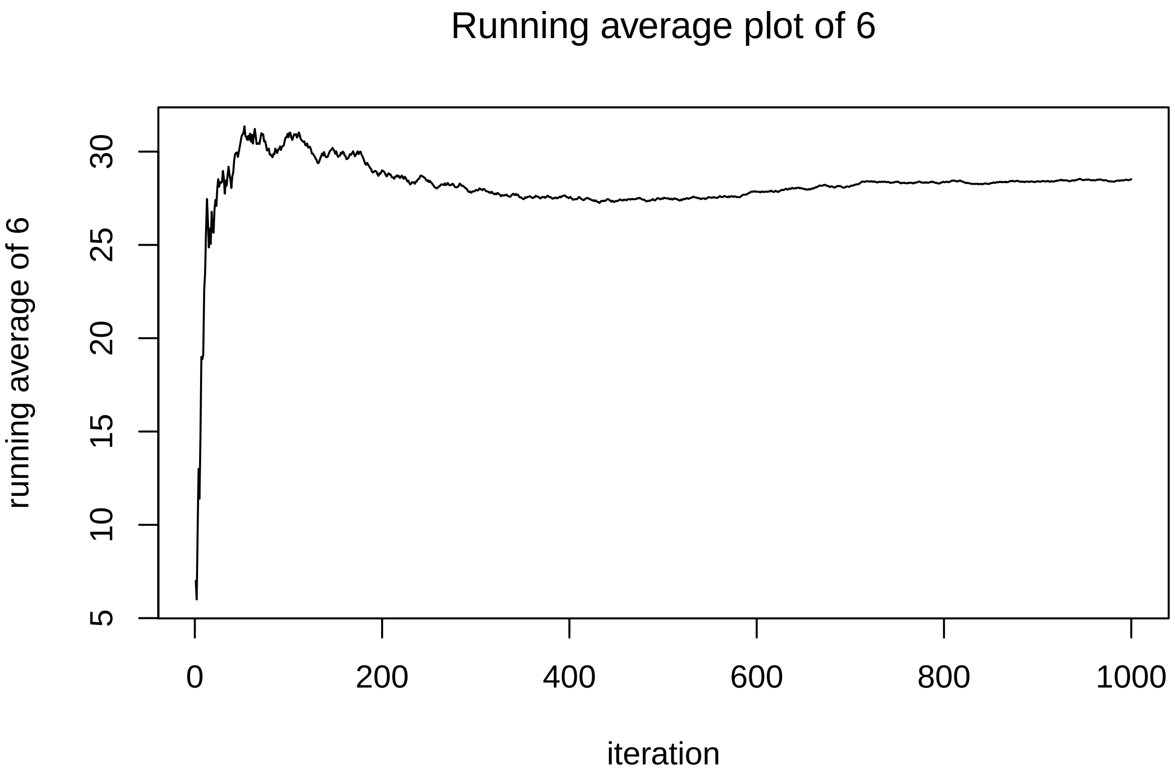


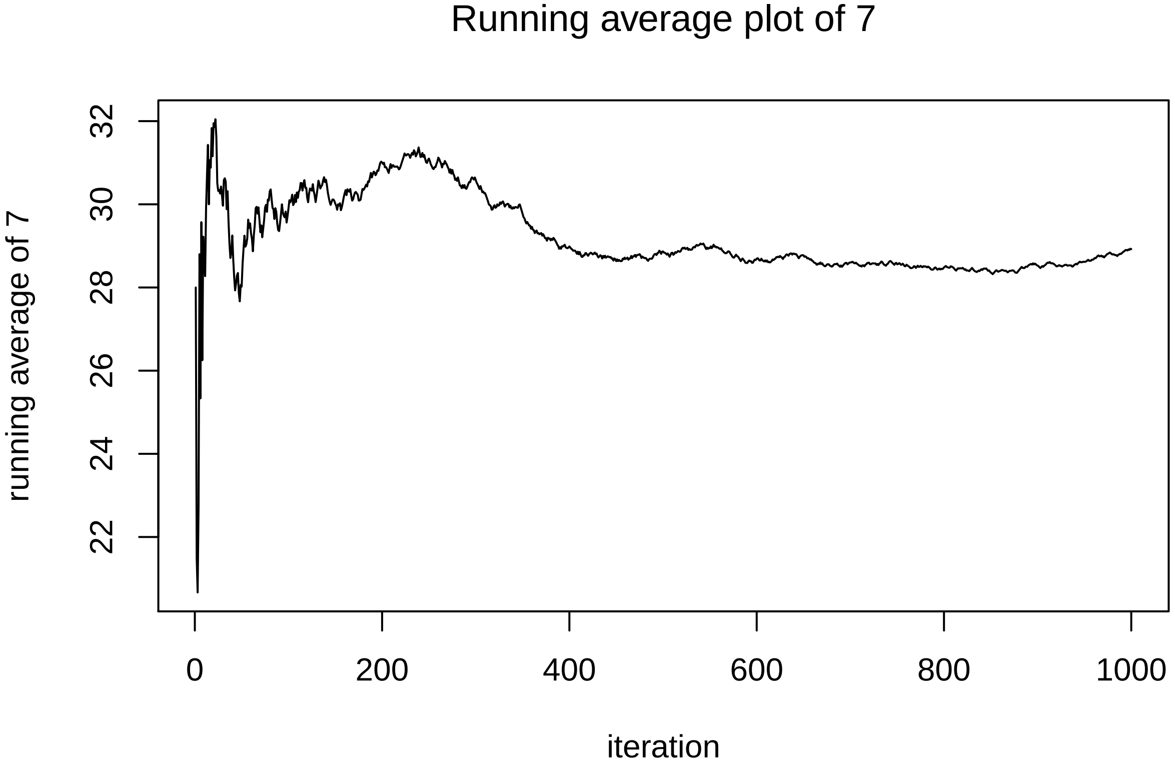

Running average plot of 2
running average of 2
40
35
30
25
0
200
400
600
800
1000
iteration

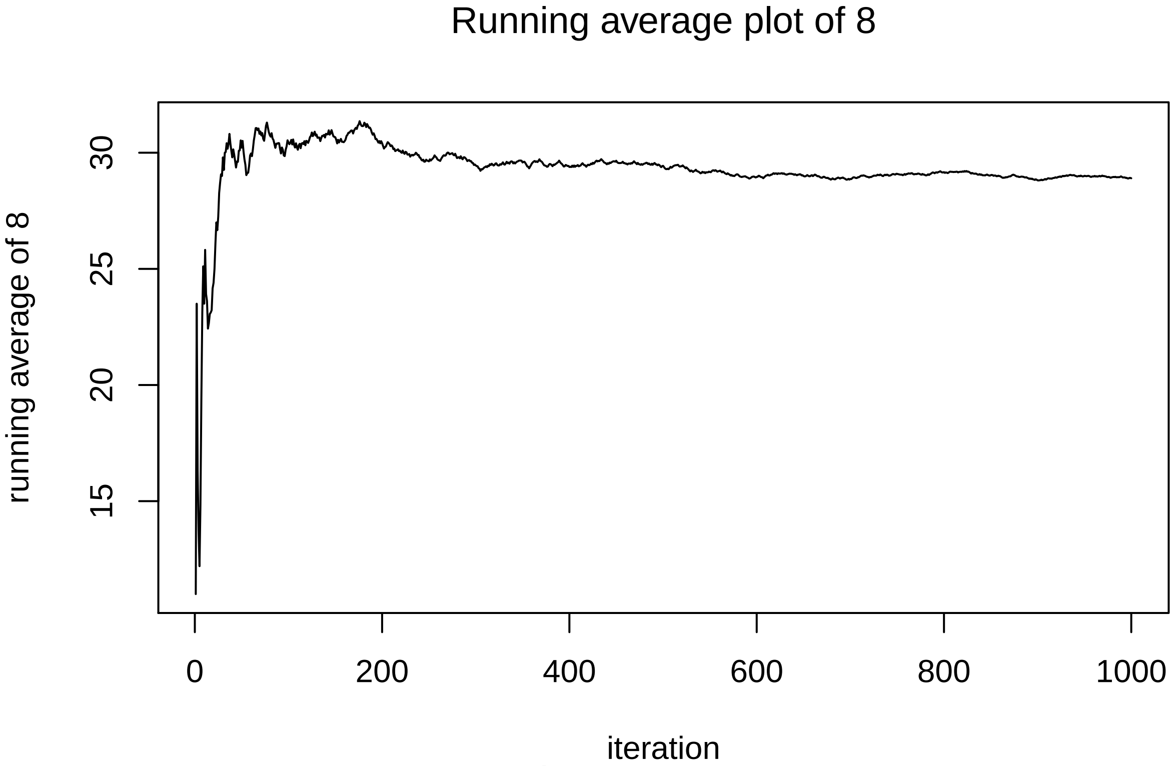

Running average plot of 3
running average of 3
30
25
20
15
10
5
0
200
400
600
800
1000
iteration

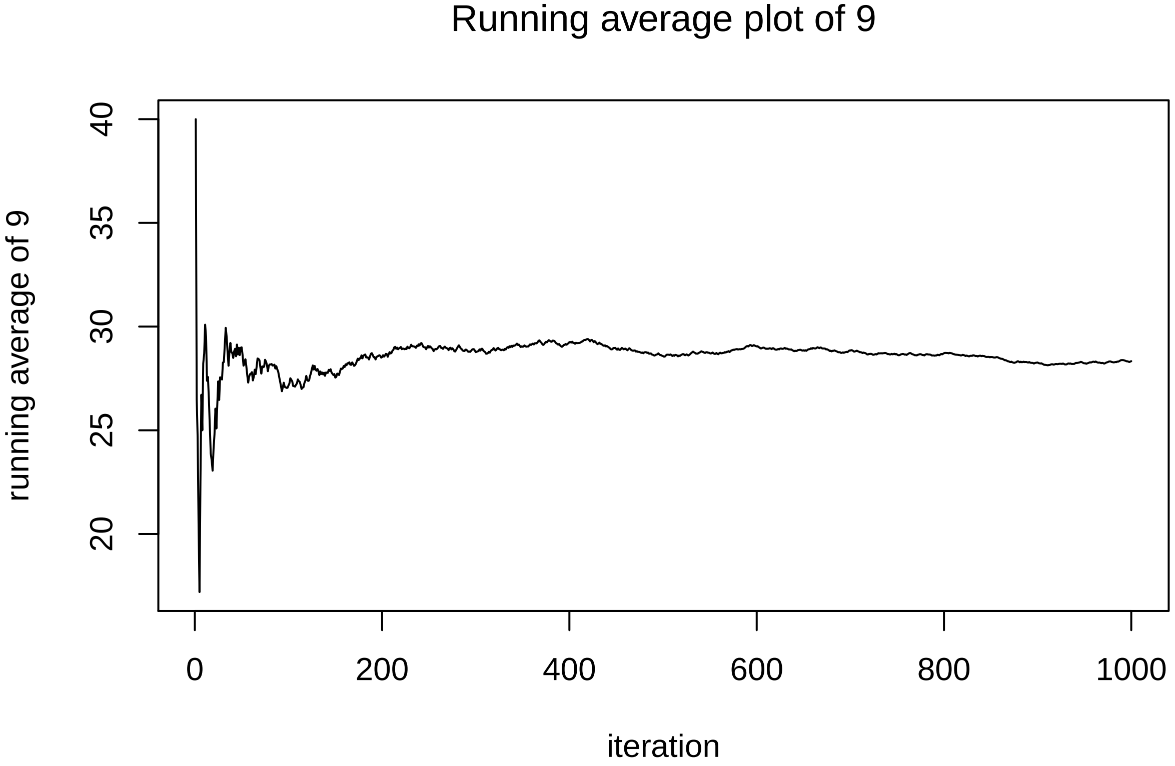

Running average plot of 4
running average of 4
35
30
25
20
0
200
400
600
800
1000
iteration

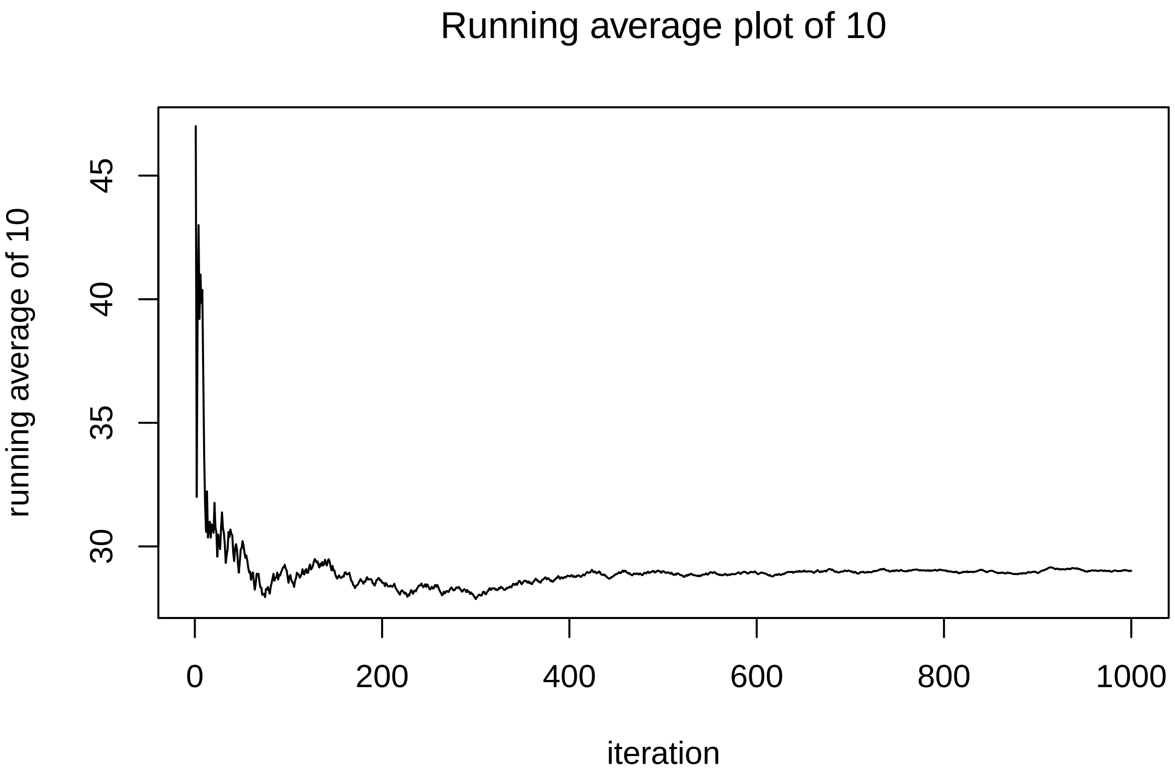

Running average plot of 5
running average of 5
30
25
20
15
10
0
200
400
600
800
1000
iteration

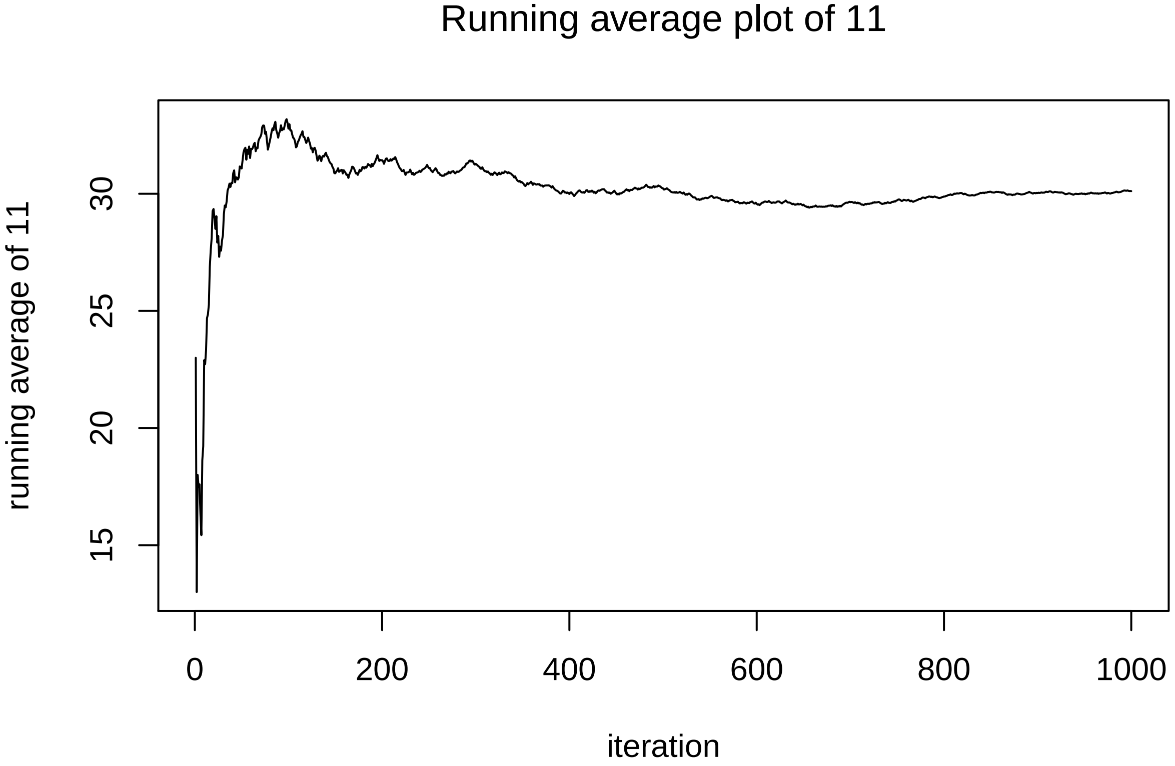

Running average plot of 6
running average of 6
5
10
15
20
25
30
0
200
400
600
800
1000
iteration

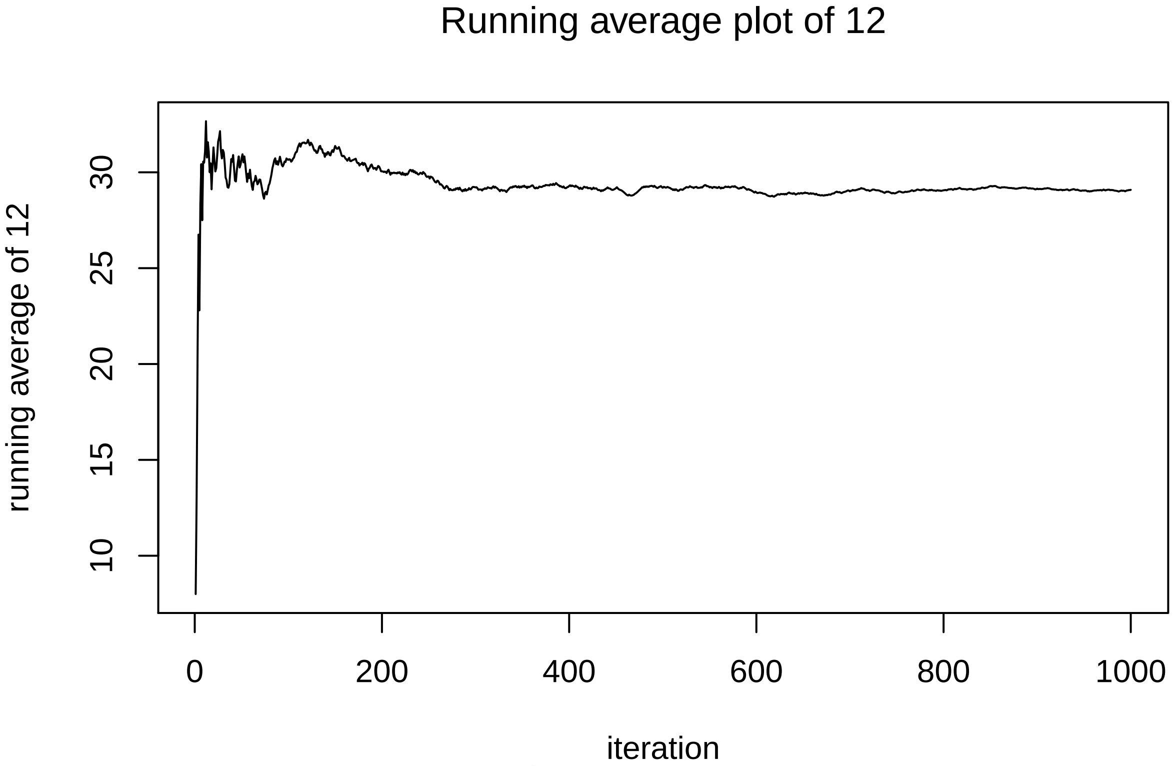

Running average plot of 7
running average of 7
22
24
26
28
30
32
0
200
400
600
800
1000
iteration

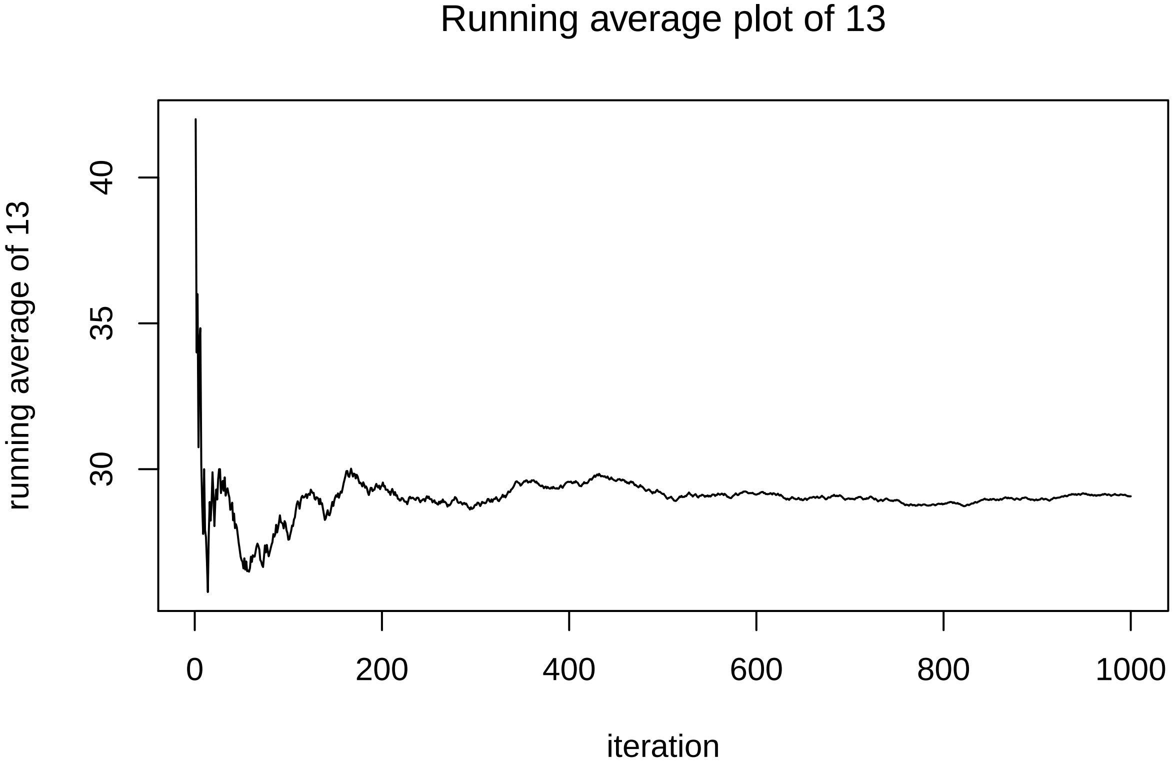

Running average plot of 8

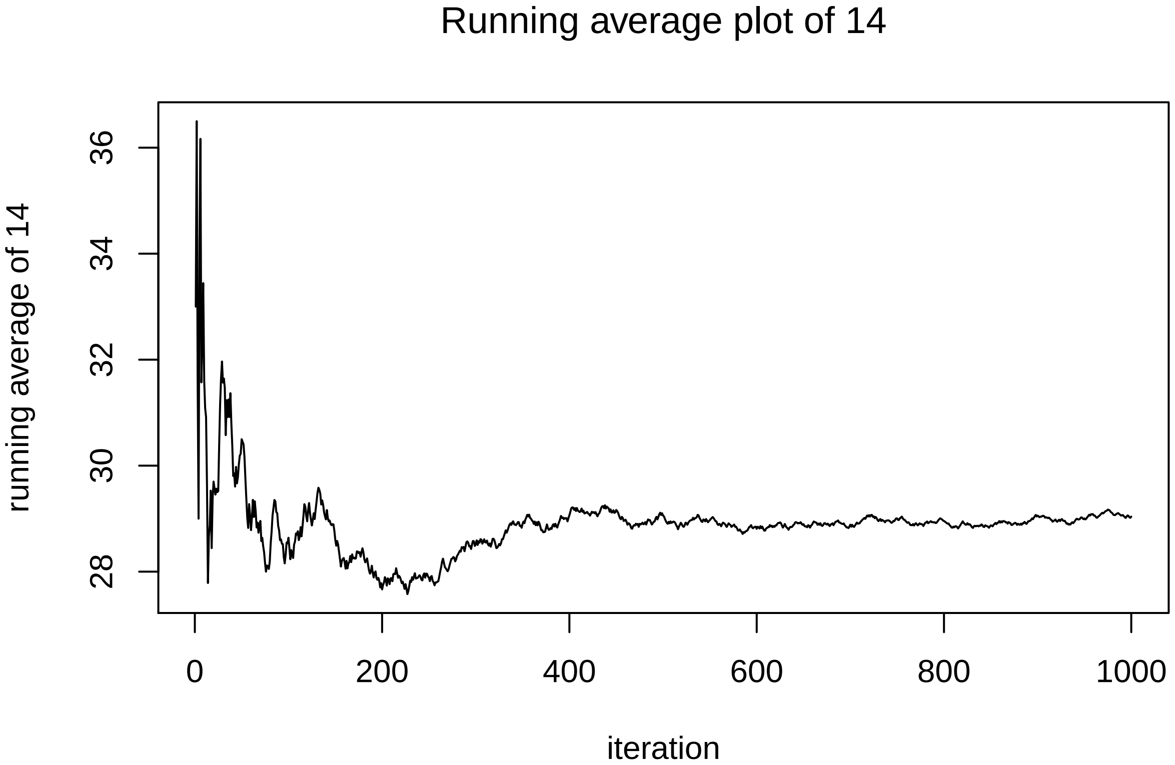

Running average plot of 9

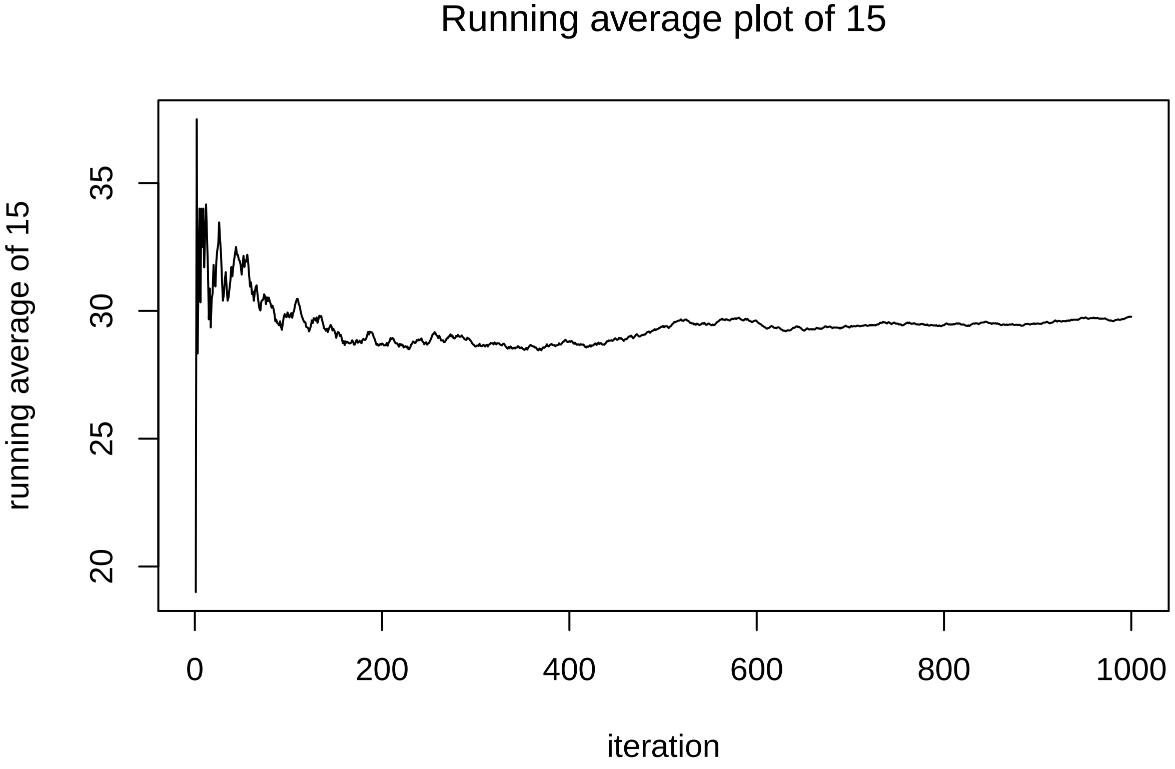

Running average plot of 10
running average of 10
45
40
35
30
0
200
400
600
800
1000
iteration

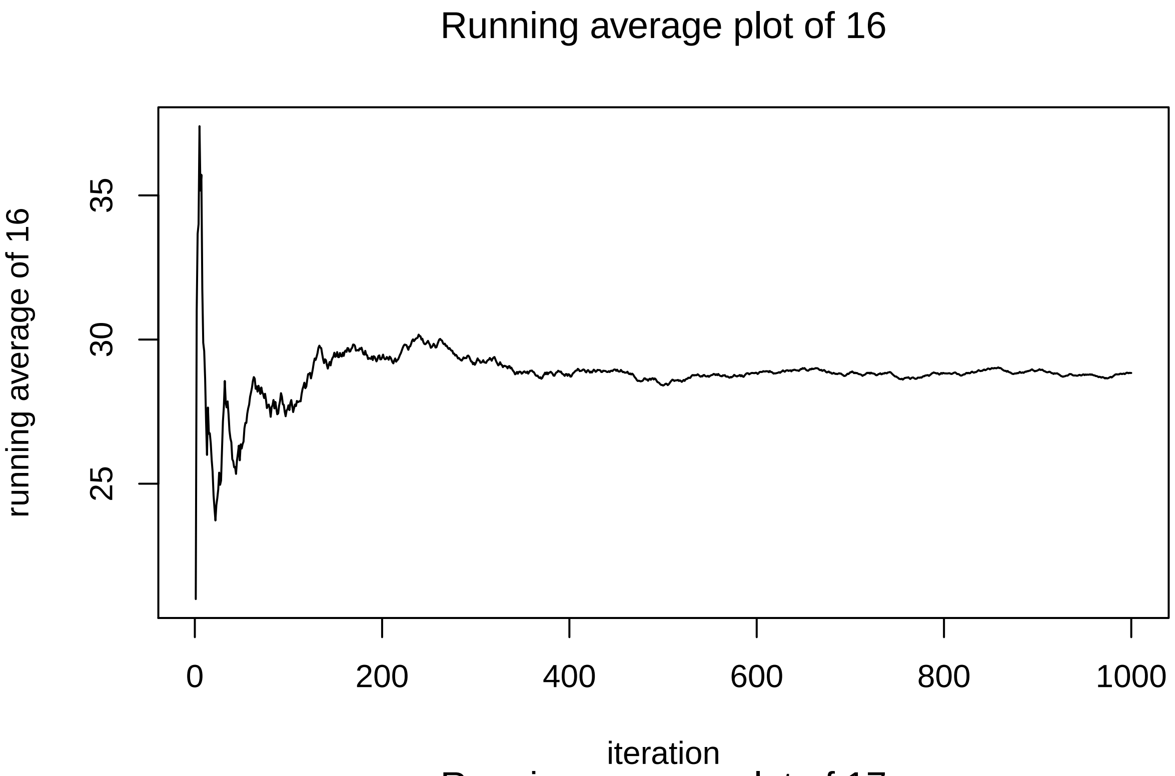

Running average plot of 11
running average of 11
30
25
20
15
0
200
400
600
800
1000
iteration

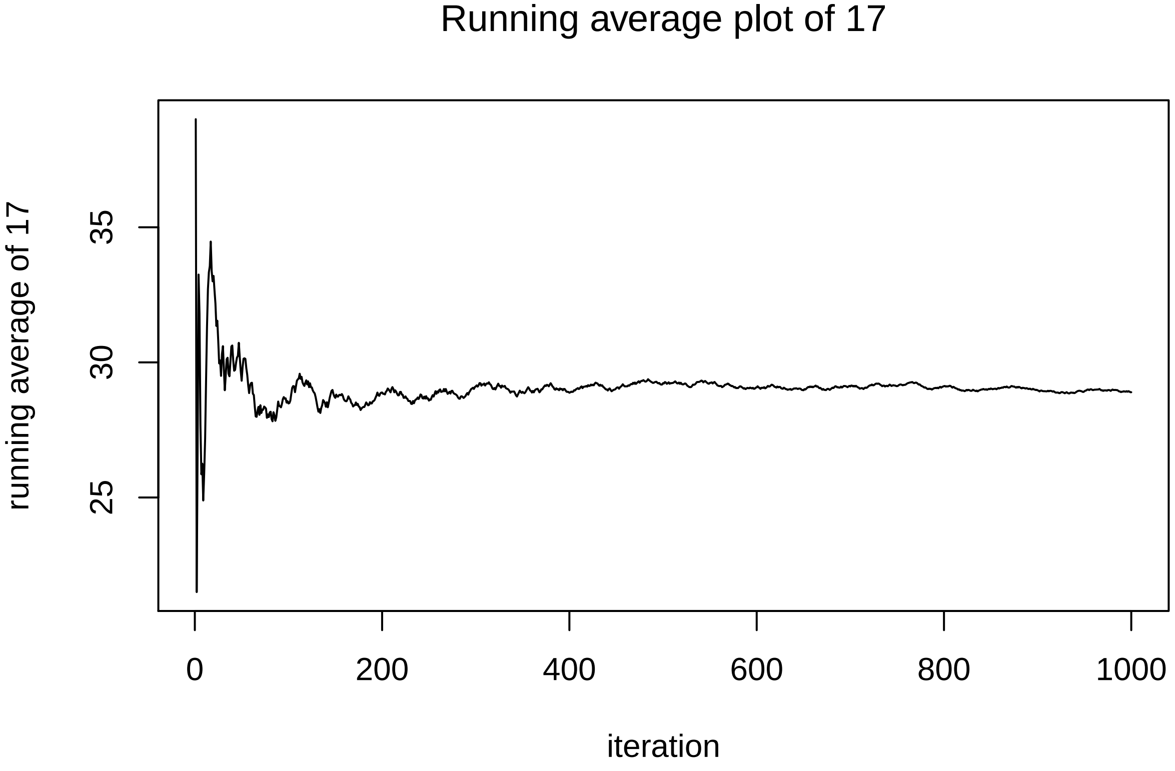

Running average plot of 12
running average of 12
10
15
20
25
30
0
200
400
600
800
1000
iteration

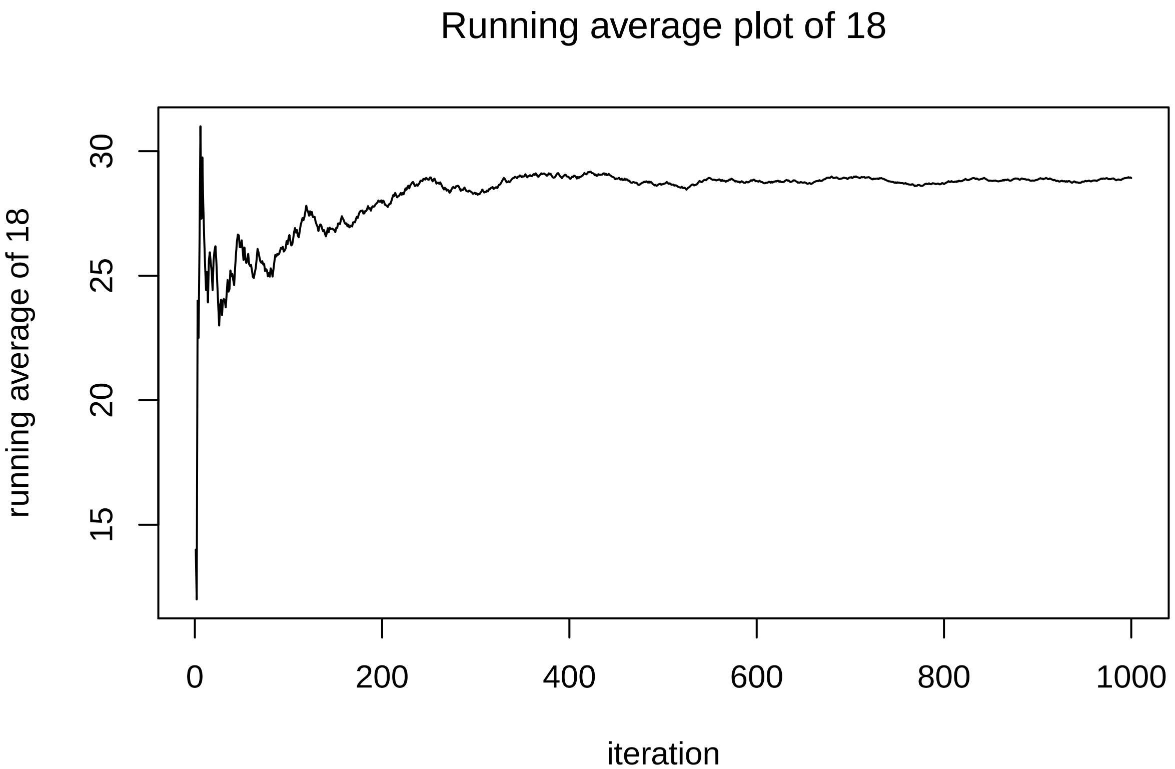

Running average plot of 13
running average of 13
30
35
40
0
200
400
600
800
1000
iteration

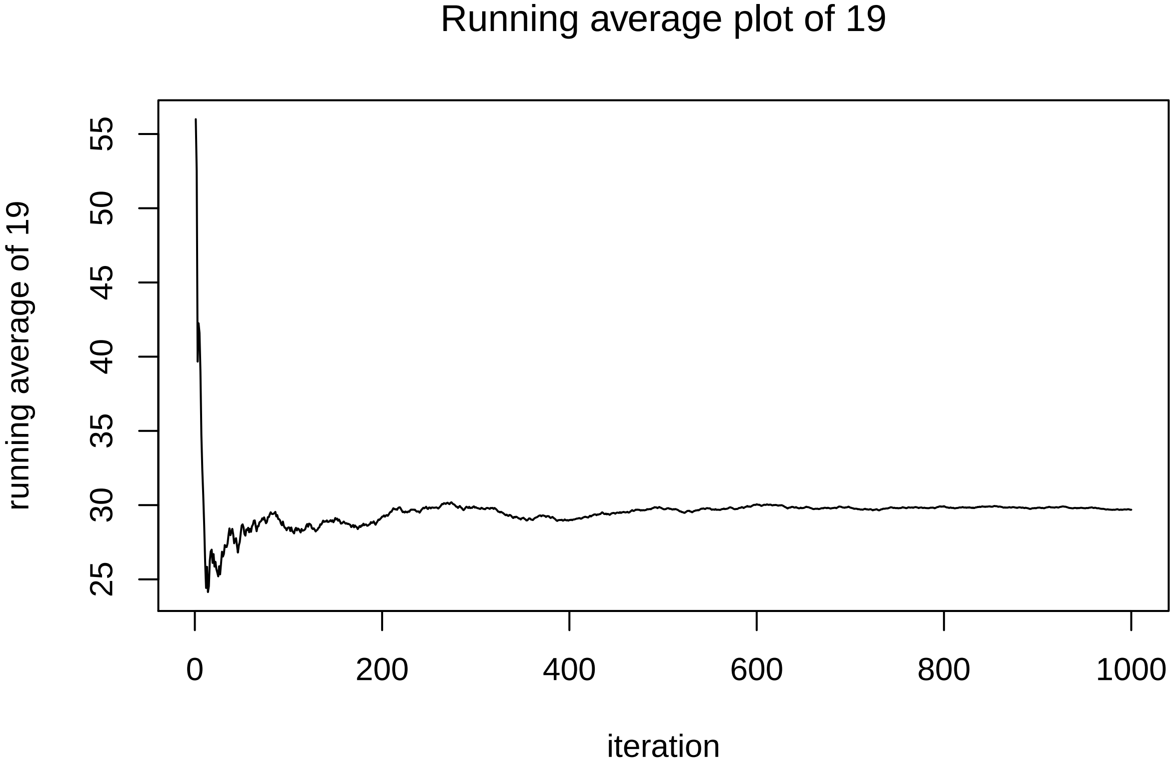

Running average plot of 14
running average of 14
28
30
32
34
36
0
200
400
600
800
1000
iteration

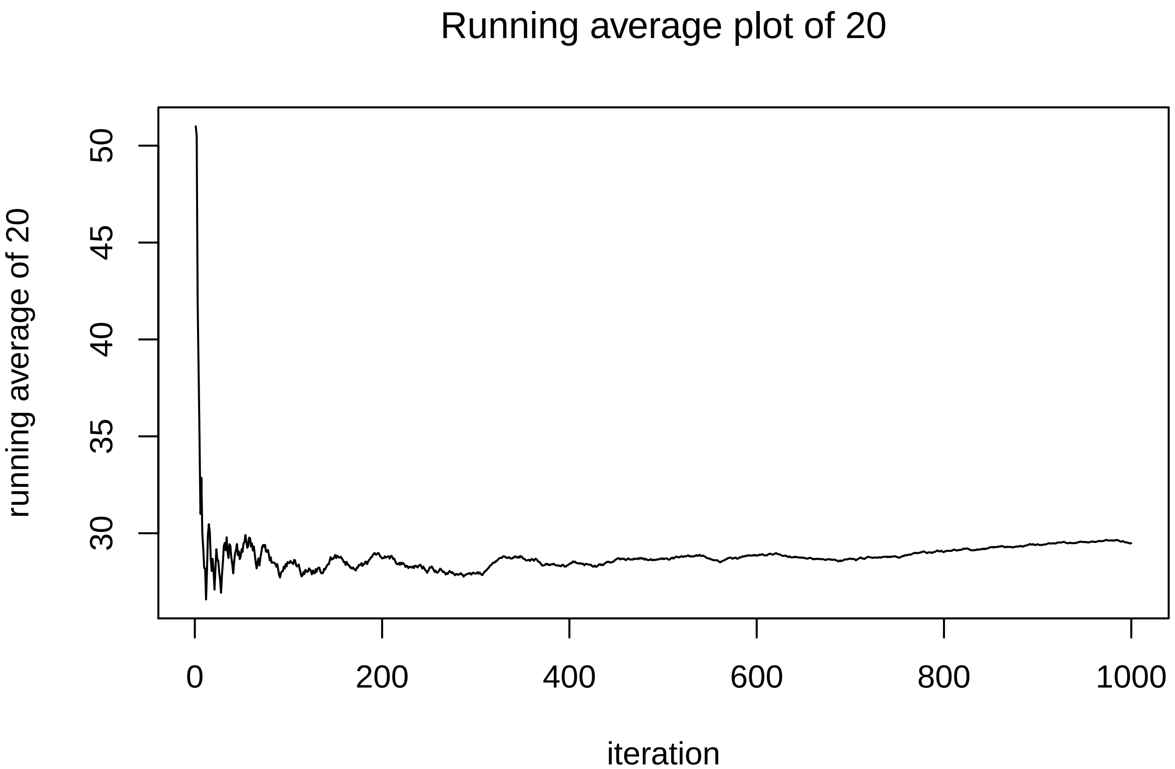

Running average plot of 15
running average of 15
20
25
30
35
0
200
400
600
800
1000
iteration

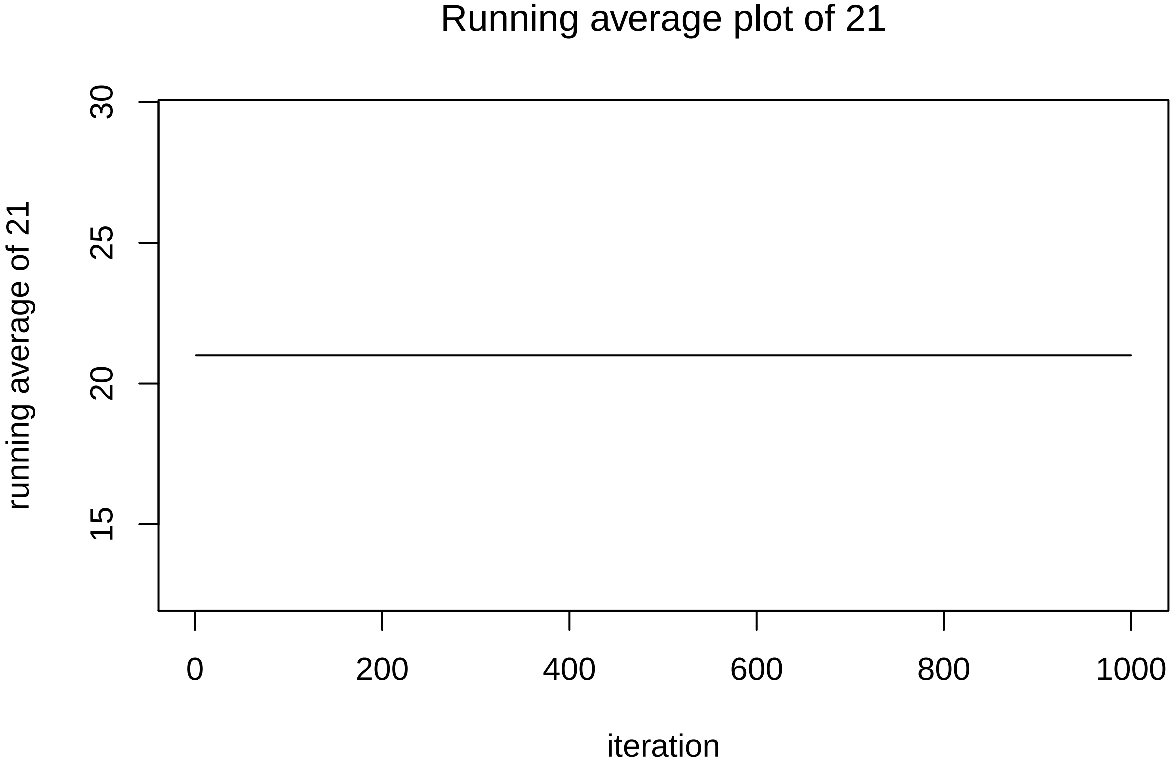

Running average plot of 16
running average of 16
25
30
35
0
200
400
600
800
1000
iteration

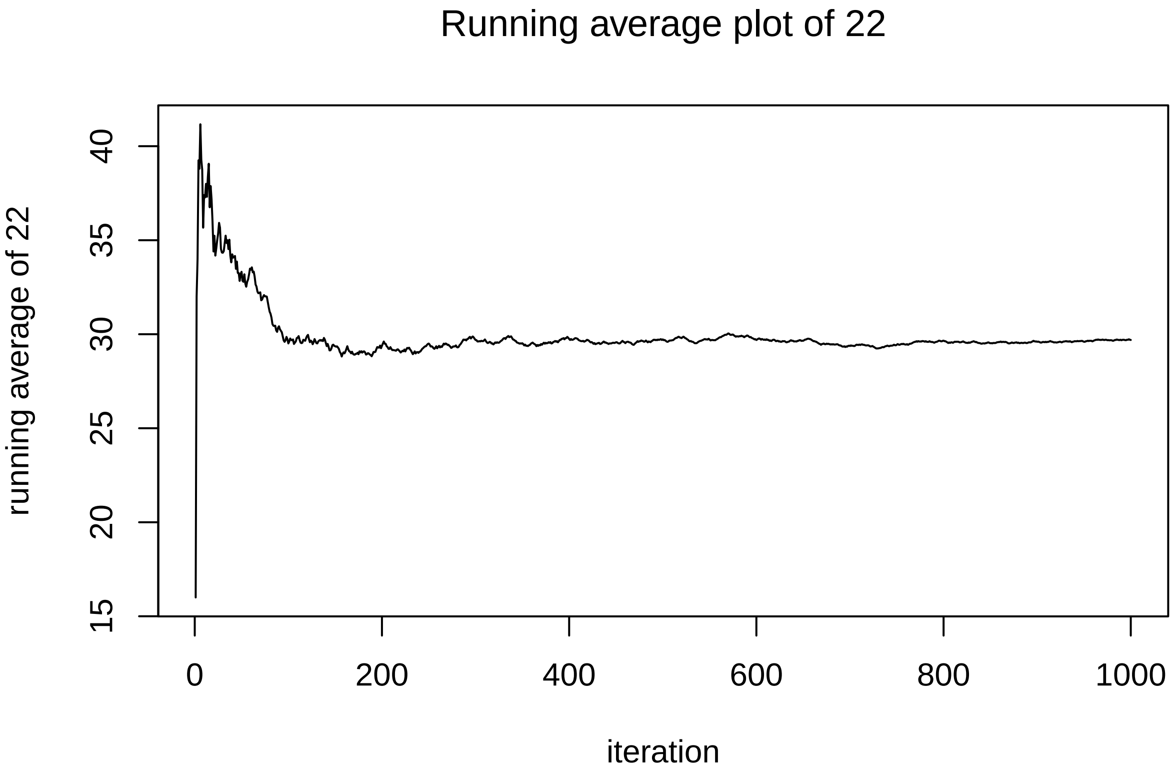

Running average plot of 17
running average of 17
25
30
35
0
200
400
600
800
1000
iteration

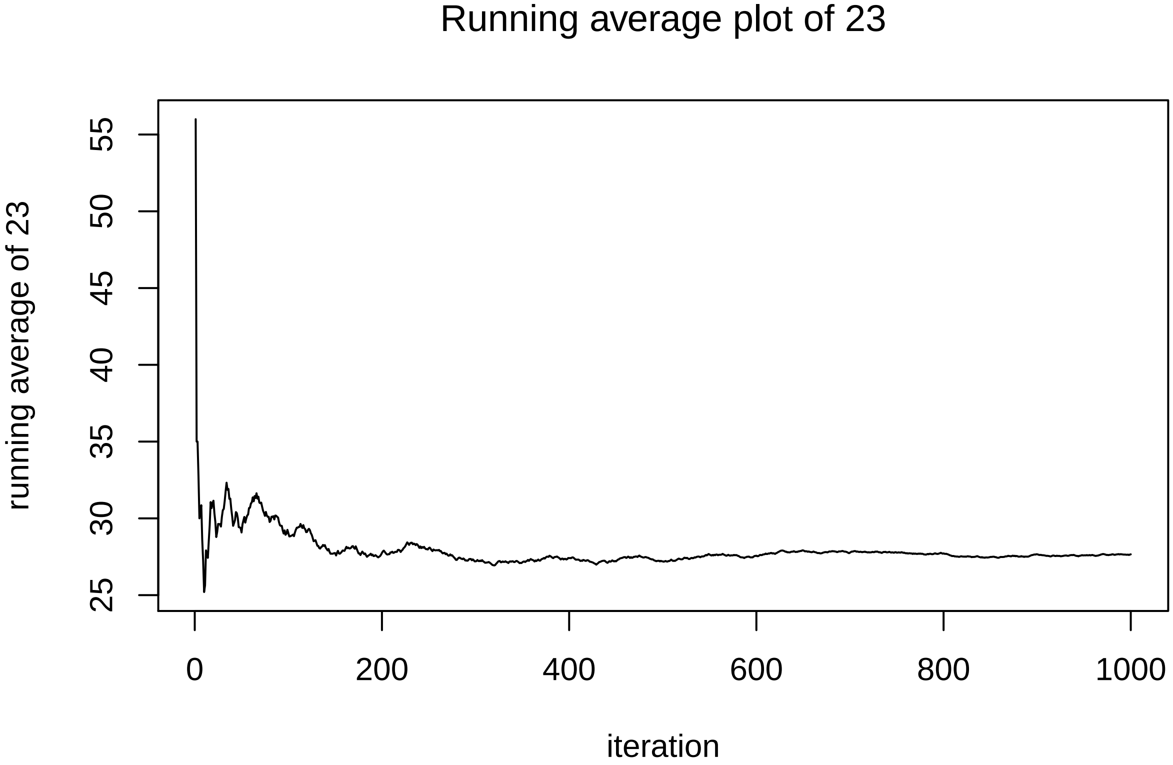

Running average plot of 18
running average of 18
15
20
25
30
0
200
400
600
800
1000
iteration

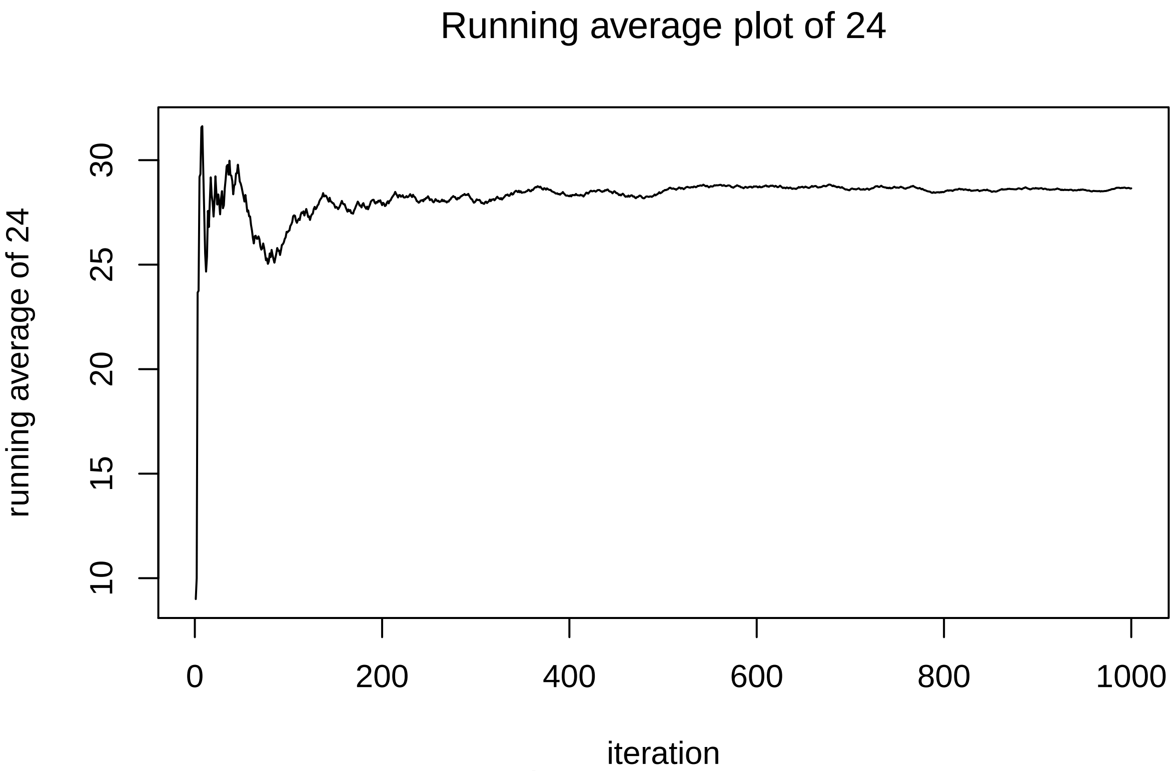

Running average plot of 19
running average of 19
25
30
35
40
45
50
55
0
200
400
600
800
1000
iteration

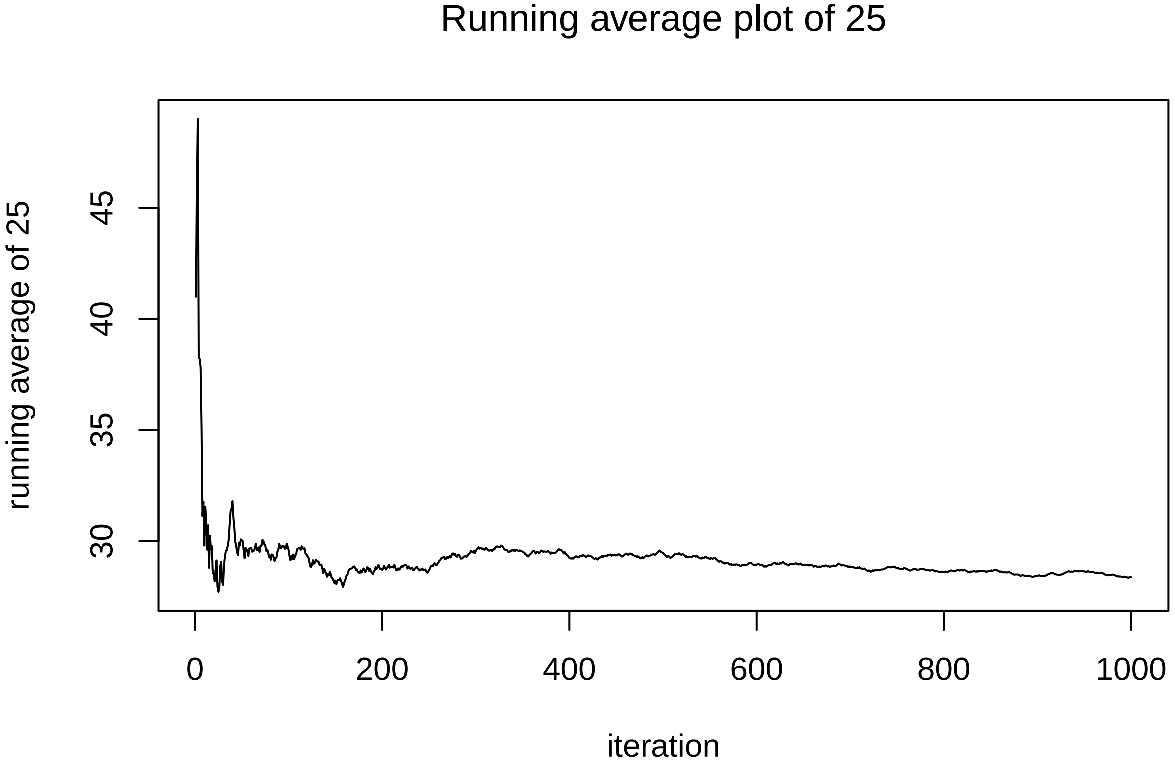

Running average plot of 20
running average of 20
50
45
40
35
30
0
200
400
600
800
1000
iteration

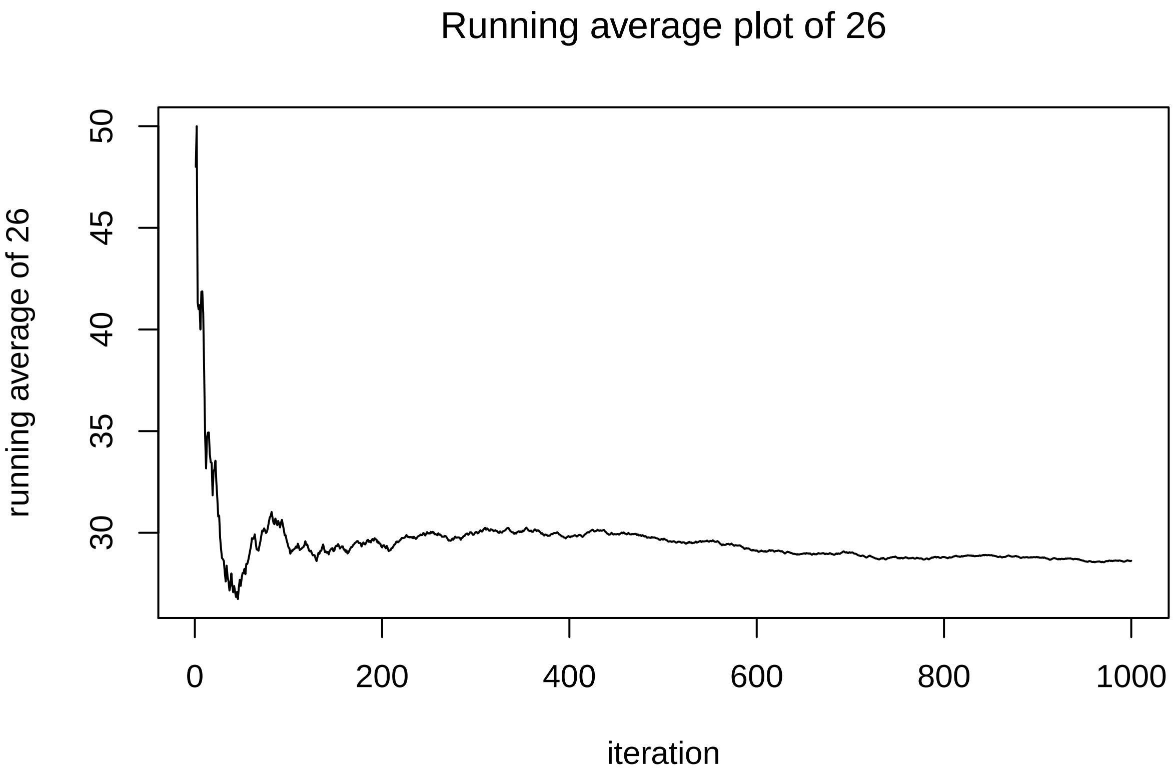

Running average plot of 21
running average of 21
30
25
20
15
0
200
400
600
800
1000
iteration

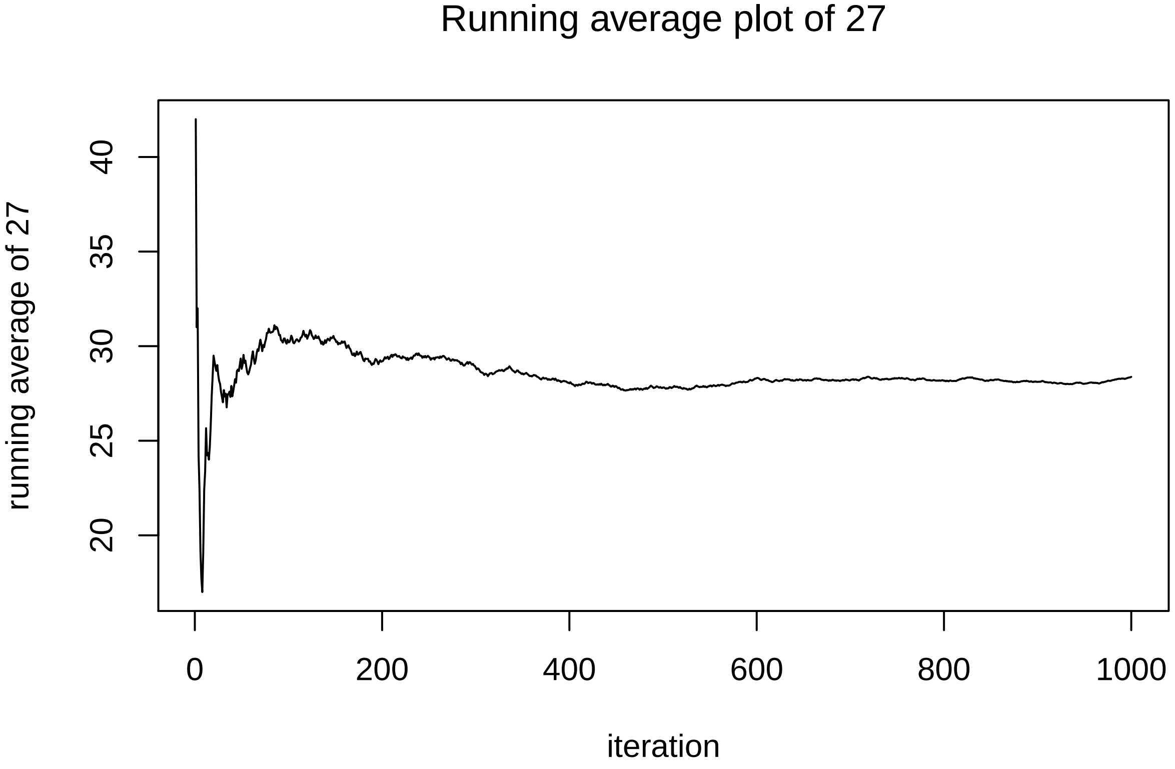

Running average plot of 22
running average of 22
15
20
25
30
35
40
0
200
400
600
800
1000
iteration

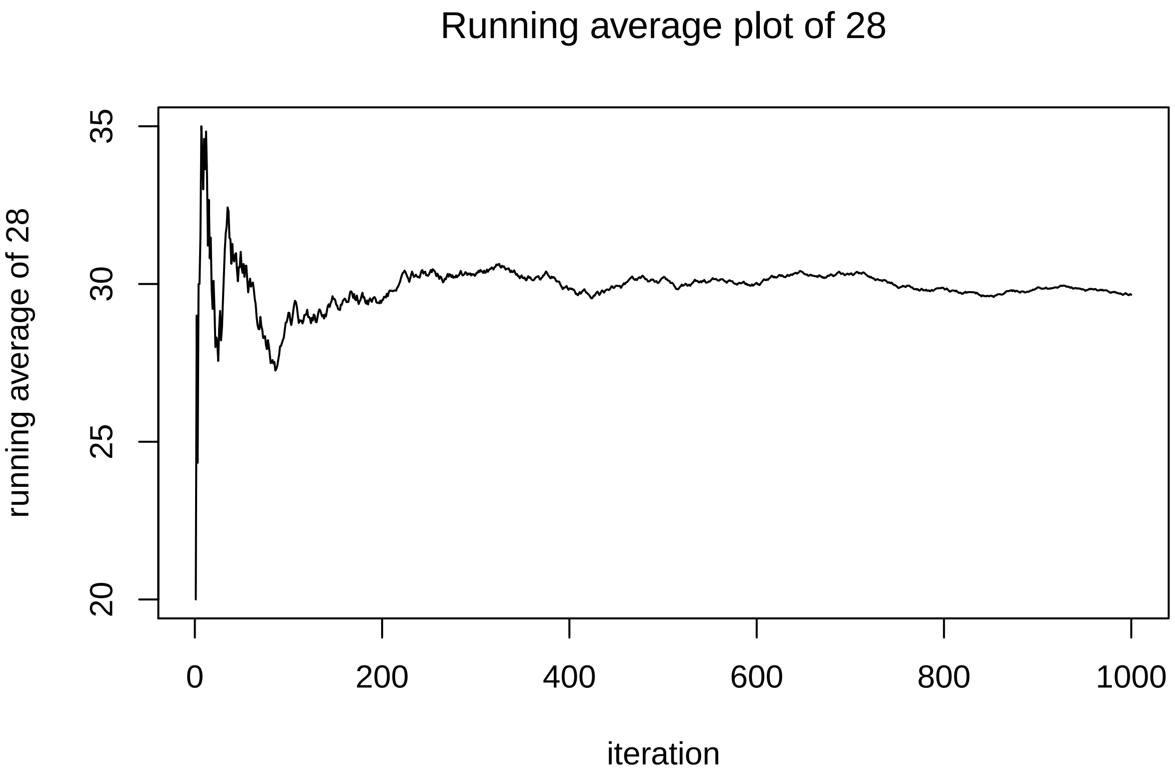

Running average plot of 23
running average of 23
25
30
35
40
45
50
55
0
200
400
600
800
1000
iteration

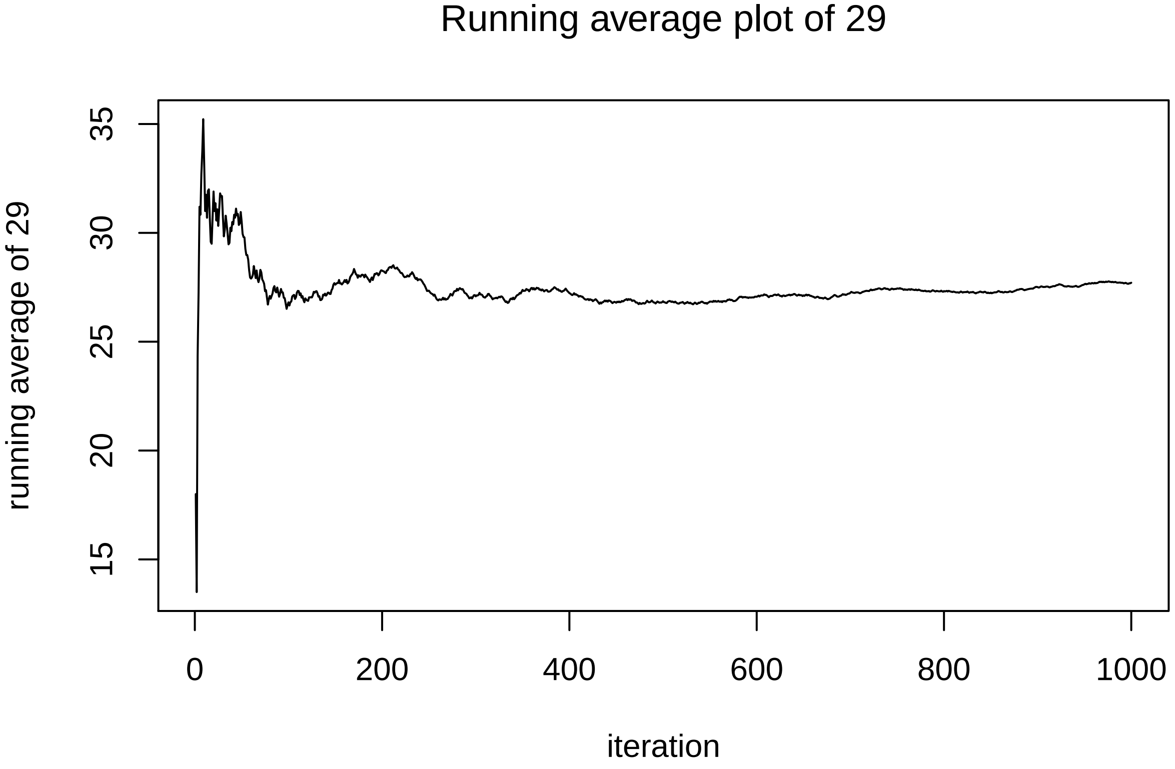

Running average plot of 24

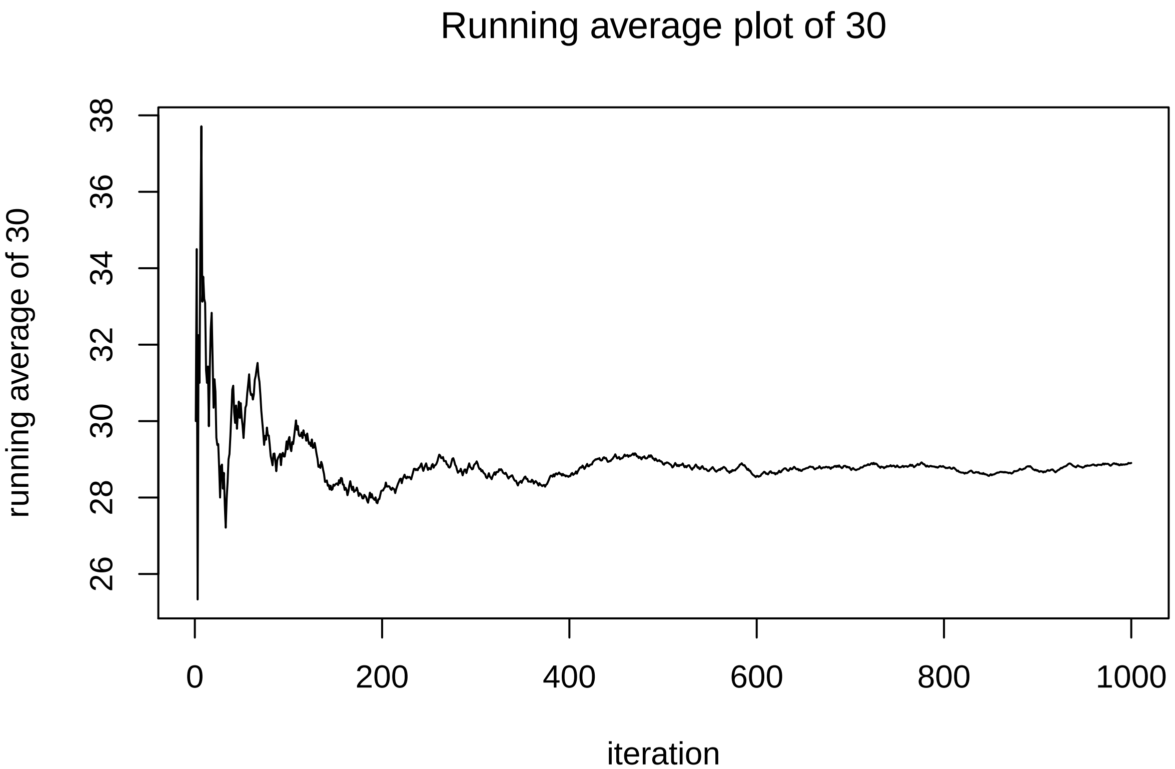

Running average plot of 25

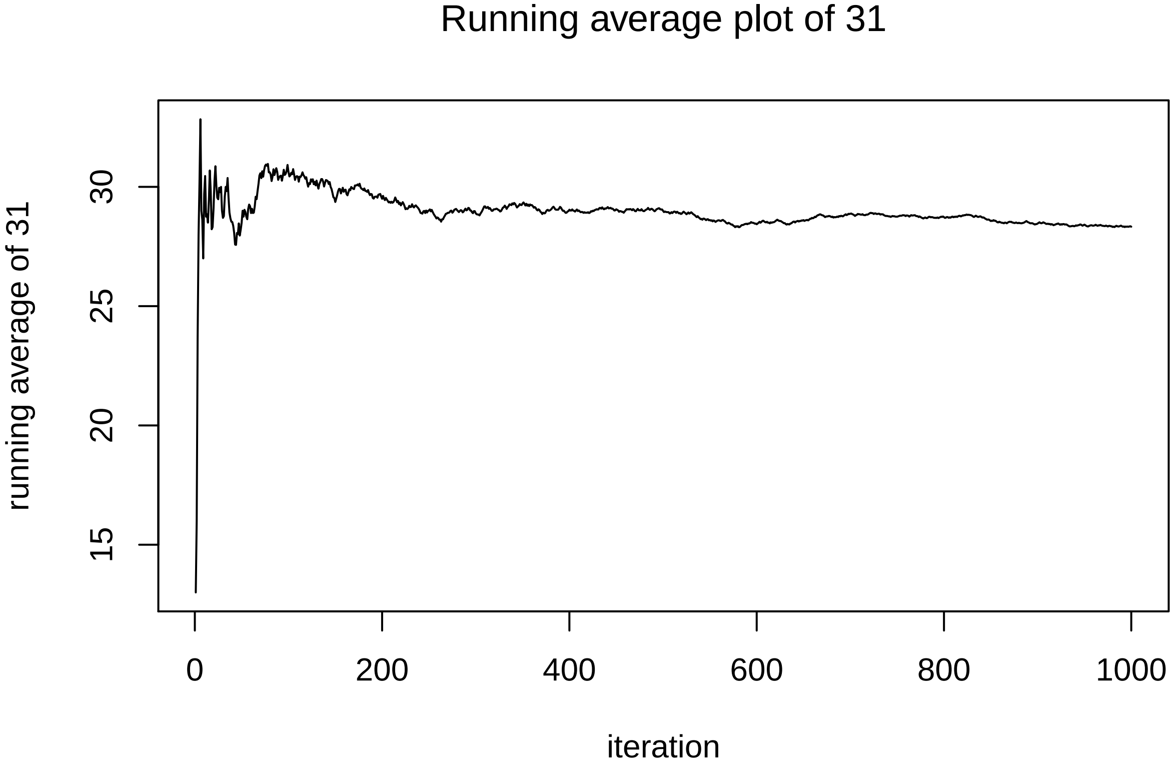

Running average plot of 26
running average of 26
50
45
40
35
30
0
200
400
600
800
1000
iteration


Running average plot of 27
running average of 27
40
35
30
25
20
0
200
400
600
800
1000
iteration

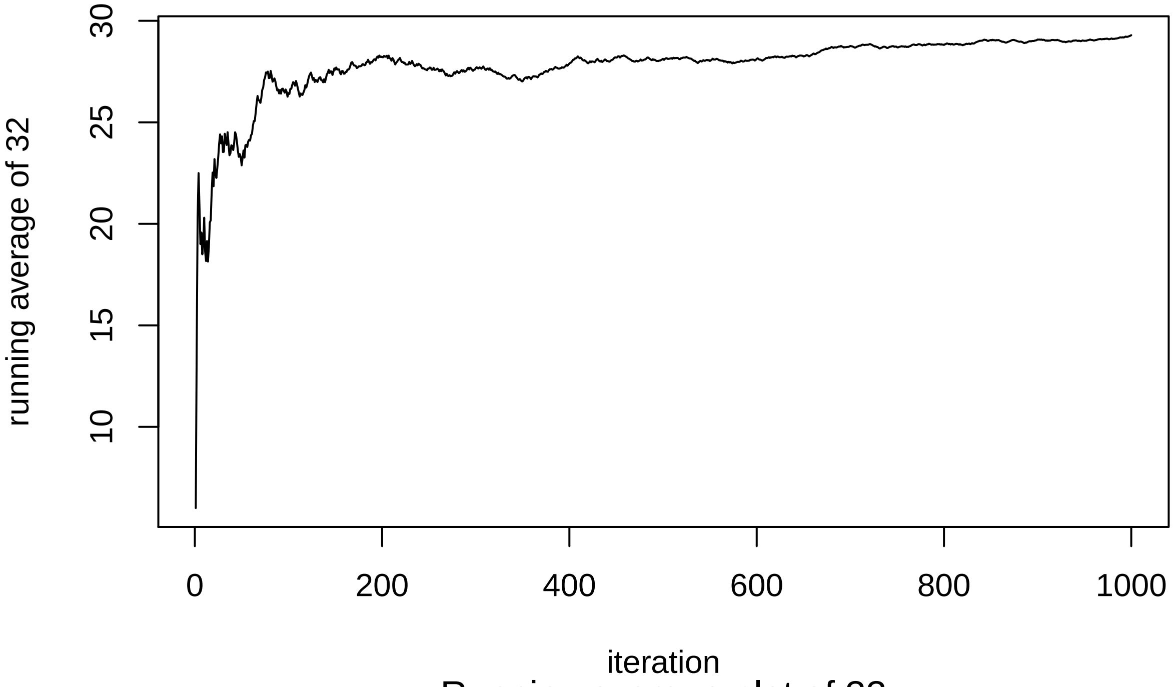

Running average plot of 28
running average of 28
20
25
30
35
0
200
400
600
800
1000
iteration


Running average plot of 29
running average of 29
15
20
25
30
35
0
200
400
600
800
1000
iteration

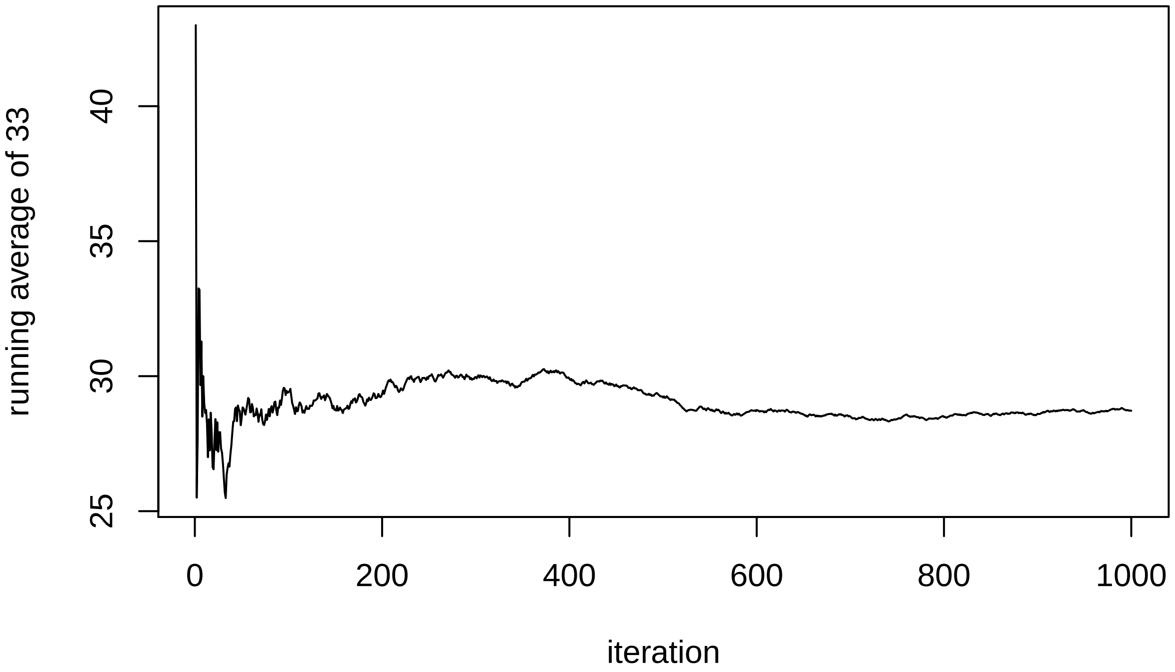

Running average plot of 30
running average of 30
26
28
30
32
34
36
38
0
200
400
600
800
1000
iteration

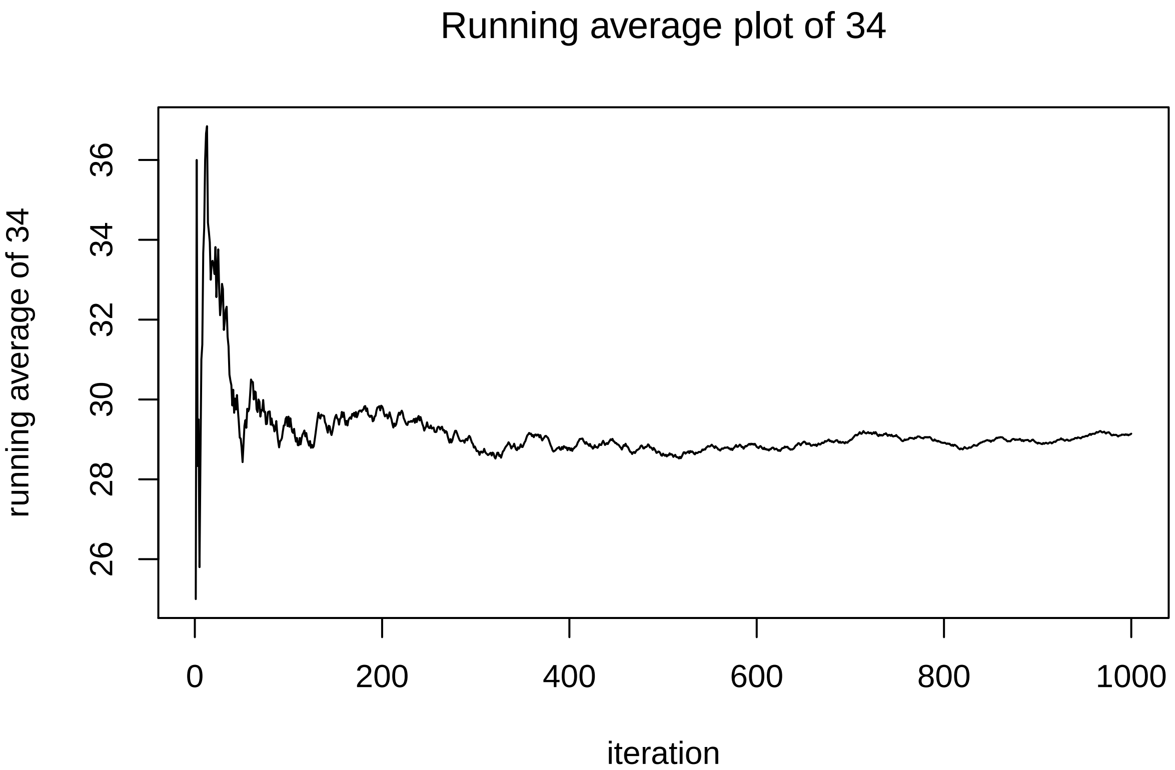

Running average plot of 31
running average of 31
15
20
25
30
0
200
400
600
800
1000
iteration

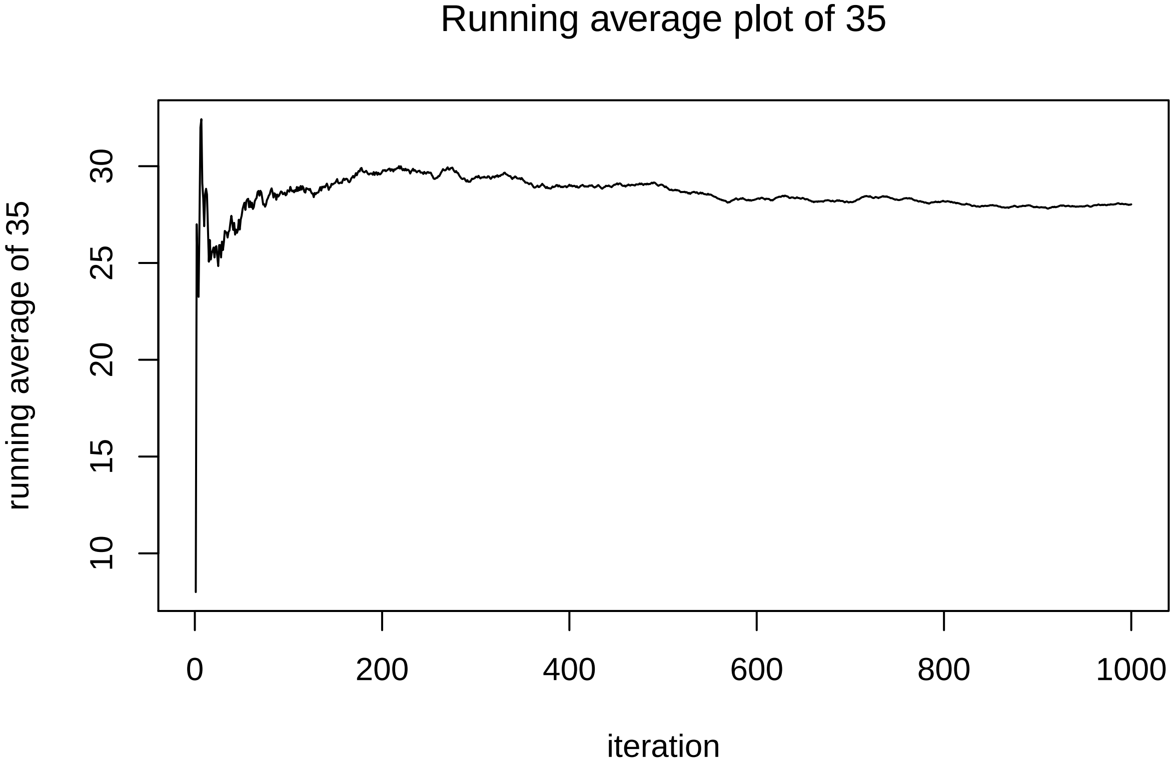
Running average plot of 32

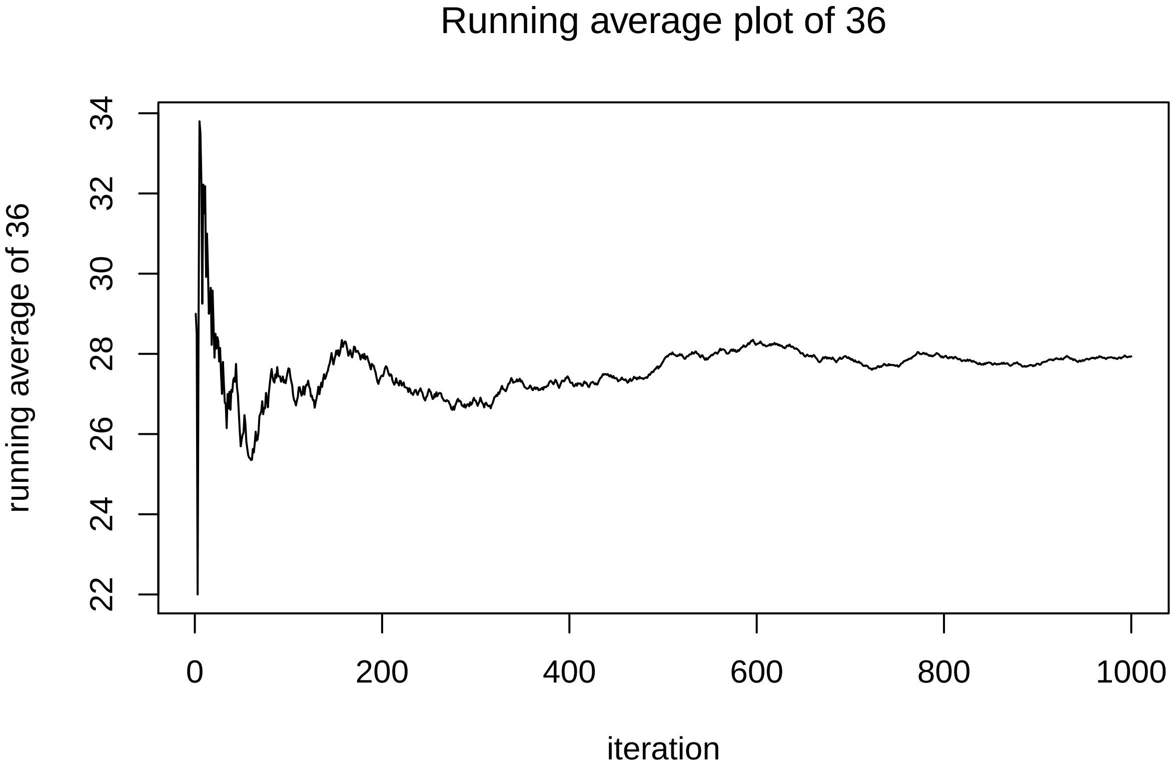


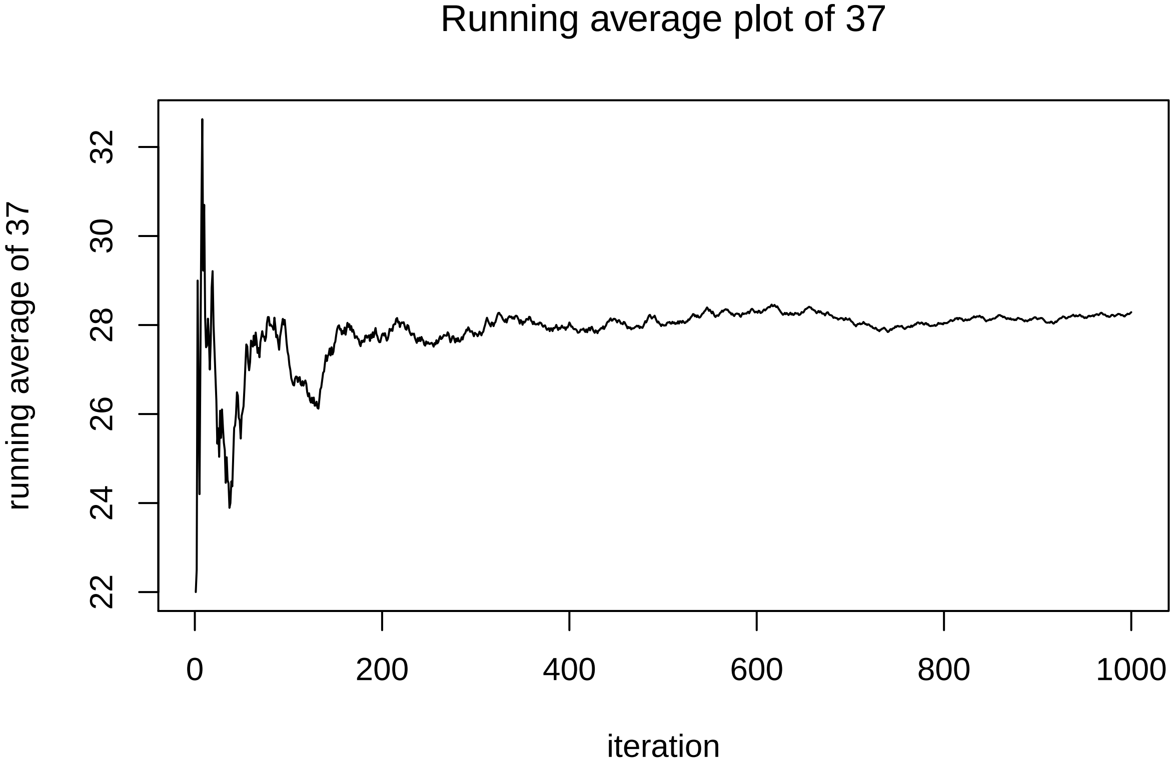
Running average plot of 33



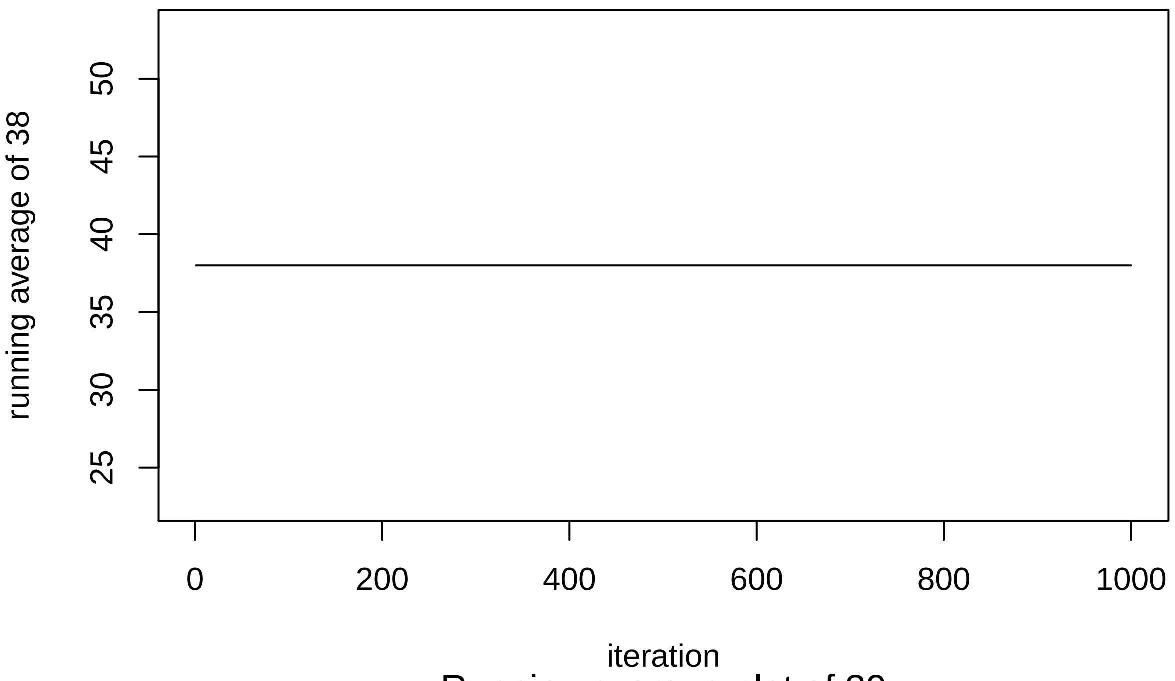

Running average plot of 34
running average of 34
26
28
30
32
34
36
0
200
400
600
800
1000
iteration


Running average plot of 35
running average of 35
10
15
20
25
30
0
200
400
600
800
1000
iteration

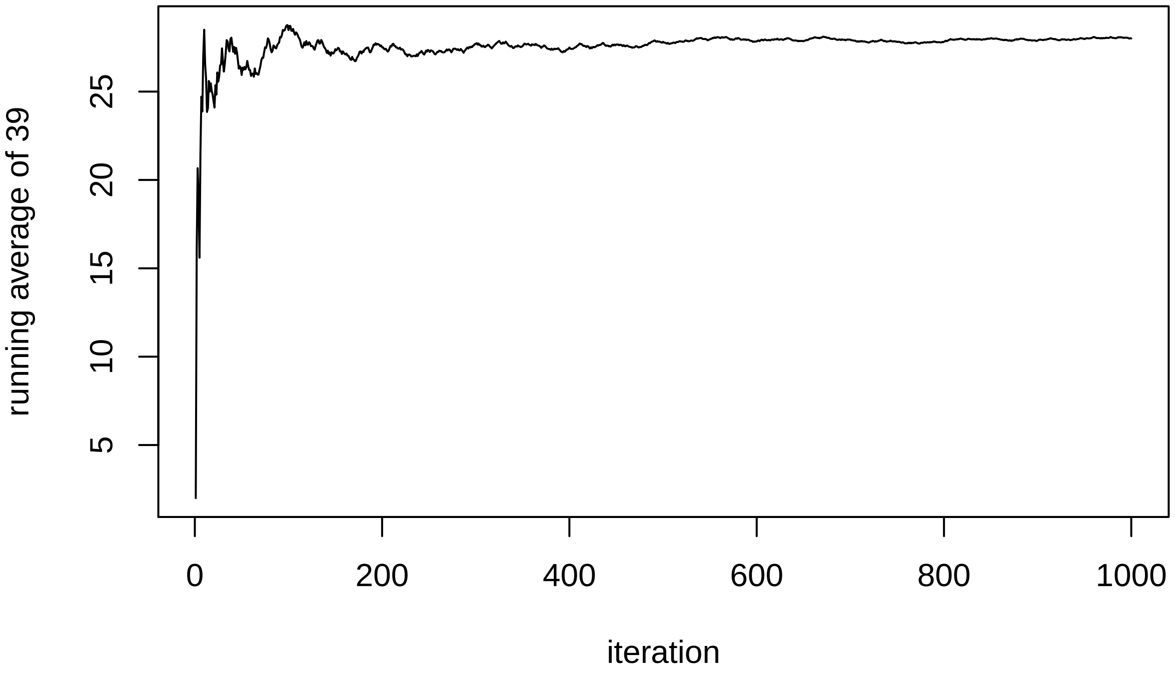

Running average plot of 36
running average of 36
22
24
26
28
30
32
34
0
200
400
600
800
1000
iteration


Running average plot of 37
running average of 37
22
24
26
28
30
32
0
200
400
600
800
1000
iteration

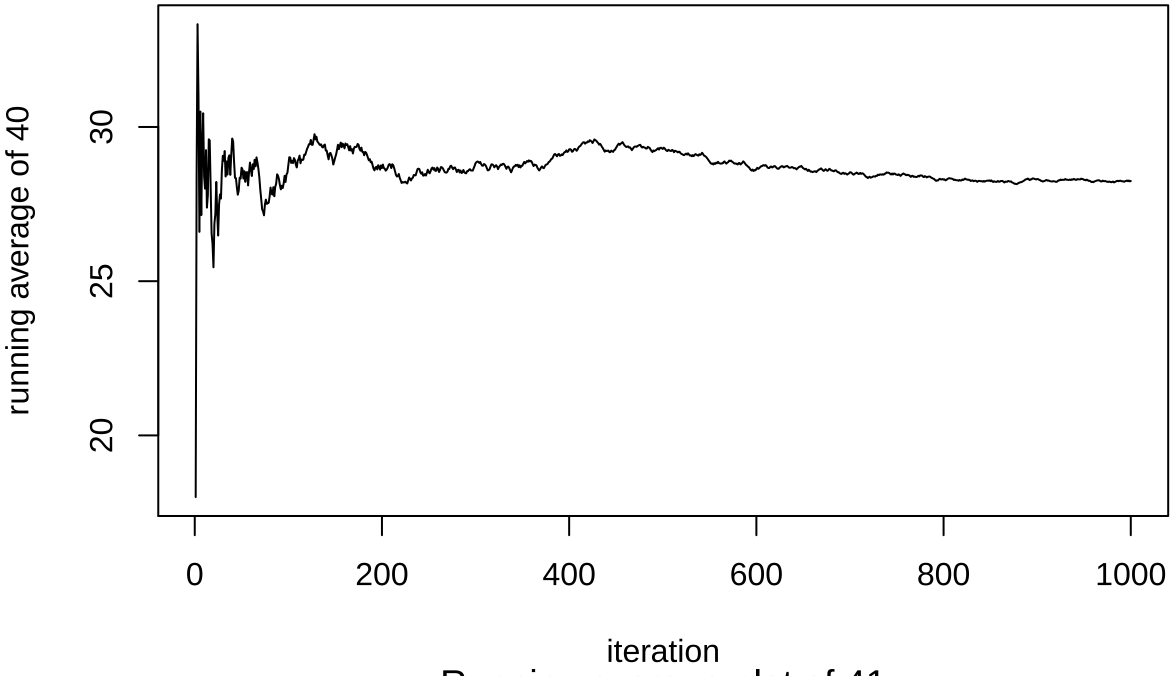
Running average plot of 38



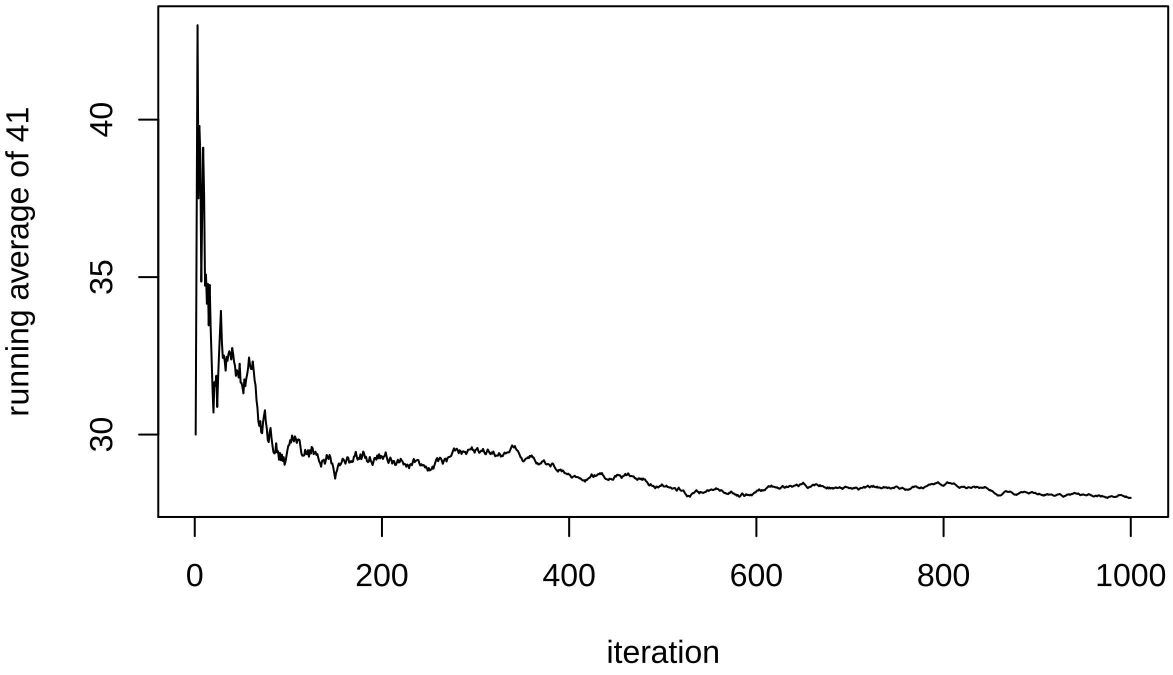
Running average plot of 39

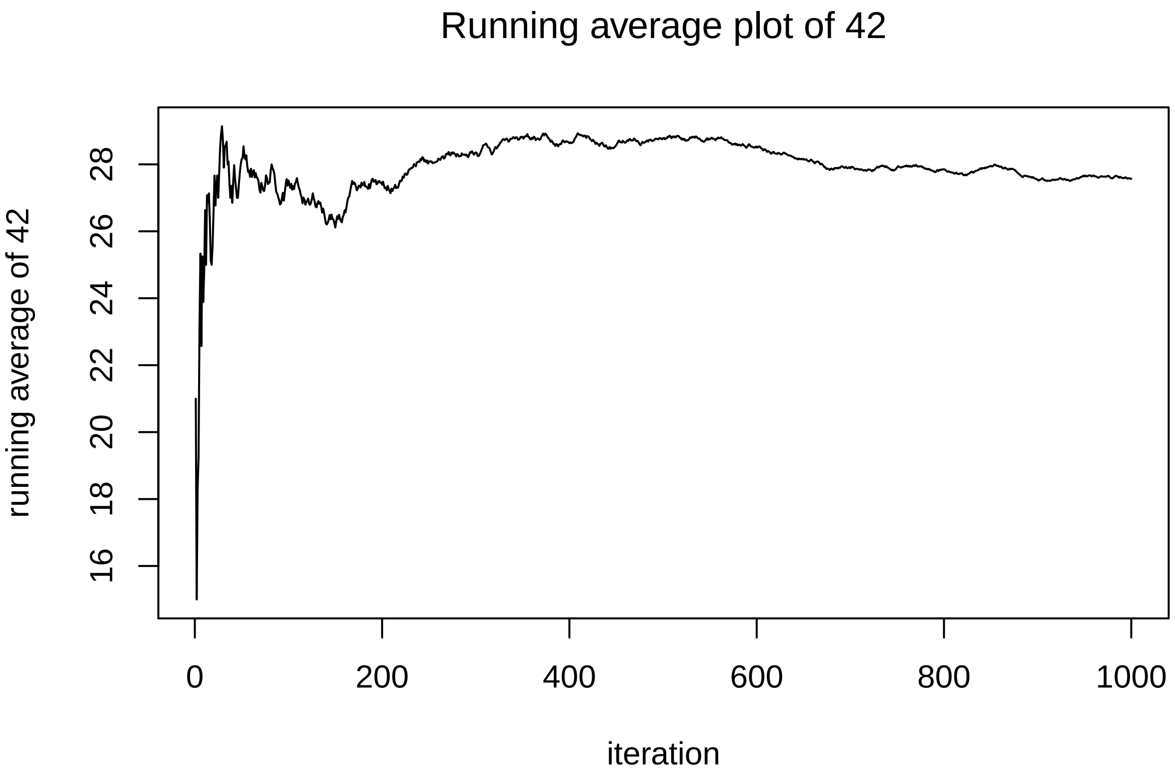


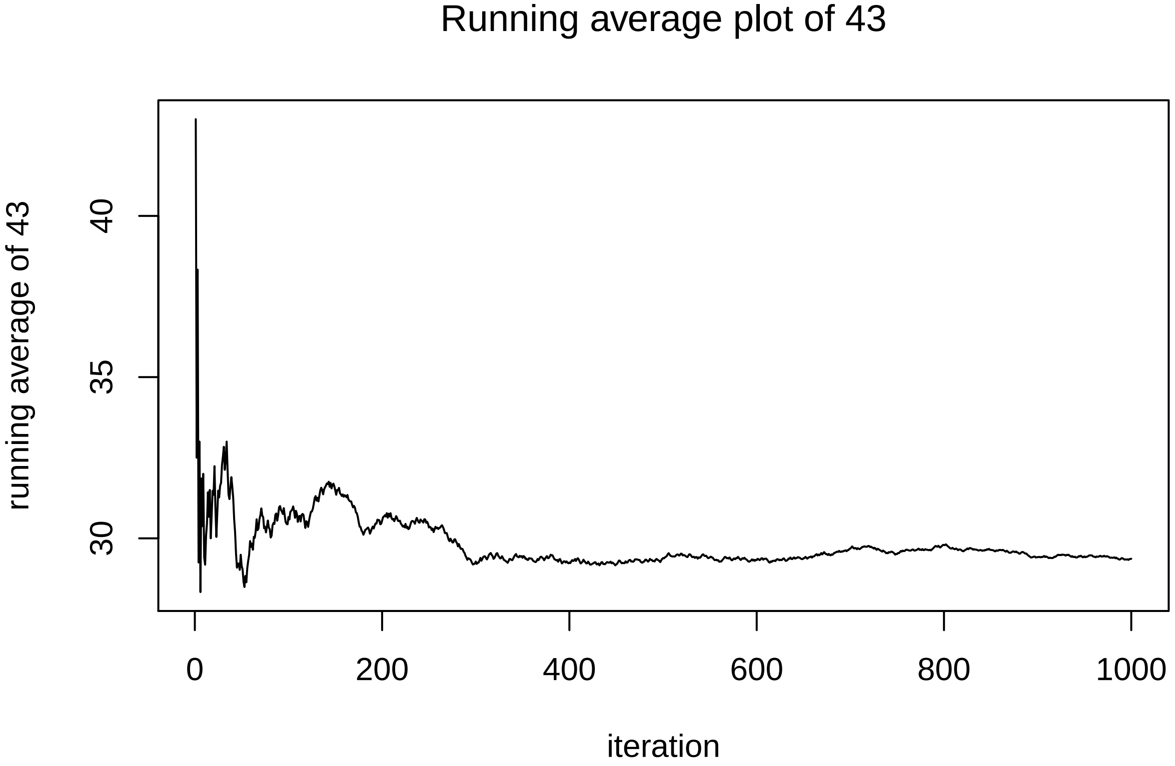
Running average plot of 40

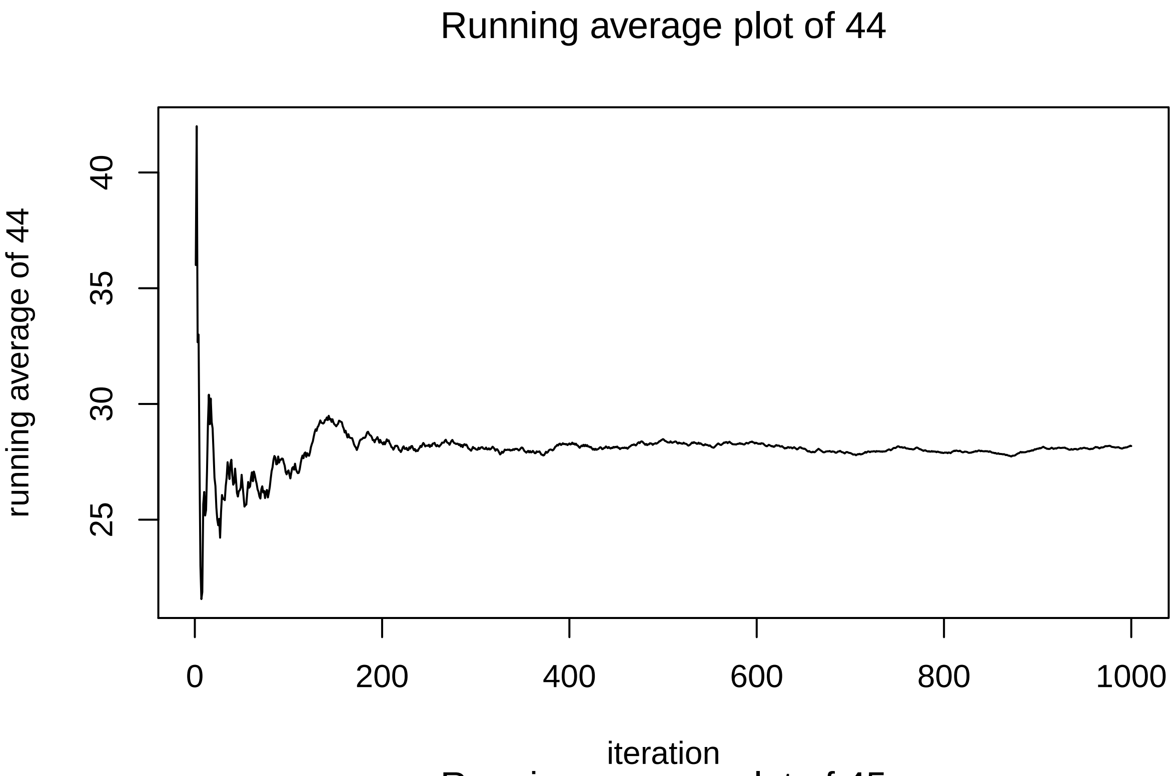


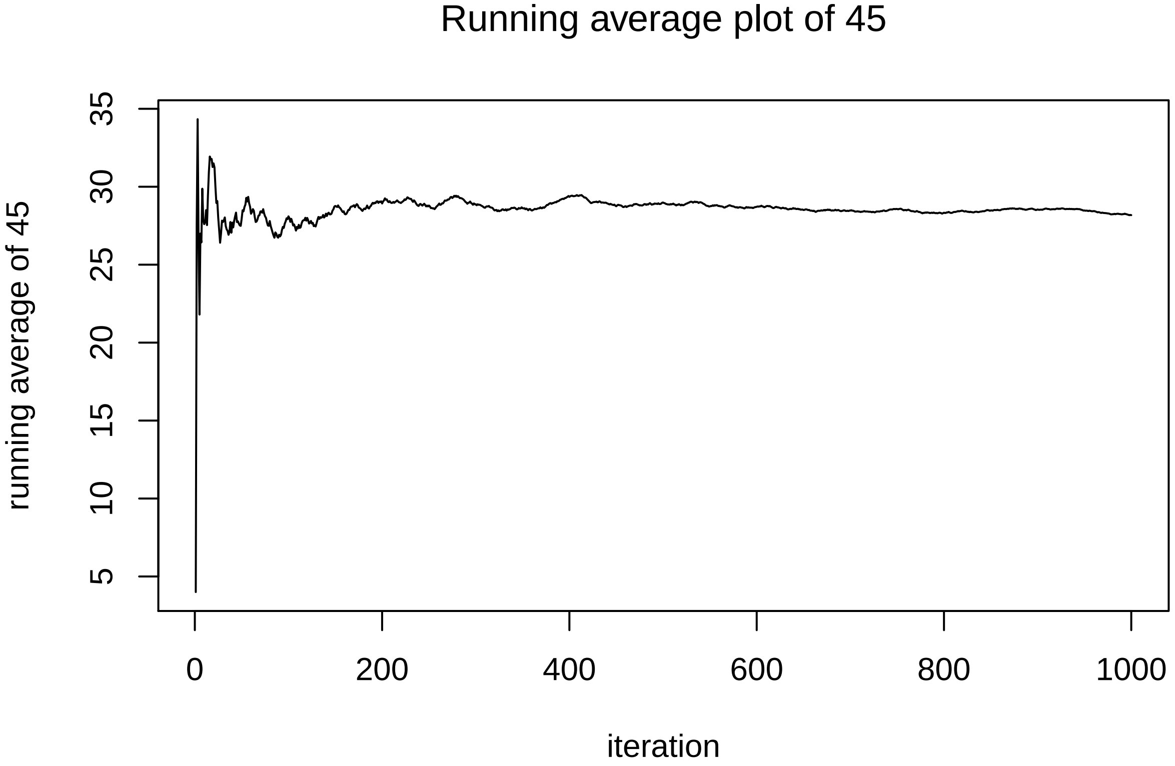
Running average plot of 41



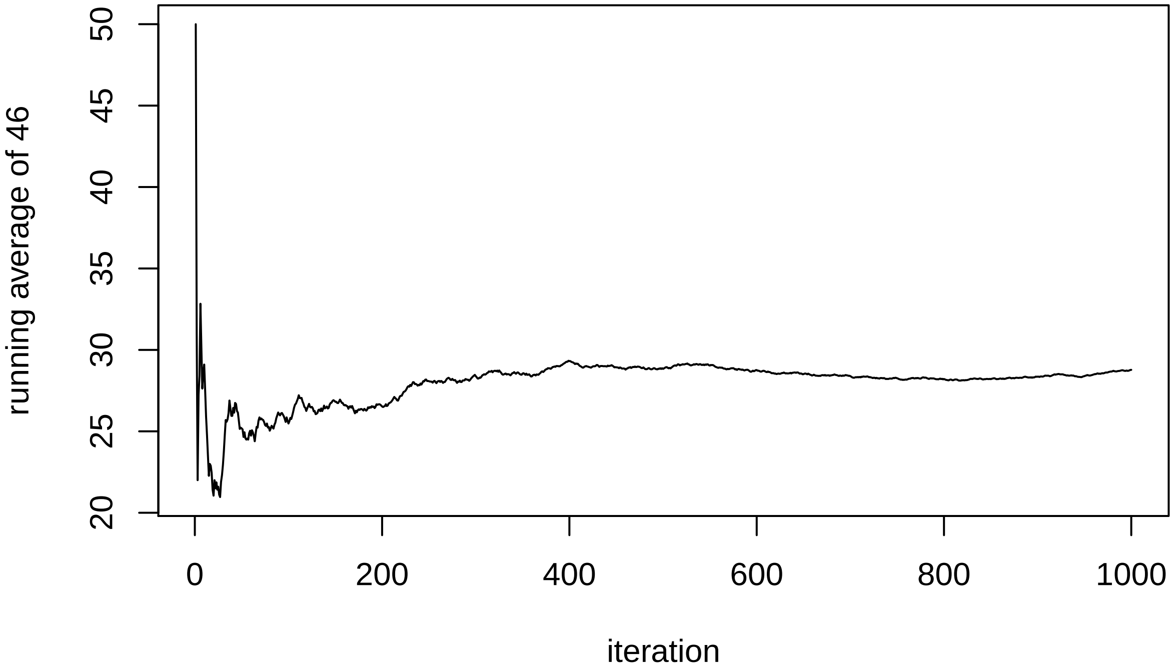

Running average plot of 42
running average of 42
16
18
20
22
24
26
28
0
200
400
600
800
1000
iteration


Running average plot of 43
running average of 43
30
35
40
0
200
400
600
800
1000
iteration

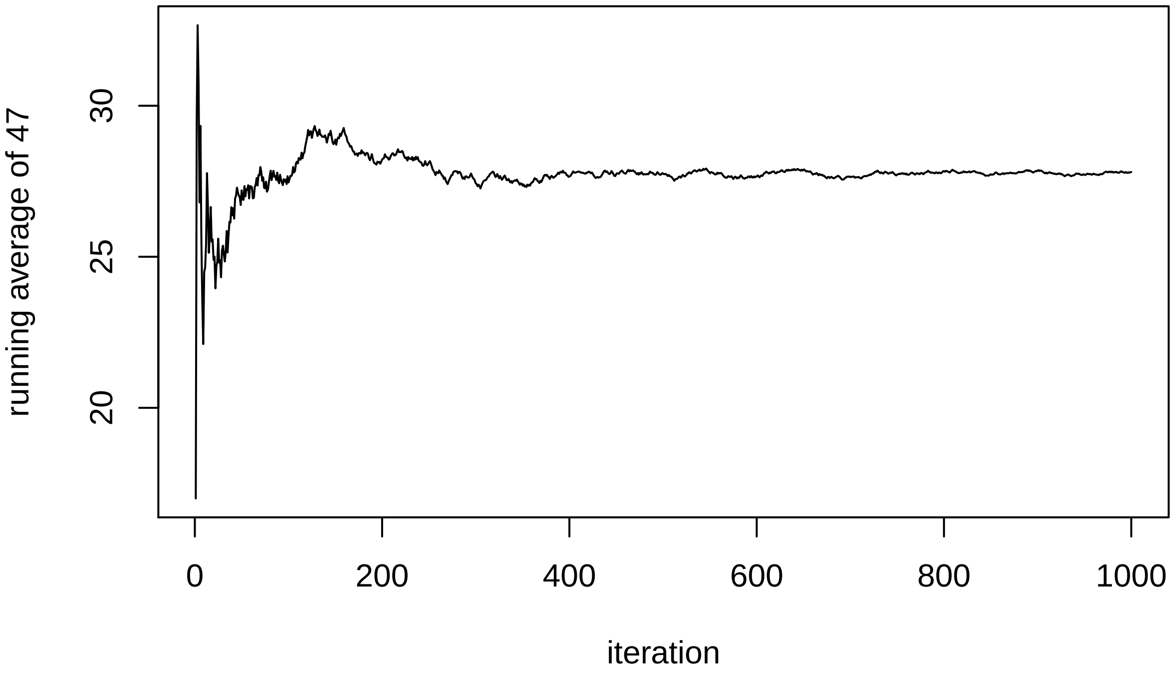

Running average plot of 44
running average of 44
25
30
35
40
0
200
400
600
800
1000
iteration

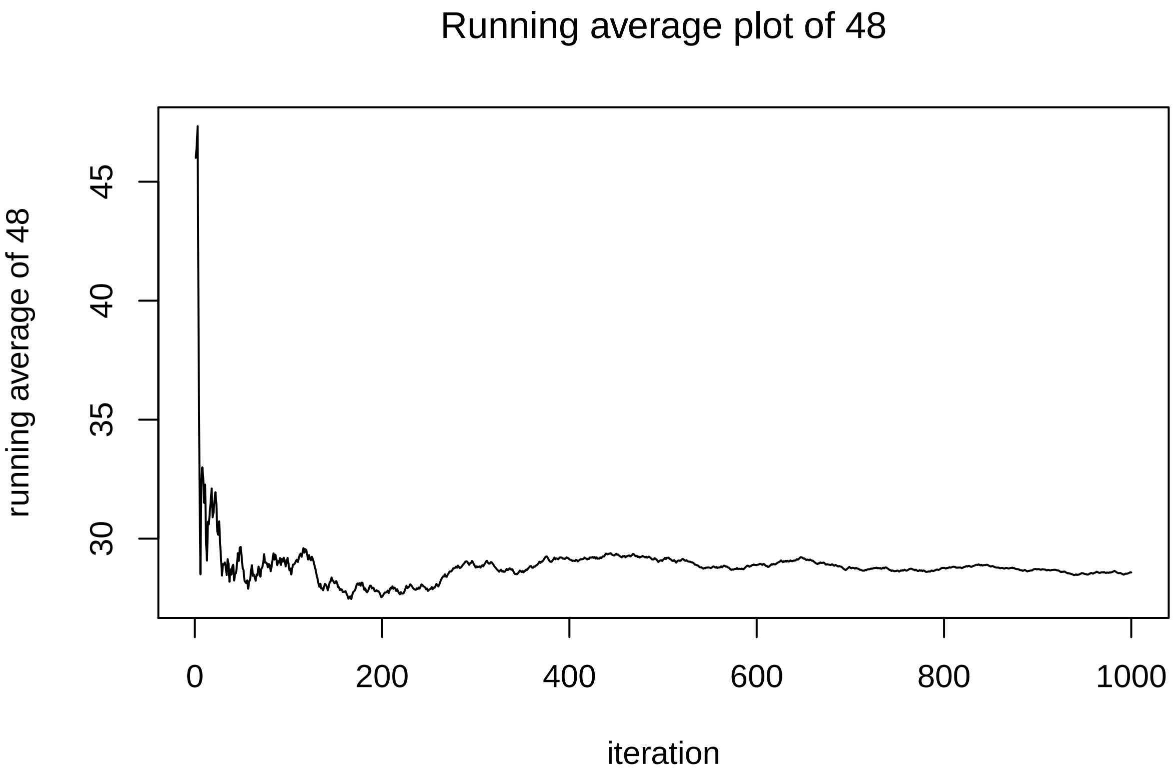

Running average plot of 45
running average of 45
5
10
15
20
25
30
35
0
200
400
600
800
1000
iteration

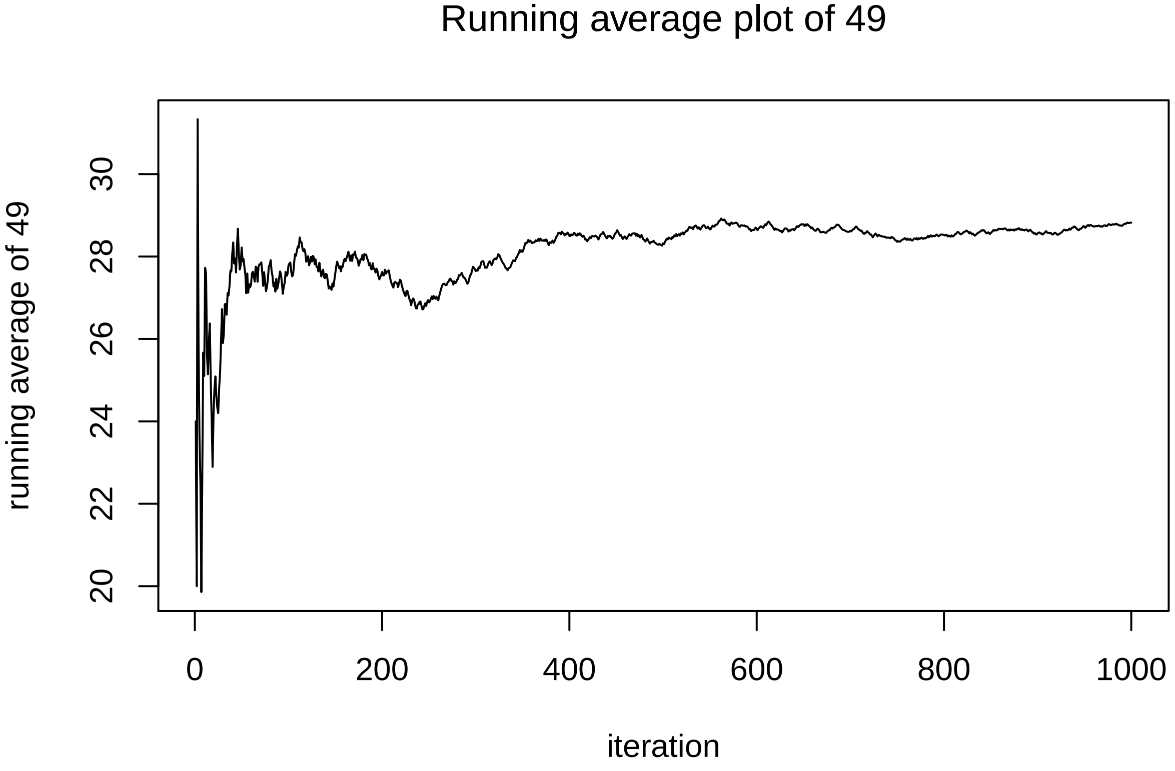
Running average plot of 46

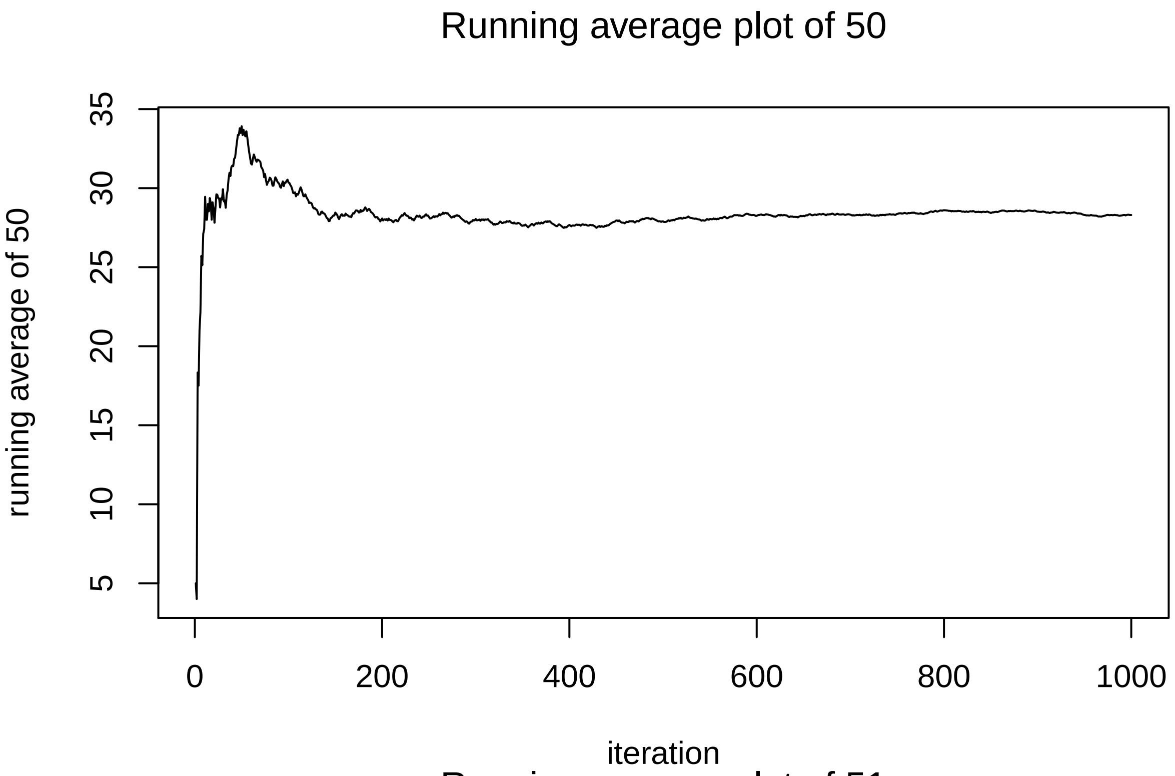


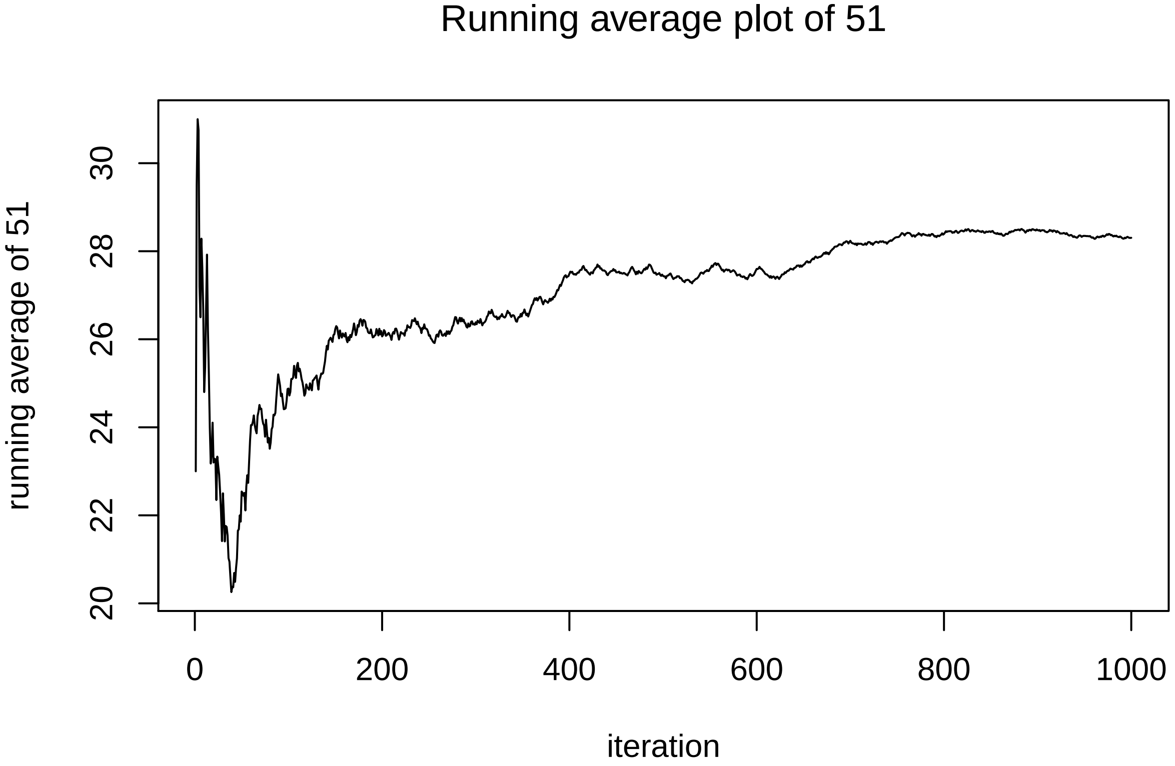
Running average plot of 47

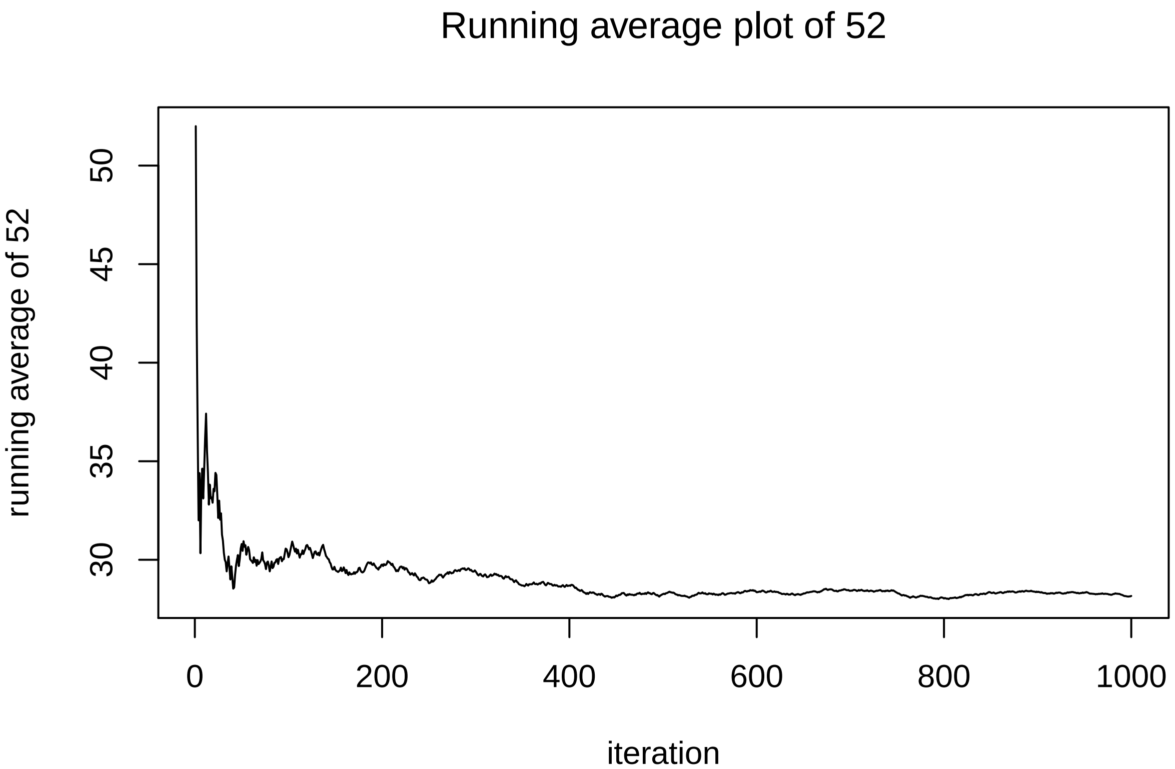


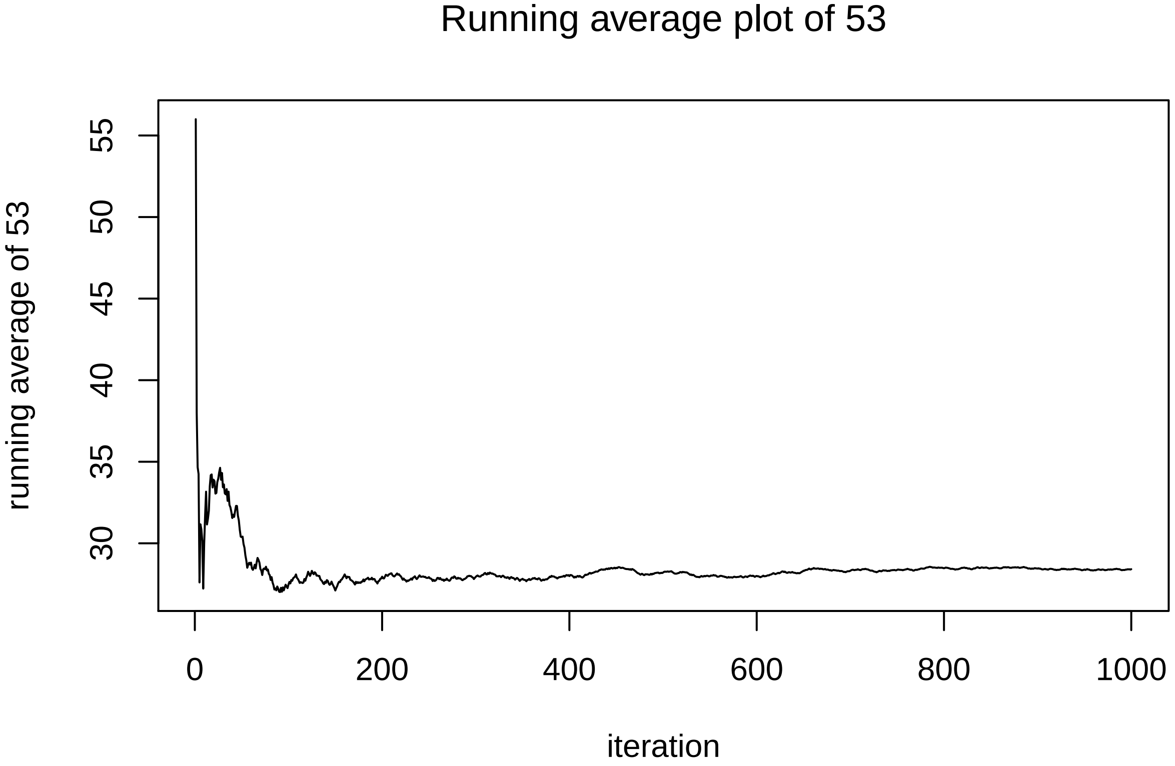

Running average plot of 48
running average of 48
45
40
35
30
0
200
400
600
800
1000
iteration

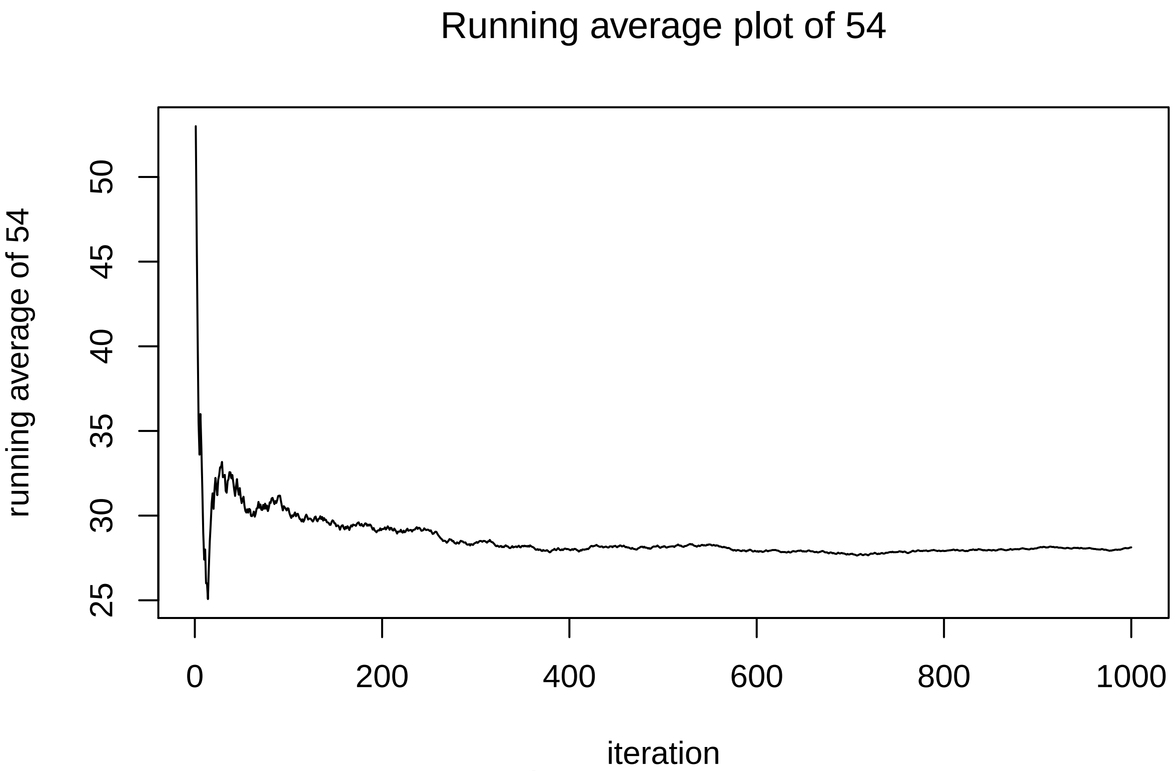

Running average plot of 49
running average of 49
30
28
26
24
22
20
0
200
400
600
800
1000
iteration

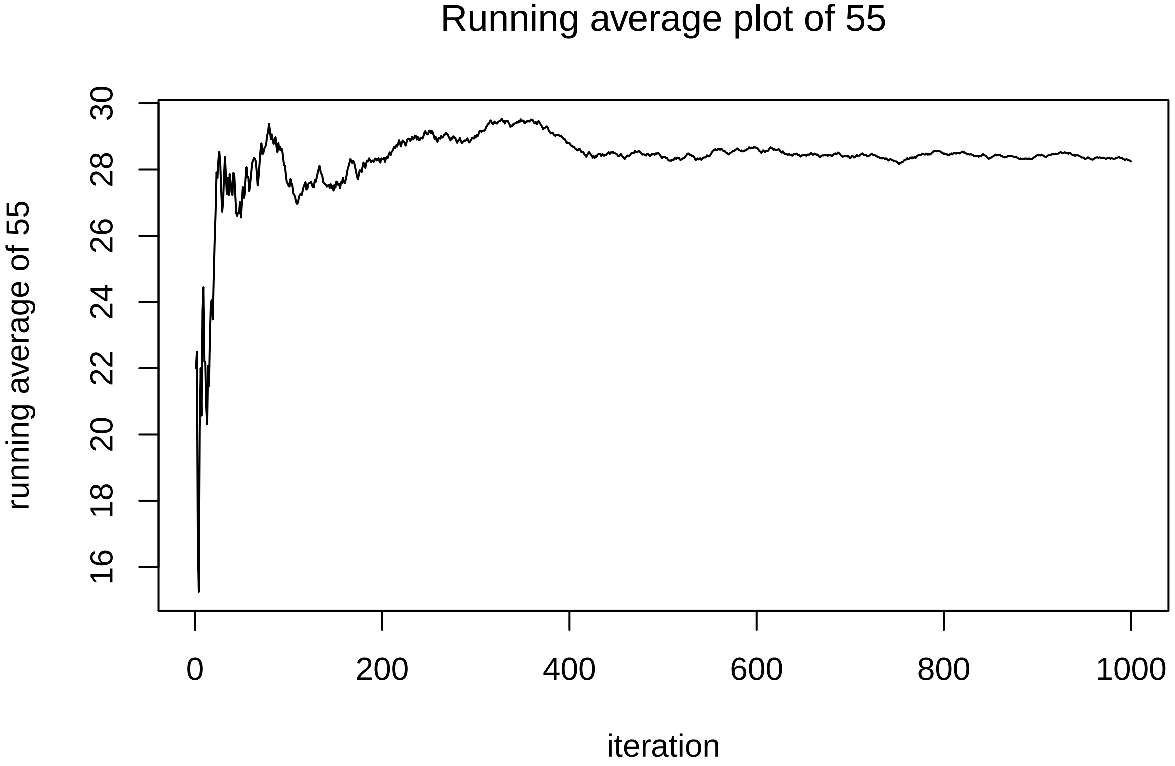

Running average plot of 50
running average of 50
5
10
15
20
25
30
35
0
200
400
600
800
1000
iteration

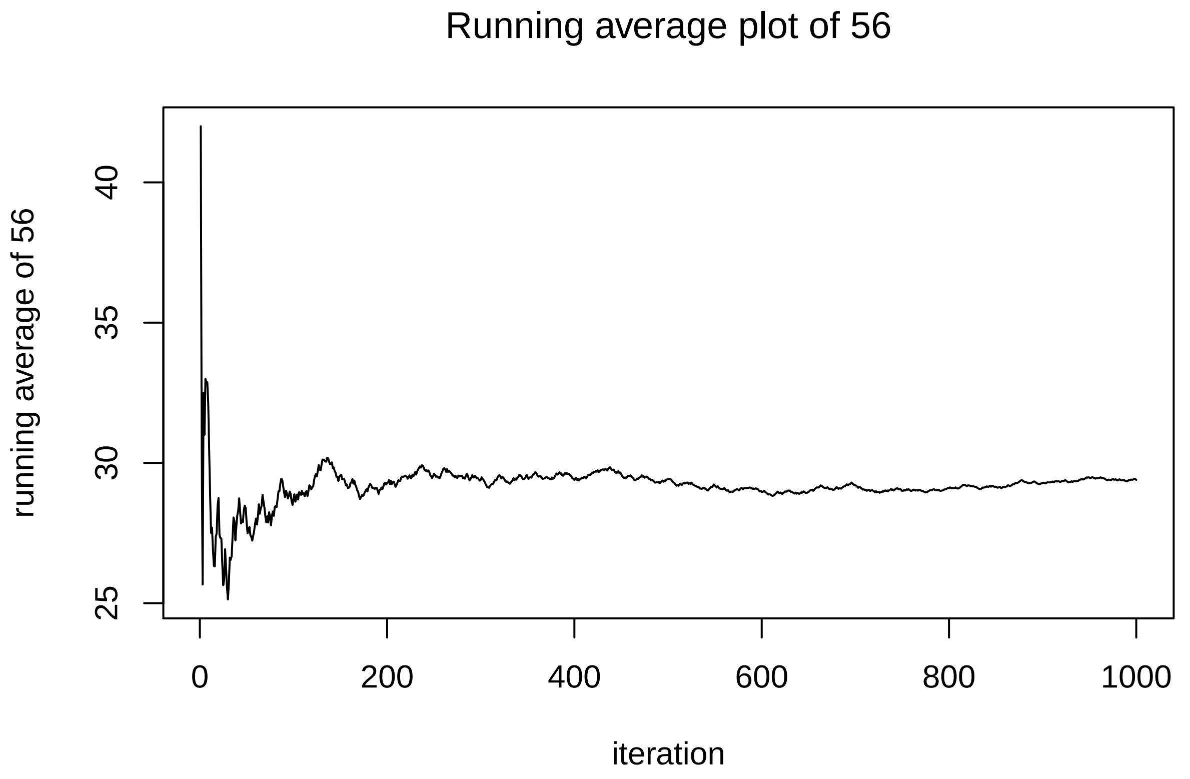

Running average plot of 51
running average of 51
20
22
24
26
28
30
0
200
400
600
800
1000
iteration


Running average plot of 52
running average of 52
50
45
40
35
30
0
200
400
600
800
1000
iteration


Running average plot of 53
running average of 53
55
50
45
40
35
30
0
200
400
600
800
1000
iteration


Running average plot of 54
running average of 54
25
30
35
40
45
50
0
200
400
600
800
1000
iteration


Running average plot of 55
running average of 55
16
18
20
22
24
26
28
30
0
200
400
600
800
1000
iteration



```
# plot(1:n.iter, run.avg[, 5], type = "l", cex = 0.5,
#      xlab = "iteration", ylab = expression(paste("running averages of", lambda[15])),
#      main = expression(paste("Running averages plot of ", lambda[15])))
```

```
# print out the linkage structure
print("File 1")
```

```
## [1] "File 1"
```

```
for (j in 1:nv[1]) {
  print(paste("lambda", 1, j, "is", res_Lambda[n.iter, j]))
}
```

```
## [1] "lambda 1 1 is 1"
## [1] "lambda 1 2 is 54"
## [1] "lambda 1 3 is 8"
## [1] "lambda 1 4 is 24"
## [1] "lambda 1 5 is 38"
## [1] "lambda 1 6 is 55"
## [1] "lambda 1 7 is 20"
## [1] "lambda 1 8 is 14"
## [1] "lambda 1 9 is 48"
## [1] "lambda 1 10 is 30"
## [1] "lambda 1 11 is 25"
## [1] "lambda 1 12 is 40"
## [1] "lambda 1 13 is 37"
## [1] "lambda 1 14 is 52"
## [1] "lambda 1 15 is 17"
## [1] "lambda 1 16 is 28"
## [1] "lambda 1 17 is 27"
## [1] "lambda 1 18 is 5"
## [1] "lambda 1 19 is 29"
## [1] "lambda 1 20 is 47"
```

```
print("File 2")
```

```
## [1] "File 2"
```

```
for(j in nv[1]:(nv[1] + nv[2])){
  print(paste("lambda", 2, j, "is", res_Lambda[n.iter, j]))
}
```

```
## [1] "lambda 2 20 is 47"
## [1] "lambda 2 21 is 21"
## [1] "lambda 2 22 is 17"
## [1] "lambda 2 23 is 44"
## [1] "lambda 2 24 is 40"
## [1] "lambda 2 25 is 15"
## [1] "lambda 2 26 is 45"
## [1] "lambda 2 27 is 51"
## [1] "lambda 2 28 is 8"
## [1] "lambda 2 29 is 52"
## [1] "lambda 2 30 is 36"
## [1] "lambda 2 31 is 10"
## [1] "lambda 2 32 is 56"
## [1] "lambda 2 33 is 22"
## [1] "lambda 2 34 is 48"
## [1] "lambda 2 35 is 42"
## [1] "lambda 2 36 is 37"
## [1] "lambda 2 37 is 47"
```

```
print("File 3")
```

```
## [1] "File 3"
```

```
for (j in (nv[1] + nv[2]):(nv[1] + nv[2] + nv[3])) {
  print(paste("lambda", 2, j, "is", res_Lambda[n.iter, j]))
}
```

```
## [1] "lambda 2 37 is 47"
## [1] "lambda 2 38 is 38"
## [1] "lambda 2 39 is 10"
## [1] "lambda 2 40 is 39"
## [1] "lambda 2 41 is 21"
## [1] "lambda 2 42 is 32"
## [1] "lambda 2 43 is 33"
## [1] "lambda 2 44 is 12"
## [1] "lambda 2 45 is 14"
## [1] "lambda 2 46 is 34"
## [1] "lambda 2 47 is 41"
## [1] "lambda 2 48 is 20"
## [1] "lambda 2 49 is 26"
## [1] "lambda 2 50 is 28"
## [1] "lambda 2 51 is 31"
## [1] "lambda 2 52 is 47"
## [1] "lambda 2 53 is 52"
## [1] "lambda 2 54 is 35"
## [1] "lambda 2 55 is 5"
## [1] "lambda 2 56 is 3"
```