LAB2_AML

Andreas C Charitos [andch552]

9/29/2019

Contents

Assignment 1

We are asked to model the behavior of a robot that walks around a ring. The ring is divided into 10 sectors. At any given time point, the robot is in one of the sectors and decides with equal probability to stay in that sector or move to the next sector. You do not have direct observation of the robot. However, the robot is equipped with a tracking device that you can access. The device is not very accurate though: If the robot is in the sector i, then the device will report that the robot is in the sectors [i-2,i+2] with equal probability. Start by defining the model which requires the following components :

- The hidden states z_t
- The observed states x_t
- The transition matrix with the probabilities for the hidden states
- The emission matrix with the probabilities for the observed states

```
## The states are :
##  State_1 State_2 State_3 State_4 State_5 State_6 State_7 State_8 State_9 State_10
## The observed states are :
##  a b c d e f g h i j
```

Next we report the Transition and emission matrices

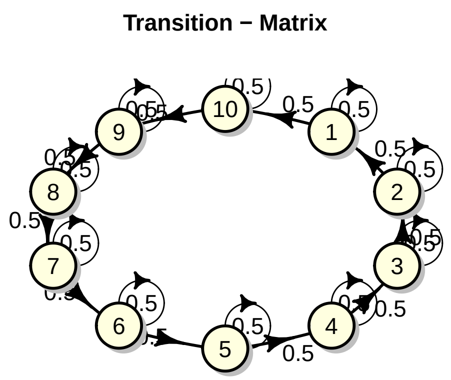
Transition Matrix

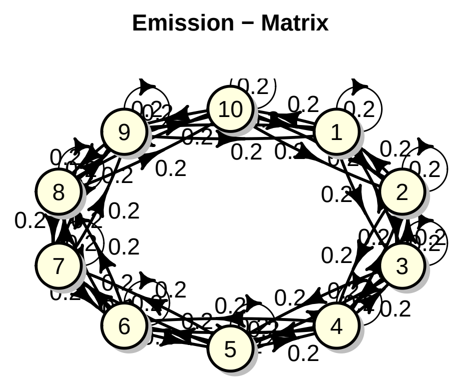
State_1	State_2	State_3	State_4	State_5	State_6	State_7	State_8	State_9	State_10
0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.5	0.5	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5
0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5

Emission Matrix

a	b	c	d	e	f	g	h	i	j
0.2	0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.2	0.2
0.2	0.2	0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.2
0.2	0.2	0.2	0.2	0.2	0.0	0.0	0.0	0.0	0.0
0.0	0.2	0.2	0.2	0.2	0.2	0.0	0.0	0.0	0.0
0.0	0.0	0.2	0.2	0.2	0.2	0.2	0.0	0.0	0.0
0.0	0.0	0.0	0.2	0.2	0.2	0.2	0.2	0.0	0.0
0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.2	0.2	0.0
0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.2	0.2
0.2	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.2
0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.2

Plot of Transmission and Emission Matrix





Assignment 2

Next we simulate from the defined model 100 timesteps

Assignment 3

We then discard the hidden states from the sample obtained above.And use the remaining observations to compute the filtered and smoothed probability distributions for each of the 100 time points.We compute also the most probable path.

Assignment 4

Now we calculate the accuracy for the filter,smooth and most probable path .

Table with the Accuracies from simulation

	Filter	Smooth	Viterbi
Accuracy	0.48	0.65	0.57

We observe that the accuracy of the smooth is better that filter and viterbi.The reasoning behind this is in the way the filtering and smoothing are calculated :

- Filtering : $p(Z^t|x^{0:t} = \frac{\alpha(Z^t)}{\sum_z t\alpha(z^t)})$
- Smoothing : $p(Z^t|x^{0:T} = \frac{\alpha(Z^t)\beta(Z^t)}{\sum_z \alpha(z^t)\beta(z^t)})$

The filter is calculating the probability of state given the observations up to this timestep while smooth is calculating the state given all the previous observations.

Assignment 5

Here we simulate 100 different simulated samples with 100 timesteps.The puspose of this procedure is to see if the that the smoothed distribution should be more accurate than the filtered distributions.

Table with the accuracies for diffrent simulations

	replicate n= 1	replicate n= 2	replicate n= 3	replicate n= 4	replicate n= 5	replicate n= 6	replicate n= 7	replicate n= 8	replicate n= 9	replicate n= 10
filter_accuracy	0.54	0.42	0.54	0.59	0.47	0.56	0.50	0.61	0.41	0.51
smooth_accuracy	0.75	0.58	0.77	0.66	0.67	0.70	0.59	0.64	0.60	0.59
viterbi_accuracy	0.50	0.50	0.63	0.47	0.67	0.54	0.51	0.51	0.59	0.51

	replicate n= 11	replicate n= 12	replicate n= 13	replicate n= 14	replicate n= 15	replicate n= 16	replicate n= 17	replicate n= 18	replicate n= 19	replicate n= 20
filter_accuracy	0.50	0.60	0.53	0.61	0.63	0.45	0.64	0.46	0.58	0.46
smooth_accuracy	0.69	0.73	0.69	0.72	0.64	0.67	0.77	0.59	0.71	0.73
viterbi_accuracy	0.48	0.44	0.64	0.66	0.51	0.39	0.42	0.62	0.48	0.65

	replicate n= 21	replicate n= 22	replicate n= 23	replicate n= 24	replicate n= 25	replicate n= 26	replicate n= 27	replicate n= 28	replicate n= 29	replicate n= 30
filter_accuracy	0.62	0.55	0.49	0.52	0.65	0.40	0.61	0.57	0.51	0.47
smooth_accuracy	0.70	0.69	0.61	0.61	0.69	0.65	0.79	0.71	0.59	0.63
viterbi_accuracy	0.44	0.49	0.51	0.42	0.52	0.44	0.52	0.44	0.44	0.47

	replicate n= 31	replicate n= 32	replicate n= 33	replicate n= 34	replicate n= 35	replicate n= 36	replicate n= 37	replicate n= 38	replicate n= 39	replicate n= 40
filter_accuracy	0.51	0.54	0.46	0.53	0.61	0.51	0.60	0.68	0.67	0.66
smooth_accuracy	0.71	0.65	0.54	0.62	0.72	0.76	0.57	0.69	0.81	0.73
viterbi_accuracy	0.53	0.51	0.33	0.58	0.40	0.59	0.34	0.51	0.60	0.44

	replicate n= 41	replicate n= 42	replicate n= 43	replicate n= 44	replicate n= 45	replicate n= 46	replicate n= 47	replicate n= 48	replicate n= 49	replicate n= 50
filter_accuracy	0.60	0.58	0.53	0.58	0.52	0.65	0.46	0.47	0.48	0.43
smooth_accuracy	0.71	0.77	0.72	0.70	0.76	0.77	0.65	0.77	0.63	0.64
viterbi_accuracy	0.48	0.41	0.46	0.63	0.49	0.49	0.49	0.38	0.34	0.50

	replicate n= 51	replicate n= 52	replicate n= 53	replicate n= 54	replicate n= 55	replicate n= 56	replicate n= 57	replicate n= 58	replicate n= 59	replicate n= 60
filter_accuracy	0.50	0.55	0.61	0.54	0.44	0.38	0.55	0.57	0.51	0.59
smooth_accuracy	0.67	0.73	0.75	0.66	0.70	0.58	0.64	0.73	0.66	0.64
viterbi_accuracy	0.47	0.64	0.66	0.43	0.43	0.54	0.53	0.49	0.44	0.46

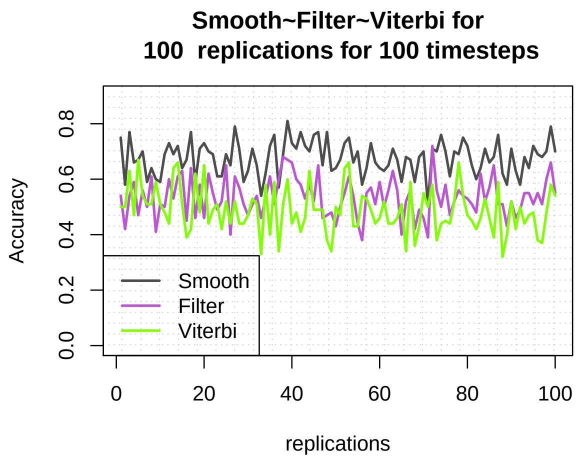
	replicate n= 61	replicate n= 62	replicate n= 63	replicate n= 64	replicate n= 65	replicate n= 66	replicate n= 67	replicate n= 68	replicate n= 69	replicate n= 70
filter_accuracy	0.50	0.56	0.63	0.56	0.40	0.52	0.56	0.42	0.49	0.46
smooth_accuracy	0.63	0.65	0.71	0.67	0.59	0.68	0.67	0.59	0.68	0.70
viterbi_accuracy	0.52	0.44	0.44	0.46	0.51	0.34	0.59	0.36	0.43	0.55

	replicate n= 71	replicate n= 72	replicate n= 73	replicate n= 74	replicate n= 75	replicate n= 76	replicate n= 77	replicate n= 78	replicate n= 79	replicate n= 80
filter_accuracy	0.39	0.72	0.56	0.50	0.58	0.47	0.52	0.56	0.54	0.53
smooth_accuracy	0.51	0.71	0.70	0.76	0.70	0.61	0.70	0.69	0.75	0.72
viterbi_accuracy	0.50	0.58	0.38	0.44	0.45	0.44	0.52	0.66	0.54	0.47

	replicate n= 81	replicate n= 82	replicate n= 83	replicate n= 84	replicate n= 85	replicate n= 86	replicate n= 87	replicate n= 88	replicate n= 89	replicate n= 90
filter_accuracy	0.51	0.48	0.62	0.52	0.57	0.65	0.51	0.51	0.43	0.52
smooth_accuracy	0.65	0.60	0.64	0.71	0.66	0.68	0.76	0.62	0.58	0.71
viterbi_accuracy	0.45	0.42	0.46	0.53	0.46	0.39	0.59	0.32	0.40	0.52

	replicate n= 91	replicate n= 92	replicate n= 93	replicate n= 94	replicate n= 95	replicate n= 96	replicate n= 97	replicate n= 98	replicate n= 99	replicate n= 100
filter_accuracy	0.46	0.49	0.55	0.55	0.51	0.55	0.51	0.60	0.66	0.55
smooth_accuracy	0.63	0.58	0.68	0.64	0.72	0.69	0.68	0.70	0.79	0.70
viterbi_accuracy	0.42	0.50	0.44	0.47	0.48	0.38	0.37	0.48	0.58	0.54

Plot of Accuracies for diffrent simulations



As we can see in general the performance of the smooth is better than filter and viterbi.

Assignment 6

Next,we generate samples with diffrent sizes in order to investigate the claim that having more timesteps we can find better where the robot is.We report the table of the accuracies and the entropy for some samples.

Table with the accuracies for diffrent sample sizes

	nSample= 10	nSample= 15	nSample= 20	nSample= 25	nSample= 30	nSample= 35	nSample= 40	nSample= 45	nSample= 50	nSample= 55
filter_accuracy	0.3	0.4000000	0.65	0.48	0.6333333	0.6000000	0.400	0.5111111	0.46	0.6000000
smooth_accuracy	0.6	0.5333333	0.75	0.52	0.8333333	0.6285714	0.625	0.7555556	0.74	0.7090909
viterbi_accuracy	0.7	0.2000000	0.45	0.28	0.6333333	0.3714286	0.550	0.4444444	0.54	0.3090909

	nSample= 60	nSample= 65	nSample= 70	nSample= 75	nSample= 80	nSample= 85	nSample= 90	nSample= 95	nSample= 100	nSample= 105
filter_accuracy	0.6000000	0.4307692	0.6000000	0.44	0.575	0.6470588	0.5333333	0.4736842	0.52	0.4000000
smooth_accuracy	0.7000000	0.6615385	0.6571429	0.48	0.750	0.6941176	0.6444444	0.7157895	0.57	0.5714286
viterbi_accuracy	0.4833333	0.3384615	0.4857143	0.40	0.600	0.3764706	0.4222222	0.4631579	0.39	0.4000000

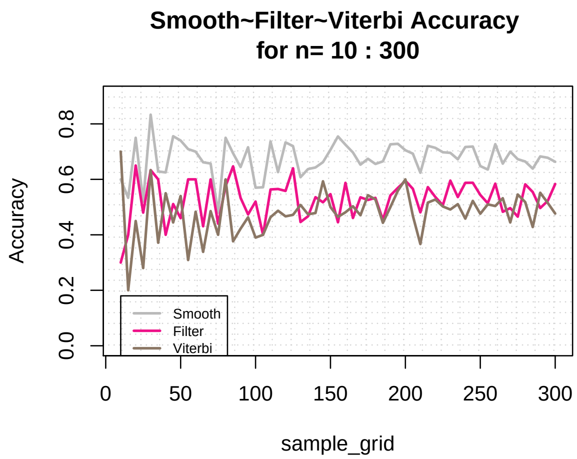
	nSample= 110	nSample= 115	nSample= 120	nSample= 125	nSample= 130	nSample= 135	nSample= 140	nSample= 145	nSample= 150	nSample= 155
filter_accuracy	0.5636364	0.5652174	0.5583333	0.640	0.4461538	0.4666667	0.5357143	0.5172414	0.5466667	0.4451613
smooth_accuracy	0.7363636	0.6260870	0.7333333	0.720	0.6076923	0.6370370	0.6428571	0.6620690	0.7066667	0.7548387
viterbi_accuracy	0.4636364	0.4869565	0.4666667	0.472	0.5076923	0.4740741	0.4785714	0.5931034	0.5000000	0.4645161

	nSample= 160	nSample= 165	nSample= 170	nSample= 175	nSample= 180	nSample= 185	nSample= 190	nSample= 195	nSample= 200	nSample= 205
filter_accuracy	0.58750	0.4606061	0.5352941	0.5257143	0.5333333	0.4540541	0.5421053	0.5692308	0.595	0.5658537
smooth_accuracy	0.72500	0.6969697	0.6529412	0.6742857	0.6555556	0.6648649	0.7263158	0.7282051	0.705	0.6926829
viterbi_accuracy	0.48125	0.5030303	0.4705882	0.5428571	0.5277778	0.4432432	0.5000000	0.5589744	0.600	0.4682927

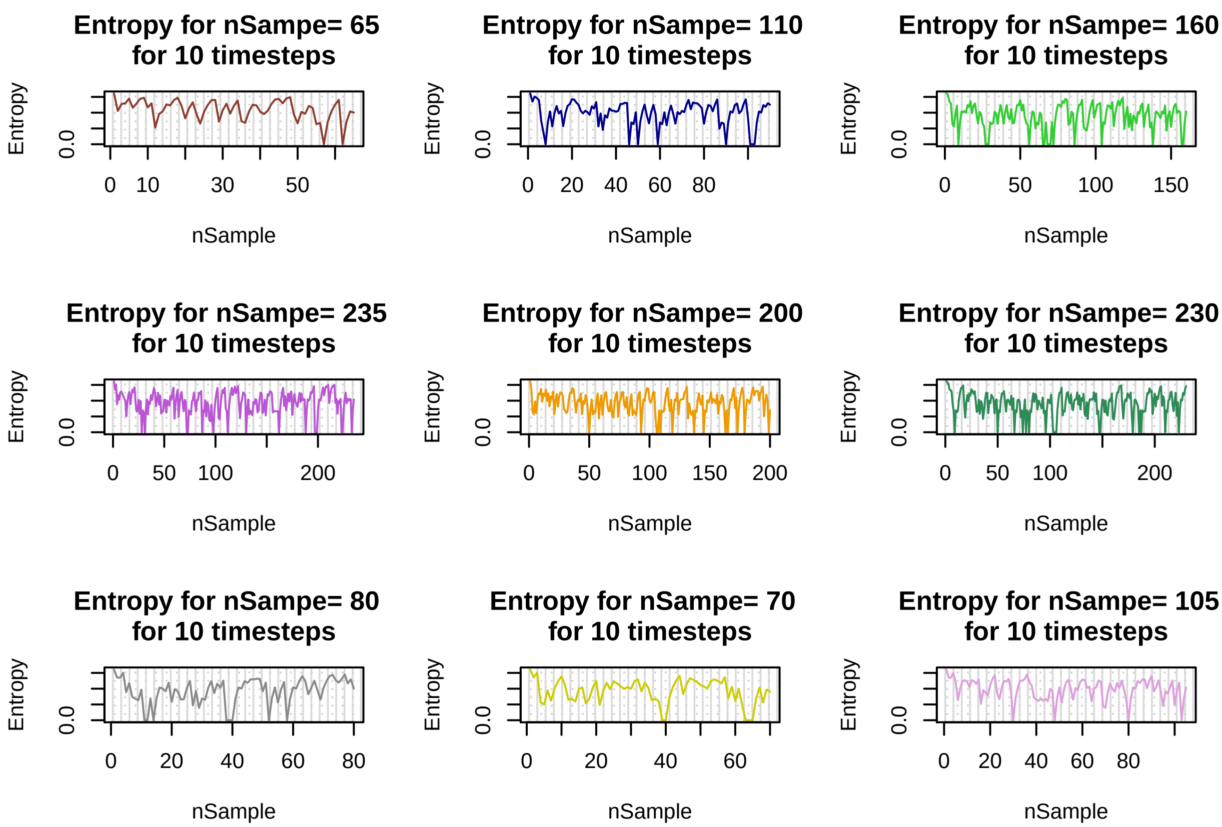
	nSample= 210	nSample= 215	nSample= 220	nSample= 225	nSample= 230	nSample= 235	nSample= 240	nSample= 245	nSample= 250	nSample= 255
filter_accuracy	0.4809524	0.5720930	0.5363636	0.5066667	0.5956522	0.5361702	0.5875000	0.5877551	0.544	0.5137255
smooth_accuracy	0.6238095	0.7209302	0.7136364	0.6977778	0.6956522	0.6723404	0.7166667	0.7183673	0.648	0.6352941
viterbi_accuracy	0.3666667	0.5162791	0.5272727	0.5022222	0.4913043	0.5106383	0.4583333	0.5224490	0.476	0.5098039

	nSample= 260	nSample= 265	nSample= 270	nSample= 275	nSample= 280	nSample= 285	nSample= 290	nSample= 295	nSample= 300
filter_accuracy	0.5846154	0.4830189	0.4962963	0.4654545	0.5821429	0.5543860	0.4965517	0.5220339	0.5833333
smooth_accuracy	0.7269231	0.6566038	0.7000000	0.6727273	0.6642857	0.6385965	0.6827586	0.6779661	0.6633333
viterbi_accuracy	0.5038462	0.5320755	0.4444444	0.5454545	0.5178571	0.4280702	0.5517241	0.5152542	0.4766667

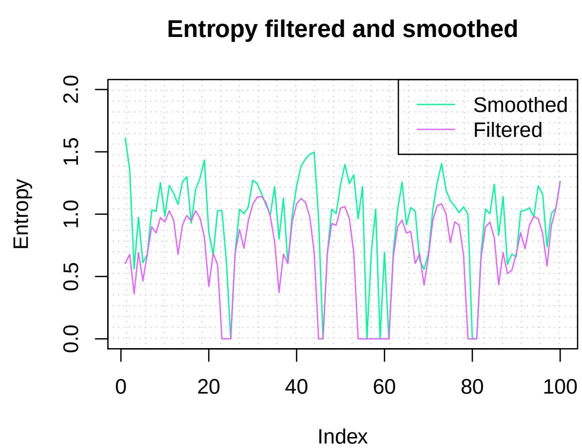
Plot of the Accuracies for diffrent samples



Plot of the Entropies for different samples



Plot of Entropies for Filter and Smoothed



So at first we ploted the entropies for different timesteps and then for a sampled simulation with 100 timesteps. As we can see there is no evidence that supports the claim that having more samples helps up to predict the state of the robot better.

Assignment 7

Finally,we take a random sample from the 100th timestep samples we created in Assignment 5 and calculate the probabilities in the 101 timestep for the hidden states.

	prob_step101
State_1	0.0000000
State_2	0.0000000
State_3	0.0000000
State_4	0.0000000
State_5	0.0000000
State_6	0.0000000
State_7	0.0636856
State_8	0.2682927
State_9	0.4363144
State_10	0.2317073

Conclusion

In conclusion we can see that on general the smooth will report better accuracy compared with the 2 other methods and the number of the samples does not provide any more information that can help us identify the robot possition better.

Appendix

Code used for Lab

```r
# Libraries
# ---------------------------------------------------------------
library(HMM)
library(seqHMM)  #another library for HMMs
library(diagram)
library(entropy)
library(dplyr)
# -------------------------------------------------------------------------
# Question 1
# --------------------------------------------------------------
names = 1:10
States = c()  # initialize the hidden states z
for (i in 1:10) {
    States = c(States, paste("State_", names[i], sep = ""))
}
States  # print the vector with the States x
Symbols = letters[1:10]  # initialie the observed states
# transmission matrix
transProbs = matrix(0, 10, 10)  # transmisison matrix-hidden
diag(transProbs) = 0.5
diag(transProbs[, -1]) = 0.5
transProbs[10, 1] = 0.5
emissionProbs = toeplitz(c(0.2, 0.2, 0.2, rep(0, 5), 0.2,
    0.2))  # emission matrix-observable
# print transition probabilities
TP = as.data.frame(transProbs)
colnames(TP) = States
knitr::kable(TP)
# print emission probabilities
EP = as.data.frame(emissionProbs)
colnames(EP) = Symbols
knitr::kable(EP)
# initialize HMM
hmm = initHMM(States, Symbols, transProbs = transProbs,
    emissionProbs = emissionProbs)
# using the diagram package
plotmat(transProbs, box.size = 0.05, box.col = "lightyellow",
    arr.length = 0.4, main = "Transition - Matrix")  # plot the transition-matrix
plotmat(emissionProbs, box.size = 0.05, box.col = "lightyellow",
    arr.length = 0.4, main = "Emission - Matrix")  # plot the emission-matrix
# -------------------------------------------------------------------------
# Question 2
# --------------------------------------------------------------
nSim = 100
sim = simHMM(hmm, nSim)  # simulate 100 timesteps
# -------------------------------------------------------------------------
# Question 3
# --------------------------------------------------------------
sim.obs = sim$observation  # discard the hidden states
```

```r
# filter
f = forward(hmm, sim.obs)
f_tab = prop.table(exp(f), margin = 2)  # marginalize over the columns so thats why we have 2 to prop t
# smooth
b = backward(hmm, sim.obs)
# b_tab=prop.table(exp(b),2)
smooth_ = posterior(hmm, sim.obs)  # or we can use the prop.table(exp(f)*exp(b),2)
# viterbi algorithm for most probable path
vit = viterbi(hmm, sim.obs)
# ------------------------------------------------------------------------
# Question 4
# -------------------------------------------------------------
# compute accuracy for filter
t.index = apply(f_tab, 2, which.max)  # marginalize over the columns
t = sapply(t.index, function(y) {
    States[y]
})
# accuracy for filter
ac1 = sum(t == sim$states)/length(sim$states)
# compute accuracy for smooth
t1.index = apply(smooth_, 2, which.max)
t1 = sapply(t1.index, function(y) {
    States[y]
})
# accuracy for smooth
ac2 = sum(t1 == sim$states)/length(sim$states)
# compute accuracy for viterbi
ac3 = sum(vit == sim$states)/length(sim$states)
accuracy_tab = cbind(ac1, ac2, ac3)
colnames(accuracy_tab) = c("Filter", "Smooth", "Viterbi")
rownames(accuracy_tab) = "Accuracy"
accuracy_tab
knitr::kable(as.data.frame(accuracy_tab))
# ------------------------------------------------------------------------
# Question 5
# -------------------------------------------------------------
accuracy_func = function(user_hmm, number_sim) {
    # --function that returns the filter ,smooth,viterbi
    # accuracy for a hmm , and number of time steps --
    simHMM = simHMM(user_hmm, number_sim)
    sim_hmm_obs = simHMM$observation  # discard the hidden states
    # filter
    forward_pass = forward(user_hmm, sim_hmm_obs)
    forward_table = prop.table(exp(forward_pass), 2)  # marginalize over the cols Margin=2 to prop tabl
    # smooth
    back_pass = backward(user_hmm, sim_hmm_obs)
    # back_table=prop.table(exp(back_pass),2)
    smooth_ = posterior(user_hmm, sim_hmm_obs)  # or prop.table(exp(forward_pass)*exp(back_pass),2)
    # viterbi algorithm for most probable path
    viterbi_res = viterbi(user_hmm, sim_hmm_obs)
    # compute accuracy for filter
    t.index = apply(forward_table, 2, which.max)  # marginalize over the columns
    t = sapply(t.index, function(y) {
```

```r
        user_hmm$States[y]
    })
    # accuracy for filter
    filter_accuracy = sum(t == simHMM$states)/length(simHMM$states)
    # compute accuracy for smooth
    t1.index = apply(smooth_, 2, which.max)
    t1 = sapply(t1.index, function(y) {
        user_hmm$States[y]
    })
    # accuracy for smooth
    smooth_accuracy = sum(t1 == simHMM$states)/length(simHMM$states)
    # compute accuracy for viterbi
    viterbi_accuracy = sum(viterbi_res == simHMM$states)/length(simHMM$states)
    # compute entropy for filter
    filter_entropy = apply(forward_table, 2, FUN = entropy.empirical)
    return(list(simHMM, accuracies = c(filter_accuracy,
        smooth_accuracy, viterbi_accuracy), entropy = list(filt_ent = filter_entropy,
        smooth_ent = smooth_entropy)))
}
# replicate the results from accuracy function 10 times
# with 100 timespters every time
nreps = 100
rep_res = replicate(nreps, accuracy_func(hmm, 100))
# create a matrix with the results from the rep_res
# --the second row of rep_res
results_mat = matrix(unlist(rep_res[2, ]), nrow = 3, ncol = nreps,
    byrow = F)
# colnames-rownames
colnames(results_mat) = paste("replicate n=", 1:nreps)
rownames(results_mat) = c("filter_accuracy", "smooth_accuracy",
    "viterbi_accuracy")
results_mat
# make table knitr::kable(as.data.frame(results_mat)) #
# don't fit page make tables
t1 <- kable(as.data.frame(results_mat[, 1:10]), format = "latex",
    booktabs = TRUE) %>% kable_styling(latex_options = "scale_down")
t2 <- kable(as.data.frame(results_mat[, 11:20]), format = "latex",
    booktabs = TRUE) %>% kable_styling(latex_options = "scale_down")
t3 <- kable(as.data.frame(results_mat[, 21:30]), format = "latex",
    booktabs = TRUE) %>% kable_styling(latex_options = "scale_down")
t4 <- kable(as.data.frame(results_mat[, 31:40]), format = "latex",
    booktabs = TRUE) %>% kable_styling(latex_options = "scale_down")
t5 <- kable(as.data.frame(results_mat[, 41:50]), format = "latex",
    booktabs = TRUE) %>% kable_styling(latex_options = "scale_down")
t6 <- kable(as.data.frame(results_mat[, 51:60]), format = "latex",
    booktabs = TRUE) %>% kable_styling(latex_options = "scale_down")
t7 <- kable(as.data.frame(results_mat[, 61:70]), format = "latex",
    booktabs = TRUE) %>% kable_styling(latex_options = "scale_down")
t8 <- kable(as.data.frame(results_mat[, 71:80]), format = "latex",
    booktabs = TRUE) %>% kable_styling(latex_options = "scale_down")
t9 <- kable(as.data.frame(results_mat[, 81:90]), format = "latex",
    booktabs = TRUE) %>% kable_styling(latex_options = "scale_down")
t10 <- kable(as.data.frame(results_mat[, 91:100]), format = "latex",
```

```r
    booktabs = TRUE) %>% kable_styling(latex_options = "scale_down")
t1
t2
t3
t4
t5
t6
t7
t8
t9
t10
# plot of the accuracies
cols = c("gray30", "mediumorchid", "chartreuse")
plot(1:nreps, results_mat[2, ], type = "l", ylim = c(0,
    0.9), col = cols[1], lwd = 2, main = paste("Smooth~Filter~Viterbi for\n ",
    nreps, " replications for 100 timesteps "), panel.first = grid(25,
    25), ylab = "Accuracy", xlab = "replications")
lines(1:nreps, results_mat[1, ], col = cols[2], lwd = 2)
lines(1:nreps, results_mat[3, ], col = cols[3], lwd = 2)
legend("bottomleft", legend = c("Smooth", "Filter", "Viterbi"),
    col = cols, lty = 1, lwd = 2)
# ---------------------------------------------------------------------
# Question 6
# -----------------------------------------------------------
sample_grid = seq(10, 300, 5)  # generate grid of timesteps
# apply the accuracy funciton to the grid
accuracy_mat = sapply(sample_grid, function(y) {
    accuracy_func(hmm, y)
})
# create a matrix with the results from the accuracy_mat
# --the second row of raccuracy_mat
res_acc_mat = matrix(unlist(accuracy_mat[2, ]), nrow = 3,
    ncol = length(sample_grid), byrow = F)
# row-column names
colnames(res_acc_mat) = paste("nSample=", sample_grid)
rownames(res_acc_mat) = c("filter_accuracy", "smooth_accuracy",
    "viterbi_accuracy")
# print the matrix
res_acc_mat
# construct a table
# knitr::kable(data.frame(res_acc_mat)) # dont't fit
# page
t11 <- kable(as.data.frame(res_acc_mat[, 1:10]), format = "latex",
    booktabs = TRUE) %>% kable_styling(latex_options = "scale_down")
t12 <- kable(as.data.frame(res_acc_mat[, 11:20]), format = "latex",
    booktabs = TRUE) %>% kable_styling(latex_options = "scale_down")
t13 <- kable(as.data.frame(res_acc_mat[, 21:30]), format = "latex",
    booktabs = TRUE) %>% kable_styling(latex_options = "scale_down")
t14 <- kable(as.data.frame(res_acc_mat[, 31:40]), format = "latex",
    booktabs = TRUE) %>% kable_styling(latex_options = "scale_down")
t15 <- kable(as.data.frame(res_acc_mat[, 41:50]), format = "latex",
    booktabs = TRUE) %>% kable_styling(latex_options = "scale_down")
t16 <- kable(as.data.frame(res_acc_mat[, 51:59]), format = "latex",
```

```r
    booktabs = TRUE) %>% kable_styling(latex_options = "scale_down")
t11
t12
t13
t14
t15
t16
# plot of the accuracies
cols1 = c("grey73", "deeppink2", "peachpuff4")
plot(sample_grid, res_acc_mat[2, ], type = "l", ylim = c(0,
    0.9), col = cols1[1], lwd = 2, main = paste("Smooth~Filter~Viterbi Accuracy \nfor n=",
    sample_grid[1], ":", rev(sample_grid)[1]), panel.first = grid(25,
    25), ylab = "Accuracy")
lines(sample_grid, res_acc_mat[1, ], col = cols1[2], lwd = 2)
lines(sample_grid, res_acc_mat[3, ], col = cols1[3], lwd = 2)
legend(x = 10, y = 0.18, legend = c("Smooth", "Filter",
    "Viterbi"), col = cols1, lty = 1, lwd = 2)
sample_indx <- sample(1:50, 9)
sample_indx
par(mfrow = c(3, 3))
for (i in sample_indx) {
    plot(accuracy_mat[, i]$entropy, type = "o", col = sample(colors(),
        1), pch = 20, ylab = "Entropy", xlab = "nSample",
        panel.first = grid(25, 25), main = paste("Entropy for nSampe=",
            sample_grid[i], " \nfor 10 timesteps"))
}
# take a sample with 100 timesteps from 4
samp1 = rep_res[, 60]
plot(samp1$entropy$filt_ent, main = "Entropy filtered and smoothed",
    type = "l", col = "mediumspringgreen", ylab = "Entropy",
    ylim = c(0, 2), panel.first = grid(25, 25))
lines(samp1$entropy$smooth_ent, col = "mediumorchid1")
legend(x = "topright", lty = 1, legend = c("Smoothed", "Filtered"),
    col = c("mediumspringgreen", "mediumorchid1"))
# ------------------------------------------------------------------------
# Question 7
# -------------------------------------------------------------
rndSample = rep_res[1, sample(1:dim(rep_res)[2], 1)]
post = posterior(hmm, rndSample[[1]]$observation)
prob_step101 = transProbs %*% post[, 100]
df = data.frame(prob_step101)
rownames(df) = States
knitr::kable(df)
# ------------------------------------------------------------------------
# Other
# -----------------------------------------------------------------
par(mfrow = c(5, 5))
for (i in 5:14) {
    plot.ts(f_tab[, i])
    pch1 = as.character(1:10)
    points(f_tab[, i], pch = pch1, cex = 1.25, font = 4,
        col = 1:10)
    legend("topleft", legend = States, col = 1:10, pch = pch1)
```

```
}
```