Supplementary materials for 2.4. Nesting behavior metric quantification

```
# Initialize variables and data structures
nest_slots = 6
days = 9 # Number of days observed
total_time = Array[nest_slots] initialized to 0

# Initialize total time for each nest slot to zero
for slot in 0 to nest_slots-1
    total_time[slot] = 0

# Function to calculate total time spent per nest slot
Function CalculateTotalTime(nest_slot_data, nest_slots, days)
    for slot in 0 to nest_slots-1
        for day in 0 to days-1
            # Add the time spent in this slot during the day to total time
            total_time[slot] += nest_slot_data[slot][day]
    return total_time

# Calculate total time spent per nest slot
total_time_per_slot = CalculateTotalTime(nest_slot_data, nest_slots, days)

# Output the results
for slot in 0 to nest_slots-1
    Print "Nest Slot:", slot, " - Total Time:", total_time_per_slot[slot], "
     seconds"
```

Figure S1. Pseudo code for calculating total time spent in a nest slot.

```
# Initialize variables and data structures
nest_slots = 6
days = 9 # Number of days observed
frequency = Array[nest_slots] initialized to 0

# Function to calculate the frequency of visits per nest slot
Function CalculateFrequency(nest_slot_data, nest_slots, days)
    for slot in 0 to nest_slots-1
        frequency[slot] = 0
        for day in 0 to days-1
            # If a visit is detected, increment the frequency count
            if nest_slot_data[slot][day] == 1
                frequency[slot] += 1
    return frequency

# Calculate frequency of visits per nest slot
frequency_per_slot = CalculateFrequency(nest_slot_data, nest_slots, days)

# Output the results
for slot in 0 to nest_slots-1
    Print "Nest Slot:", slot, " - Frequency:", frequency_per_slot[slot], "visits"
```

Figure S2. Pseudo code for calculating number of visits to a nest slot.

```
# Initialize variables and data structures
nest_slots = 6
time_intervals = 9 # Study conducted from 1 PM to 9 PM
days = 9
time_spent = Array[nest_slots][time_intervals][days]

# Initialize an array for total time spent per time interval
total_time_per_interval = Array[nest_slots][time_intervals]

# Function to calculate total time spent per time interval
Function CalculateTotalTime(time_spent, nest_slots, time_intervals, days)
    for slot in 0 to nest_slots-1
        for interval in 0 to time_intervals-1
            total_time_per_interval[slot][interval] = 0
            for day in 0 to days-1
                total_time_per_interval[slot][interval] += time_spent[slot][interval][day]
    return total_time_per_interval

# Calculate total time spent per time interval
total_time_per_interval = CalculateTotalTime(time_spent, nest_slots, time_intervals, days)

# Output the results
for slot in 0 to nest_slots-1
    Print "Nest Slot:", slot
    for interval in 0 to time_intervals-1
        Print "Time Interval", interval + 1, "-", interval + 2, "PM:", total_time_per_interval[slot][
     interval], "seconds"
```

Figure S3. Pseudo code for calculating hourly time spent in the six nest slots.

```
# Initialize variables and data structures
nest_slots = 6
max_frames = Maximum number of frames across all nest slots
presence_data = Array[nest_slots][max_frames]

# Initialize an array to store simultaneous presence counts
simultaneous_counts = Array[0...nest_slots] initialized to 0

# Function to count simultaneous presence in each frame
Function CountSimultaneousPresence(presence_data, nest_slots, max_frames)
    for frame in 0 to max_frames-1
        # Sum the number of occupied nest slots in the current frame
        simultaneous_presence = 0
        for slot in 0 to nest_slots-1
            if presence_data[slot][frame] == 1
                simultaneous_presence += 1

        # Increment the count for the number of simultaneous presences
        simultaneous_counts[simultaneous_presence] += 1
    return simultaneous_counts

# Calculate the simultaneous presence counts
simultaneous_counts = CountSimultaneousPresence(presence_data, nest_slots,
     max_frames)

# Output the results
for count in 0 to nest_slots
    Print "Number of slots occupied:", count, " - Frames:", simultaneous_counts[
     count]
```

Figure S4. Pseudo code for calculating number of birds simultaneously using the six nest slots.