A low-cost solution for documenting, tracking, and verifying cage-level animal husbandry tasks using wireless QR scanners and cloud-based spreadsheets

Elizabeth A. Hobson

Department of Biological Sciences, University of Cincinnati

elizabeth.hobson@uc.edu

ORCID: 0000-0003-1523-6967

All animal activities are approved by the University of Cincinnati IACUC protocol 21-02-23-01

Abstract

Animal care is a critical component underlying successful behavioral and cognition experiments. Technological solutions for documentation and verification of care can aid in monitoring that activities are completed according to standard operating procedures and ensure that no individuals are overlooked. Here, I summarize a low-cost, flexible, and easy to use system that I developed to document and monitor care of animals for our research group. The system enables real-time and remote-enabled verification that critical daily tasks have been completed for every cage, and helps us monitor our longer-term tasks to make sure that our care team is adhering to our set schedule. The main materials and components needed to implement this system are QR codes, a thermal laminator, a QR scanner, a computer to manage data input, and a database into which the data are scanned and summarized. There are four main steps to setting up our system: (1) purchase a QR scanner, (2) generate and print QR codes, (3) set up the data input spreadsheet, and (4) add summarization and verification capability to the spreadsheet. Paired with simple scripts in a cloud-based spreadsheet, scanned QR code data can then be easily summarized in real time to provide verification of care. The flexibility of the system allows it to be customized to a large range of species.

Keywords

Animal care, paperless documentation, verification, QR code

A low-cost solution for documenting, tracking, and verifying cage-level animal husbandry tasks using wireless QR scanners and cloud-based spreadsheets

Behavioral and cognition experiments are frequently conducted with animals in controlled lab environments. Animals involved in these experiments need to be cared for on a regular schedule and documenting this care is often required by each university's Institutional Animal Care and Use Committee (IACUC) or similar regulatory bodies. Requirements for documentation of care have helped improve transparency, consistency, and the quality of animal care (National Research Council 2011). High quality care is especially important in behavioral and cognitive experiments, where we usually want animals to be healthy, active, and behaving as normally as possible when they participate in experiments. However, the effort involved in documenting details of animal care can be time consuming. An animal care tracking system can aid in facilitating the documentation of animal care activities but can additionally serve as a method for validating that care has been received for each animal on the correct schedule.

Technological solutions can reduce the burden of documenting animal care activities. Quick response (QR) codes are widely used in many fields in biology. QR codes can encode urls as quick links to open websites, can link to online data entry forms, or can contain simple text. Pairing QR codes with smartphone/smart device technology has become a popular method to document animal care (e.g., Green et al. 2017), track specimens (e.g., Diazgranados and Funk 2013), or facilitate data entry (e.g., Oteyo and Toili 2020).

Although using technology to more easily document animal care helps with reporting, some method to verify that care has been provided to each animal on the proper schedule would also be valuable.

Verification would help double-check that no animals have been accidentally overlooked and that care (especially tasks that do not occur on a daily basis) are following the intended schedule. A verification system like this would help improve the consistency of care while reducing animal care staff stress about potentially missing critical care. This method would be especially useful when many animals are being cared for across multiple cages, tanks, or enclosures and multiple categories of care are completed for each animal on different time scales (for example when some care needs to be completed every day, but other care needs to be completed once a week).

I developed a new system to document and validate animal care activities. My group recently started working with a new system in the lab and now have about 40 adult Northern bobwhite quail (*Colinus virginianus*) housed in about 20 cages, approved for use in behavioral and cognition experiments by the University of Cincinnati IACUC protocol 21-02-23-01. The population is cared for by several lab members and has tasks that need to be completed on different time intervals. For example, daily each cage's food and water needs to be changed and the health of all the birds needs to be checked, weekly we clean the trays, and at least once every two weeks we provide enrichment.

My system is a low-cost, flexible, and easy to use method that facilitates documentation of our animal husbandry activities, enables real-time and remote-enabled verification that critical daily tasks have been completed for every cage, and helps us monitor our longer-term tasks to make sure that our care team is adhering to our set schedule.

Here, I summarize the system, which we have been using and fine-tuning in the lab since summer 2021. The main materials and components needed to implement this system are QR codes, a thermal laminator, a QR scanner, a computer to manage data input, and a database into which the data are scanned and summarized. There are four main steps to setting up the system, which are described

below: (1) purchase a QR scanner, (2) generate and print QR codes, (3) set up the data input spreadsheet, and (4) add summarization and verification capability to the spreadsheet.

Step 1: Purchase QR scanner device

Data is input into the system with a handheld QR scanner. Wireless QR scanners are available for between 30-100$USD. I used a Tera 2D Barcode Scanner (Model D5100, https://amzn.to/3AM4Zvh) which cost about 45$USD at time of purchase in 2021. For QR scanning capability, look for a "2D" barcode scanner ("1D" scanners can only read barcodes). These scanners are usually rechargeable with a USB cable; the more expensive models often come with a charging cradle that functions as a stand. The battery life is quite good on some of these models; in our lab, we charge our scanner usually only once every 2-3 weeks. Different models vary in their range – in our case, our model is able to transmit to a central computer from an adjacent room.

Step 2: Generate and print QR codes

Custom QR codes can be freely generated using the R package "qrcode" (Teh and Onkelinx 2021). QR codes can also be generated using several free websites for those not comfortable with R. In our system, each code contains the cage ID and the category or task that is being scanned, separated by a space. See Box 1 for the R script I used to generate tags for 20 cages, each with seven different categories of tasks or activities that we scan and track. In my lab, we use the categories "good.health" and "PROBLEM" to enter data on health checks, "cleaned.water.food", "spot.cleaned", and "full.clean" to track different types of cleaning activities, "enrichment" to record when birds are given enrichment items like an alfalfa feeder or a dust bath, and "NOTE" to add a row to the database into which we can enter any extra notes by hand after scanning.

Box 1. R script to generate custom QR tags

```
# load packages
library(tidyverse)
library(qrcode)
library(png)

# create vector of cage IDs (forcing two-digits for single digit numbers)
cages <- str_pad(seq(1:20), 2, pad = "0")

# create vector of categories for tags
checks <- c("good.health", "PROBLEM", "cleaned.water.food", "spot.cleaned", "full.clean",
"enrichment", "NOTE")

# create dataframe with all cage IDs and all categories of tag checks
cages.checks <- expand.grid(cages, checks)
colnames(cages.checks) <- c("CageID", "Check.Type")
cages.checks <- cages.checks %>% arrange(CageID, Check.Type)

# create full text labels for QR codes (use space as separator between cageID and task category)
cages.checks$Label <- paste("Cage", cages.checks$CageID, cages.checks$Check.Type, sep=" ")

# loop to create png images for each QR tag
for(i in 1:nrow(cages.checks)){
  #specify path & file name for each png image
  mypath <- file.path("qrs.spot_full.clean", paste(cages.checks$Label[i], ".png", sep = ""))
  #create png using qrcode_gen from package
  png(file=mypath)
  qrcode::qrcode_gen(cages.checks$Label[i])
  dev.off()
}
```

After the image files for each QR code have been generated, they can be compiled into a table with one row of codes per cage. I used Microsoft Word to compile the tables: I found that this approach, while somewhat time-consuming, provided the easiest method for color and layout customization. I used a colored background that corresponded to each code's category to help more quickly differentiate categories and reduce the chances of scanning the wrong code (see Fig 1). We have good luck scanning QR codes that are sized at 0.75 inches. To avoid mis-scanning adjacent codes, we notched out a paper card with a 1 inch opening and use it to cover all QR codes except the desired one when scanning.

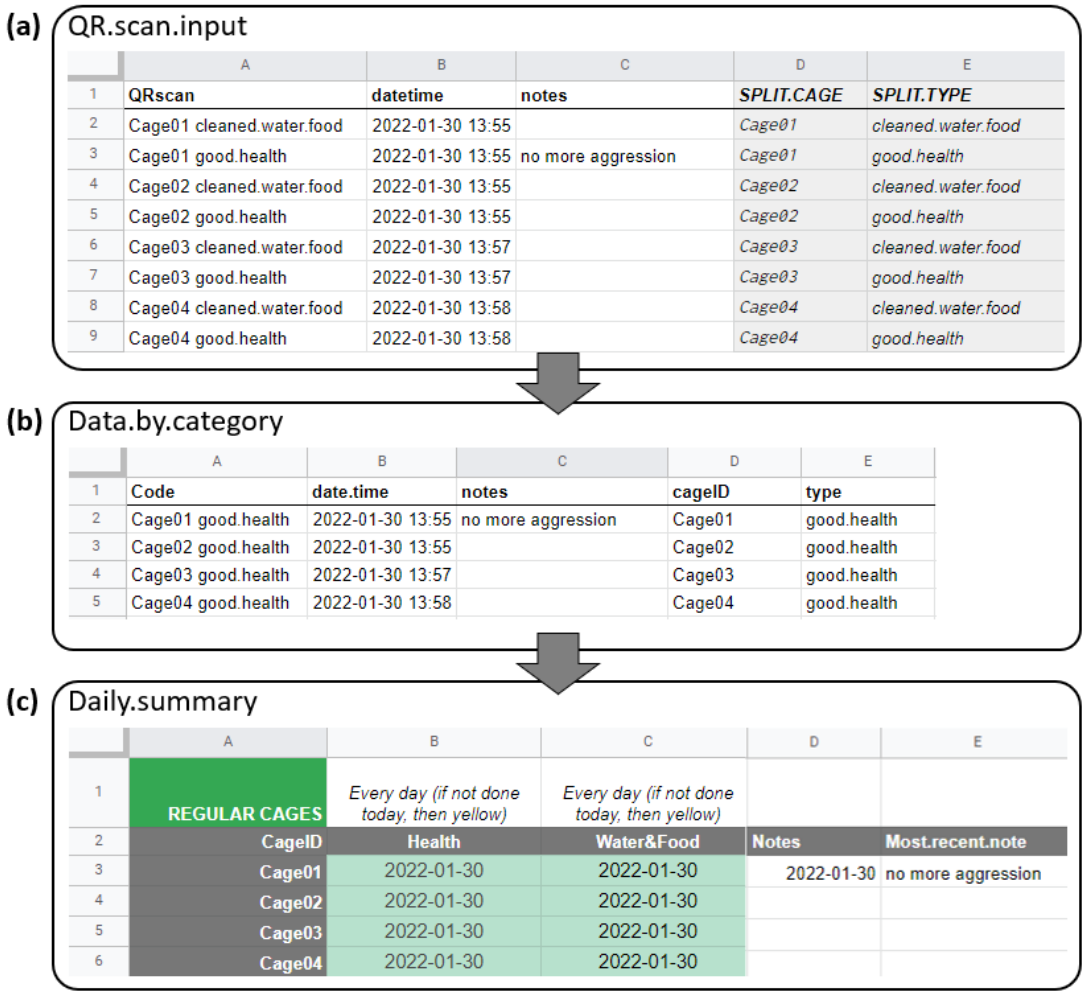
Fig 1. Example cage QR tag with QR codes for six animal care categories and one for notes.

Once printed, these tags are cut out, with each cage's tags in a single strip and laminated for long-term durability. I used an inexpensive thermal laminator to protect the tags. Tag strips were laminated with 5 mil laminator pouches. Laminators are widely available for 20-60$USD (e.g., Apache AL13 Thermal Laminator, https://amzn.to/32KW7JR). An optional paper trimmer greatly facilitates this step (e.g., Worklion Paper Cutter with Security Blade, https://amzn.to/3ubVpR3). I used zip ties to secure the tags to the front of each cage.

Step 3: Set up central hub and spreadsheet for data input

Data from the QR scanner are transmitted to a central computer via USB. Many kinds of computers can be used as the central hub: at a minimum, the computer needs a USB port, an internet connection, and the ability to access Google Sheets. In my lab we use a basic desktop computer on which we do other kinds of data entry. Small single-board computers like the Raspberry Pi would be a very low-cost option and should work well as a hub (e.g., a Raspberry Pi Zero with case and accessories costs around 60$USD, https://amzn.to/3HtmpiL, and would only need a monitor added to make it functional for the QR scanning system).

Figure 2. Google spreadsheet into which QR tags are scanned and data are summarized, showing the three types of sheets necessary for the system. Panel (a) shows the format of the data input sheet. Panel (b) shows how data from the spreadsheet in panel a are filtered by category. Panel (c) shows the daily summary sheet, updated by date of last scan per cage and per category as well as the date of the most recent note and the contents of the most recent note per cage.



(a) QR.scan.input

	A	B	C	D	E
1	QRscan	datetime	notes	SPLIT.CAGE	SPLIT.TYPE
2	Cage01 cleaned.water.food	2022-01-30 13:55		Cage01	cleaned.water.food
3	Cage01 good.health	2022-01-30 13:55	no more aggression	Cage01	good.health
4	Cage02 cleaned.water.food	2022-01-30 13:55		Cage02	cleaned.water.food
5	Cage02 good.health	2022-01-30 13:55		Cage02	good.health
6	Cage03 cleaned.water.food	2022-01-30 13:57		Cage03	cleaned.water.food
7	Cage03 good.health	2022-01-30 13:57		Cage03	good.health
8	Cage04 cleaned.water.food	2022-01-30 13:58		Cage04	cleaned.water.food
9	Cage04 good.health	2022-01-30 13:58		Cage04	good.health

(b) Data.by.category

	A	B	C	D	E
1	Code	date.time	notes	cageID	type
2	Cage01 good.health	2022-01-30 13:55	no more aggression	Cage01	good.health
3	Cage02 good.health	2022-01-30 13:55		Cage02	good.health
4	Cage03 good.health	2022-01-30 13:57		Cage03	good.health
5	Cage04 good.health	2022-01-30 13:58		Cage04	good.health

(c) Daily.summary

	A	B	C	D	E
1	REGULAR CAGES	Every day (if not done today, then yellow)	Every day (if not done today, then yellow)		
2	CageID	Health	Water&Food	Notes	Most.recent.note
3	Cage01	2022-01-30	2022-01-30	2022-01-30	no more aggression
4	Cage02	2022-01-30	2022-01-30		
5	Cage03	2022-01-30	2022-01-30		
6	Cage04	2022-01-30	2022-01-30		

I set up a Google Sheet to scan the QR codes into, to summarize the scans, and to set up a system to easily double-check and proof that care is being provided to each cage on the correct schedule. An example of the whole spreadsheet, with individual sheets, the script to automatically add the date and time, and the summary table is available at https://bit.ly/34vKAi6. This file can be copied and modified for each lab's individual requirements as necessary.

My system uses a Google Sheet workbook containing three types of sheets (Fig. 2).

The first sheet ("QR.scan.input") is the location into which the QR tags are scanned and data are inputted (Fig. 2a). This sheet has five main columns. Column A ("QRscan") contains the scanned QR tags. To scan the QR code, select the next empty cell in the QRscan column. Activate the wireless QR scanner and start scanning tags. Each scan will add the text associated with each QR tag to a new row in the QRscan column. Column B ("datetime") automatically updates with the date and time each individual QR code is scanned. This automatic date is added by using the Script Editor to write and save a Google Apps Script snippet (Box 2). Column C ("notes") provides an area to add any notes by hand after scanning. Column D contains code to automatically split the information in the first column to separate the cage numbers and the categories of tasks, which then appear in Columns D and E.

Box 2. Google Apps Script snippet to automatically add date/time (from tutorial by Dan Nguyen, http://blog.danwin.com/how-to-automatically-timestamp-a-new-row-in-google-sheets-using-apps-script/). From Google Sheets, select "Extensions" then "Apps Script", paste the code below into the script editor, and save it as a Code.gs file named "datetime auto adder**".**

```
var SHEET_NAME = 'QRscan.input';
var DATETIME_HEADER = 'datetime';

function getDatetimeCol(){
  var headers = SpreadsheetApp.getActiveSpreadsheet().getSheetByName(SHEET_NAME).ge
tDataRange().getValues().shift();
  var colindex = headers.indexOf(DATETIME_HEADER);
  return colindex+1;
}

function onEdit(e) {
  var ss = SpreadsheetApp.getActiveSheet();
  var cell = ss.getActiveCell();
  var datecell = ss.getRange(cell.getRowIndex(), getDatetimeCol());
  if (ss.getName() == SHEET_NAME && cell.getColumn() == 1 && !cell.isBlank() && dat
ecell.isBlank()) {
    datecell.setValue(new Date()).setNumberFormat("yyyy-MM-dd hh:mm");
  }
};
```

The second type of sheet serves as an intermediary between the first sheet, where data are input, and the third sheet, where data are summarized. This second kind of sheet filters all the input data by category in separate sheets (see Box 3). Figure 2b shows how data from the spreadsheet in Figure 2a are filtered by category (here, showing the "HEALTH" category). Each category of QR codes that are tracked should get its own filtered sheet. In our system, we use 6 intermediary sheets: a health check, logs of water/food changes, spot cleaning, full cleaning, and enrichment, and a sheet for extra notes. These intermediary sheets should not be edited during use of the system and are just used to facilitate connecting the input sheet to the summary sheet.

Box 3. Google sheet code to automatically filter data by category from data input sheet. This is an example showing how "health" data are filtered. We want to have the "good health", "problem", and "recovering" scans all compiled in this sheet so that we can double-check that health was assessed every day. In cell A2 of the "HEALTH" intermediary sheet, the following code is entered, which will filter and import all relevant data from the data inputted into Sheet1. The category-only data are in column E of Sheet 1.

```
=filter(Sheet1!A2:E14995, (Sheet1!E2:E14995="good.health")+(Sheet1!
E2:E14995="problem")+(Sheet1! E2:E14995="recovering"))
```

The final type of sheet is the summary and verification sheet, shown in Figure 2c. This sheet uses the cage IDs (which were split into separate columns in D and E from the combined cage ID and category information Column A of the input sheet) to summarize the date of the last scan by cage and category (see details below).

Step 4: Add summarization and checking capability to the spreadsheet

Our summary sheet uses several methods to summarize and validate animal care. The summary sheet shows the date of the most recent QR scan by cage and category, then conditional formatting rules control and update the color of the cells based on the time since the last scan and the interval at which each category of care needs to be completed (Fig. 2c). This sheet uses the cage IDs to query each

category of task (in the filtered sheets) to find the most recent scan of each cage for each category (see Box 4). Column A contains these cage IDs (using the same format as in the two Column Ds in Fig. 2a and 2b).

Box 4. Google sheet code to return the date of the most recent scan by cage and category. This is an example showing how "health" data are filtered. The example below is code in Cell B3 in Figure 2c, showing how the intermediary sheet "HEALTH" (Fig. 2b) is queried for the most recent scan completed for Cage01 (named in Cell A3 of Figure 2c). Similar code is used for all cages in this category, with other categories calling the corresponding other intermediary sheets.

```
=large(filter(HEALTH!$B$2:$B,HEALTH!$D$2:$D=A4),1)
```

To simplify validating the care schedule, I added conditional formatting to automatically update the color of each cell in the summary table. To set or change the conditional formatting rules, select "Format" then "Conditional formatting" in the Google Sheet to view the rule menu. In "Apply to range" specify the cells which the rule applies to (here, conditional formatting is applied consistently across all cages within each care category). In "Format rule", choose "Custom formula is", then add the conditional formatting desired and choose the response color to use when the rule is satisfied. For an example of our two rules for color-coding to identify whether a task has or has not been completed each day, see Box 5. In our system, we set each cage's cell to green when each care category has been completed within the time limit set by the conditional formatting. Tasks which still need to be completed are set to yellow. For example, Figure 2c shows that all the health checks and changes of food and water have been completed "today" for all cages; if a cage has not been checked or had these changed on the current date, the cell color turns yellow.

Box 5. Setting cell-level conditional formatting. This is an example showing how "health" summary data are conditionally formatted to update the color of the cells based on when each cage was last scanned. The example below are two conditional formatting rules applied to Cell B3 in Figure 2c. Similar code is used for all cages in this category, with other categories calling the corresponding other intermediary sheets.

If Cage01 has been scanned today, update the cell color to green:
`=DATEDIF(TODAY(),B4,"D")=0`

If Cage 01 has not been scanned today and needs care, update the cell color to yellow:
`=DATEDIF(B4,TODAY(),"D")>0`

171

172 This summary table provides a quick and easy way to verify that care has been completed for each cage.
173 Any cage that was accidentally missed can be very quickly and easily identified and care can be
174 completed. This system also greatly aids in monitoring the schedule for tasks that are completed on a
175 different cycle. We find it especially useful in monitoring our enrichment schedule, which needs to be
176 offered to each cage at least once every two weeks but does not occur on a specific day each week.

177 **Use and troubleshooting**

178 In practice, we use this system as a two-step verification of care. The care team completes all care
179 activities for the day. Our last step is to scan all the cages to document the type of care provided. This
180 cage-by-cage scanning allows us to easily proof that care is complete: for example, before scanning each
181 tag, we do a final check that the health of all individuals is good, and that water and food have been
182 cleaned and replaced. If everything looks good, those QR codes for that particular cage are scanned
183 before moving on to the next cage. Any accidentally missed water or food containers can be identified
184 and rectified at this stage. At the end of scanning, we check the summary sheet to make sure that all
185 cage cells across all categories of care are green, which provides an additional verification that care to all

cages has been completed, and no cages have been accidentally overlooked. Any cage with a cell that is not colored green is re-checked and re-scanned.

The most common errors we have seen in using this system are (1) running out of new rows in the input sheet, (2) adding new rows to the input sheet but forgetting to continue the splitting code, (3) adding cage ID or category codes that do not match the criteria in the spreadsheet, (4) accidentally scanning QR codes into a sheet other than the input sheet, and (5) missing scans. To address running out of new rows, just go to the bottom of the input sheet and add the desired number of new rows. To fix errors with cells not having the splitting code, when new cells are added, just drag the splitting code in Column D to apply it to all of the new rows in that column. To ensure cage ID or category codes match, check the QR code generation procedures to make sure that the codes generated match the desired cage ID and category formats and check the filtering criteria in the intermediary and summary sheets. To correct erroneously scanning QR codes into a sheet other than the data input sheet, put a warning label on the scanner handset (when discovered, the “undo” function can also reset cells to the proper contents and cages can be re-scanned with the data input sheet). To address scans that are not properly received by the central computer, the QR codes just need to be re-scanned. In our experience, missed scans are relatively rare, but most commonly occur for the very first scan of a session.

Conclusions

Our animal care monitoring system is low cost and flexible way to document and verify animal care. Paired with simple scripts in a cloud-based spreadsheet, the scanned QR code data can be summarized to provide real-time verification of care that can be checked by any member of the team with access to the Google Sheet and an internet connection. Our team has found this system to be very reassuring both to people completing care as well as supervisors monitoring care. As care is being completed, the summary sheet can be checked while workers are still in the animal care room to double-check that care

has been provided to all cages and quickly address any deficits. Remotely, other team members can access the spreadsheets and easily verify whether care has been provided each day.

This system is highly customizable and can be adapted for a wide range of species receiving many different types of care. Any research group caring for many individuals across separate cages, tanks, or enclosures, and especially when multiple categories of care are required and when care activities are completed on a non-daily schedule, would benefit from a system like this one.

Acknowledgments

Thanks to Sanjay Prasher and Alexis Williams, who helped design an early prototype of the system described here and to Sanjay Prasher, Claire O'Connell, Chelsea Carminito, and Xavier Francis, who helped test and refine the system. During preparation of this work, EAH was supported by NSF IOS 2015932.

References

Diazgranados M, Funk VA. 2013. Utility of QR codes in biological collections. PhytoKeys. 25(25):21. doi:10.3897/PHYTOKEYS.25.5175.

Green T, Smith T, Hodges R, Fry WM. 2017. A simple and inexpensive way to document simple husbandry in animal care facilities using QR code scanning. Laboratory Animals. 51(6):656–659. doi:10.1177/0023677217718004.

National Research Council. 2011. Guide for the Care and Use of Laboratory Animals, 8th Edition. Washington D.C.

Oteyo IN, Toili MEM. 2020. Improving Specimen Labelling and Data Collection in Bio-science Research using Mobile and Web Applications. Open Computer Science. 10(1):1–16. doi:10.1515/COMP-2020-0002/MACHINEREADABLECITATION/RIS.

Teh V, Onkelinx T. 2021. qrcode: Generate QRcodes with R. Version 0.1.4.