DATA MUSTER

Using the DMMongoDB Package to Retrieve Data

It is now possible to very easily and quickly access a range of historical data from the DataMuster database via the DMMongoDB R package. The following provides a very quick overview on how to access and use the package. To give it a go you will need to contact Lauren and she will provide you with a username and password to provide read access. Please note this package is currently only for internal use and aims to make it easier for students and staff to search and analyse DataMuster data. It is hoped the R tools will support your project goals. We are also seeking feedback to identify new and better methods and understanding that will help unravel the drivers of cattle production across all of our research projects.

To start with you will need to download and install the DMMongoDB package. This can be done by accessing the Precision Livestock Management GitHub repository using the R devtools package. Make sure you have devtools installed and then run the install_github command as shown below. We recommend you use RStudio to develop and run your R scripts. If you need assistance installing R, RStudio or loading packages then let me know. All of the software is open source so is freely available.

```
library(devtools)

install_github("PrecisionLivestockManagement/DMMongoDB")

library(DMMongoDB)
```

Once you have loaded the the DMMongoDB library you can use the RStudio help to find out more about the package. You will find the help information listed under the RStudio help tab and then go to the home page and click on packages, if installed correctly DMMongoDB will be listed. The pacakges are arranged in alphabetical order. The DMMongoDB help provides details on each of the functions that are currently available.

Before you run any of the DMMongoDB functions you will firstly need to setup a username and password. If you run dmaccess("username"), you will then be prompted for your password. Your credentials will be written to your secure password store on your computer and will be called each time you run one of the DMMongoDB functions. This provides secure access to the DataMuster database without having any published user credentials. Please note that "username" should be the username that you have been assigned.

The first function to run should be propsearch(), this will pull in a list of all RFID tags that are currently assigned to a property and paddock, it will also return a list of management tags and paddocks associated with those RFID tags. These tag numbers provide a good starting point to do further searches using the RFID tag numbers. For example a dataframe of management tags and paddocks associated with the a list of RFID tags for Belmont paddock 66 can be pulled in using:

```
tags <- propsearch("Belmont", paddock="66")
```

If the paddock is left blank te function will return details of all cattle from the property.

To pull in the daily cattle weight data and assign to a list which includes RFID tags plus property name you can use the dailywts function. This function has three parameters:

1. a list of RFIDs that you want to pull in,
2. a start date (formatted as date),
3. an end date (again formatted as date),
4. and the minumum number of weights that will be be accepted for the returned values for each individual animal.

If no start, end or minumum number is included it pulls in all values. To search for all animals from Belmont that have at least 10 daily weights and using the RFID tage from the previous search (e.g. tags) from 1st July to 31st December 2019 the following code would be run.

```
start <- as.Date("01/07/2018", format="%d/%m/%Y")

end <- as.Date("31/12/2018", format="%d/%m/%Y")

values <- 10

dailydata <- dailywts(tags$RFID, start, end, values)
```

This search returns a list of RFID tag numbers, plus the property name for each RFID tag and an inidividual dataframe, including all the dates and weights, for each animal. While these data are well organised, using an S3 object class, they are not very easy to use for overall analyses. It is very quick and easy to create a single dataframe that includes all of the necessary information to do further analyses using the dplyr package.

```
library(dplyr)

alldata <- bind_rows(dailydata$DailyWeights, .id = "RFID")
```

These data are now in a single dataframe and ready for further analyses and a small section of the data can be seen below:

RFID	Date	Weight
942 000002679208	2018-07-01 08:58:48	648
942 000002679208	2018-07-02 09:31:16	646
942 000002679208	2018-07-03 13:40:16	664
942 000002679208	2018-07-04 12:24:03	662
942 000002679208	2018-07-05 11:24:52	660
942 000002679208	2018-07-06 16:04:16	666
942 000002679208	2018-07-07 15:11:39	672
942 000002679208	2018-07-08 12:45:14	662
942 000002679208	2018-07-09 13:16:18	654
942 000002679208	2018-07-10 16:06:50	664
942 000002679208	2018-07-11 11:29:22	658
942 000002679208	2018-07-12 13:32:49	670
942 000002679208	2018-07-13 12:29:17	672
942 000002679208	2018-07-14 11:02:13	666
942 000002679208	2018-07-15 10:56:27	666
942 000002679208	2018-07-15 16:38:04	646
942 000002679208	2018-07-15 16:48:56	648
942 000002679208	2018-07-15 17:08:22	654
942 000002679208	2018-07-16 10:20:19	666
942 000002679208	2018-07-16 12:49:11	658
942 000002679208	2018-07-16 17:33:20	650
942 000002679208	2018-07-17 09:47:10	642
942 000002679208	2018-07-18 14:06:40	612

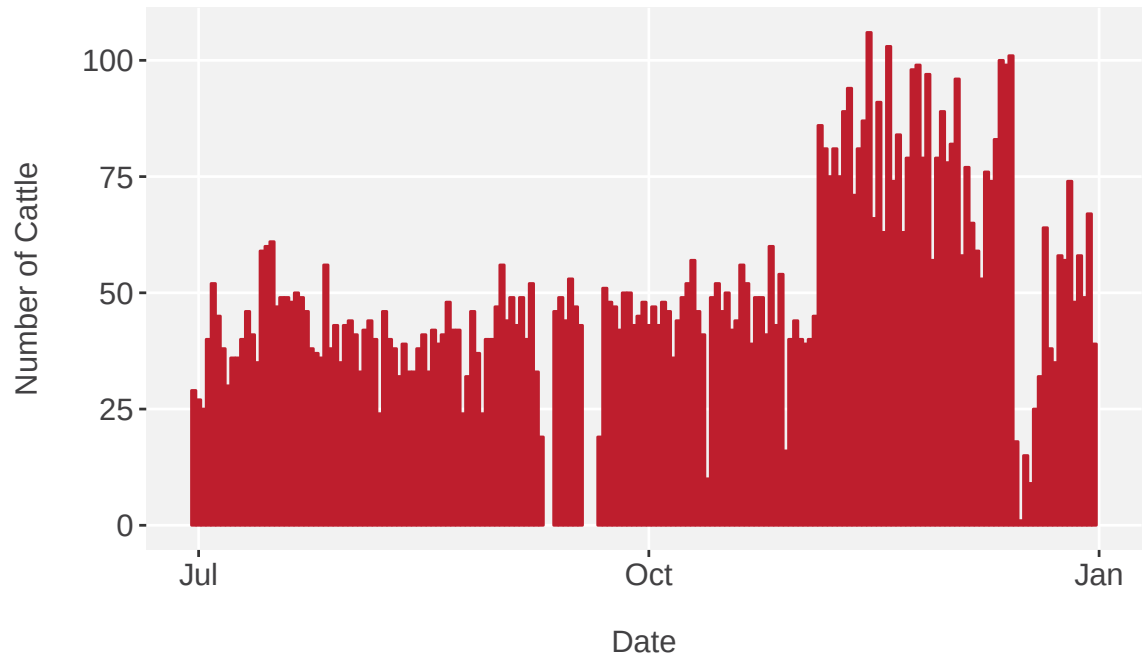
The dataframe provides a good starting point, however, with a few lines more code it is possible to start to organise the data and begin to explore patterns. Most of the sorting and organising can be done using the dplyr package to build up sub-sets of data. For example the following code can generate a table that counts how many animals on each day used an ALMS.

```
alldailydata <- alldata %>% group_by(Date = as.Date(Date)) %>% tally()
```

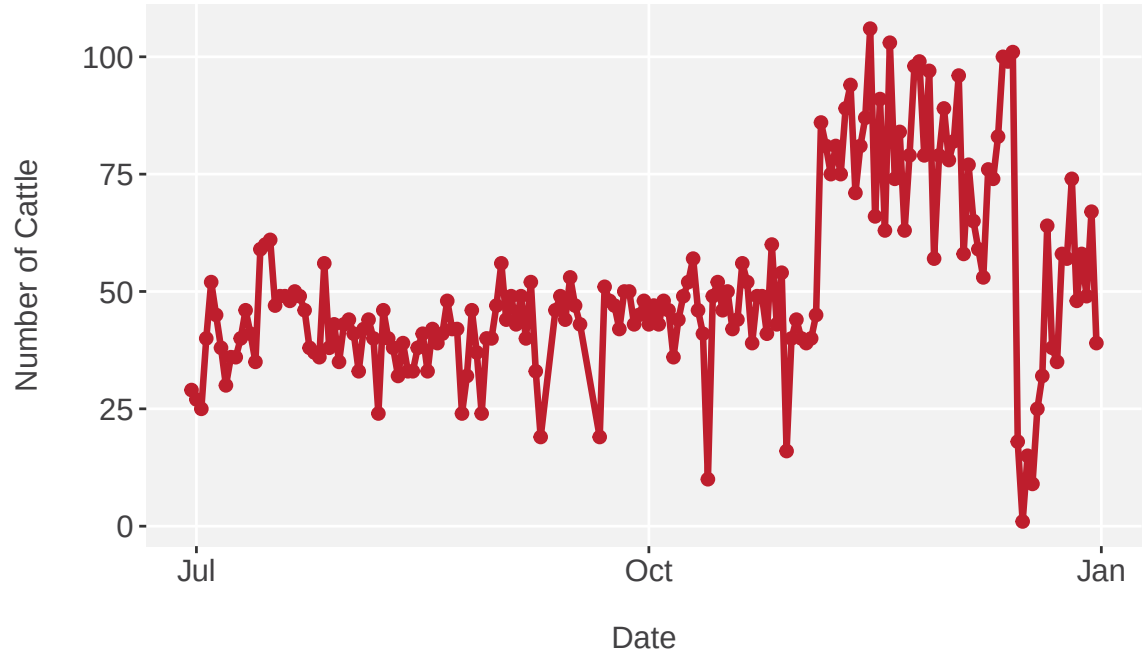
Note that previously the Date column included date and time. The as.Date(Date) command converts all of the date and time values to date values. It is possible to now count the number of animals that used the ALMS on each day using the tally() command. Again the first few rows of the new dataframe are below with the n column being the count of the daily data:

Date	n
2018-06-30	29
2018-07-01	27
2018-07-02	25
2018-07-03	40
2018-07-04	52
2018-07-05	45

The dataframe called alldailydata can very easily be used to plot animal useage over time using the ggplot2 package and the gg_col function:



Or if preferred a line might graph might be better using the gg_line function:



This document aims to provide some basic help to get started using DMMongoDB. The weeklywts() function can be used like the dailywts() function but provides the weekly average weight data. The base DMMongoDB package can provide the data which, if organising on time can be used to derive association metrics between pairs of cattle. The automated updates to the DataMuster database will now be occuring every ten minutes and will includes the seperate data on RFID time reads. These new functions will allow a quick check of cattle access and provide better data to determine measures of association.

The propcattlecheck() function does a check of all cattle that have crossed an ALMS with the date, time, RFID number and their weight within a predefined period. The period is days before present, where 0 is the default to return all cattle today. If no number is set it will only return cattle from today. So the following will search Belmont for the previous two days:

```
checkBelmont <- propcattlecheck("Belmont", days = 0)
```

This results in a dataframe:

RFID	Date	Weight	TimeSinceLastVisit
982 123707586335	2019-02-19 12:58:29	227.22	17.32971 mins
982 123708458246	2019-02-19 12:51:15	299.95	24.56305 mins
982 123707586442	2019-02-19 12:50:49	269.72	24.99638 mins
982 123707586329	2019-02-19 12:49:42	161.24	26.11305 mins
982 123707586292	2019-02-19 12:46:42	207.67	29.11305 mins
942 000002679208	2019-02-19 12:41:09	632.00	34.66305 mins

DATA MUSTER

RFID	Date	Weight	TimeSinceLastVisit
982 125003974215	2019-02-19 12:40:44	510.00	35.07971 mins
982 123708458258	2019-02-19 12:33:41	258.95	42.12971 mins
982 123707586327	2019-02-19 12:31:09	196.06	44.66305 mins
942 000002765009	2019-02-19 12:19:50	546.00	55.97971 mins
982 123707586290	2019-02-19 12:19:01	268.00	56.79638 mins
982 123707586297	2019-02-19 12:19:01	374.30	56.79638 mins
982 123707586321	2019-02-19 12:19:01	371.01	56.79638 mins
982 123707586330	2019-02-19 12:19:00	303.64	56.81305 mins
982 123476836221	2019-02-19 12:16:23	572.00	59.42971 mins
982 123476836221	2019-02-19 12:16:19	354.00	59.49638 mins
982 123476836229	2019-02-19 12:16:08	530.00	59.67971 mins
982 123708458258	2019-02-19 12:13:03	247.41	62.76305 mins
982 123707586325	2019-02-19 12:12:48	314.06	63.01305 mins
982 123707586443	2019-02-19 12:01:54	274.49	73.91305 mins
982 123707586323	2019-02-19 11:49:59	305.49	85.82971 mins
982 123707586441	2019-02-19 11:33:30	287.18	102.31305 mins
942 000007192945	2019-02-19 11:23:26	530.00	112.37971 mins
982 123707585395	2019-02-19 11:23:12	588.00	112.61305 mins
982 123707585395	2019-02-19 11:23:08	419.00	112.67971 mins
982 123707585395	2019-02-19 11:23:05	676.00	112.72971 mins
942 000002680070	2019-02-19 11:22:58	284.00	112.84638 mins
942 000002680070	2019-02-19 11:22:56	664.00	112.87971 mins
942 000002765009	2019-02-19 11:22:44	626.00	113.07971 mins
942 000002765009	2019-02-19 11:22:41	608.00	113.12971 mins
942 000002679490	2019-02-19 11:22:33	622.00	113.26305 mins
942 000002679490	2019-02-19 11:22:26	700.00	113.37971 mins
982 123476836234	2019-02-19 11:22:13	648.00	113.59638 mins
942 000007133920	2019-02-19 11:22:05	650.00	113.72971 mins
942 000002679947	2019-02-19 11:21:50	778.00	113.97971 mins
982 123707586316	2019-02-19 11:15:53	215.97	119.92971 mins
942 000002769775	2019-02-19 11:06:15	558.00	129.56305 mins
982 123707586315	2019-02-19 10:52:54	170.21	142.91305 mins
982 123707586441	2019-02-19 10:52:54	246.19	142.91305 mins
982 123707586452	2019-02-19 10:36:23	288.14	159.42971 mins
982 123707586307	2019-02-19 10:35:48	221.80	160.01305 mins
982 123707586327	2019-02-19 10:33:22	194.23	162.44638 mins
982 123707586301	2019-02-19 09:47:27	536.68	208.36305 mins
982 123707586330	2019-02-19 09:47:27	301.93	208.36305 mins
982 123707586335	2019-02-19 09:43:37	232.19	212.19638 mins
942 000007140345	2019-02-19 09:34:11	499.00	221.62971 mins
982 123708458264	2019-02-19 09:31:58	314.91	223.84638 mins
982 123707586296	2019-02-19 09:05:01	306.85	250.79638 mins
982 123707586314	2019-02-19 09:05:01	378.06	250.79638 mins
982 123707586332	2019-02-19 09:05:01	227.77	250.79638 mins
982 123707586443	2019-02-19 09:05:01	271.88	250.79638 mins
982 123707586291	2019-02-19 08:56:17	271.82	259.52971 mins
982 123708458264	2019-02-19 08:56:17	307.67	259.52971 mins

RFID	Date	Weight	TimeSinceLastVisit
982 123707586315	2019-02-19 08:50:11	162.72	265.62971 mins
982 123707586395	2019-02-19 08:47:36	275.36	268.21305 mins
982 123707586327	2019-02-19 08:41:17	185.45	274.52971 mins
982 123707585852	2019-02-19 08:40:28	289.47	275.34638 mins
982 123707586441	2019-02-19 08:34:41	285.75	281.12971 mins
982 123707586321	2019-02-19 08:32:49	263.95	282.99638 mins
982 123707586390	2019-02-19 08:32:49	391.35	282.99638 mins
982 123707586297	2019-02-19 08:32:04	267.28	283.74638 mins
982 123707586486	2019-02-19 08:28:32	290.03	287.27971 mins
982 123707586470	2019-02-19 08:28:21	285.32	287.46305 mins
942 000002680000	2019-02-19 08:26:15	518.00	289.56305 mins
982 123707586442	2019-02-19 08:21:10	260.42	294.64638 mins
942 000002679490	2019-02-19 08:19:47	656.00	296.02971 mins
942 000002680070	2019-02-19 08:19:13	606.00	296.59638 mins
982 123707586455	2019-02-19 08:17:55	217.58	297.89638 mins
942 000007192945	2019-02-19 08:17:32	496.00	298.27971 mins
982 123707586290	2019-02-19 08:15:19	265.29	300.49638 mins
982 123708458246	2019-02-19 08:15:04	292.00	300.74638 mins
942 000007022343	2019-02-19 08:11:43	489.00	304.09638 mins
982 123707586301	2019-02-19 07:45:37	228.84	330.19638 mins
982 123707586329	2019-02-19 07:44:24	151.67	331.41305 mins
982 123707586292	2019-02-19 07:44:13	196.40	331.59638 mins
982 123707586319	2019-02-19 07:44:13	234.54	331.59638 mins
982 123707586307	2019-02-19 07:41:56	213.20	333.87971 mins
982 123707586335	2019-02-19 07:40:44	222.62	335.07971 mins
982 123707586323	2019-02-19 06:59:37	295.53	376.19638 mins
982 123707586452	2019-02-19 06:42:16	282.71	393.54638 mins
982 123708458258	2019-02-19 06:40:20	242.06	395.47971 mins
982 123707586330	2019-02-19 06:34:44	285.59	401.07971 mins
982 123707586443	2019-02-19 06:30:30	256.32	405.31305 mins
982 123707586452	2019-02-19 05:54:31	278.47	441.29638 mins
982 123708458258	2019-02-19 05:54:11	236.35	441.62971 mins
942 000002720466	2019-02-19 00:07:40	556.00	788.14638 mins

The RFIDcattlecheck() function allows an individual or list of RFID numbers to be checked and to again look back over several days. The following can be used to check a single RFID from CQIRP (it is possible to provide a list of RFID numbers if several cattle need to be checked) to get a list of visits over the last week.

```
checkCQIRP <- RFIDcattlecheck("982 123498173907", days = 7)
```

These checkCQIRP dataframe shown below provides information on the frequency of visits to water from cow RFID "982 123498173907" over the last seven days:

DATA MUSTER

RFID	Date	Weight	TimeSinceLastVisit
982 123498173907	2019-02-19 10:06:51	445.29	3.149408 hours
982 123498173907	2019-02-18 07:30:44	431.77	29.751352 hours
982 123498173907	2019-02-17 10:43:40	440.67	50.535797 hours
982 123498173907	2019-02-17 07:25:43	437.56	53.834963 hours
982 123498173907	2019-02-16 09:56:53	441.17	75.315519 hours
982 123498173907	2019-02-15 11:18:59	421.23	97.947186 hours
982 123498173907	2019-02-15 10:38:16	435.42	98.625797 hours
982 123498173907	2019-02-14 11:48:53	435.85	121.448852 hours
982 123498173907	2019-02-14 08:13:01	436.70	125.046630 hours
982 123498173907	2019-02-13 06:19:07	426.66	150.944963 hours