Logbook and supplementary registrations at the Kappel farm

Anders Ringgaard Kristensen

PigIT Report No. 3 • April 2014

Logbook and supplementary registrations at the Kappel farm

Anders Ringgaard Kristensen

PigIT report No. 3 • April 2014

Centre for Herd-oriented Education, Research and Development
Department of Large Animal Sciences
University of Copenhagen

Contents

Chapter 1

Introduction

The Kappel family farm is one of the farms used for data collection in the PigIT project. It is situated in Boddum, Thy, in the northwestern part of Jutland. The sensor data collected from the farm are automatically transferred to the central PigIT database located at Aarhus University. In addition to the sensor data, a range of supplementary data is collected from the herd using various data sources.

The objective of this report is to provide a systematic overview of the supplementary data collected at the farm and to describe how to migrate the supplementary data into the PigIT database. The general data model for supplementary data is described in a separate note (Kristensen, 2014).

Chapter 2

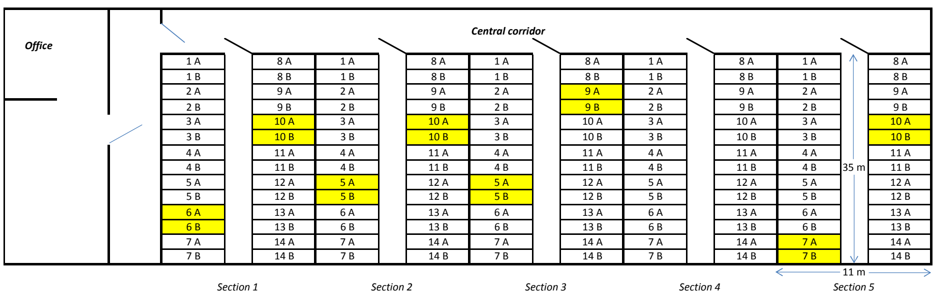
Farm and data

2.1 The finisher unit

Data collection takes place in a finisher unit consisting of five sections each containing 28 pens. Two pens share the same feed dispenser and the same water nipple, so a section is equipped with 14 feed dispensers and 14 water nipples. A schematic diagram of the unit can be seen as Figure 2.1. A schematic diagram of two pens sharing a feed dispenser and a water nipple is shown in Figure 2.2. The feeding system is a liquid feeding system.

Sensor registrations and continuous video recordings are collected from 4 pens in each of Sections 1, 2, 3 and 5 of the unit. Those experimental pens are shown in yellow color in Figure 2.1. Sensor registrations include for each pen the temperature (thermometers) at two locations. For each two pens (sharing a feed dispenser) the water consumption (flow meter) and the feed consumption (from the feeding computer) are recorded. At section level the temperature and humidity are retrieved from the climate computer.

Video recordings are continuously collected from two cameras over each pen. One of the cameras is primarily intended for weight assessment whereas the main



Office
Central corridor
1 A 1 B 2 A 2 B 3 A 3 B 4 A 4 B 5 A 5 B 6 A 6 B 7 A 7 B
8 A 8 B 9 A 9 B 10 A 10 B 11 A 11 B 12 A 12 B 13 A 13 B 14 A 14 B
35 m
11 m
Section 1
Section 2
Section 3
Section 4
Section 5

Figure 2.1: Overview of the five sections with pen names as used in the PigIT database. The yellow pens are those followed by the project.

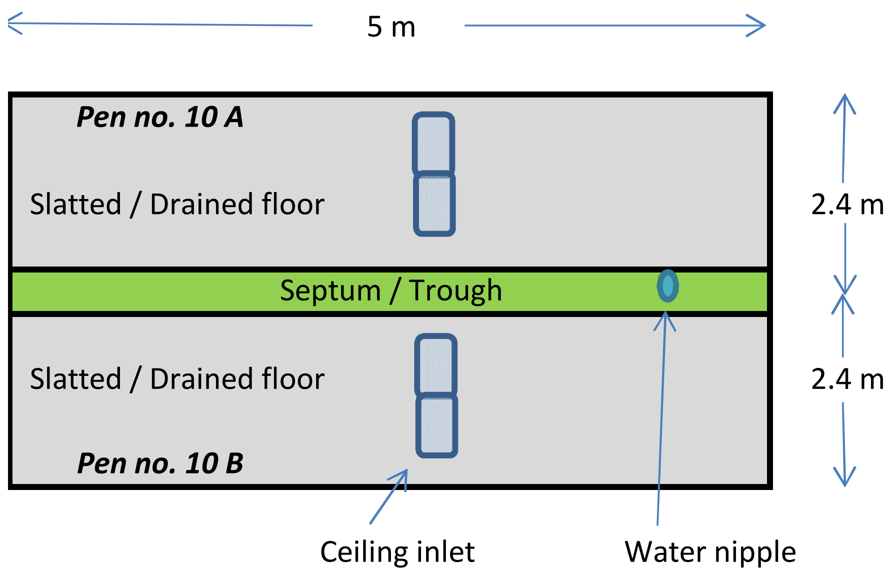


Figure 2.2: Schematic diagram of two pens sharing a feed dispenser and a water nipple. Pens 10A and 10B of Section 5 are used as an example of pen layout.

purpose of the other camera is activity monitoring.

2.2 Overview of supplementary data

Basically, three kinds of data are collected at the Kappel farm (in addition to the automatic registrations):

- A text log maintained by the VSP technician. It is organized according to day and new entries are added at the top. The text is written in Danish.
- A spreadsheet file maintained by the VSP technician.
- The commercial herd database from DLBR IT. Data is entered by the herd staff by a PDA and includes (of relevance to the PigIT project):
  - Daily pen level observations of tail biting, diarrhea and fouling.
  - Data on insertion of pigs into pens (in selected pens with individual pig weight at insertion).
  - Data on dead pigs at pen level.
  - Data on pigs removed from pens.
  - Data on manual weighing of pigs.
  - Data on feedstuffs.

Chapter 3

The logbook and the spreadsheet file

The logbook will be saved in Word format to the DropBox folder so that it is available to the scientists of PigIT. No attempts will be done to integrate it in the PigIT database.

The spreadsheet file contains results of different tests of weighing equipment (manual pig weight and feeding system) as well as a check of the feed consumption at dispenser level against total consumption at system level. It will just be saved in the DropBox folder in Excel format without any attempt to incorporate it in the database.

Chapter 4

Data from the DLBR database

The relevant parts of the DLBR database will be extracted regularly and moved to the PigIT database.

4.1 Relevant tables for the PigIT project

The following tables of the DLBR database are of relevance to the PigIT project.

`PigItVentilData` The table contains the daily pen level observations of tail biting, diarrhea and fouling.

`PigItVentil` The table identifies the PigIT feed dispensers (one for each of the 16 experimental pens) in the standard feed dispenser table of the herd.

`SVentil` The standard table of feed dispensers of the herd.

`SMaengde_Af_Dyr` The standard table containing information about all movements of animals. For the 16 experimental pens, the registrations are at feed dispenser level (i.e. 2 pens sharing a feed dispenser).

`SIndivid_Vaegt` Data at individual pig level of manual weighings in the herd.

`SFoderEmne` Feedstuff table.

`SFoderemneKonvert` A table linking the feedstuffs defined in the feeding system with feedstuffs in the feedstuff table.

`SFoderforbrug` A table with feed consumption at dispenser and feedstuff level.

In the following subsections, the above mentioned tables are described.

4.1.1 `PigItVentilData`

The table contains the daily pen level observations of tail biting, diarrhea and fouling. It has been designed with direct reference to the originally developed paper form shown in Figure B.1. It has the fields shown in Table 4.1.

Table 4.1: Fields of the table `PigItVentilData`

Field	Type	Explanation
Id	uniqueidentifier	Primary key
VentilId	uniqueidentifier	Foreign key, identification of pen
Timestamp	datetime	Time of registration (date)
Ajourini	nvarchar(32)	Initials of the observer
Alle	tinyint	0: Nothing observed 1: Something observed
Diarre	tinyint	0: No diarrhea 1: Diarrhea
DKlatter	tinyint	Number of spots
DBehandling	tinyint	1: No treatment 2: pig treated 3: pen treated 4: section treated
Halebid	tinyint	0: No tail biting 1: Tail biting
HGrisNr	nvarchar(11)	Identification of pig (not used)
HBehandling	tinyint	5: No treatment 6: pigs removed 7: “sinner” removed 8: rooting material provided
Svineri	tinyint	0: No fouling 1: Fouling
SAndel	tinyint	9: 25 percent of lying area 10: 50 percent of lying area 11: All
SBehandling	tinyint	12: No treatment 13: Medication 14: Temperature or sprinkling adjusted

Table 4.2: Fields of the table `PigItVentil`

Field	Type	Explanation
Id	uniqueidentifier	Primary key
VentilId	uniqueidentifier	Foreign key, identification of feed dispenser

Table 4.3: Fields of the table `SVentil`

Field	Type	Explanation
ID	uniqueidentifier	Primary key, identification of dispenser
VentilNr	int	Dispenser number
VentilType	int	(Always 0)
ProcesUdstyr	char(10)	BD: “Big Dutchman” - not PigIT dispenser H: Right pen for PigIT dispensers V: Left pen for PigIT dispensers
StedID	uniqueidentifier	Location (always the same)
ProdgrenNr	int	Production branch (always 7200)

4.1.2 `PigItVentil`

The table is simply a list of the 16 experimental pen identifications from the standard feed dispensers defined in the herd database (in the table `SVentil`). The fields are described in Table 4.2.

4.1.3 `SVentil`

The table lists all feed dispensers defined in the herd. Each dispenser is shared between two pens. In order to be able to distinguish the two pens shared by the 8 dispensers of the experimental pens, eight additional “logical dispensers” have been defined in the table. The fields are listed in Table 4.3.

The dispensers (and thus the pens) are in the DLBR IT database numbered consecutively across section, whereas the dispensers and pens in the PigIT database are numbered within section. The table `FeedDispenserPigPens` in the PigIT database translates the two numbering systems so that the pens sharing a dispenser can be matched two by two. The translation from pen to pen is discussed in Section 6.1.2.

4.1.4 `SMaengde_Af_Dyr`

The table contains information about all movements of animals. For the 16 experimental pens, the registrations are currently at feed dispenser level (i.e. 2 pens sharing a feed dispenser), but in the future the registrations will be at pen level. It contains numerous fields that are not used and which have no relevance for PigIT. It is a very general table which in DLBR IT is used for all kinds of movements of pigs either individually or in groups. The relevant fields are listed in Table 4.4.

Table 4.4: Fields of the table `SMaengde_Af_Dyr`

Field	Type	Explanation
ID	uniqueidentifier	Primary key, identification of event
OmsaetnIndKode	int	1: Bought 2: Sold 3: Dead 4: To weaners 6: To finishers 7: To gilts 8: To boars
OmsaetnAfgKode	int	(Same values as OmsaetnInd-Kode)
OmsaetnDato	int	Date of movement YYYYM-MDD
OmsaetnAntal	int	Number of pigs
OmsaetnVgtMgde	float	Total weight
DyrGruppe	int	4: Weaners 6: Finishers
InternFlytningUngdyrID	uniqueidentifier	Identification of pig
VentilFraID	uniqueidentifier	Moved *from* dispenser
VentilTilID	uniqueidentifier	Moved *to* dispenser
Ajourtid	datetime	Timestamp of last update
Ajourini	nvarcha(32)	Initials of employee

Table 4.5: Fields of the table `SIndivid_Vaegt`

Field	Type	Explanation
ID	uniqueidentifier	Primary key, identification of weighing
GrisID	uniqueidentifier	Identification of pig
Vaegt	float	Weight in kg
Tid	datetime	Timestamp (date) of the weighing

Table 4.6: Fields of the table `SFoderEmne`

Field	Type	Explanation
ID	uniqueidentifier	Primary key, identification of feedstuff
FoderkodeParti	int	An integer defining the kind of feed
FoderNavnLang	char(30)	A descriptive name
Egenskabindhold	text	A string describing the nutritional content*
Ajourtid	datetime	The time for last update

* See description in the text.

Pigs entering a PigIT experimental pen with manual weighing are registered individually with weight in this table, and if a pig is removed it is also registered individually with weight. Pigs entering PigIT experimental pens *without* manual weighing are *not* registered individually.

4.1.5 `SIndivid_Vaegt`

This table contains manual weighing of individual pigs. The fields are listed in Table 4.5. The field `GrisID` refers to the field `InternFlytningUngdyrID` of the table `SMaengde_Af_Dyr` so that the feed dispenser of the pig can be identified.

4.1.6 `SFoderEmne`

This table is the feedstuff table of the herd. It contain numerous fields of which only few are of relevance for PigIT and many are not used. The relevant fields are listed in Table 4.6.

The most interesting field is `Egenskabindhold` which contains detailed information about the nutritional contents of the feedstuffs. It is a string of arbitrary length with a special format. It consists of substrings separated by the character "&". Each substring describes a property of the feedstuff (eg. the energy or protein content) and it has three fields separated by the character ";". The first field (three digits) identifies the property, the second (two digits) is not used and the third holds the numerical value (the decimal point is a comma). An example (in excerpt) of a string for the feedstuff "technical fat" is

Table 4.7: Fields of the table `SFoderemneKonvert`

Field	Type	Explanation
ID	uniqueidentifier	Primary key, identification of feedstuff
Niveau	char(1)	Always “E”
Kodeparti	int	An integer defining the kind of feed
Navn	char(30)	A descriptive name
KodeFF	char(11)	A code identifying the feedstuff in the feeding system
PartiFF	char(2)	Unclear meaning
NavnFF	char(50)	Another descriptive name
Sortering	char(20)	Not used
Status	int	Seems always to be 0
Markering	int	Seems to be 0, 1 or 2

```
    001;00;99,5 &161;00;3,82914572864322& ... 481;00;0&
```

The interpretation is that Property 1 (dry matter percentage) is 99.5, Property 161 (energy) is 3.829 Scandinavian Feed Units, and Property 461 is 0.

A full list of property codes and names is shown in Appendix D.

4.1.7 `SFoderEmneKonvert`

This table links the feedstuffs defined in the feeding system to the feedstuffs of the table `SFoderEmne`. It is the field `Kodeparti` that must be searched for in the other table. `SFoderEmneKonvert` has the fields shown in Table 4.7.

4.1.8 `SFoderForbrug`

The table contains information about feed consumption at feedstuff and dispenser level. Typically it is at monthly intervals. Relevant fields are shown in Table 4.8.

Table 4.8: Fields of the table `SFoderForbrug`

Field	Type	Explanation
ID	uniqueidentifier	Primary key, identification of record
TidFra	int	First date of the interval YYYYMMDD
TidTil	int	Last date of the interval YYYYMMDD
ForbrugKg	float	Consumption of feedstuff in kg
ForbrugFEs	float	Consumption of feedstuff in FEs (often empty)
DyrGruppe	int	Animal group 6 = finishers
FoderEmneID	uniquidentifier	Forteign key - identifies feedstuff in SFoderEmne
VentilID	uniquidentifier	Forteign key - identifies dispenser in SVen-til
Ajourtid	datetime	Time of last update

Chapter 5

Extracting data from the DLBR database

In this subsection the database is accessed through R using the RODBC library. The variable `kappel` is a database connection returned by the `odbcConnect` command of the library.

5.1 Extracting supplementary registrations

As mentioned, the supplementary registrations of diarrhea, tail biting and fouling are in the table `PigItVentilData`. The dispenser identification used in that table are linked to the dispenser identities in the standard herd database table `SVentil` in a small table called `PigItVentil`. The following code examples show first how to obtain a list of the 16 pens and then how to extract the supplementary registrations.

5.1.1 Feed dispensers

A list of PigIT pens is achieved by merging the PigItVentil table with the SVentil table as follows:

```
> dispensers = sqlQuery(kappel, "SELECT VentilNr, Procesudstyr
                         FROM SVentil, PigItVentil
                         WHERE PigItVentil.VentilId = SVentil.ID")
```

And the resulting list is:

```
> dispensers
   VentilNr Procesudstyr
1         2    H
2         2    V
3         8    H
4         8    V
5        16    V
6        16    H
```

```
7         23   H
8         23   V
9         29   H
10        29   V
11        37   H
12        37   V
13        58   V
14        58   H
15        63   H
16        63   V
```

A pen is defined by dispenser number and the labels "H" and "V" for right and left pen, respectively.

5.1.2 Diarrhea

A diarrhea case is indicated by the field `Diarre = 1` in the table. The following code extracts all diarrhea cases (including their characteristics). Afterwards, it creates a subset of registrations exclusively regarding Dispenser 63, right pen ("H").

```
> diarrhea =  sqlQuery(kappel,
+  "SELECT Timestamp, DKlatter, DBehandling, VentilNr, ProcesUdstyr
+   FROM PigItVentilData, SVentil
+   WHERE PigItVentilData.VentilId = SVentil.ID AND Diarre = 1
+   ORDER BY Timestamp")
> length(diarrhea[,1])
[1] 76
> diarrhea63H = diarrhea[(diarrhea$VentilNr == 63 &
+                        substr(diarrhea$ProcesUdstyr, 1, 1) %in% c("H")),]
> diarrhea63H
    Timestamp DKlatter DBehandling VentilNr ProcesUdstyr
4  2013-10-21        1           1       63    H
8  2013-10-25        1           1       63    H
10 2013-10-27        1           1       63    H
13 2013-10-31        3           3       63    H
```

As it is seen, there is a total of 76 diarrhea registrations. Out of them, 4 refer to Dispenser 63, pen "H", and on the 31st of October the pen has been treated.

5.1.3 Tail biting

A case of tail biting is indicated by the field `Halebid = 1` in the table. The following code extracts all cases of tail biting (including their characteristics). Afterwards, it creates a subset of registrations exclusively regarding Dispenser 63, right pen ("H").

```
> tailBite =  sqlQuery(kappel,
+   "SELECT Timestamp, HGrisNr, HBehandling, VentilNr, ProcesUdstyr
+   FROM SVentil, PigItVentilData
+   WHERE PigItVentilData.VentilId = SVentil.ID AND Halebid = 1
+   ORDER BY Timestamp")
> length(tailBite[,1])
[1] 8
> tailBite63H = tailBite[(tailBite$VentilNr == 63 &
+                       substr(tailBite$ProcesUdstyr, 1, 1) %in% c("H")),]
> tailBite63H
   Timestamp HGrisNr HBehandling VentilNr ProcesUdstyr
3 2013-10-27      NA           5       63    H
```

```
5 2013-11-01       NA            5        63    H
6 2013-11-04       NA            5        63    H
```

As it is seen, there is a total of 8 observations of tail biting. Out of them, 3 refer to Dispenser 63, pen "H", but there has been no treatment.

5.1.4 Fouling

A case of pen fouling is indicated by the field `Svineri = 1` in the table. The following code extracts all cases of pen fouling (including their characteristics). Afterwards, it creates a subset of registrations exclusively regarding Dispenser 63, right pen ("H").

```
> fouling =  sqlQuery(kappel,
+   "SELECT Timestamp, SAndel, SBehandling, VentilNr, ProcesUdstyr
+   FROM SVentil, PigItVentilData
+   WHERE PigItVentilData.VentilId = SVentil.ID AND Svineri = 1
+   ORDER BY Timestamp")
> length(fouling[,1])
[1] 87
> fouling63H = fouling[(fouling$VentilNr == 63 &
+                          substr(fouling$ProcesUdstyr, 1, 1) %in% c("H")),]
> fouling63H
    Timestamp SAndel SBehandling VentilNr ProcesUdstyr
19 2013-10-22      9          12       63    H
74 2013-11-18      9          12       63    H
77 2013-11-19      9          12       63    H
85 2013-12-16      9          12       63    H
```

As it is seen, there is a total of 87 observations of fouling. Out of them, 4 refer to Dispenser 63, pen "H", but there has been no treatment.

5.2 Extracting pig data

The data discussed in this subsection concerns data about moving pigs to and from the PigIT pens.

5.2.1 Pigs inserted

Pigs inserted into the finisher unit are listed in the table `SMaengde_Af_Dyr`. In order to extract a list of pigs, sorted by date, inserted into the PigIT experimental pens, we need to merge with `PigItVentil` and `SVentil`. The following R command will extract such a list including number of pigs, dispenser identification, weight at insertion and date. Afterwards, it creates a subset of pigs inserted at feed dispenser number 8 and counts the total number.

```
> pigsInserted = sqlQuery(kappel,
+   "SELECT OmsaetnAntal, VentilNr, ProcesUdstyr, OmsaetnVgtMgde, OmsaetnDato
+   FROM SMaengde_Af_Dyr, SVentil, PigItVentil
+   WHERE PigItVentil.VentilId = SVentil.ID
+   AND VentilTilID = SVentil.ID ORDER BY OmsaetnDato")
> pigsInserted[1:5,]
  OmsaetnAntal VentilNr ProcesUdstyr OmsaetnVgtMgde OmsaetnDato
1            1        2    V                     25    20120926
```

```
2             1         2   V                       19    20120926
3             1         2   V                       21    20120926
4             1         2   V                       22    20120926
5             1         2   V                       22    20120926
> disp8 = pigsInserted[pigsInserted$VentilNr == 8,]
> length(disp8$VentilNr)
[1] 72
```

In other words, a total of 72 pigs have been inserted at Dispenser 8 during the period. It should be noticed that currently only the dispenser (“`VentilNr`”) is registered on insertion. It is not known whether the pig is inserted into the left or the right pen belonging to the dispenser.

5.2.2 Pigs dead or removed

Dead or removed pigs from the PigIT experimental pens are extracted in a similar way from `SMaengde_Af_Dyr`. Dead pigs have `OmsaetnAfgKode = 3`. The following code extracts and lists them all (there seems to be 9):

```
> pigsDead = sqlQuery(kappel,
+   "SELECT OmsaetnAntal, VentilNr, ProcesUdstyr, OmsaetnVgtMgde, OmsaetnDato
+   FROM SMaengde_Af_Dyr, SVentil, PigItVentil
+   WHERE PigItVentil.VentilId = SVentil.ID
+   AND VentilFraID = SVentil.ID
+   AND OmsaetnAfgKode = 3 ORDER BY OmsaetnDato")
> pigsDead
  OmsaetnAntal VentilNr ProcesUdstyr OmsaetnVgtMgde OmsaetnDato
1             1         2   V                     30.0    20121011
2             1         8   V                     28.0    20121101
3             1         8   V                     39.0    20121109
4             1         8   V                     41.0    20121111
5             1         2   V                     29.0    20130212
6             1         2   V                     47.0    20130219
7             1         8   V                     66.5    20130222
8             1        23   V                     50.0    20130615
9             1        23   V                     66.0    20130622
```

Pigs removed from a pen will have `OmsaetnAfgKode = 6`. A total of 4 pigs seem to have been removed from the PigIT pens:

```
> pigsMoved = sqlQuery(kappel,
+   "SELECT OmsaetnAntal, VentilNr, ProcesUdstyr, OmsaetnVgtMgde, OmsaetnDato
+   FROM SMaengde_Af_Dyr, SVentil, PigItVentil
+   WHERE PigItVentil.VentilId = SVentil.ID AND VentilFraID = SVentil.ID
+   AND OmsaetnAfgKode = 6 ORDER BY OmsaetnDato")
> pigsMoved
  OmsaetnAntal VentilNr ProcesUdstyr OmsaetnVgtMgde OmsaetnDato
1             1         8   V                     23.0    20121101
2             1         2   V                     30.5    20130115
3             1         8   V                     30.0    20130115
4             1        23   V                     31.0    20130607
```

5.2.3 Pig stock in a pen over time

Given the information about pigs inserted and dead and removed pigs it is possible to calculate the number of pigs present at a given dispenser on a given day. An

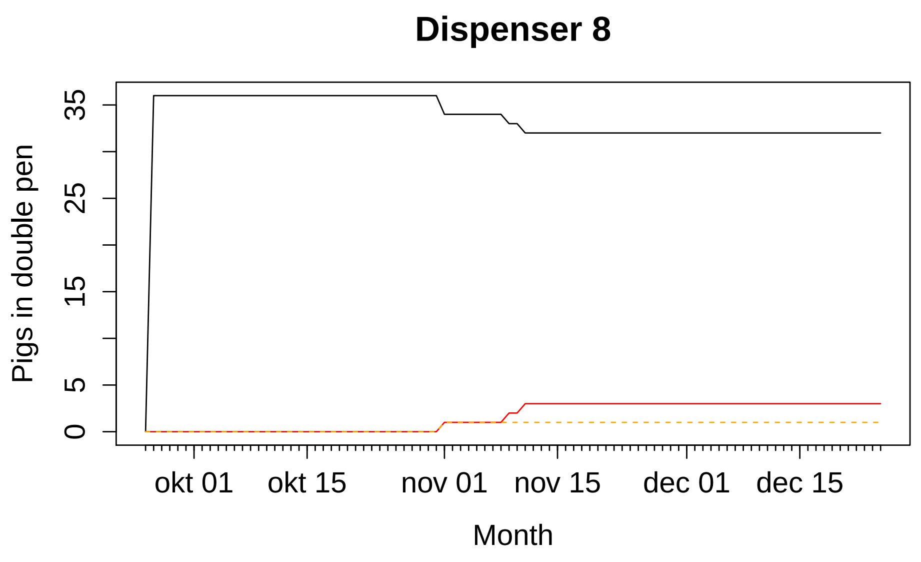


Figure 5.1: Pig stock over time for a batch of pigs at Dispenser 8. Black line is the stock, red line is cumulated number of dead pigs and dashed orange is cumulated pigs removed.

example is shown in Figure 5.1 for a batch at Dispenser 8. The data of the plot has been extracted by use of an R function written for the purpose. The source code is seen in Appendix C.1. It returns a data frame where each row corresponds to a date. The variables are date, total stock, number of dead pigs and number of removed pigs.

5.3 Extracting data on manual weighing of pigs

A special activity in the Kappel herd is manual weighing of pigs in selected pens of one of the sections. The pigs in those pens are weighed regularly and all pigs in the entire section are weighed individually at insertion and at the end of the growing period. The manual weighing data is in the table `SIndivid_Vaegt` and by merging with information in `SMaengde_Af_Dyr` the location (at dispenser level) of the pigs is obtained. By also matching the dispenser identity from PigItVentil the data can be limited to PigIT pens.

The following code extracts all manual weighings in the PigIT pens and lists the first 10 observations.

```
> individualWeighings = pigsInserted = sqlQuery(kappel,
+   "SELECT GrisID, Vaegt, Tid, OmsaetnAntal, VentilNr,
+   ProcesUdstyr, OmsaetnVgtMgde, OmsaetnDato
+   FROM SMaengde_Af_Dyr, SIndivid_Vaegt, SVentil, PigItVentil
+   WHERE GrisID = InternFlytningUngdyrID
+   AND PigItVentil.VentilId = SVentil.ID
+   AND VentilTilID = SVentil.ID ORDER BY OmsaetnDato, VentilNr, Tid, GrisID")
> # Convert pig ID to unique integer
> individualWeighings$Pig = as.integer(individualWeighings$GrisID)
> # Skip original ID
> individualWeighings = individualWeighings[,-1]
> # Print the first 10
> individualWeighings[1:10,]
   Vaegt         Tid OmsaetnAntal VentilNr ProcesUdstyr OmsaetnVgtMgde OmsaetnDato Pig
1     23 2012-09-26            1        2   V                       23    20120926 099
2     27 2012-09-26            1        2   V                       27    20120926 016
3     27 2012-09-26            1        2   V                       27    20120926 088
4     20 2012-09-26            1        2   V                       20    20120926 198
5     26 2012-09-26            1        2   V                       26    20120926 360
6     22 2012-09-26            1        2   V                       22    20120926 029
7     23 2012-09-26            1        2   V                       23    20120926 342
8     31 2012-09-26            1        2   V                       31    20120926 076
9     24 2012-09-26            1        2   V                       24    20120926 059
10    23 2012-09-26            1        2   V                       23    20120926 293
```

One way of visualizing the manual weighing data is by plotting individual growth curves for pigs belonging to a given dispenser and batch. Plots for two dispensers of a batch are shown in Figure 5.2. The plots have been created by use of the R function `plotWeigtByDispenseAndPig` shown in Appendix C.1.

In pens were only the weight at insertion and at the end of the growing period are known, the plots will, of course, be simpler as illustrated in Figure 5.3

5.4 Visualizing supplementary registrations with other data

A function called `addObservationsToPlot` has been written. It adds observations of diarrhea, tail biting and/or fouling to existing plots. The souce code is shown in Appendix C.1.

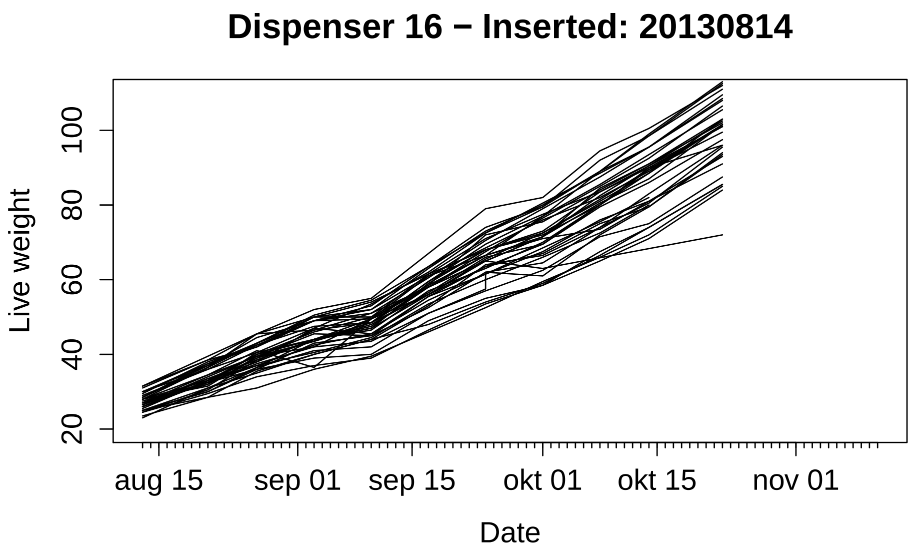
Figure 5.4 shows an example of combining plots of pig stock and supplementary registrations of diarrhea and fouling (there were no tail biting observations in the double pen).

Figure 5.5 shows an example of combining plots of individual pig weights for a recently inserted batch with observations of diarrhea.

5.5 Data on feeding and feedstuffs

The PigIT database stores daily feed consumption records at dispenser and feedstuff level in the table PigIt.Feeding. Each record links uniquely to a feedstuff listed as a record in the table `PigIt.Feed`. Each feedstuff in the table has a field called `FeedComponentNumber` which can be used to link the feedstuff in the PigIt database to the feedstuffs (with nutritional contents) in the DLBR database table `SFoderEmne` (through the translation table `SFoderemneKonvert`).

The procedure is to link feedstuffs in table `PigIt.Feed` with feedstuffs in table `SFoderemneKonvert` by matching the fields `FeedComponentNumber` in `PigIt.Feed` and `KodeFF` in `SFoderemneKonvert`. Then, the matched



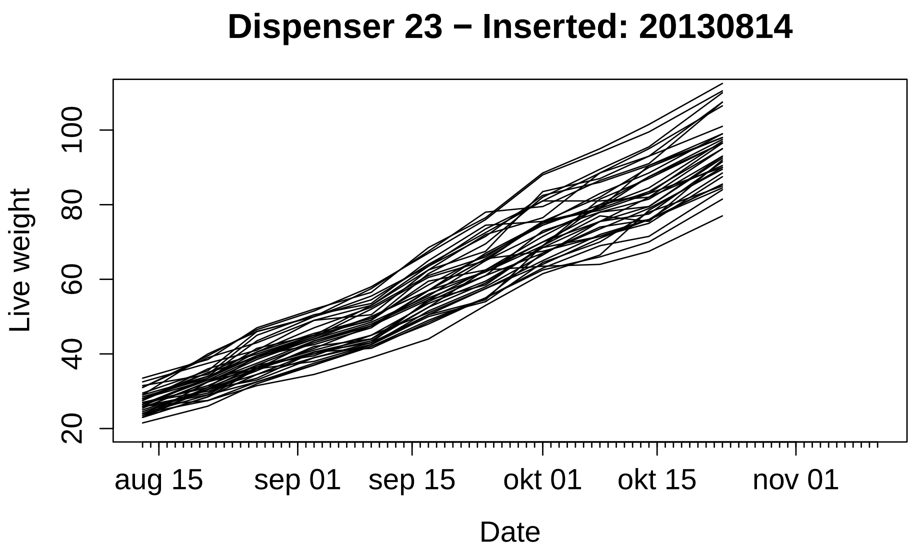


Figure 5.2: Individual growth curves for two dispensers in a batch.

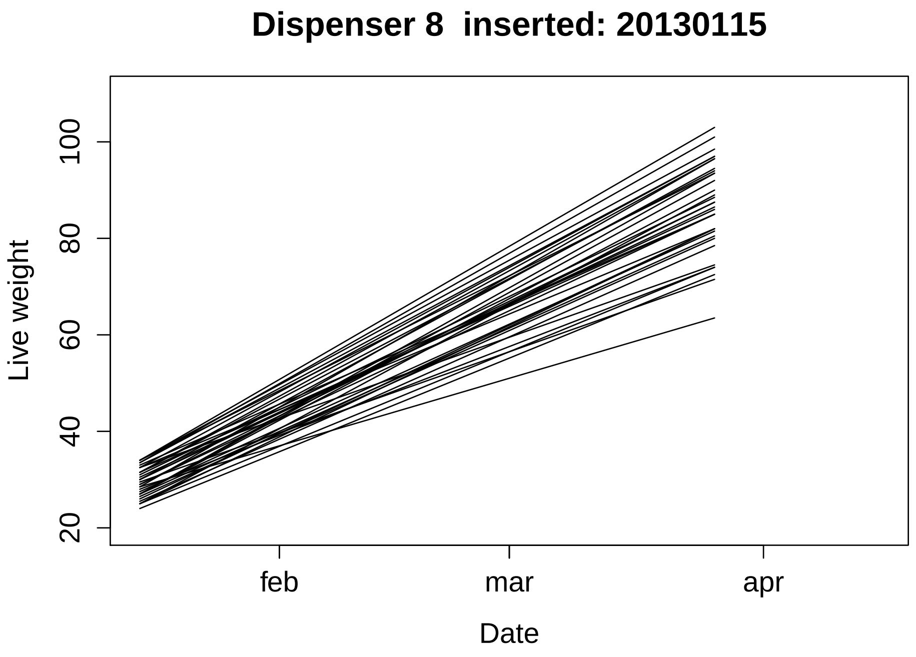


Figure 5.3: Individual growth curves for Dispenser 8 with only two weighings.

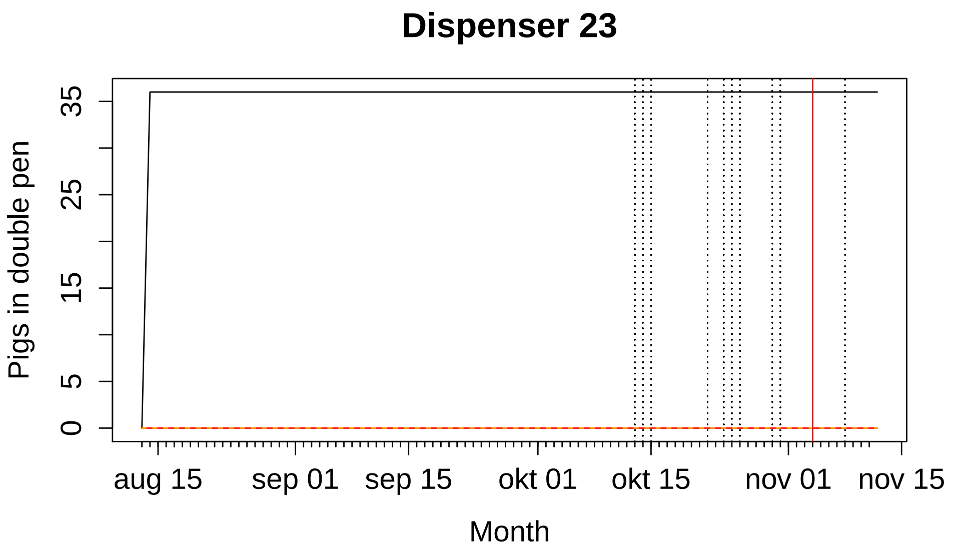


Figure 5.4: Pig stock for Dispenser 23 combined with observations of diarrhea and fouling (both pens belonging to Dispenser 23. A red line corresponds to an observation of diarrhea and a black dashed line corresponds to fouling.

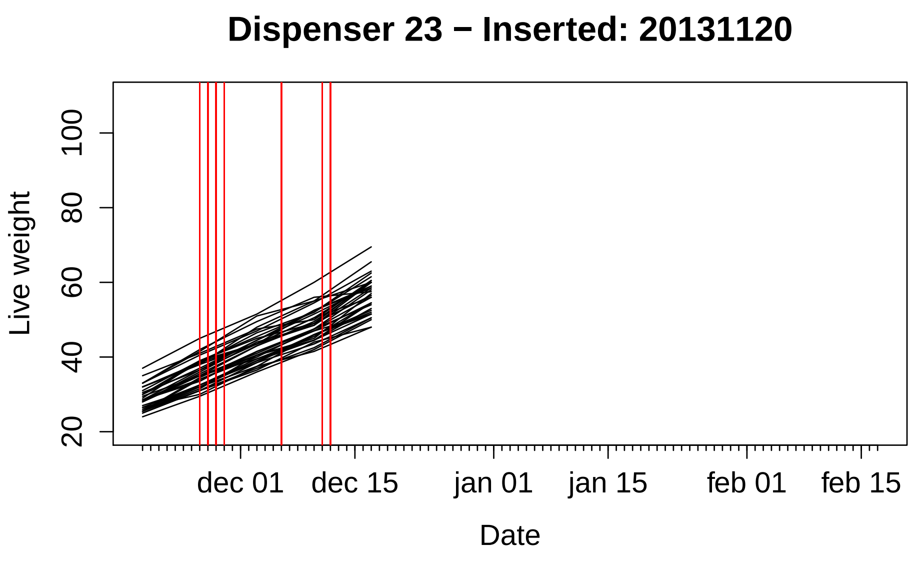


Figure 5.5: Individual growth curves for Dispenser 23 combined with observations of diarrhea (both pens belonging to Dispenser 23. The red lines correspond to observations of diarrhea on a given day.

feedstuff is looked up in the `SFoderEmne` table by matching the fields `Kodeparti` and `Foderkodeparti`. This match is not unique, because a new record is created in `SFoderEmne` each time the contents of a feedstuff is edited. The feedstuff in the PigIT database is, therefore, linked to *the most recent record in `SFoderEmne` before the `CreateTime` in `PigIt.Feed`*. The following code will do the trick:

```
pigItFeedsKappelAnalysis =
  sqlQuery(kappel,
           "SELECT pigitdb.PigIt.Feed.CreateTime, FeedName,
           FeedComponentNumber, Navn, Kodeparti, KodeFF, PartiFF,
           NavnFF, Markering, FoderNavnLang, Ajourtid, EgenskabIndhold
           FROM pigitdb.PigIt.Feed, SFoderemneKonvert, SFoderEmne
           WHERE FeedComponentNumber = KodeFF AND pigitdb.PigIt.Feed.DataBaseId = 1
           AND Kodeparti = FoderkodeParti AND
           Ajourtid IN
           (SELECT max(Ajourtid) FROM SFoderEmne WHERE Kodeparti = FoderkodeParti
           AND Ajourtid < pigitdb.PigIt.Feed.CreateTime)")
pigItFeedsKappelAnalysis
```

This results in a data frame where the nutritional content of the feedstuff is represented as a string as described in Section 4.1.6. In order to parse those strings and interpret them as numerical characteristics, a function called `addAnalysis` has been written. The source code is seen in section C.1. Calling the function on the feedstuff data frame created above is done simply by reading the codes and property names from a CSV file and then calling the function on the data frame. In this example, the nutritional codes 1, 9, 165 and 185 are included:

```
> codes = read.table("P:/PigIT/DBEgenskaber.csv", sep = ";", header = TRUE)
> nf = addAnalysis(pigItFeedsKappelAnalysis, codes,  c(1, 9, 165, 185))
> nf[1:5, c(2, 6, 12:15)]
  FeedName KodeFF Toerstof %  Protein     FEsv  FEsv/100kg
1 SOYA MOL     14        88.5   42.856 1.084746 100.9747336
2 SOYAMOLE     39        88.5   42.856 1.084746 100.9747336
3 SOYA MOL      4        88.5   42.856 1.084746 100.9747336
4 SP LILLE     35        75.0    0.000 1.585366   0.1357287
5  SP STOR     36        75.0    0.000 1.585366   0.1357287
```

Chapter 6

Migrating data to the PigIT database

The data model for the supplementary registrations in the PigIT database is described in a separate report (Kristensen, 2014). In the following sections it is described how to fill the various tables of the PigIT database from the tables of the DLBR database.

6.1 Basic information

6.1.1 Maintaining a table of observers

Most registrations in the DLBR database have an identification of the employee responsible for the individual observation. The identification is usually in the form of the initials of the employee. These identifications are maintained in the PigIT database, but they are stored in a separate table called `PigIt.Observer`. Therefore, whenever a registration is added to one of the main tables, it must be checked that the observer exists in the `PigIt.Observer` table. If the entry does not exist it must be created. An R function, `getObserver`, looking the observer up in the table and adding an entry if he/she does not exist has been written. It is shown in Appendix C.1. The function is called by all routines copying observations from the DLBR database to the PigIT database.

6.1.2 Matching pens across databases

In the PigIT database the pens are numbered within sections as shown in Figure 2.1. In the DLBR database, the pens are identified by dispenser number (the field `VentilNr`) and whether it is the "left" or "right" pen (the field `ProcesUdstyr`) sharing the dispenser as shown in Table 4.3. The two numbering systems can be partially matched through the PigIT table `FeedDispenserPigPens` which for each pen in Figure 2.1 provides the dispenser number. The translation from

Table 6.1: Translation table from pen numbers in the DLBR database to pen numbers in the PigIT database

DLBR database		PigIT database
Dispenser	"Left" (V) or "Right" (H)	Pen number
2	H	1.10B
2	V	1.10A
8	H	1.6A
8	V	1.6B
16	V	2.10A
16	H	2.10B
23	H	2.5A
23	V	2.5B
29	H	3.9B
29	V	3.9A
37	H	3.5A
37	V	3.5B
58	V	5.10A
58	H	5.10B
63	H	5.7A
63	V	5.7B

"left" and "right" to "A" and "B", however cannot be established automatically. Therefore, the translation table shown as Table 6.1 has been established.

In order to simplify the translation, the separate table `DLBR_PenLink` has been created in the PigIT database (Kristensen, 2014). It translate the uniqueidentifier ID of pens in the DLBR database to pens in the PigIT database according to Table 6.1. The R script matching the pens and entering the IDs into the table is shown in Appendix C.2.

6.2 Farmer observations

The daily farmer observations of diarrhea, tail biting and fouling are stored in the table `PigIt.FarmerObservations` in the PigIT database. The observations as such are in the `PigItVentilData` table in the DLBR database.

An R script transferring the farmer observations from the DLBR database to the PigIT database is shown in Appendix C.3. It involves the following steps:

1. Identification of the pre-defined events, properties and treatments from the relevant tables in the PigIT database.

2. Reading all observations from the `PigItVentilData` table in the DLBR database.

3. Inserting all observations into the `PigIt.FarmerObservations` in the PigIT database.

4. Optionally printing of some test queries.

It should be noticed that all events (including the "Nothing to report") event can be observed simultaneously. The fact that "Nothing to report" can occur simultaneously with the other events must be considered as a BUG in the software used for registration. Nevertheless, the erroneous double registrations are maintained in the PigIT database.

6.3 Pig movements

The pig movements in the DLBD database are stored in the `SMaengde_Af_Dyr` table. In the PigIT database they are stored in the table `PigIt.PigMovements`. An R script transferring the pig movements from the DLBR database to the PigIT database is shown in Appendix C.4. It involves the following steps:

1. Reading inserted, dead and removed pigs. Later there will also be pigs slaughtered.

2. For each of the events "Pigs inserted", "Pigs died" and "Pigs removed":

   (a) Identifying the event in the `PigIt.ObservedEvent` table.

   (b) Inserting the observations into the `PigIt.PigMovements` table.

6.4 Individual weighings

The individual weighings in the DLBR database are stored in `SIndivid_Vaegt`. The identity of individual pigs (linking them to a specific dispenser) are stored in the table `SMaengde_Af_Dyr`. Two tables are created in the PigIT database (see Kristensen, 2014, for details):

- `PigIt.ManualWeighings` and

- `PigIt.IndividualPigs`.

The first-mentioned table contains the weighings as such whereas the latter links pigs to a specific pen. Since, however, only the dispenser is known, all pigs belonging to a given dispenser are linked to the same pen.

An R script transferring the pig weighings from the DLBR database to the PigIT database is shown in Appendix C.5. It involves the following steps:

1. Reading all weighings from the table `SIndivid_Vaegt` in the DLBR database.

2. Extracting all unique pig IDs in the list.
3. Inserting the unique pigs into the `PigIt.IndividualPigs` table:
   (a) Looking up their location (pen) in the `SMaengde_Af_Dyr`
   (b) Translating the DLBR pen ID to PigIT pen ID
   (c) Writing the pig to the `PigIt.IndividualPigs` table.
4. Inserting all weighings into the `PigIt.ManualWeighings` table.

6.5 Feeding and feedstuffs

The rather complicated representation of feedstuffs in the DLBR database has already been described in Section 4.1.6 and Section 5.5 describes how to link the feedstuffs with feeds in the DLBR database.

The migration of feeding information to the PigIT database involves translation of feedstuff information from the table `SFoderEmne` to the format used in the table `PigIt.FeedAnalysis` in the PigIT database. Furthermore, the feed consumption at dispenser level stored in the `SFoderforbrug` is transferred to the table `PigIt.FeedConsumptionPeriod` in the PigIT database. An R script transferring the feed information from the DLBR database to the PigIT database is shown in Appendix C.6. It involves the following steps:

1. Reading the file with property codes (as also listed in Appendix D).
2. Copying the PigIT feedstuffs to the DLBR database.
3. Matching PigIT feedstuffs with DLBR feedstuffs as described in Section 5.5.
4. Calling the function `addAnalysis` for interpretation of nutritional contents (see Appendix C.1).
5. For each feedstuff:
   (a) Inserting it into the `PigIt.FeedAnalysis` table.
   (b) Inserting it into the `DLBR_Feedstuffs` table if it does not exist.
6. Reading feed consumptions at dispenser level from the table `SFoderforbrug`.
7. Calling the function `addAnalysis` for interpretation of nutritional contents (see Appendix C.1).
8. For each record:
   (a) Inserting it into the `PigIt.FeedConsumptionPeriod` table.
   (b) Inserting the feedstuff in question into the `DLBR_Feedstuffs` table if it does not exist. In that case it is also added to the table called `PigIt.FeedAnalysis`.

Bibliography

Kristensen, A. R., 2014. Data model for supplementary registrations in the PigIT project. PigIT Report 2, The PigIT project, Department of Large Animal Sciences, University of Copenhagen.

Appendix A

Example of log book text

20/12-2013 JOH

Alle stier i sektion to er optalte og afstemte. De manglende 7 grise i stierne er indtastet på håndterminalen med dato og vægt - men ikke hvilke stier de kommer fra!!

Peter er forelagt problemet og mener, at vi kan spore aktuel gris til aktuel sti vha. fodercomputeren idet antal i sti ændres heri, hver gang en gris vejes ud af en sti.

Jeg har bedt dem om at taste sti-nr fremover, hver gang de udvejer en gris.

Pga. stop på foderanlæg (stoppet snegl) har jeg ikke kontrolleret vejecellerne herpå.

Besætningens egne notater/logbog vedr. uregelmæssigheder er kopieret og indsat nederst i denne logbog - hvor disse fremover kan findes.

Sikkerhedskopi af databasen er udlæst og findes i mappen "JOH" på skrivebordet på Kappels computer.

29/11-2013 JOH

Enkelt-dyrvægt samt blandetanke kontrolleret og fundet helt ok.

Alle stier i sektion to er optalte og afstemte - dels med Claus' hjælp og ikke mindst Mette Schmidt.

Det er aftalt med Claus, at selvom der kun er øremærker i grise i de fire vejestier, så registreres der afgang/flytninger fra alle stier i sektionen!!

Claus var så ganske tilfreds med "Mini-Gris" angående Svineri, Diarre og Halebid.

Øremærkerne i de fire vejestier fungerer indtil videre - men de af Niels Peder Bådsgård påførte tal med tusch, synes snart at forsvinde - vi håber chippen holder...

Alle kameralinser er rengjorte.

Jeg talte med Kristian Kappel, og han lod ingen tvivl om, at han efterlyser i den grad viden om, hvad videooptagelserne bruges til - om de bruges - og hvad fører det til . . . .

JEP er orienteret herom telefonisk og vil kontakte Erik Jørgensen, Foulum desangående:

Den 27/11 jep Telefonmøde

Fra: Jette Pedersen
Sendt: 26. november 2013 15:37
Til: Jens Ove Hansen; Henriette Steinmetz; Erik Jørgensen (Erik.Jorgensen@agrsci.dk)
Emne: Referat fra telefonmødet i går

Dagsorden:

1. **Status på opsætning af vandmålere og videodata**

   Opsætning af vandmålere er gået fint og opsamling af videodata kører nu. Dog er den centrale server gået ned, så det vil give et kortvarigt hul i videodata.

   Data bliver pt brugt primært til at kigge på vægte.

2. **Fejlbehæftede øremærker**

   Sidst sending af øremærker - dem vi har sat i til hold 4 har været meget fejlbehæftede. Konkret har vi ikke kunne læse øremærkerne ved udvejningen og vi har været nød til at udskrift flere undervejs fra vejestierne. Mærkerne var en blød version som ikke var 100% tætte - en fejl i produktionen, som iflg firmaet skulle være specielt for de som blev leveret først på sommeren, mens de som er blevet i august fungere som de skal. Fra VSP har Niels Peter Baadsgaard taget initiativ til et fælles møde med firmaet, Agrosoft og VSP for at undersøge om det er korrekt.

3. **Nyt hold i gang**

   Niels Ove var onsdag i sidst uge oppe og starte et nyt hold op, fik sat øremærker i vejeholdene (og kun dem) den "gamle type". Det tager ca 20-25 min at sætte øremærker i og veje en sti.

   Mærkning af grisen

   Erik ønskede at mærkerne blev lidt mindre, de er svære at veje med kamera når der er farve ned af siden på grisen. Aftalen er at Erik melder tilbage til Jens Ove om han evt skal til at anvende spritpenne næste gang der skal mærkes op.

4. **Evt**

   Tidsforbrug, Jens Ove samler op på hvor meget ekstra tid der bruges på at deltage i forsøget:

   - Vejning - alle grise ind og ud

- Ugentlige vejninger
- Mærkning af grise
- Restringer af svineri mv.

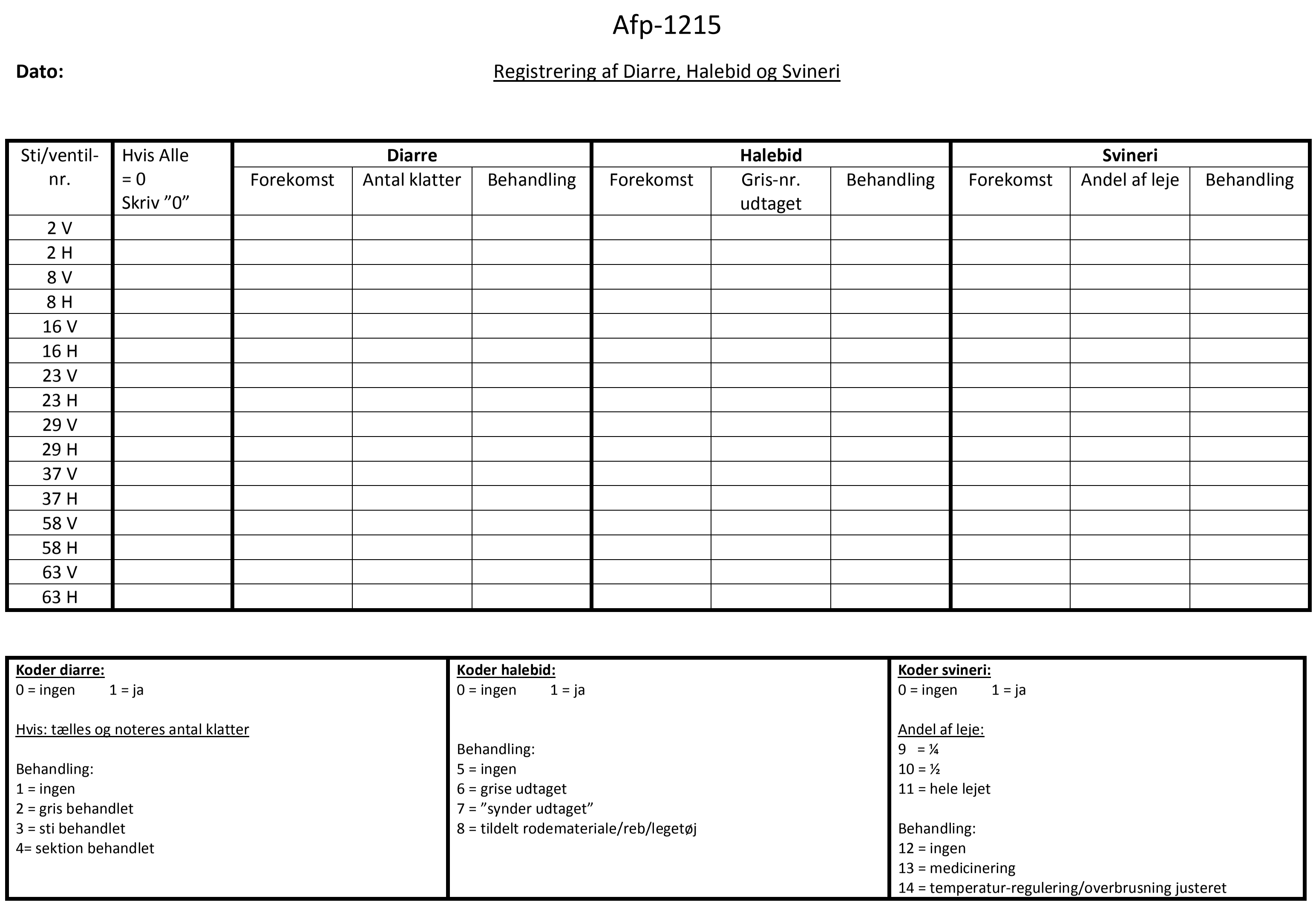
Logbogen er kommet i Dropbox, og den skal sammen med logbog, kopi af stemninger af dyr og foder lægger derover hver 14.dag.

Jens Ove har præciseret overfor Klaus at det er vigtig de skriver hvilke foder de sætter grisene på og hvornår de skrifter faser.

Appendix B

Registration form used as basis for the supplementary registrations

The paper form originally developed for the supplementary registrations at the Kappel farm is shown as Figure B.1.



Afp-1215

Dato:

Registrering af Diarre, Halebid og Svineri

Sti/ventil-nr.	Hvis Alle = 0 Skriv "0"	**Diarre**			**Halebid**			**Svineri**		
		Forekomst	Antal klatter	Behandling	Forekomst	Gris-nr. udtaget	Behandling	Forekomst	Andel af leje	Behandling
2 V										
2 H										
8 V										
8 H										
16 V										
16 H										
23 V										
23 H										
29 V										
29 H										
37 V										
37 H										
58 V										
58 H										
63 V										
63 H										

Koder diarre:	Koder halebid:	Koder svineri:
0 = ingen 1 = ja	0 = ingen 1 = ja	0 = ingen 1 = ja
Hvis: tælles og noteres antal klatter		Andel af leje: 9 = ¼ 10 = ½ 11 = hele lejet
Behandling: 1 = ingen 2 = gris behandlet 3 = sti behandlet 4= sektion behandlet	Behandling: 5 = ingen 6 = grise udtaget 7 = "synder udtaget" 8 = tildelt rodemateriale/reb/legetøj	Behandling: 12 = ingen 13 = medicinering 14 = temperatur-regulering/overbrusning justeret

Figure B.1: The form designed for manual registrations at the Kappel farm.

Appendix C

Source code

C.1 Functions defined in R

```r
######################################################################
# Function calculating the actual stock of pigs at a given dispenser #
# from date of insertion and a specified number of days forward.     #
######################################################################
# pigsInserted:  A data frame with all inserted pigs in the herd     #
# pigsDead:      A data frame with all dead pigs in the herd         #
# pigsMoved:     A data frame with all removed pigs in the herd      #
# insertionDate: The insertion date (as Date format)                 #
# dispenser:     The dispenser number                                #
# days:          The number of days                                  #
######################################################################
getStock = function(pigsInserted, pigsDead, pigsMoved,
                    insertionDate, dispenser, days) {
  ins = pigsInserted[pigsInserted$VentilNr == dispenser,]
  ins$EventDate = as.Date(paste(ins$OmsaetnDato, "", sep=""), format = "%Y%m%d")
  ins = ins[ins$EventDate == insertionDate,]
  deads = pigsDead[pigsDead$VentilNr == dispenser,]
  if (length(deads$OmsaetnDato) > 0) {
    deads$EventDate = as.Date(paste(deads$OmsaetnDato, "", sep=""), format = "%Y%m%d")
  }
  rem = pigsMoved[pigsMoved$VentilNr == dispenser,]
  if (length(rem$OmsaetnDato) > 0) {
    rem$EventDate = as.Date(paste(rem$OmsaetnDato, "", sep=""), format = "%Y%m%d")
  }
  minDay = min(ins$EventDate)
  sto = data.frame(Day = (minDay-1), Stock = 0, Dead = 0, Removed = 0)
  row = 2
  for (i in minDay:(minDay+days)) {
    insDay = ins[ins$EventDate == i,]
    if (length(insDay$EventDate) > 0) {
      inserted = sum(insDay$OmsaetnAntal)
      sto[row-1, 2] = 0
      sto[row-1, 3] = 0
      sto[row-1, 4] = 0
    }
    else inserted = 0
    dieDay = deads[deads$EventDate == i,]
    if (length(dieDay$EventDate) > 0) dead = sum(dieDay$OmsaetnAntal)
    else dead = 0
    movDay = rem[rem$EventDate == i,]
    if (length(movDay$EventDate) > 0) moved = sum(movDay$OmsaetnAntal)
    else moved = 0
    sto[row, 1] = as.Date(i, origin = as.Date("1970/1/1"))
    sto[row, 2] = sto[row-1, 2] + inserted - dead - moved
    sto[row, 3] = sto[row-1, 3] + dead
    sto[row, 4] = sto[row-1, 4] + moved
    row = row + 1
  }
  return(sto)
}

######################################################################
# Function finding all insertion days for a pen (or a dispenser)     #
# The result is rerturned as a vector of Dates                       #
######################################################################
# pigsInserted:  A data frame with all inserted pigs in the herd     #
# dispenser:     The dispenser number                                #
# pen:           A vector specifying either "H" for right pen, "V"   #
#                for left pen or both. The allowed values are c("H"),#
#                c("V") or c("H", "V").                              #
######################################################################
getInsertionDates = function(pigsInserted, dispenser, pen) {
  ins = pigsInserted[pigsInserted$VentilNr == dispenser &
                     substr(pigsInserted$ProcesUdstyr, 1, 1) %in% pen,]
  ins$EventDate = as.Date(paste(ins$OmsaetnDato, "", sep=""), format = "%Y%m%d")
  return(unique(ins$EventDate))
}


######################################################################
# Function plotting individual growth curves of individual pigs.     #
# A plot is created for each combination of batch and dispenser.     #
# The function may optionally add vertical lines for observation of  #
# diarrhea, tail biting and fouling                                  #
######################################################################
# weights: A data frame with all individual weighigs                 #
# diarrhea: An optiional data frame with diarrhea observations       #
# tail: An optional data frame with tail biting observations         #
# foul: An optional data frame with fouling observations             #
######################################################################
plotWeigtByDispenseAndPig = function(weights, diarr = NA,
                                     tail = NA, foul = NA) {
  batches = unique(weights$OmsaetnDato)
  disps = unique(weights$VentilNr)
  for (b in batches) {
    thisBatch = weights[weights$OmsaetnDato == b, ]
    for (d in disps) {
      batchPen = thisBatch[thisBatch$VentilNr == d, ]
      if (length(batchPen$Pig > 0)) {
        pigs = unique(batchPen$Pig)
        for (p in 1:length(pigs)) {
          thisPig = batchPen[batchPen$Pig == pigs[p], ]
          if (p == 1) {
            x1 = as.Date(thisPig$Tid[1])
            x2 = x1 + 90
#            xl = c(thisPig$Tid[1], thisPig$Tid[1] + 90*24*60*60)
            plot(as.Date(thisPig$Tid), thisPig$Vaegt, type = "l",
                 main = paste("Dispenser", d, "- Inserted:", thisPig$OmsaetnDato[1]),
                 ylim = c(20, 110), xlim = c(x1, x2), xlab = "Date", ylab = "Live weight")
            axis(side = 1, at = seq(x1, x2, by = 1), labels = FALSE, tcl = -0.2)

          }
          else {
            lines(as.Date(thisPig$Tid), thisPig$Vaegt)
          }
        }
      }
      if (is.data.frame(diarr)) addObservationsToPlot(diarr, d, c("V", "H"), 1, "red")
      if (is.data.frame(tail)) addObservationsToPlot(tail, d, c("V", "H"), 2)
      if (is.data.frame(foul)) addObservationsToPlot(foul, d, c("V", "H"), 3)
    }
  }
}


########################################################################
# Function adding vertical lines to an existing plot. A vertical      #
# A vertical line corresponds to a pen specific observation on a      #
# given day. Typically, the observation is either diarrhea, tail      #
# biting or fouling. The data of the x-axis of the plot must be of    #
# type Date.                                                           #
########################################################################
# observations:  A data frame with daily penwise observations         #
# dispenser:     The dispenser number (integer)                        #
# pen:           A vector specifying either "H" for right pen, "V"    #
#                for left pen or both. The allowed values are c("H"),#
#                c("V") or c("H", "V").                                #
# lineType:      The value of the lty parameter                        #
# lineColor:     The value of the col parameter                        #
########################################################################
addObservationsToPlot = function(observations, dispenser, pen,
                                 lineType, lineColor = "black") {
  obsPen = observations[(observations$VentilNr == dispenser &
                       substr(observations$ProcesUdstyr, 1, 1) %in% pen),]
  if (length(obsPen$Timestamp) > 0) {
    length(obsPen$Timestamp)
    for (i in 1:length(obsPen$Timestamp)) {
      date = as.Date(obsPen$Timestamp)
      abline(v = date, lty = lineType, col = lineColor)
    }
  }
}


########################################################################
# Function parsing the string EgenskabIndhold in feedstuff records.   #
# The function splits the string up into substring, parses each       #
# substring and creates variables corresponding to the nutritional    #
# components requested. By default the very long string               #
# EngenskabIndhold is removed from the data frame. The indexes in     #
# the include vector MUST be in ascending order!                      #
########################################################################
# feeds: A data frame with feedstuffs including EgenskabIndhold       #
# kodes: A data frame with names and codes for the properties         #
# include: An integer vector defining the nutr. codes to include      #
# remove: TRUE if the variable EgenskabIndhold should be deleted      #
########################################################################
addAnalysis = function(feeds, kodes, include, remove = TRUE) {
  usedKodes = kodes[kodes$Nr. %in% include,]
  newCols = c()
  for (i in 1:length(usedKodes[,2])) {
    newCols[i] = toString(usedKodes[i ,2])
  }
  nc = length(feeds[1,])
  feeds[newCols] = 1.0 - 1.0
  for (i in 1:length(feeds[,1])) {
    str = toString(feeds$EgenskabIndhold[i])
    posts = strsplit(str, "&")
    c = 0
    for (j in 1:length(posts[[1]])) {
      entries = strsplit(posts[[1]][j], ";")
      thisCode = strtoi(entries[[1]][1])
      if (thisCode %in% include) {
        c = which(include == thisCode)
        cont = entries[[1]][3]
        cont = gsub(",",".",cont)
        feeds[i, nc+c] = as.double(cont)
      }
    }
  }
  if (remove) {
    return(feeds[, -which(colnames(feeds)=="EgenskabIndhold")])
  }
  else {
    return(feeds)
  }
}

#######################################################################
# Function looking up the ID of an employee at a specified farm.      #
# If the employee doesn't exist, he/she is added to the Observer      #
# table.                                                              #
#######################################################################
# farm: A uniqueidentifier (as a string) specifying the farm          #
# initials: The initials used in the DLBR database for the employee   #
#######################################################################
getObserver = function(farm, initials) {
  str = paste("SELECT Id FROM PigIt.Observer WHERE FarmId = '",
              farm,"' AND Initials = '", initials,"'", sep ="")
  obs = sqlQuery(pigIt, str)
  observer = toString(obs[[1]])
  # If observer doesn't exist add him/her to table
  if (observer == "") {
    str = paste("INSERT INTO PigIt.Observer (Id, FarmId, Initials)
                VALUES (NEWID(), '", farm, "', '", initials,"')",
                sep ="" )
    ins = sqlQuery(pigIt, str)
  }
  else {
    # The observer already existed - return ID
    return(observer)
  }

  # Now the observer has been created - return ID
  str = paste("SELECT Id FROM PigIt.Observer WHERE FarmId = '",
              farm,"' AND Initials = '", initials,"'", sep ="")
  obs = sqlQuery(pigIt, str)
  observer = toString(obs[[1]])
  return(observer)
}
```

C.2 R script for matching pens

```
####################################################################################
# Basic setup for the Kappel farm
####################################################################################
```

```
# Read pens (dispensers) from the DLBR database
dispensers = sqlQuery(kappel, "SELECT SVentil.ID, VentilNr,
                      Procesudstyr FROM SVentil, PigItVentil
                      WHERE PigItVentil.VentilId = SVentil.ID")
dispensers

# Read pens from the PigIt database
pens =  sqlQuery(pigIt, "SELECT * FROM PigIt.PigPen")
pens

# Establish a map from DLBR pen IDs to PigIt pen IDs
for (i in 1:length(dispensers[,1])) {
  if (dispensers$VentilNr[i] == 2) {
    if (substr(dispensers$Procesudstyr[i], 1, 1) %in% c('H')) {
      dispensers$ppn[i] = '1.10B'
    }
    else {
      dispensers$ppn[i] = '1.10A'
    }
  }
  else {
    if (dispensers$VentilNr[i] == 8) {
      if (substr(dispensers$Procesudstyr[i], 1, 1) %in% c('H')) {
        dispensers$ppn[i] = '1.6A'
      }
      else {
        dispensers$ppn[i] = '1.6B'
      }
    }
    else {
      if (dispensers$VentilNr[i] == 16) {
        if (substr(dispensers$Procesudstyr[i], 1, 1) %in% c('H')) {
          dispensers$ppn[i] = '2.10B'
        }
        else {
          dispensers$ppn[i] = '2.10A'
        }
      }
      else {
        if (dispensers$VentilNr[i] == 23) {
          if (substr(dispensers$Procesudstyr[i], 1, 1) %in% c('H')) {
            dispensers$ppn[i] = '2.5A'
          }
          else {
            dispensers$ppn[i] = '2.5B'
          }
        }
        else {
          if (dispensers$VentilNr[i] == 29) {
            if (substr(dispensers$Procesudstyr[i], 1, 1) %in% c('H')) {
              dispensers$ppn[i] = '3.9B'
            }
            else {
              dispensers$ppn[i] = '3.9A'
            }
          }
          else {
            if (dispensers$VentilNr[i] == 37) {
              if (substr(dispensers$Procesudstyr[i], 1, 1) %in% c('H')) {
                dispensers$ppn[i] = '3.5A'
              }
              else {
                dispensers$ppn[i] = '3.5B'
              }
            }
            else {
              if (dispensers$VentilNr[i] == 58) {
                if (substr(dispensers$Procesudstyr[i], 1, 1) %in% c('H')) {
                  dispensers$ppn[i] = '5.10B'
                }
                else {
                  dispensers$ppn[i] = '5.10A'
                }
              }
              else {
                if (dispensers$VentilNr[i] == 63) {
                  if (substr(dispensers$Procesudstyr[i], 1, 1) %in% c('H')) {
                    dispensers$ppn[i] = '5.7A'
                  }
                  else {
                    dispensers$ppn[i] = '5.7B'
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}

# Convert ID to string
for (i in 1:length(dispensers[,1])) {
  pn = dispensers$ppn[i]
  pen = pens[pens$PigPenName == pn,]
  dispensers$PigIT_PenId[i] = toString(pen$Id[1])
}
dispensers

# Enter the map into the table
for (i in 1:length(dispensers[,1])) {
  str = paste("INSERT INTO DLBR_PenLink (DLBR_PenId, PigIT_PenId) VALUES ('",
              dispensers$ID[i], "','", dispensers$PigIT_PenId[i], "')", sep = "")
  ins = sqlQuery(pigIt, str)
  print(ins)
}
```

C.3 Migrating farmer observations

```
##########################################################################################
# Identify possible events, properties and treatments
##########################################################################################

# The "No event"
noEv = sqlQuery(pigIt, "SELECT Id FROM PigIt.ObservedEvents
                WHERE EventName = 'Nothing to report'")
noEvent = toString(noEv[[1]])
str = paste("SELECT Id FROM PigIt.ObservedEventProperties
            WHERE PropertyName = 'Nothing to observe' ",
            "AND EventId ='", toString(noEvent[[1]]), "'", sep ="" )
noProp = sqlQuery(pigIt, str)
```

```
noProperty = toString(noProp[[1]])
str = paste("SELECT Id FROM PigIt.ObservedEventTreatment
            WHERE TreatmentName = 'No treatment' ",
            "AND EventId ='", toString(noEvent[[1]]), "'", sep ="" )
noTreat = sqlQuery(pigIt, str)
noTreatment = toString(noTreat[[1]])

# The diarrhea event
diarEv = sqlQuery(pigIt, "SELECT Id FROM PigIt.ObservedEvents
                  WHERE EventName = 'Diarrhea'")
diarEvent = toString(diarEv[[1]])
str = paste("SELECT Id FROM PigIt.ObservedEventProperties
            WHERE PropertyName = 'Number of spots' ",
            "AND EventId ='", diarEvent, "'", sep ="" )
spotProp = sqlQuery(pigIt, str)
spotProperty = toString(spotProp[[1]])
str = paste("SELECT Id FROM PigIt.ObservedEventTreatment
            WHERE TreatmentName = 'No treatment' ",
            "AND EventId ='", diarEvent, "'", sep ="" )
noDTreat = sqlQuery(pigIt, str)
noDTreatment = toString(noDTreat[[1]])
str = paste("SELECT Id FROM PigIt.ObservedEventTreatment
            WHERE TreatmentName = 'Pig treated' ",
            "AND EventId ='", diarEvent, "'", sep ="" )
pigDTreat = sqlQuery(pigIt, str)
pigDTreatment = toString(pigDTreat[[1]])
str = paste("SELECT Id FROM PigIt.ObservedEventTreatment
            WHERE TreatmentName = 'Pen treated' ",
            "AND EventId ='", diarEvent, "'", sep ="" )
penDTreat = sqlQuery(pigIt, str)
penDTreatment = toString(penDTreat[[1]])
str = paste("SELECT Id FROM PigIt.ObservedEventTreatment
            WHERE TreatmentName = 'Section treated' ",
            "AND EventId ='", diarEvent, "'", sep ="" )
sectionDTreat = sqlQuery(pigIt, str)
sectionDTreatment = toString(sectionDTreat[[1]])
# Crerate a vector with diarrhea treatments
diarTreat = c(noDTreatment, pigDTreatment, penDTreatment, sectionDTreatment)
diarTreat

# The tail bite event
tailEv = sqlQuery(pigIt, "SELECT Id FROM PigIt.ObservedEvents
                  WHERE EventName = 'Tailbite'")
tailEvent = toString(tailEv[[1]])
str = paste("SELECT Id FROM PigIt.ObservedEventProperties
            WHERE PropertyName = 'Nothing to observe' ",
            "AND EventId ='", tailEvent, "'", sep ="" )
noTProp = sqlQuery(pigIt, str)
noTProperty = toString(noTProp[[1]])
str = paste("SELECT Id FROM PigIt.ObservedEventTreatment
            WHERE TreatmentName = 'No treatment' ",
            "AND EventId ='", tailEvent, "'", sep ="" )
noTTreat = sqlQuery(pigIt, str)
noTTreatment = toString(noTTreat[[1]])
str = paste("SELECT Id FROM PigIt.ObservedEventTreatment
            WHERE TreatmentName = 'Pigs removed' ",
            "AND EventId ='", tailEvent, "'", sep ="" )
remTTreat = sqlQuery(pigIt, str)
remTTreatment = toString(remTTreat[[1]])
str = paste("SELECT Id FROM PigIt.ObservedEventTreatment
            WHERE TreatmentName = 'Sinner removed' ",
            "AND EventId ='", tailEvent, "'", sep ="" )
```

```
sinTTreat = sqlQuery(pigIt, str)
sinTTreatment = toString(sinTTreat[[1]])
str = paste("SELECT Id FROM PigIt.ObservedEventTreatment
            WHERE TreatmentName = 'Rooting material provided' ",
            "AND EventId ='", tailEvent, "'", sep ="" )
rootTTreat = sqlQuery(pigIt, str)
rootTTreatment = toString(rootTTreat[[1]])
# Create a vector with tail bite treatments
tailTreat = c(rep("Not relevant", 4),
              noTTreatment, remTTreatment, sinTTreatment, rootTTreatment)

# The fouling event
foulEv = sqlQuery(pigIt, "SELECT Id FROM PigIt.ObservedEvents
                  WHERE EventName = 'Fouling'")
foulEvent = toString(foulEv[[1]])
str = paste("SELECT Id FROM PigIt.ObservedEventProperties
            WHERE PropertyName = 'Percent of lying area' ",
            "AND EventId ='", foulEvent, "'", sep ="" )
foulProp = sqlQuery(pigIt, str)
foulProperty = toString(foulProp[[1]])

str = paste("SELECT Id FROM PigIt.ObservedEventTreatment
            WHERE TreatmentName = 'No treatment' ",
            "AND EventId ='", foulEvent, "'", sep ="" )
noFTreat = sqlQuery(pigIt, str)
noFTreatment = toString(noFTreat[[1]])
str = paste("SELECT Id FROM PigIt.ObservedEventTreatment
            WHERE TreatmentName = 'Medication' ",
            "AND EventId ='", foulEvent, "'", sep ="" )
medFTreat = sqlQuery(pigIt, str)
medFTreatment = toString(medFTreat[[1]])
str = paste("SELECT Id FROM PigIt.ObservedEventTreatment
            WHERE TreatmentName = 'Temperature or sprinkler adjusted' ",
            "AND EventId ='", foulEvent, "'", sep ="" )
tempFTreat = sqlQuery(pigIt, str)
tempFTreatment = toString(tempFTreat[[1]])
# Create a vector with fouling treatments
foulTreat = c(rep("Not relevant", 11),
              noFTreatment, medFTreatment, tempFTreatment)
# The pre-defined percentages of dirty lying area
foulProp = c(rep(-1, 8), 25, 50, 100)
foulProp

# Read all farmer observations from the DLBR database
observations = sqlQuery(kappel, "SELECT * FROM PigItVentilData")
observations[1:10,]

# The Kappel farm
FarmID = "15979332-9D92-4F37-B33C-653F30F91C4D"

# Copy observations and insert them into PigIt
for (i in 1:length(observations[,1])) {
  str = paste("SELECT PigIT_PenId FROM DLBR_PenLink
              WHERE DLBR_PenId = '", toString(observations$VentilId[i]),"'",
              sep ="")
  pigItPen = sqlQuery(pigIt, str)
  pen = toString(pigItPen[[1]])
  # Identify the observer
  observer = getObserver(FarmID, observations$Ajourini[i])
  # Insert events where nothing is observed (sometimes conflicts)
  if (observations$Alle[i] == 0) {
    str = paste("INSERT INTO PigIt.FarmerObservations
                (Id, DatabaseId, PenID, ObservationDate, ObserverId, ",
                "EventId, EventPropertyValue, EventTreatmentId) ",
                "VALUES (NEWID(), 1, '", pen, "', '",
                observations$Timestamp[i], "', '", observer, "', '",
                noEvent, "', NULL, '", noTreatment, "')", sep="")
    ins = sqlQuery(pigIt, str)
  }
  # Insert diarrhea events
  if (observations$Diarre[i] == 1) {
    str = paste("INSERT INTO PigIt.FarmerObservations
                (Id, DatabaseId, PenID, ObservationDate, ObserverId, ",
                "EventId, EventPropertyValue, EventTreatmentId) ",
                "VALUES (NEWID(), 1, '", pen, "', '",
                observations$Timestamp[i], "', '", observer, "', '",
                diarEvent, "', ", observations$DKlatter[i], ", '",
                diarTreat[observations$DBehandling[i]], "')", sep="")
    ins = sqlQuery(pigIt, str)
  }
  # Insert tailbite events
  if (observations$Halebid[i] == 1) {
    str = paste("INSERT INTO PigIt.FarmerObservations
                (Id, DatabaseId, PenID, ObservationDate, ObserverId, ",
                "EventId, EventPropertyValue, EventTreatmentId) ",
                "VALUES (NEWID(), 1, '", pen, "', '",
                observations$Timestamp[i], "', '", observer, "', '",
                tailEvent, "', NULL, '",
                tailTreat[observations$HBehandling[i]], "')", sep="")
    ins = sqlQuery(pigIt, str)
  }
  # Insert fouling events
  if (observations$Svineri[i] == 1) {
    str = paste("INSERT INTO PigIt.FarmerObservations
                (Id, DatabaseId, PenID, ObservationDate, ObserverId, ",
                "EventId, EventPropertyValue, EventTreatmentId) ",
                "VALUES (NEWID(), 1, '", pen, "', '",
                observations$Timestamp[i], "', '", observer, "', '",
                foulEvent, "', ", foulProp[observations$SAndel[i]], ", '",
                foulTreat[observations$SBehandling[i]], "')", sep="")
    ins = sqlQuery(pigIt, str)
  }
}

# Some test quesries - can be skipped
diar = sqlQuery(pigIt, "SELECT PigPenName, ObservationDate,
                EventPropertyValue, TreatmentName
                FROM PigIt.PigPen, PigIt.FarmerObservations,
                PigIt.ObservedEventTreatment, PigIt.ObservedEvents
                WHERE PigIt.PigPen.Id = PenId AND
                EventTreatmentId = PigIt.ObservedEventTreatment.Id
                AND PigIt.FarmerObservations.EventId = PigIt.ObservedEvents.Id AND
                PigIt.ObservedEvents.EventName = 'Diarrhea'" )
diar

tail = sqlQuery(pigIt, "SELECT PigPenName, ObservationDate,
                EventPropertyValue, TreatmentName FROM
                PigIt.PigPen, PigIt.FarmerObservations,
                PigIt.ObservedEventTreatment, PigIt.ObservedEvents
                WHERE PigIt.PigPen.Id = PenId AND
                EventTreatmentId = PigIt.ObservedEventTreatment.Id
                AND PigIt.FarmerObservations.EventId = PigIt.ObservedEvents.Id AND
                PigIt.ObservedEvents.EventName = 'Tailbite'" )
tail
```

```
foul = sqlQuery(pigIt, "SELECT PigPenName, ObservationDate,
                EventPropertyValue, TreatmentName FROM
                PigIt.PigPen, PigIt.FarmerObservations,
                PigIt.ObservedEventTreatment, PigIt.ObservedEvents
                WHERE PigIt.PigPen.Id = PenId AND
                EventTreatmentId = PigIt.ObservedEventTreatment.Id AND
                PigIt.FarmerObservations.EventId = PigIt.ObservedEvents.Id AND
                PigIt.ObservedEvents.EventName = 'Fouling'" )
foul

# List conflicts where "Everything OK" has been entered simultaneously with
# a diarrhea, tail bite or fouling event.
observations[observations$Alle == 0 & observations$Diarre == 1,]
observations[observations$Alle == 0 & observations$Svineri == 1,]
observations[observations$Alle == 0 & observations$Halebid == 1,]
```

C.4 Migrating pig movements

```
# Read list of inserted pigs from DLBR
pigsInserted = sqlQuery(kappel,
                        "SELECT OmsaetnAntal, VentilNr, VentilId,
                        ProcesUdstyr, OmsaetnVgtMgde, OmsaetnDato,
                        Ajourtid, Ajourini
                        FROM SMaengde_Af_Dyr, SVentil, PigItVentil
                        WHERE PigItVentil.VentilId = SVentil.ID
                        AND VentilTilID = SVentil.ID ORDER BY OmsaetnDato")
pigsInserted[1:5,]

# Read list of dead pigs from DLBR
pigsDead = sqlQuery(kappel,
                    "SELECT OmsaetnAntal, VentilNr, VentilId,
                    ProcesUdstyr, OmsaetnVgtMgde, OmsaetnDato,
                    Ajourtid, Ajourini
                    FROM SMaengde_Af_Dyr, SVentil, PigItVentil
                    WHERE PigItVentil.VentilId = SVentil.ID
                    AND VentilFraID = SVentil.ID
                    AND OmsaetnAfgKode = 3 ORDER BY OmsaetnDato")
pigsDead

# !!!! Slaughter event not available yet !!!!


# Read list of removed pigs from DLBR
pigsMoved = sqlQuery(kappel,
                     "SELECT OmsaetnAntal, VentilNr, VentilId,
                     ProcesUdstyr, OmsaetnVgtMgde, OmsaetnDato,
                     Ajourtid, Ajourini
                     FROM SMaengde_Af_Dyr, SVentil, PigItVentil
                     WHERE PigItVentil.VentilId = SVentil.ID
                     AND VentilFraID = SVentil.ID
                     AND OmsaetnAfgKode = 6 ORDER BY OmsaetnDato")
pigsMoved

# Identify event "Pigs inserted"
insEv = sqlQuery(pigIt, "SELECT Id FROM PigIt.ObservedEvents
                 WHERE EventName = 'Pigs inserted'")
insEvent = toString(insEv[[1]])
insEv

# Copy insertions to PigIt database
```

```
for (i in 1:length(pigsInserted[,1])) {
  # Get PigIt pen ID
  str = paste("SELECT PigIT_PenId FROM DLBR_PenLink WHERE DLBR_PenId = '",
              toString(pigsInserted$VentilId[i]),"'", sep ="")
  pigItPen = sqlQuery(pigIt, str)
  pen = toString(pigItPen[[1]])
  # Get observer ID
  observer = getObserver(FarmID, pigsInserted$Ajourini[i])
  # Get event date
  EventDate = as.Date(paste(pigsInserted$OmsaetnDato[i], "", sep=""),
                      format = "%Y%m%d")
  EventDate
  # Insert insertion
  str = paste("INSERT INTO PigIt.PigMovements (Id, DatabaseId, PenID,
              EventDate, ObserverId, ", "EventId, NumberOfPigs,
              TotalPigWeight, UpdateTime) ", "VALUES (NEWID(), 1, '",
              pen, "', '", EventDate, "', '", observer, "', '",
              insEvent, "', ", pigsInserted$OmsaetnAntal[i], ", ",
              pigsInserted$OmsaetnVgtMgde[i], ", '",
              pigsInserted$Ajourtid[i], "')", sep="")
  ins = sqlQuery(pigIt, str)
}

# Identify dead event
deadEv = sqlQuery(pigIt, "SELECT Id FROM PigIt.ObservedEvents
                  WHERE EventName = 'Pigs died'")
deadEvent = toString(deadEv[[1]])
deadEvent

# Copy dead pigs to PigIt database
for (i in 1:length(pigsDead[,1])) {
  # Get PigIt pen ID
  str = paste("SELECT PigIT_PenId FROM DLBR_PenLink
              WHERE DLBR_PenId = '", toString(pigsDead$VentilId[i]),"'", sep ="")
  pigItPen = sqlQuery(pigIt, str)
  pen = toString(pigItPen[[1]])
  # Get observer ID
  observer = getObserver(FarmID, pigsDead$Ajourini[i])
  # Get event date
  EventDate = as.Date(paste(pigsDead$OmsaetnDato[i], "", sep=""),
                      format = "%Y%m%d")
  EventDate
  # Insert deaths
  str = paste("INSERT INTO PigIt.PigMovements (Id, DatabaseId, PenID,
              EventDate, ObserverId, EventId, NumberOfPigs,
              TotalPigWeight, UpdateTime) VALUES (NEWID(), 1, '",
              pen, "', '", EventDate, "', '", observer, "', '",
              deadEvent, "', ", pigsDead$OmsaetnAntal[i], ", ",
              pigsDead$OmsaetnVgtMgde[i], ", '", pigsDead$Ajourtid[i], "')",
              sep="")
  ins = sqlQuery(pigIt, str)
}

# Identify remove event
remEv = sqlQuery(pigIt, "SELECT Id FROM PigIt.ObservedEvents
                 WHERE EventName = 'Pigs removed'")
remEvent = toString(remEv[[1]])
remEvent

for (i in 1:length(pigsMoved[,1])) {
  # Get PigIt pen ID
  str = paste("SELECT PigIT_PenId FROM DLBR_PenLink
              WHERE DLBR_PenId = ’", toString(pigsMoved$VentilId[i]),"’",
              sep ="")
  pigItPen = sqlQuery(pigIt, str)
  pen = toString(pigItPen[[1]])
  # Get observer ID
  observer = getObserver(FarmID, pigsMoved$Ajourini[i])
  # Get event date
  EventDate = as.Date(paste(pigsMoved$OmsaetnDato[i], "", sep=""),
                      format = "%Y%m%d")
  EventDate
  # Insert removals
  str = paste("INSERT INTO PigIt.PigMovements (Id, DatabaseId, PenID,
              EventDate, ObserverId, EventId, NumberOfPigs,
              TotalPigWeight, UpdateTime) VALUES (NEWID(), 1, ’",
              pen, "’, ’", EventDate, "’, ’", observer, "’, ’",
              remEvent, "’, ", pigsMoved$OmsaetnAntal[i], ", ",
              pigsMoved$OmsaetnVgtMgde[i], ", ’", pigsMoved$Ajourtid[i], "’)",
              sep="")
  ins = sqlQuery(pigIt, str)
}
```

C.5 Migrating manual weighings

```
# Read list of individual weighings in the DLBR database
individualWeighings = sqlQuery(kappel,
                               "SELECT GrisID, Vaegt, Tid, OmsaetnAntal,
                               VentilNr, VentilId, ProcesUdstyr,
                               OmsaetnVgtMgde, OmsaetnDato
                               FROM SMaengde_Af_Dyr, SIndivid_Vaegt,
                               SVentil, PigItVentil
                               WHERE GrisID = InternFlytningUngdyrID
                               AND PigItVentil.VentilId = SVentil.ID
                               AND VentilTilID = SVentil.ID
                               ORDER BY OmsaetnDato, VentilNr, Tid, GrisID")

# Find the pig IDs
pigs = unique(individualWeighings$GrisID)
pigs

# Put all pigs in the pig table
for (i in 1:length(pigs)) {
  # Find pen/dispenser and insertion date
  str = paste("SELECT VentilTilID, OmsaetnDato FROM SMaengde_Af_Dyr,
              SVentil, PigItVentil WHERE PigItVentil.VentilId = SVentil.ID
              AND VentilTilID = SVentil.ID AND InternFlytningUngdyrID = ’",
              toString(pigs[i]),"’", sep ="")
  penDate = sqlQuery(kappel, str)
  # Find pen in PigIt database
  str = paste("SELECT PigIT_PenId FROM DLBR_PenLink WHERE DLBR_PenId = ’",
              toString(penDate$VentilTilID[1]),"’", sep ="")
  pigItPen = sqlQuery(pigIt, str)
  pen = toString(pigItPen[[1]])
  EventDate = as.Date(paste(penDate$OmsaetnDato[1], "", sep=""),
                      format = "%Y%m%d")
  str = paste("INSERT INTO PigIt.IndividualPigs (Id, DatabaseId, PenID,
              InsertionDate) VALUES (’", pigs[i], "’, ", 1, ", ’",
              pen, "’, ’", EventDate, "’)", sep="")
  ins = sqlQuery(pigIt, str)
}
```

```
# Copy all weighings to PigIt
for (i in 1:length(individualWeighings[,1])) {
  str = paste("INSERT INTO PigIt.ManualWeighings
              (Id, PigId, PigWeight, ObservationTime) ",
              "VALUES (NEWID(),'", individualWeighings$GrisID[i],
              "', ", individualWeighings$Vaegt[i],", '",
              individualWeighings$Tid[i], "')", sep="")
  ins = sqlQuery(pigIt, str)
  print(ins)
}
```

C.6 Migrating and matching feedstuffs

```
# Read codes from file

codes = read.table("P:/PigIT/DBEgenskaber.csv", sep = ";", header = TRUE)
codes

# Read the Feed table in the PigIT database
allFeeds = sqlQuery(pigIt, "SELECT * FROM PigIt.Feed WHERE DataBaseId = 1")
allFeeds

# Write them to the Kappel database
# Create the feed table in the DLBR database
tabMap = sqlQuery(kappel, "CREATE TABLE Feed
                 (Id uniqueidentifier NOT NULL, CreateTime datetime NOT NULL,
                  UpdateTime datetime NOT NULL, DataBaseId int NOT NULL,
                  FeedName nvarchar(max) NOT NULL, FeedComponentNumber nvarchar(max) NOT NULL,
                  FeedEnergy decimal(18, 2) NOT NULL,
                  PRIMARY KEY(Id))")
tabMap

# Copy all feeds from PigIT to DLBR database
for (i in 1:length(allFeeds[,1])) {
  str = paste("INSERT INTO Feed
               (Id, CreateTime, UpdateTime, DataBaseId, FeedName, FeedComponentNumber, FeedEnergy) ",
              "VALUES ('", toString(allFeeds$Id[i]), "', '", toString(allFeeds$CreateTime[i]),
              "', '", toString(allFeeds$UpdateTime[i]),
              "', ", allFeeds$DataBaseId[i], ", '", toString(allFeeds$FeedName[i]),
              "', '", toString(allFeeds$FeedComponentNumber[i]),
              "', ", allFeeds$FeedEnergy,")", sep="")
  ins = sqlQuery(kappel, str)
  print(ins)
}

# Match PigIt feedstuffs with DLBR feedstuffs as described in separate note
pigItFeedsRaw =
  sqlQuery(kappel,
           "SELECT SFoderEmne.ID, Feed.Id, Feed.CreateTime,
           FeedName, FeedComponentNumber, Navn, Kodeparti, KodeFF, PartiFF, FoderKodeParti,
           NavnFF, Markering, FoderNavnLang, Ajourtid, Ajourini, EgenskabIndhold
           FROM Feed, SFoderemneKonvert, SFoderEmne
           WHERE FeedComponentNumber = KodeFF AND Feed.DataBaseId = 1
           AND Kodeparti = FoderKodeParti AND
           Ajourtid IN
           (SELECT max(Ajourtid) FROM SFoderEmne WHERE Kodeparti = FoderkodeParti
           AND Ajourtid < CreateTime)")
pigItFeedsRaw

# Interpret string EgenskabIndhold as described in separate note
pigItFeeds = addAnalysis(pigItFeedsRaw, codes,
                             c(1, 9, 101, 103, 107, 109, 111, 113,
                               115, 117, 119, 121, 123, 165), FALSE)
pigItFeeds

# Remove blanks ...
colnames(pigItFeeds) = make.names(colnames(pigItFeeds))
colnames(pigItFeeds)


# Insert feed properties in analysis table and enter matching DLBR feedstuff in table
for (i in 1:length(pigItFeeds[,1])) {

  # Create a new uniueidentifier
  a = sqlQuery(pigIt, "SELECT NEWID()")
  AnalysisId = toString(a[[1]])

  # Insert feed analysis
  str = paste("INSERT INTO PigIt.FeedAnalysis
               (Id, FeedId, DryMatter, CrudeProtein, EnergyFEsv, Lysine, Methionine,
                Cysteine, Threonine, Tryptophan, Isoleucine, Leucine, Histidine,
                Phenylalanine, Tyrosine, Valine) ",
              "VALUES ('", AnalysisId, "', '", toString(pigItFeeds$Id[i]),
              "', ", pigItFeeds$Toerstof...[i],
              ", ", pigItFeeds$Protein.[i], ", ", pigItFeeds$FEsv.[i],
              ", ", pigItFeeds$Lysin.[i], ", ", pigItFeeds$Methionin.[i],
              ", ", pigItFeeds$Cystin.[i], ", ", pigItFeeds$Treonin.[i],
              ", ", pigItFeeds$Tryptofan.[i], ", ", pigItFeeds$Isoleucin.[i],
              ", ", pigItFeeds$Leucin.[i], ", ", pigItFeeds$Histidin.[i],
              ", ", pigItFeeds$Fenylalanin.[i], ", ", pigItFeeds$Tyrosin.[i],
              ", ", pigItFeeds$Valin.[i], ")", sep="")
  ins = sqlQuery(pigIt, str)
  print(ins)

  # Look the feedstuff up in the table with DLBR feedstuffs
  str = paste("SELECT DLBR_FeedId FROM DLBR_Feedstuff WHERE NutritionalContents LIKE '",
              pigItFeeds$EgenskabIndhold[i], "'",  sep ="" )

  feId = sqlQuery(pigIt, str)
  feId
  feedId = toString(feId[[1]])

  # Insert it in DLBR feedstuffs if it doesn't exist
  if (feedId == "") {

    # Find observer
    observer = getObserver(FarmID, pigItFeeds$Ajourini[i])

    # Insert into table
    str = paste("INSERT INTO DLBR_Feedstuff
               (DLBR_FeedId, AnalysisId, UpdateTime, UpdateResponsible,
                FeedName, NutritionalContents, CodeLot) ",
                "VALUES ('", pigItFeeds$ID[i], "', '", AnalysisId,
                "', '", pigItFeeds$Ajourtid[i], "', '",
                observer, "', '", pigItFeeds$FeedName[i], "', '",
                pigItFeeds$EgenskabIndhold[i], "', ",
                pigItFeeds$FoderKodeParti[i], ")", sep="")
    ins = sqlQuery(pigIt, str)
    print(paste(i, "inserted", pigItFeeds$ID[i], ins))
    feedId = toString(pigItFeeds$ID[i])
  }
}
```

```
# Read list of feed consumptions from DLBR database
feedConsRaw = sqlQuery(kappel, "SELECT SFoderEmne.ID, TidFra, TidTil,
  VentilNr, ProcesUdstyr, Forbrug_Kg, FoderNavnLang, SFoderEmne.Ajourtid,
  SFoderEmne.Ajourini, EgenskabIndhold, FoderKodeParti
  FROM SFoderforbrug, SVentil, PigItVentil, SFoderEmne
  WHERE PigItVentil.VentilId = SVentil.ID
  AND FoderEmneID = SFoderEmne.ID
  AND SFoderforbrug.VentilID = SVentil.ID ORDER BY VentilNr, FoderNavnLang, TidFra")

# Trim feed names
feedConsRaw$FeedName = gsub("^\\s+|\\s+$", "", feedConsRaw$FoderNavnLang)

# Interpret string EgenskabIndhold as described in separate note
feedCons = addAnalysis(feedConsRaw, codes,
                       c(1, 9, 101, 103, 107, 109, 111, 113,
                         115, 117, 119, 121, 123, 165), FALSE)

# Remove blanks ...
colnames(feedCons) = make.names(colnames(feedCons))
colnames(feedCons)

# Copy feed consumptions to PigIt table
for (i in 1:length(feedCons[,1])) {
  # Convert dates
  FromDate = as.Date(paste(feedCons$TidFra[i], "", sep=""), format = "%Y%m%d")
  ToDate = as.Date(paste(feedCons$TidTil[i], "", sep=""), format = "%Y%m%d")

  # Look the feedstuff up
  str = paste("SELECT DLBR_FeedId FROM DLBR_Feedstuff
              WHERE NutritionalContents LIKE '",
              feedCons$EgenskabIndhold[i], "'",  sep ="" )

  feId = sqlQuery(pigIt, str)
  feId
  feedId = toString(feId[[1]])
  feedId

  # Insert it if it doesn't exist. In that case an analysis entry must also be created
  if (feedId == "") {

    # Create uniqueidentifier
    a = sqlQuery(pigIt, "SELECT NEWID()")
    AnalysisId = toString(a[[1]])
    print(AnalysisId)

    # Insert feed analysis - FeedId will be NULL!
    str = paste("INSERT INTO PigIt.FeedAnalysis
                (Id, DryMatter, CrudeProtein, EnergyFEsv, Lysine, Methionine,
                Cysteine, Threonine, Tryptophan, Isoleucine, Leucine, Histidine,
                Phenylalanine, Tyrosine, Valine) ",
                "VALUES ('", AnalysisId, "', ", feedCons$Toerstof...[i],
                ", ", feedCons$Protein.[i], ", ", feedCons$FEsv.[i],
                ", ", feedCons$Lysin.[i], ", ", feedCons$Methionin.[i],
                ", ", feedCons$Cystin.[i], ", ", feedCons$Treonin.[i],
                ", ", feedCons$Tryptofan.[i], ", ", feedCons$Isoleucin.[i],
                ", ", feedCons$Leucin.[i], ", ", feedCons$Histidin.[i],
                ", ", feedCons$Fenylalanin.[i], ", ", feedCons$Tyrosin.[i],
                ", ", feedCons$Valin.[i], ")", sep="")
    ins = sqlQuery(pigIt, str)
    print(ins)

    # Get observer
    observer = getObserver(FarmID, feedCons$Ajourini[i])

    # Insert into table with DLBR feedstuffs
    str = paste("INSERT INTO DLBR_Feedstuff
                (DLBR_FeedId, AnalysisId, UpdateTime, UpdateResponsible,
                FeedName, NutritionalContents, CodeLot) VALUES ('",
                feedCons$ID[i], "', '", AnalysisId, "', '",
                feedCons$Ajourtid[i], "', '", observer, "', '",
                feedCons$FeedName[i], "', '", feedCons$EgenskabIndhold[i], "', ",
                feedCons$FoderKodeParti[i], ")", sep="")
    ins = sqlQuery(pigIt, str)
    print(paste(i, "inserted", feedCons$ID[i], ins))
    feedId = toString(feedCons$ID[i])
  }

  # Get the analysis ID for this feedstuff
  str = paste("SELECT AnalysisId FROM DLBR_Feedstuff WHERE DLBR_FeedId = '",
              feedId, "'",  sep ="" )
  a = sqlQuery(pigIt, str)
  AnalysisId = toString(a[[1]])
  print(AnalysisId)

  # Get dispenser ID matched by name (which is a number)
  str =  paste("SELECT Id FROM PigIt.FeedDispenser
               WHERE DatabaseId = 1 AND FeedDispenserName = '",
               toString(feedCons$VentilNr[i]), "'", sep ="" )
  veId = sqlQuery(pigIt, str)
  ventilId = toString(veId[[1]])
  ventilId
  # Insert feed consumption record
  str = paste("INSERT INTO PigIt.FeedConsumptionPeriod ",
              "(Id, AnalysisId, FirstDay, LastDay, DispenserId, ConsumedAmount) ",
              "VALUES (NEWID(), '", AnalysisId, "', '", FromDate,"', '",
              ToDate, "', '", ventilId, "', ", feedCons$Forbrug_Kg[i], ")", sep="")
  ins = sqlQuery(pigIt, str)
  print(ins)
}
```

Appendix D

Properties of feedstuffs

```
Nr.;Navn paa egenskab
1;Toerstof %
2;Raaaske
3;Enhed
4;Version 110831
5;Vand
8;Genberegn: Slet
9;Protein
10;Raaprotein
11;St.f. raaprotein
12;Kvaelstof
14;Gl.f. raaprotein
15;Fedt
16;Raafedt
17;Ford. raafedt
18;Fedtsyre %afRaaf
19;Jodtal
20;Jodtalsprod
21;FaecesFo.OrgStof
23;LFK-kontrol
24;Sukkerarter
25;Kulhydrat
26;Traestof
27;Ford. traestof
28;NFE
29;Ford. NFE
30;Organisk stof
31;Letford.Kulhydr
32;FerMenterbKulhy
33;Opl. fibre
34;Uopl. fibre
35;Sum fibre
36;Stivelse
37;Sukker
38;Sukk+stiv
39;Ford. koef.
40;FK-raaprot.(Gl.)
41;FK-raafedt (Gl.)
42;FK-traestof
43;FK-NFE
44;FK-P 0FytTilsat
45;EFOS
46;EFNi
47;UTSi
```

```
48;EFOSi
49;FerMentKulhydso
50;FE-korr.faktor
51;FK-raaprot.(Ny)
52;FK-raafedt (Ny)
53;I-Faktor, %
54;MIKROMINERALER
55;Mineraler
56;Calcium
57;Fosfor
58;Ford.P 0FytTils
59;Kalium
60;Natrium
61;Klorid
62;Magnesium
63;Jern, mg
64;Kobber, mg
65;Mangan, mg
66;Zink, mg
67;Jod, mg
68;Selen, mg
69;Svovl
70;FytaseFYT,1000e
71;Ford.P Fyt 60%
72;Ford.P Fyt 100%
73;Ford.P Fyt 150%
74;Ford.P Fyt 200%
75;Vitaminer,kgts
76;A-vit., 1000 IE
77;B1-thiamin, mg
78;B2-ribofl, mg
79;B6-pyridox, mg
80;B12-cyanoc, mcg
81;B3-Niacin, mg
82;D-Panto.syre,mg
83;Biotin, mg
84;Folinsyre, mg
85;Cholin, mg
86;C-vitamin, mg
87;D-vit., 1000 IE
88;E-vit(A-tf.),mg
89;K-vitamin, mg
98;Aminosyrer,raap
99;Cystin+methion.
100;St.f. cys+meth
101;Cystin
102;St.f. cystin
103;Fenylalanin
104;St.f. fenylalan
105;Fenylala+tyr
106;St.f. fenyl+tyr
107;Histidin
108;St.f. histidin
109;Isoleucin
110;St.f. isoleucin
111;Leucin
112;St.f. leucin
113;Lysin
114;St.f. lysin
115;Methionin
116;St.f. methionin
117;Treonin
```

```
118;St.f. treonin
119;Tryptofan
120;St.f. tryptofan
121;Tyrosin
122;St.f. tyrosin
123;Valin
124;St.f. valin
135;Andre stof,kgts
136;Zinkbac, mg
137;Spiramycin, mg
138;Tylosin, mg
139;Virginiamyc, mg
140;Flavomycin, mg
142;Salinomycin, mg
143;Avilamycin, mg
144;Benzoesyre
145;MicroAid, mg
146;Bioplus 2B, mg
147;Sangrovit, mg
148;ÂÌkologi, %
149;Omlaegnings-%
160;Energi
161;FE (Aktuel)
163;MJ OE
164;MJ NE
165;FEsv
166;FEso
167;FEsv Kontrol
168;FEso Kontrol
169;FEs (EFOS-syst)
170;Elektrolytbal
181;FE/100kg
185;FEsv/100kg
186;FEso/100kg
187;FEsvKontr/100kg
188;FEsoKontr/100kg
189;FEs/100kg
198;Aktuel pris
199;Forventet pris
200;Gl.f. cys+meth.
202;Gl.f. cystin
204;Gl.f. fenylalan
206;Gl.f. fen+tyr
208;Gl.f. histidin
210;Gl.f. isoleucin
212;Gl.f. leucin
214;Gl.f. lysin
216;Gl.f. methionin
218;Gl.f. treonin
220;Gl.f. tryptofan
222;Gl.f. tyrosin
224;Gl.f. valin
239;Ford.koef.2
240;FK-cyst+meth
241;FK-cystin
242;FK-fenylalanin
243;FK-fenylal+tyro
244;FK-histidin
245;FK-isoleucin
246;FK-leucin
247;FK-lysin
248;FK-methionin
```

```
249;FK-treonin
250;FK-tryptofan
251;FK-tyrosin
252;FK-valin
341;FK-cys forhold
342;FK-fen forhold
344;FK-his forhold
345;FK-iso forhold
346;FK-leu forhold
347;FK-lys forhold
348;FK-met forhold
349;FK-tre forhold
350;FK-try forhold
351;FK-tyr forhold
352;FK-val forhold
471;FK-P FytNiv 60%
472;FK-P FytNiv100%
473;FK-P FytNiv150%
474;FK-P FytNiv200%
480;FytaseFTU,1000e
481;B.xylanase,1000
800;Ledetekster
801;FORD. INDHOLD
802;INDHOLD
803;
804;
805;
```

PigIT Report No. 3 • April 2014
http://www.pigit.net/publications/PigIT-Report3.pdf

Centre for Herd-oriented Education, Research and Development
Department of Large Animal Sciences
University of Copenhagen