Glucose: gather - mutate - ifelse

Introduction

There are different ways to organise a dataset. Statisticians almost always prefer the *long* format, while in other circumstances a *wide* format can make an overview easier. The long format contain one row for each observation, i.e. if observations are made at 4 time points for 14 horses this will lead to 4*14 rows of data. In the wide format instead each row represents one horse and the 4 measurements made on the same horse are shown in different columns and with different column names.

Plots and data manipulations, like producing new variables and sorting datasets is also part of the exercise.

Before you start, make sure you have installed and loaded the `tidyverse` package.

```
library(tidyverse)
```

Read the glucose data set

This dataset contains glucose levels of Icelandic horses[1]. Observations are made for 14 horses that get one of three different diets. Glucose and insulin levels are measured at a baseline, and after 6, 12 and 24 hours. The dataset is in the wide format.

```
glucose <- read_csv("Glucose.csv")
```

Use mutate to create new variables

We are interesting in investigating changes from the baseline glucose level at time point 24 hours. We compute both the glucose difference (`Glucose_diff`) and the ratio between measurement at time point 24 and the baseline (\texttt{Glucose_ratio}). To add these new variables, we use `mutate`:

```
Glucose_change <- glucose %>%
  mutate(Glucose_diff = Glucose_24 - Glucose_baseline,
         Glucose_ratio = Glucose_24 / Glucose_baseline)
```

Plot your data

A boxplot can illustrate how the difference of glucose levels differs with diet.

```
Glucose_change %>%
  ggplot(aes(x = Diet,
             y = Glucose_diff)) +
  geom_boxplot()
```

[1] This is a simulated dataset based on a realistic experiment setup

The plot can be improved by setting more informative names on the x- and y-axes using `labs` for the x-and y-axis and \texttt{scale_x_discrete) to name the three diets (categories on the x-axis):

```
Glucose_change %>%
  ggplot(aes(x = Diet,
             y = Glucose_diff)) +
  geom_boxplot() +
  labs(y = "Difference in Glucose",
       x = "Sugar Content in Diet") +
  scale_x_discrete(labels = c("High", "Medium", "Low"))
```

Exercise

1. Make boxplots for the glucose ratio computed above.
2. Compute differences between time point 12 and the baseline for insulin and make boxplots for this variable.

Use gather to bring your dataset a long format

Gather all observation into one column with numeric data. We keep `ID` and `Diet` for each row and use the remaining column headers to produce a *key* for each row.

```
glucose_tidy <- glucose %>%
  gather(key, measurement, -ID, -Diet)
```

This means that observations can now be found in one column, while the *key* indicates which variable value is given. See the first entries of the data set:

```
head(glucose_tidy)
```

In the next step we separate the key into the variable name and the time point. Note that you need to have names that are easy to split and that these names originate from the column names in the original data set. In this case the names are separated by an underscore (_) which is passed to the program with the `sep = "\\_"` statement.

```
glucose_tidy <- glucose %>%
  gather(key,
         measurement,
         -ID, -Diet) %>%
  separate(key,
           into = c("variable", "time"),
           sep = "\\_")
```

```
head(glucose_tidy)
```

Usually we do not want to have Insulin and Glucose in the same column. To split it into two columns according to variable name the statement `spread` can be used:

```
glucose_tidy <- glucose %>%
  gather(key,
         measurement,
         -ID, -Diet) %>%
  separate(key,
           into = c("variable", "time"),
           sep = "\\_") %>%
  spread(variable, measurement) # Create columns for each value of variable,
                                # and put the values from measurement in these cells
```

The first observations in the new dataset are seen below. We have now 5 columns: `ID`, `Diet`, `time`, `Glucose`, `Insulin`.

```
head(glucose_tidy)
```

If we want to rename the time point now called baseline to 0 we can use the `mutate` statement and change the variable `time` if (and only if) it is equal to baseline. The `ifelse` statement first contains the condition (if `time == "baseline"`), then the value that is given if the condition is true (`0`) and then the value if the condition is not true (`time`, i.e. the other values of time).

```
glucose2 <- glucose_tidy %>%
  mutate(time = ifelse(time == "baseline", 0, time))
```

To inspect the dataset we can use `View`.

```
View(glucose2)
```

In this case `time` is a character (i.e. text) and `ID` is numeric (i.e. numbers), as can be seen by running `str(glucose2)`. Reasonably, `ID` should be a factor and time could be either a factor (a categorical variable) or a numeric variable. Let's set it to a numeric variable here, even if it probably would be best to use as a factor in a statistical analysis.

```
glucose2 <- glucose_tidy %>%
  mutate(time = ifelse(time == "baseline", 0, time)) %>%
  mutate(time = parse_number(time),
         ID = as.factor(ID))
```

Run `str(glucose2)` to check the new variable types.

By using `arrange` we can bring the data into a logical order, e.g. sorted by Treatment (Diet), Animal ID, and time in the order 0, 6, 12, and 24.

```
glucose2 <- glucose_tidy %>%
  mutate(time = ifelse(time == "baseline", 0, time)) %>%
  mutate(time = parse_number(time),
         ID = as.factor(ID)) %>%
  arrange(Diet, ID, time)
```

Further examples of how to reshape data from wide to long, and vice versa, can be found in Chapter 5.11 of *Modern Statistics with R*.

Plot glucose data

A line plot can be used to illustrate the time series for each horse. To get individual lines for each horse use the `group=ID` statement. Different colors are used for the three different diets.

```
glucose2 %>%
  ggplot(aes(x = time,
             y = Glucose,
             group = ID,
             color = Diet)) +
  geom_line()
```

It might also be good to include a mean line that illustrates how the mean for each diet changes over time. This can be obtained by `stat_summary`. Line width (`lwd`) is here set to 1.2 to get a thicker line for this mean.

```
glucose2 %>%
  ggplot(aes(x = time,
             y = Glucose,
             group = ID,
             color = Diet)) +
  geom_point() +
  stat_summary(aes(group = Diet),
               fun.y = mean,
               geom = "line",
               lwd = 1.2)
```

Exercise

3. Make a similar plot for the insulin data.

Solutions to exercices

1.

```
Glucose_change %>%
  ggplot(aes(x = Diet,
             y = Glucose_ratio)) +
  geom_boxplot() +
  labs(y = "Glucose ratio",
       x = "Sugar Content in Diet") +
  scale_x_discrete(labels = c("High", "Medium", "Low"))
```

2.

```
Glucose_change <- glucose %>%
  mutate(Glucose_diff = Glucose_24 - Glucose_baseline,
         Glucose_ratio = Glucose_24 / Glucose_baseline,
         Insulin_diff = Insulin_12 - Insulin_baseline)

Glucose_change %>%
  ggplot(aes(x = Diet,
             y = Insulin_diff)) +
  geom_boxplot() +
  labs(y = "Difference in Insulin",
       x = "Sugar Content in Diet") +
  scale_x_discrete(labels = c("High", "Medium", "Low"))
```

3.

```
glucose2 %>%
  ggplot(aes(x = time,
             y = Insulin,
             group = ID,
             color = Diet)) +
  geom_line()
```

```
glucose2 %>%
  ggplot(aes(x = time,
             y = Insulin,
             group = ID,
             color = Diet)) +
  geom_point() +
  stat_summary(aes(group = Diet),
               fun.y = mean,
               geom = "line",
               lwd = 1.2)
```