Coding Tasks on dplyr, tidyr and ggplot2 - 17th May 2021

Mario Keller

May 10, 2021

Contents

1 Introduction

A good bioinformatician not only produces huge amounts of data, but is also able to cleanse and adjust these in order to then produce plots that are ready for publication. In today's coding task we will have a look on 3 R packges that are useful for the mentioned tasks.

The first of these packages is dplyr, which offers a grammar for data manipulation. The second package is tidyr, which helps you to clean up your data. The third package is ggplot2, which helps your to create good looking graphics that can be adjusted down to the smallest detail.

We will not fully explore the complete functionality of these packages but have a look on the most important functions. For further details check out the websites of the packages that are placed as a hyperlink in the package names above.

2 Required libraries

The following libraries and data are required in today's coding tasks.

```r
library(dplyr)
library(tidyr)
library(ggplot2)
library(gridExtra)

#Adjust the path for your system
TPM.df <-readRDS("data/TPM.df.rds")
```

3 dplyr

One of the most important aspects about the dplyr packages is the so-called pipe operator %>%. This operator allows you to chain together sets of operations in a very intuitive fashion. Here is one example where we create a data frame, for which we calculate the mean for each row and afterwards the mean of the computed row means:

```r
#With nested parentheses.
mean(rowMeans(data.frame(x=1:10, y=1:10)))
```

```
## [1] 5.5
```

```r
#With %>%
data.frame(x=1:10, y=1:10) %>% rowMeans %>% mean
```

```
## [1] 5.5
```

As you can see, using the %>% operator is much more intuitive than using nested parentheses.

In addition the dplyr package offers additional functions that help you to filter your data, add new columns, adjust existing columns and arrange your data frames.

3.1 Task1 - Level: Beginner

Use the function filter() to filter for patients, whose Gene1 TPM values in brain and liver are > 150 and use afterwards arrange() to order the patients in descending order by their Gene1 Brain TPM value. Use the %>% operator and continue with the filtered data frame.

Hint: You should end up with 731 patients.

3.2 Task2 - Level: Advanced

Create a new column called sumVar which contains the sum of Gene1_Brain, Gene2_Brain, Gene1_Liver and Gene2_Liver for each patient. Afterwards filter for patients where sumVar > 500 and subsequently remove the sumVar column with select().

Helpful functions are mutate(), rowwise() and c_across(). Use the %>% operator.

Hint: Check out the example in ?c_across. You should end up with 679 patients. The data frame is now a tibble, which is just an advanced data frame.

4 tidyr

If you did not solve Task2 load the data.frame stored in "TPM.df.tidyr.rds" and continue with this data frame.

The tidyr package offers different functions for cleaning up your data frames such as changing them from a wide to long format nad vice versa.

4.1 Task3 - Level: Beginner

We will now mimic a scenario where for one of the patients the Gene1_Brain value is missing. Enter for the patient in row 3 an NA in the Gene1_Brain column (e.g. TPM.df[3,2] <- NA or TPM.df$Gene1_Brain[3] <- NA). Use replace_na() to replace the NA in the Gene1_Brain column with 361.0915.

Hint: replace() needs a list that contains the replacement values for each column.

4.2 Task4 - Level: Advanced

At the moment we have data frame in a wide format. However, we want it to be in a long format. Use pivot_longer() to produce a data frame where each combination of patient, gene and tissue is a single row.

Hint: You can split the names of the columns 2 to 5 by "_" and use "Gene" and "Tissue" as new column names. In addition the values column could be called "TPM".

This is how the data frame should look like:

```
## # A tibble: 8 x 4
##   Patien.ID Gene  Tissue   TPM
##   <fct>     <chr> <chr>  <dbl>
## 1 164       Gene1 Brain  394. 
## 2 164       Gene2 Brain   88.1
## 3 164       Gene1 Liver  189. 
## 4 164       Gene2 Liver   82.4
## 5 747       Gene1 Brain  362. 
## 6 747       Gene2 Brain   71.2
## 7 747       Gene1 Liver  195. 
## 8 747       Gene2 Liver   91.9
```

4.3 Task5 - Level: Advanced

We can also go from a long format to a wide format. Use pivot_wider() to make two columns out of the "Gene" column.

Hint: The parameters "names_from" and "values_from" are helpful.

This is how the data frame should look like:

```
## # A tibble: 6 x 4
##   Patien.ID Tissue Gene1 Gene2
##   <fct>     <chr>  <dbl> <dbl>
## 1 164       Brain   394.  88.1
## 2 164       Liver   189.  82.4
## 3 747       Brain   362.  71.2
## 4 747       Liver   195.  91.9
## 5 842       Brain   361.  73.7
## 6 842       Liver   170.  76.4
```

5 ggplot2

If you did not solve Task5 load the data.frame stored in "TPM.df.ggplot2.rds" and continue with this data frame.

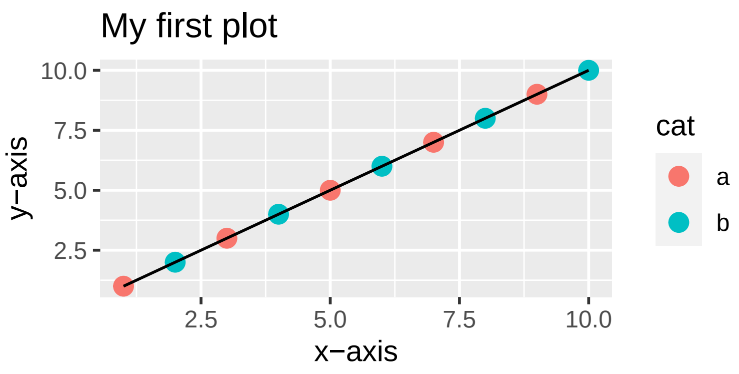
The ggplot2 package offers a modular structure that allows you to generate plots layer by layer. The most important function is ggplot() that initializes a ggplot object in which we declare the input data frame and set the plot aesthetics. The aesthetics are defined within ggplot() by using aes(). Aesthetics tell ggplot which columns contain the values on the x- and y-axis or which column should be used as a color or fill. After we have initialized a ggplot object with ggplot() we can add layers with the plus sign (+). Here are some examples of layers:

- geom_point() for scatter plots
- geom_line() for line plots
- geom_col() for bar charts
- geom_histogram() for histograms
- geom_density() for density plots

In addition there are different functions that can be used to adjust the x- and y-axis, to set labels or to set a user-defined color-scale.

Here is a simple example that shows how to create a scatter plot and to color the points by a certain column in the data.frame

```r
#Create the data frame
data.frame(column1=1:10, column2=1:10, cat=factor(rep(c("a","b"),5), levels=c("a", "b"))) %>%
  #Initialize the ggplot object
  ggplot(., aes(x=column1, y=column2, color=cat)) +
  #Add a layer with points
  geom_point(size=3) +
  #Add a layer with a line
  geom_line(col="black") +
  #Re-name the axes and set a title
  labs(x="x-axis", y="y-axis", title="My first plot")
```



Once you have internalized the ggplot philosophy it is very easy to create beautiful plots that nicely represent your results.

5.1 Task6 - Level: Beginner

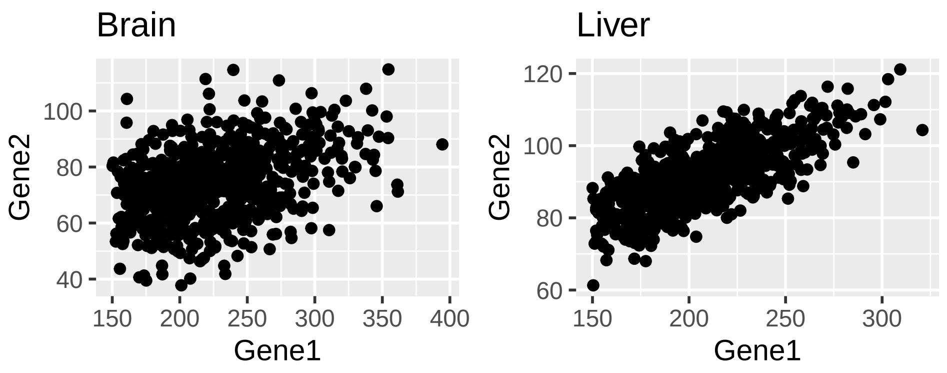
We want to make for each of the two tissues a scatter plot to show the correlation between Gene1 and Gene2. One easy way is to first subset the data.frame to have a single data frame for each of the tissues, create the plot and store it in a variable. Subset the data.frame once for the Brain tissue and once for the Liver tissue and store the two data frames as Brain.TPM.df and Liver.TPM.df

Hint: The subset() function is very useful.

5.2 Task7 - Level: Beginner

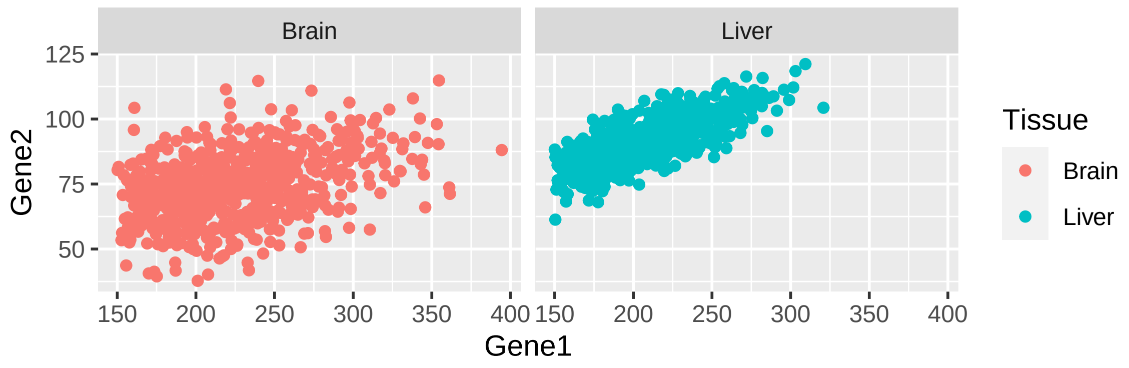
Use the two data frames to create two scatter plots using geom_point() with Gene1 on the x- and Gene 2 on the y-axis and store them as Brain.plot and Liver.plot. Afterwards use grid.arrange() from the gridExtra package to combine the two plots into a single plot side-by-side.

Hint: x and y are defined in aes(). You can tell grid.arrange whether you want the plots side by side or on top of each other (check out ncol and nrow).



5.3 Task8 - Level: Advanced

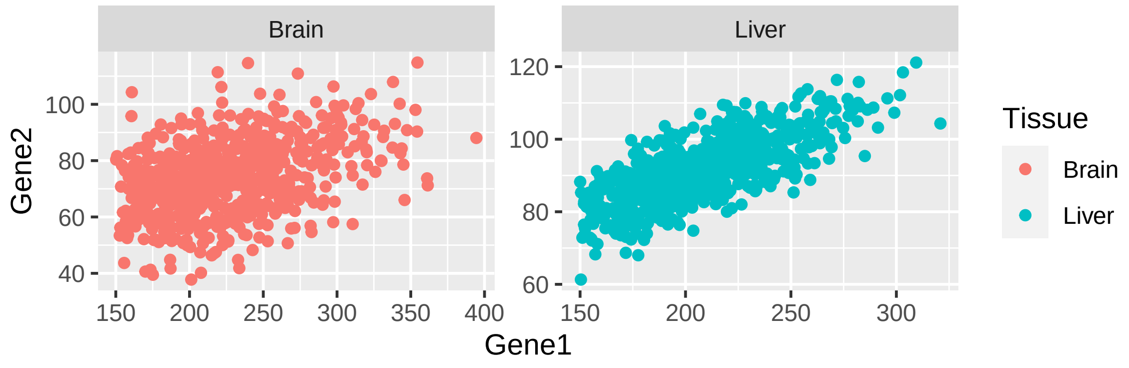
An alternative to the above approach is to use facet_wrap() from the ggplot2 package to automatically create subplots. Use the TPM.df with facet_wrap() to create two subplots based on the Tissue column. In addition, color the the points by Tissue.



5.4 Task9 - Level: Advanced

As you can see the two plots share the same x- and y-axis. Try to set for each plot individual axes.

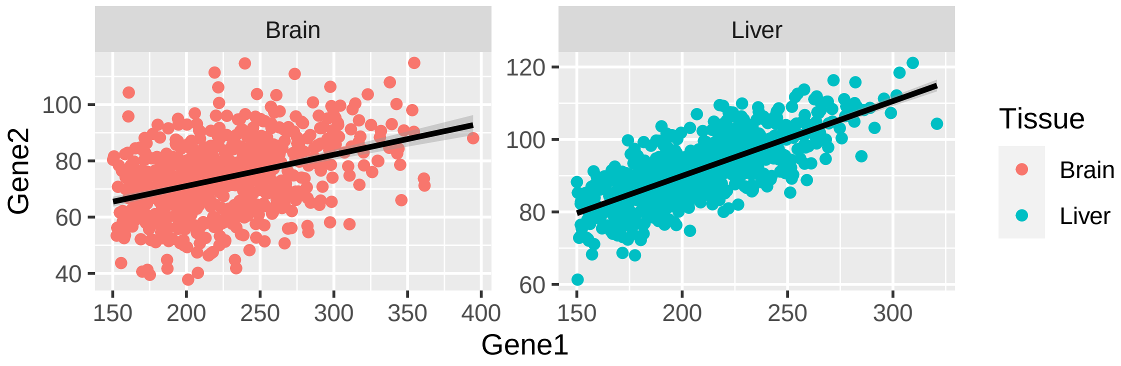
Hint: Check out ?facet_wrap to find out how you can create for each plot individual x- and y-axes.



5.5 Task10 - Level: Advanced

When we try to look at correlations a regression line is often helpful. Use geom_smooth() to add a black regression line as a new layer.

Hint: Check out ?geom_smooth and look for the correct "method". lm could be useful.


Brain
Liver
Gene2
Gene1
Tissue
Brain
Liver