Graphics Choices

Data and Computing Fundamentals

Simple Graphics for Gene Expression

In the 1980s, the National Cancer Institute developed a set of 60 cancer cell lines, called NCI60. The original purpose was for screening potential anti-cancer drugs. A recent overview in *Nature* describes some of the ongoing work with these cell lines.

Here you will examine gene expression in these cell lines using data described in Staunton *et al.*. More than 41,000 probes were used for each of the 60 cell lines.

For convenience, the data are provided by the `DCI` package in two data tables `NCI60` and `NCI60cells`.[1]

`NCI60` is somewhat large — 41,078 probes by 60 cell lines. Each of those 2,454,680 entries is a measure of how much a particular gene was expressed in one cell line.

In this exercise, you're going to look at just one of the probes. The question you will make a graphic to address is:

Is the probe's gene expressed in different amounts across different types of cancer?

This activity is intended to be about the construction of informative graphics, not the mechanics of transfiguring data. To save time, the following set of commands should be copied into your report, so that you can execute them to create glyph-ready data.

You should be able to read and understand the commands, but you don't have to construct them from scratch.[2]

Before running the commands, you'll need to assign a value to `probeSelected`. Pick one of these probes to look at, it doesn't matter which one.[3]

A_32_P33263
CP
FSD1
A_32_P52227
A_23_P135646
FLJ38359
ZNF682
A_24_P936145
ALS2CR4
DZIP1L
LOC492303
ZNF277P
CBFA2T3
MUC13
A_32_P168727

```
probeSelected <- "Your probe name"
```

[1] `NCI60` and `NCI60cells` are lightly re-organized from the form provided by Staunton *et al.*.

[2] A few lines involve data cleaning, which you have not yet studied. Don't worry about the lines where `as.character()` or `gsub()` appear.

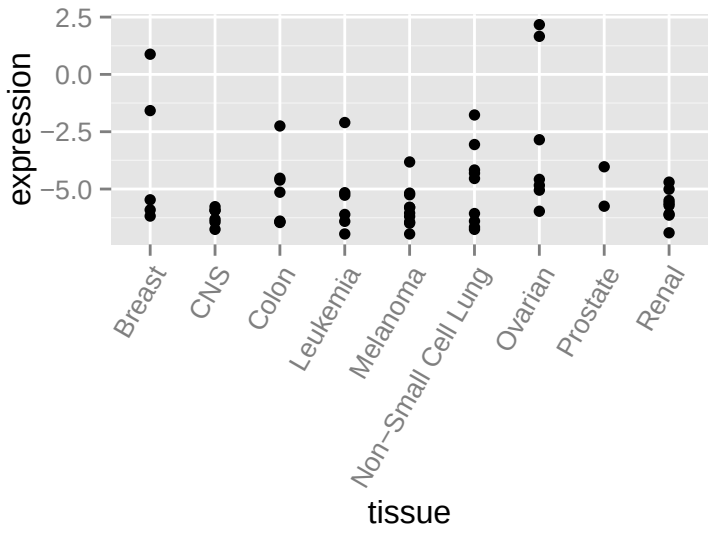
[3] In reality, the choice of probe makes a great deal of difference. These particularly probes were chosen based on a screening test. In an upcoming part of the course, you'll look at how to deal with multiple variables, such as the 41,078 variables here.

```
Narrow <-
  NCI60 %>%
  tidyr::gather( cellLine, expression, -Probe ) %>%
  mutate( cellLine=as.character(cellLine) )
JustTheTypes <-
  NCI60cells %>%
  mutate( cellLine=as.character(cellLine) ) %>%
  mutate( cellLine=gsub("\\:",".", cellLine)) %>%
  select(cellLine, tissue )
Narrow <- inner_join( Narrow, JustTheTypes )
MyProbe <-
  Narrow %>%
  filter( Probe==probeSelected )
SummaryStats <-
  MyProbe %>%
  group_by( tissue ) %>%
  summarise( mid=mean( expression ),
             se=sd( expression )/sqrt(n()))
```

The `MyProbe` and `SummaryStats` data frames are in glyph-ready form.

We're ready for a simple graphic: Make a scatter plot of the expression versus tissue type:

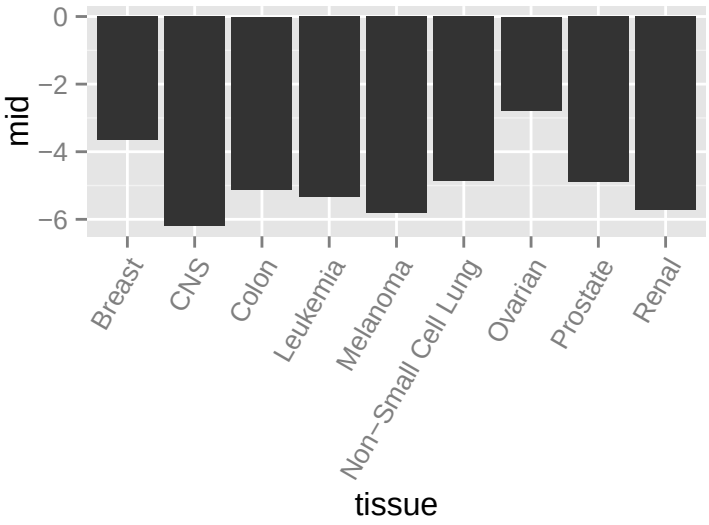
```
ggplot( MyProbe, aes(x=tissue, y=expression)) +
  geom_point() +
  theme(axis.text.x=element_text(angle=60,hjust=1))
```



It's hard to see, at a glance, the differences between expressions in the various tissue types. Therefore, this is not a good graphic.

It's very common in the professional literature to display data like these as a bar chart. It might look like this:

```
pp<- ggplot( SummaryStats, aes(x=tissue, y=mid )) +
  geom_bar( stat="identity" ) +
  theme(axis.text.x=element_text(angle=60,hjust=1))
pp
```



Before going on, decide what you like and don't like about this plot. Write it down

Then continue ...

... but not before you've written down your opinions ...

... Really!

... Are you ready now?

... You've really written something?

... Then go on.

There are several bad features of this plot:

- Too much ink.
- The order of the levels of `tissue` is alphabetical. It's unlikely that the mechanisms of cancer consider what we call different types of cancer in English. So the x-axis order is being wasted.
- The precision of the feature (mean expression) is not shown. How would the viewer know whether this spread is just the result of random variation?

Here are several suggestions for improving the graphic:

1. Lighten up on the color. Perhaps `alpha=.2`. Or perhaps a dot plot rather than a bar chart.
2. Reorder the tissue types.

   One way to do this is to pick a quantity that you want to dictate the order of groups. For instance, the mean expression.

   - Create a data table of that quantity for each group (as with `SimpleStats`).
   - Reorder the categories based on the quantity:

   ```
   SummaryStats2 <-
     SummaryStats %>%
     mutate( tissue=reorder( tissue, mid ) )
   ```

   Use `disc()` to order the other way.

3. Show a statistical measure of the variation.

   Without going into details of statistical method, a very common way to present the imprecision in an estimated quantity is with a "standard error." For quantities such as the mean, the standard error has a simple form (but not necessarily a form that will be obvious to you). Here's the calculation that created `SimpleStats` with the mean and the "standard error of the mean."

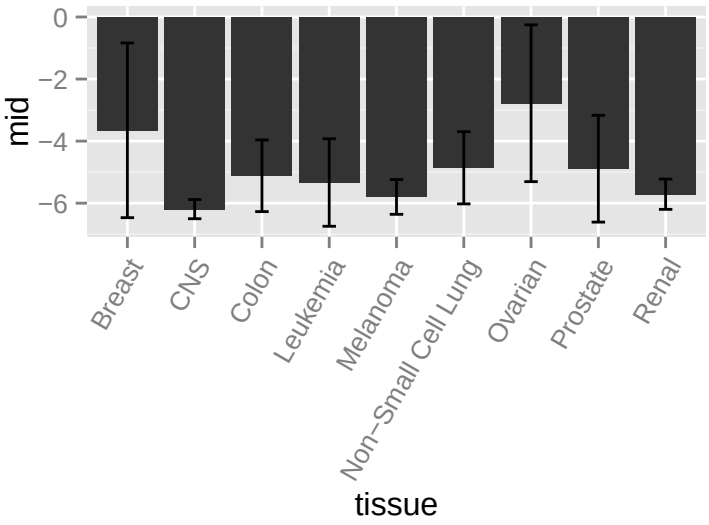
```
SimpleStats <-
  MyProbe %>%
  group_by( tissue ) %>%
  summarise( mid=mean( expression ),
         se=sd(expression)/sqrt(n()) )
```

4. Show the expression value for each of the individual cases in `MyProbe`.

5. Use a different modality, e.g. a dot plot, a box-and-whiskers plot (with `notch=TRUE`), a violin plot.

The conventional solution is not the best: show the bars along with the measure of precision.

```
ggplot( SimpleStats, aes(x=tissue, y=mid )) +
  geom_bar( stat="identity" ) +
  geom_errorbar(aes( ymax=mid+2*se,ymin=mid-2*se),width=.2) +
  theme(axis.text.x=element_text(angle=60,hjust=1))
```



Statisticians refer to such things as "dynamite plots," in a way intended to be pejorative. You can find a better way to present these data.

Explication of the Data Transfiguration

The `NCI60` table is in wide format: each of the cell lines is its own variable. You're going to turn it into a narrow format.

1. Look at the variable names in `NCI60`. How many are there?

2. Run this command:

```
Narrow <-
  NCI60 %>%
  gather( cellLine, expression, -Probe ) %>%
  mutate( cellLine=as.character(cellLine) )
```

Look at a small random set of the cases in `Narrow`. How many variables are there now?

Add in the cell types (with some technical corrections to make names match)

```
JustTheTypes <-
  NCI60cells %>%
  mutate( cellLine=as.character(cellLine) ) %>%
  mutate( cellLine=gsub("\\:",".", cellLine)) %>%
  select(cellLine, tissue )
```

Now, add the kind of cell line the each case belongs to. This information is in the `NCI60cells` table. The variables already have matching names.

```
Narrow <- inner_join( Narrow, JustTheTypes )
```

Using your probe, filter out only the cases involving that probe, for instance:

The result, `MyProbe` has 60 to 600 cases. It should look like this:

```
##         Probe      cellLine expression tissue
## 1 A_32_P33263       BR.MCF7      -1.58 Breast
## 2 A_32_P33263 BR.MDA_MB_231      -5.47 Breast
## 3 A_32_P33263     BR.HS578T      -6.18 Breast
## 4 A_32_P33263     BR.BT_549      -5.91 Breast
```

Now, `MyProbe` will look like this:

```
##         Probe      cellLine expression tissue
## 1 A_32_P33263       BR.MCF7      -1.58 Breast
## 2 A_32_P33263 BR.MDA_MB_231      -5.47 Breast
## 3 A_32_P33263     BR.HS578T      -6.18 Breast
```