
Data science fundamentals 01: Visualize and explore (Sept. 21, 2017)

Recorded Stream

Unfortunately the audio was not captured, but the video was. I don't see why it would make any sense to put the video out without any audio, but I thought we shouldn't give any special privileges to people without hearing disabilities :-)

Data exploration is the art of looking at the data, rapidly generating hypotheses, quickly testing them, then repeating again and again and again. The goal of data exploration is to generate many promising leads that we can later explore in more depth. Visualization is key for data exploration, but not sufficient. We need data transformation and exploratory data analysis which we will cover in this section. Modeling is important too but we will explore it later.

Classwork/Homework: Get familiar with HANES and MIMIC3 data sets - we will be extensively using them.

Prerequisites

Install package tidyverse and load it.

```r
install.packages("tidyverse")
library(tidyverse)
```

Recall data frames. We analyzed HANES data frame and plotted two numerical variables. Among the variables in HANES are:

1. A1C - Hemoglobin percentage in patients
2. UACR - Urine Albumin/Creatinine Ratio in patients

Also recall that when we plotted it using log transformation, we saw two patient groups.

Creating a `ggplot`

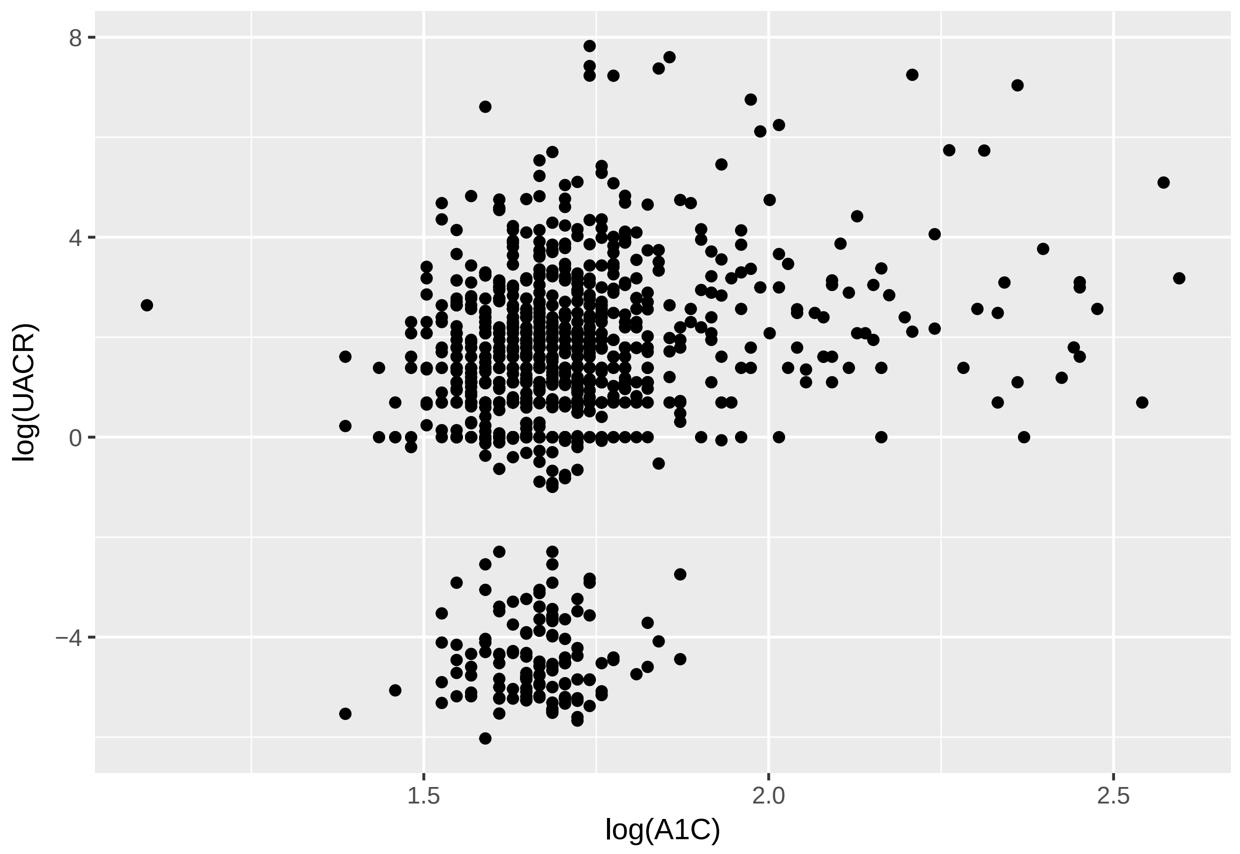
We will first import the curated HANES data set in RStudio:

```r
# Load the package RCurl
library(RCurl)
# Import the HANES data set from GitHub; break the string into two for readability
# (Please note this readability aspect very carefully)
URL_text_1 <- "https://raw.githubusercontent.com/kannan-kasthuri/kannan-kasthuri.github.io"
URL_text_2 <- "/master/Datasets/HANES/NYC_HANES_DIAB.csv"
# Paste it to constitute a single URL
URL <- paste(URL_text_1,URL_text_2, sep="")
HANES <- read.csv(text=getURL(URL))
# Rename the GENDER factor for identification
HANES$GENDER <- factor(HANES$GENDER, labels=c("M","F"))
# Rename the AGEGROUP factor for identification
HANES$AGEGROUP <- factor(HANES$AGEGROUP, labels=c("20-39","40-59","60+"))
# Rename the HSQ_1 factor for identification
HANES$HSQ_1 <- factor(HANES$HSQ_1, labels=c("Excellent","Very Good","Good", "Fair", "Poor"))
# Rename the DX_DBTS as a factor
HANES$DX_DBTS <- factor(HANES$DX_DBTS, labels=c("DIAB","DIAB NO_DX","NO DIAB"))
# Omit all NA from the data frame
HANES <- na.omit(HANES)
# Observe the structure
str(HANES)
```

```
## 'data.frame':    1112 obs. of  23 variables:
##  $ KEY              : Factor w/ 1527 levels "133370A","133370B",..: 28 43 44 53 55 70 84 90 100 107
##  $ GENDER           : Factor w/ 2 levels "M","F": 1 1 1 1 1 1 1 1 1 1 ...
##  $ SPAGE            : int  29 28 27 24 30 26 31 32 34 32 ...
##  $ AGEGROUP         : Factor w/ 3 levels "20-39","40-59",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ HSQ_1            : Factor w/ 5 levels "Excellent","Very Good",..: 2 2 2 1 1 3 1 2 1 3 ...
##  $ UCREATININE      : int  105 53 314 105 163 150 46 36 177 156 ...
##  $ UALBUMIN         : num  0.707 1 8 4 3 2 2 0.707 4 3 ...
##  $ UACR             : num  0.00673 2 3 4 2 ...
##  $ MERCURYU         : num  0.37 0.106 0.487 2.205 0.979 ...
##  $ DX_DBTS          : Factor w/ 3 levels "DIAB","DIAB NO_DX",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ A1C              : num  5 5.2 4.8 5.1 4.3 5.2 4.8 5.2 4.8 5.2 ...
##  $ CADMIUM          : num  0.2412 0.1732 0.0644 0.0929 0.1202 ...
##  $ LEAD             : num  1.454 1.019 0.863 1.243 0.612 ...
##  $ MERCURYTOTALBLOOD: num  2.34 2.57 1.32 14.66 2.13 ...
##  $ HDL              : int  42 51 42 61 52 50 57 56 42 44 ...
##  $ CHOLESTEROLTOTAL : int  184 157 145 206 120 155 156 235 156 120 ...
##  $ GLUCOSESI        : num  4.61 4.77 5.16 5 5.11 ...
##  $ CREATININESI     : num  74.3 73 80 84.9 66 ...
##  $ CREATININE       : num  0.84 0.83 0.91 0.96 0.75 0.99 0.9 0.84 0.93 1.09 ...
##  $ TRIGLYCERIDE     : int  156 43 108 65 51 29 31 220 82 35 ...
##  $ GLUCOSE          : int  83 86 93 90 92 85 72 87 96 92 ...
##  $ COTININE         : num  31.5918 0.0635 0.035 0.0514 0.035 ...
##  $ LDLESTIMATE      : int  111 97 81 132 58 99 93 135 98 69 ...
##  - attr(*, "na.action")= 'omit' Named int  2 15 16 24 26 28 33 34 35 39 ...
##   ..- attr(*, "names")= chr  "2" "15" "16" "24" ...
```

We create a `ggplot` as follows:

```r
# Load the tidyverse library
library(tidyverse)
# Make a ggplot
ggplot(data = HANES) +
geom_point(mapping = aes(x = log(A1C), y = log(UACR)))
```



With ggplot2, we plot with the function `ggplot()`. `ggplot()` creates a coordinate system that we can add layers to. The first argument of `ggplot()` is the dataset to use in the graph. So `ggplot(data = HANES)` creates an empty graph.

We add one or more layers to `ggplot()`. The function `geom_point()` adds a layer of points to the plot, which creates a scatterplot. ggplot2 comes with many geom functions that each add a different type of layer to a plot.

Each geom function in ggplot2 takes a mapping argument. This defines how variables in our dataset are mapped to visual properties. The mapping argument is always paired with `aes()`, and the x and y arguments of `aes()` specify which variables to map to the x and y axes. ggplot2 looks for the mapped variable in the data argument, in this case, HANES.

The general graphing template is given by

```r
ggplot(data = <DATA>) +
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

Classwork/Homework:

1. Run `ggplot(data = HANES)`. What do you see?

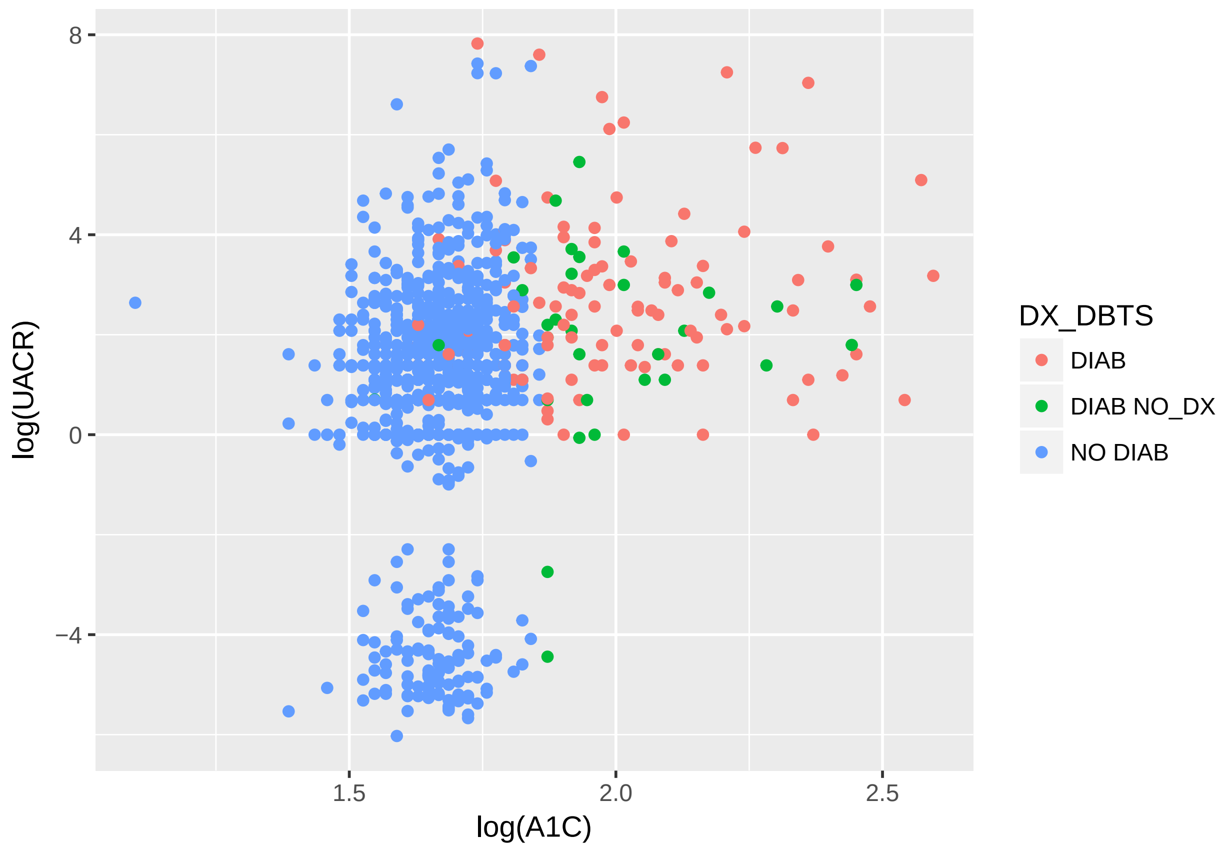
2. How many rows are in HANES? How many columns?
3. What does the DX_DBTS variable describe?
4. Make a scatterplot of HDL vs A1C.

Aesthetic mappings

We can add a third variable, like DX_DBTS, to a two dimensional scatterplot by mapping it to an **aesthetic**. An aesthetic is a visual property of the objects in the plot. Aesthetics include things like the size, the shape, or the color of the points. We can display a point in different ways by changing the values of its aesthetic properties.

We can convey information about the data by mapping the aesthetics in the plot to the variables in HANES dataset. For example, we can map the colors of our points to the DX_DBTS variable to reveal the diabetes status of each person.

```r
# Make a ggplot with asthetic color for the variable DX_DBTS
ggplot(data = HANES) +
geom_point(mapping = aes(x = log(A1C), y = log(UACR), color=DX_DBTS))
```

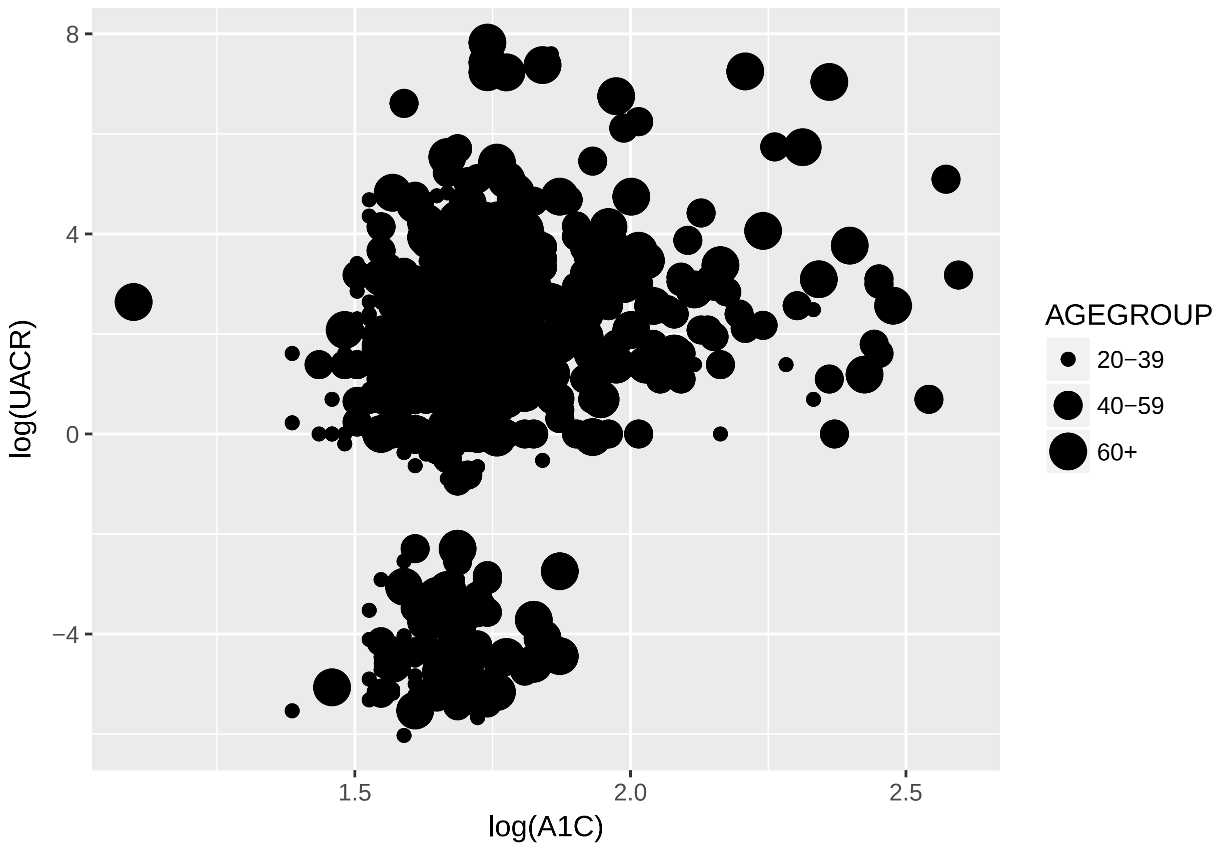


To map an aesthetic to a variable, associate the name of the aesthetic to the name of the variable inside `aes()`. ggplot2 will automatically assign a unique level of the aesthetic (here a unique color) to each unique value of the variable, a process known as scaling. ggplot2 will also add a legend that explains which levels correspond to which values.

The colors reveal many things about this data set. We see that the two clusters are entirely made up of people who don't have diabetes. More, importantly, it reveals people who are diagnosed with diabetes or have one, but not diagnosed.

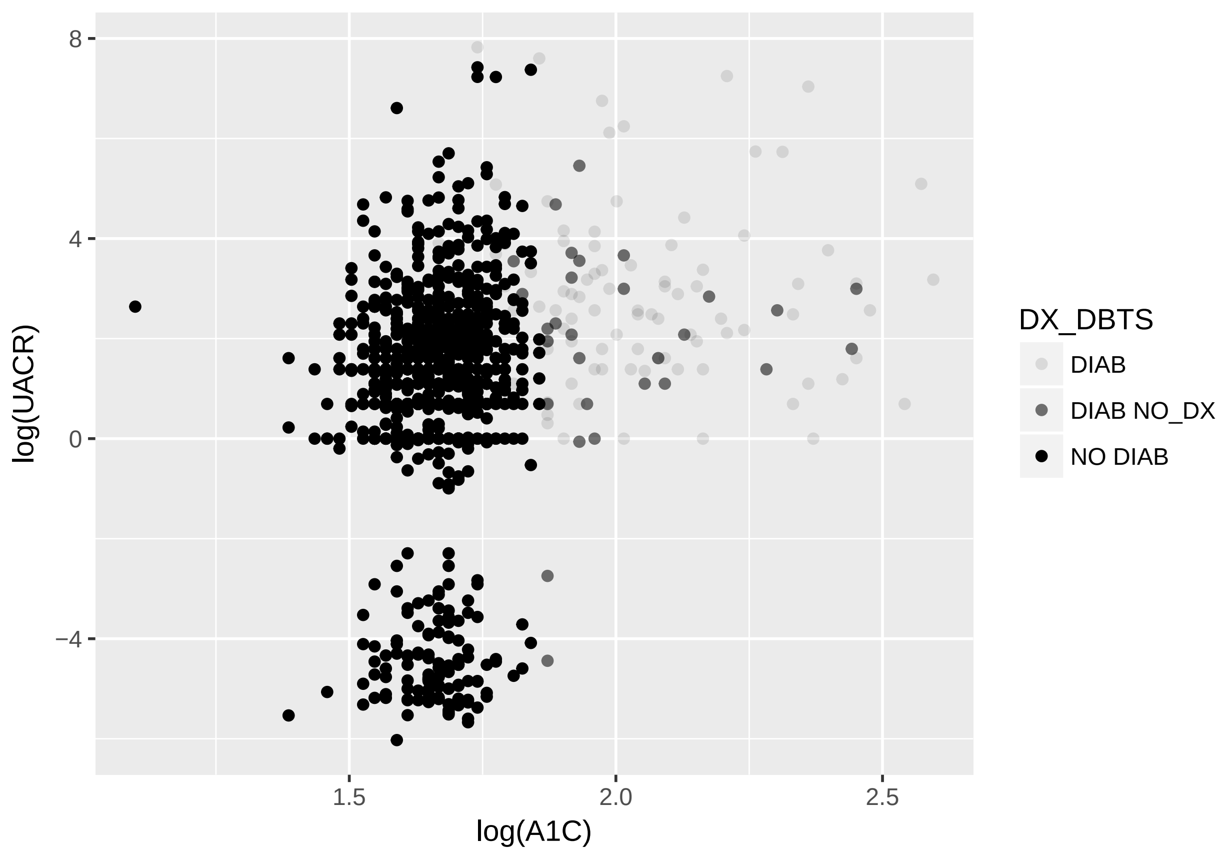
In the above example, we mapped DX_DBTS to the color aesthetic, but we could have mapped AGEGROUP to the size aesthetic in the same way. In this case, the exact size of each point would reveal its AGEGROUP affiliation. We get a warning here, because mapping an unordered variable (class) to an ordered aesthetic (size) is not a good idea.

```r
# Make a ggplot with asthetic size for the variable DX_DBTS
ggplot(data = HANES) +
geom_point(mapping = aes(x = log(A1C), y = log(UACR), size=AGEGROUP))
```



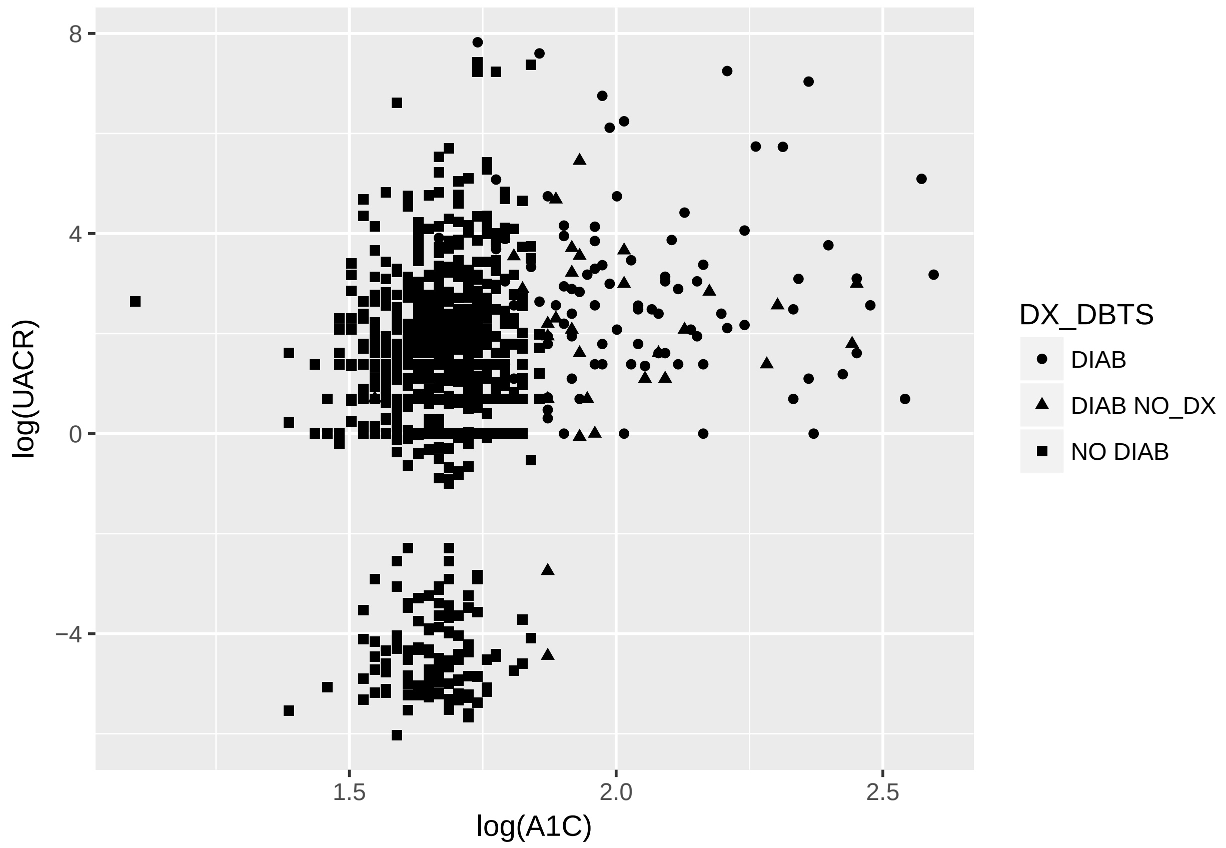
Or we could have mapped DX_DBTS to the alpha aesthetic, which controls the transparency of the points:

```r
# Make a ggplot with asthetic alpha for the variable DX_DBTS
ggplot(data = HANES) +
geom_point(mapping = aes(x = log(A1C), y = log(UACR), alpha=DX_DBTS))
```



or the shape of the points:

```
# Make a ggplot with asthetic shape for the variable DX_DBTS
ggplot(data = HANES) +
geom_point(mapping = aes(x = log(A1C), y = log(UACR), shape=DX_DBTS))
```



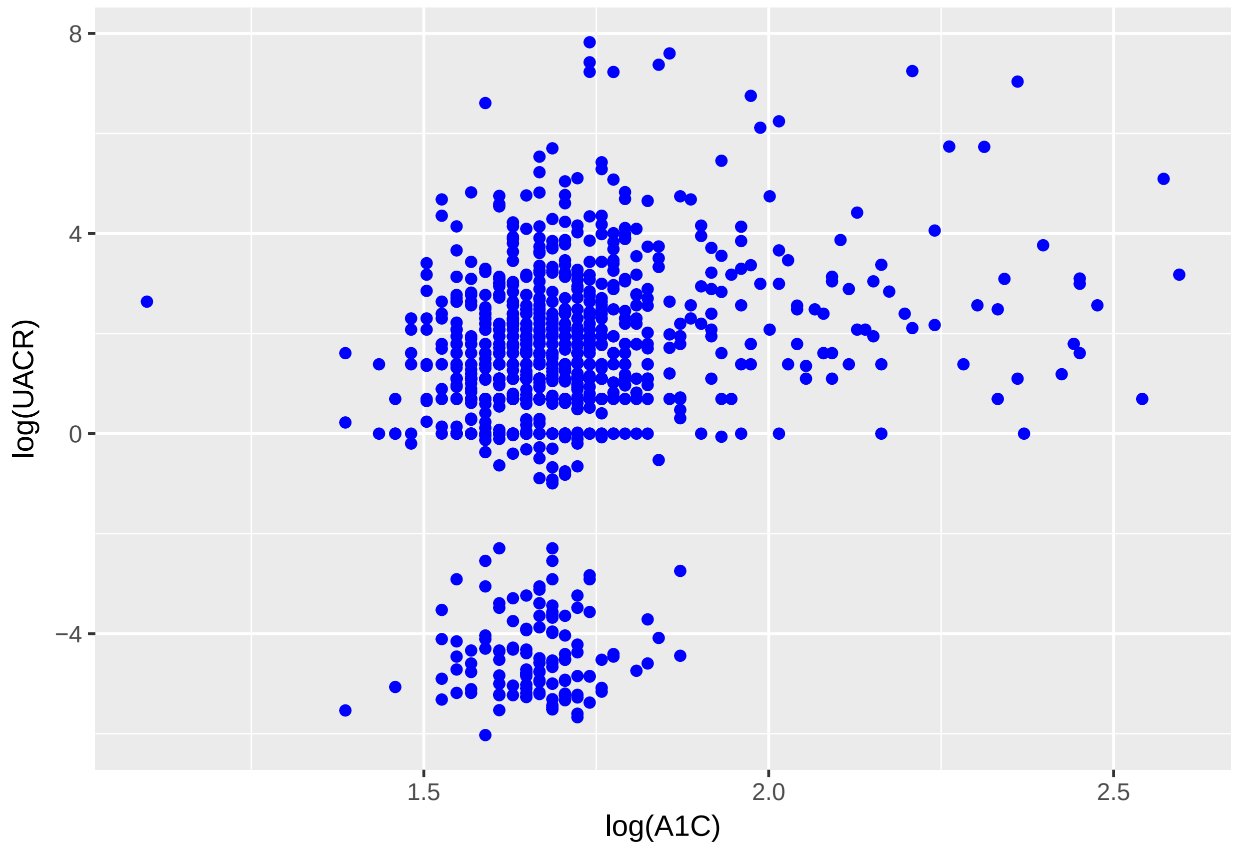
Note: ggplot2 will only use six shapes at a time. By default, additional groups will go unplotted when we use the shape aesthetic.

For each aesthetic, we use `aes()` to associate the name of the aesthetic with a variable to display. The `aes()` function gathers together each of the aesthetic mappings used by a layer and passes them to the layer's mapping argument. The syntax highlights a useful insight about x and y: the x and y locations of a point are themselves aesthetics, visual properties that we can map to variables to display information about the data.

Once we map an aesthetic, ggplot2 takes care of the rest. It selects a reasonable scale to use with the aesthetic, and it constructs a legend that explains the mapping between levels and values. For x and y aesthetics, ggplot2 does not create a legend, but it creates an axis line with tick marks and a label. The axis line acts as a legend; it explains the mapping between locations and values.

We can also set the aesthetic properties of our geom manually. For example, we can make all of the points in our plot red:

```
# Make a ggplot with asthetic shape for the variable DX_DBTS
ggplot(data = HANES) +
geom_point(mapping = aes(x = log(A1C), y = log(UACR)), color="blue")
```



Classwork/Homework:

1. What's gone wrong with this code? Why are the points not blue?

```
# Make a ggplot with asthetic shape for the variable DX_DBTS
ggplot(data = HANES) +
geom_point(mapping = aes(x = log(A1C), y = log(UACR), color="blue"))
```

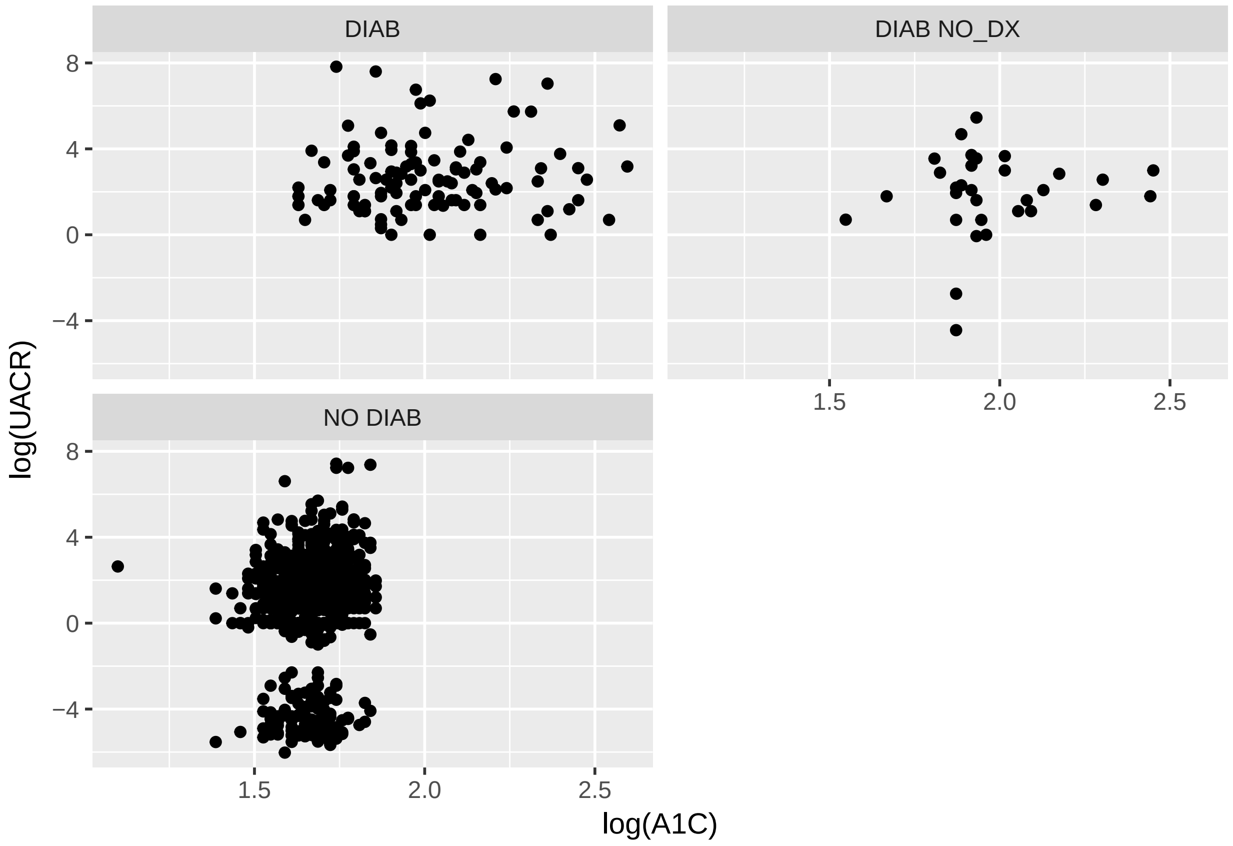
2. Which variables in HANES are categorical? Which variables are continuous? How can we see this information?
3. Map a continuous variable to color, size, and shape. How do these aesthetics behave differently for categorical vs. continuous variables?
4. What happens if we map the same variable to multiple aesthetics?
5. What does the stroke aesthetic do? What shapes does it work with? 6.What happens if we map an aesthetic to something other than a variable name, like aes(colour = A1C < 5)?

Facets

One way to add additional variables is with aesthetics. Another way, particularly useful for categorical variables, is to split plots into what are known as **facets**, subplots that each display one subset of the data.

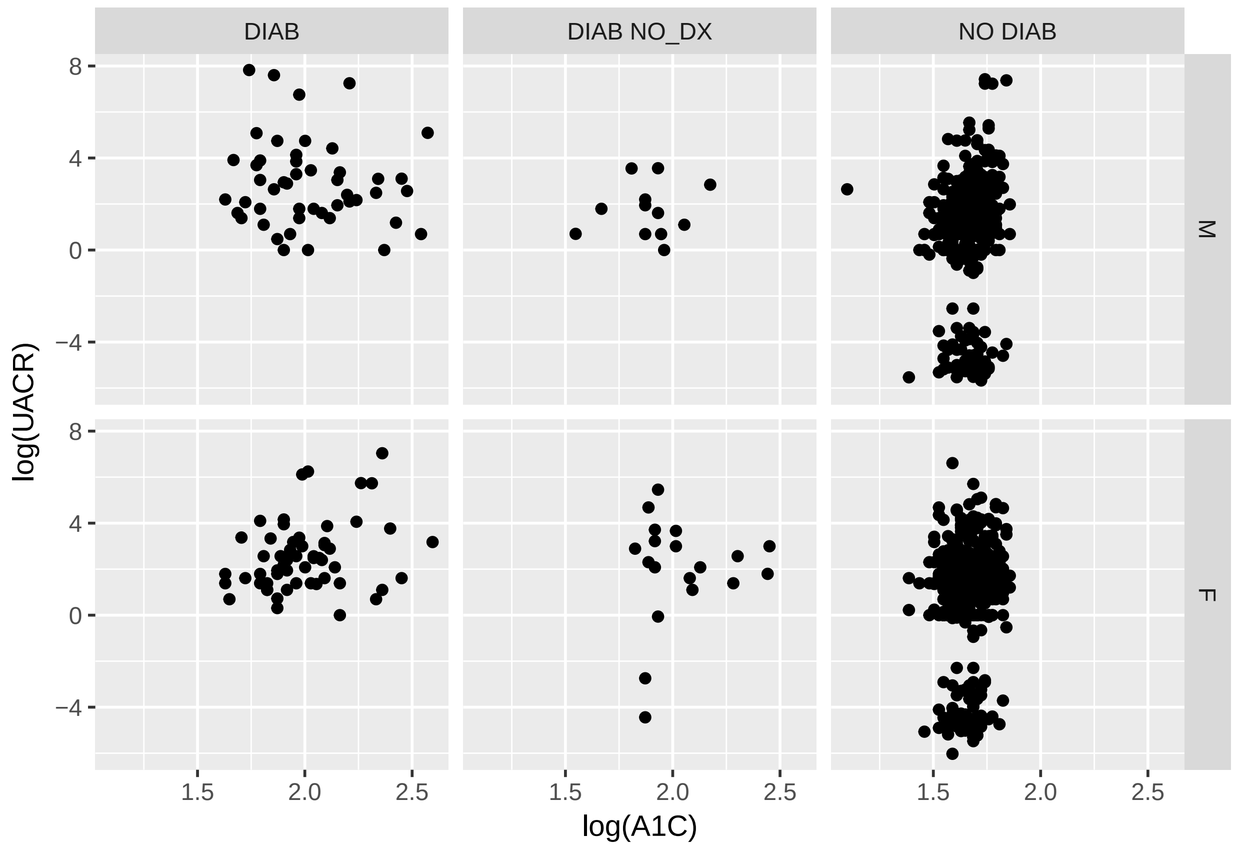
To facet a plot by a single variable, the function `facet_wrap()` is useful. The first argument of `facet_wrap()` should be a formula, which we create with ~ followed by a variable name (here "formula" is the name of a data structure in R, not a synonym for “equation”). The variable that we pass to `facet_wrap()` should be discrete.

```
# Make a ggplot with facets
ggplot(data = HANES) +
geom_point(mapping = aes(x = log(A1C), y = log(UACR))) +
facet_wrap(~ DX_DBTS, nrow = 2)
```



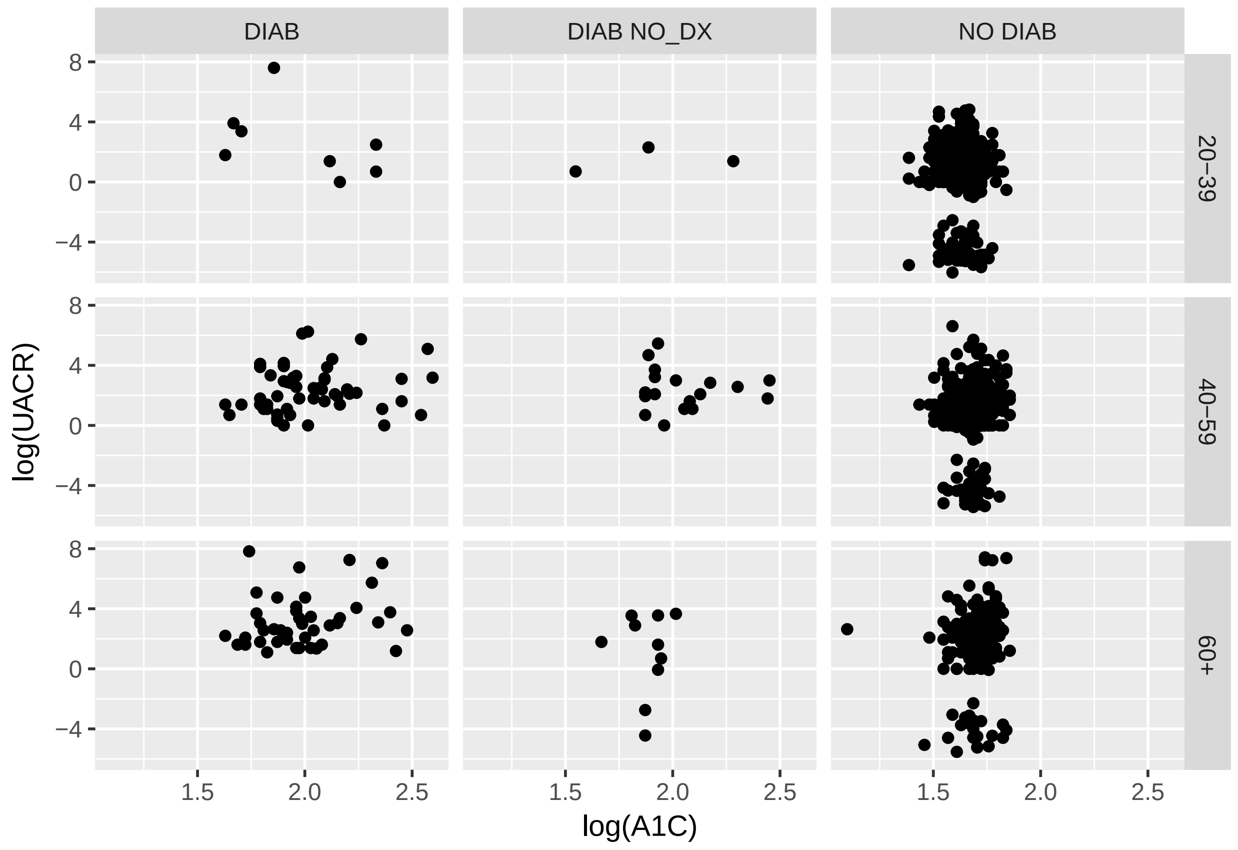
To facet our plot on the combination of two variables, we should add `facet_grid()` to our plot call. The first argument of `facet_grid()` is also a formula. This time the formula should contain two variable names separated by a ~.

```
# Make a ggplot with facet grid - GENDER vs DX_DBTS
ggplot(data = HANES) +
geom_point(mapping = aes(x = log(A1C), y = log(UACR))) +
facet_grid(GENDER ~ DX_DBTS)
```



We see that both males and females have almost the same distribution. However, plotting the facet grid with AGEGROUP vs DX_DBTS, reveals,

```
# Make a ggplot with facet grid - AGEGROUP vs DX_DBTS
ggplot(data = HANES) +
geom_point(mapping = aes(x = log(A1C), y = log(UACR))) +
facet_grid(AGEGROUP ~ DX_DBTS)
```



there are hardly people with diabetes in the age group 20-39, while there are lots of them with no diabetes who have small UACR values.

However, inspecting the age group 60+, we see plenty of them with diabetes and the population with no diabetes and small UACR values had seem to diminish, suggesting, low UACR may contribute to diabetes more than A1C. This is an example of **directed insight** obtained through visualization.

Classwork/Homework:

1. What happens if you facet on a continuous variable?
2. What plots does the following code make? What does `.` do?

```r
ggplot(data = HANES) +
geom_point(mapping = aes(x = log(HDL), y = log(CHOLESTEROLTOTAL))) +
facet_grid(AGEGROUP ~ .)

ggplot(data = HANES) +
geom_point(mapping = aes(x = log(UALBUMIN), y = log(GLUCOSE))) +
facet_grid(. ~ DX_DBTS)
```

3. Derive directed insights from the above plots.
4. Take the first faceted plot in this section:

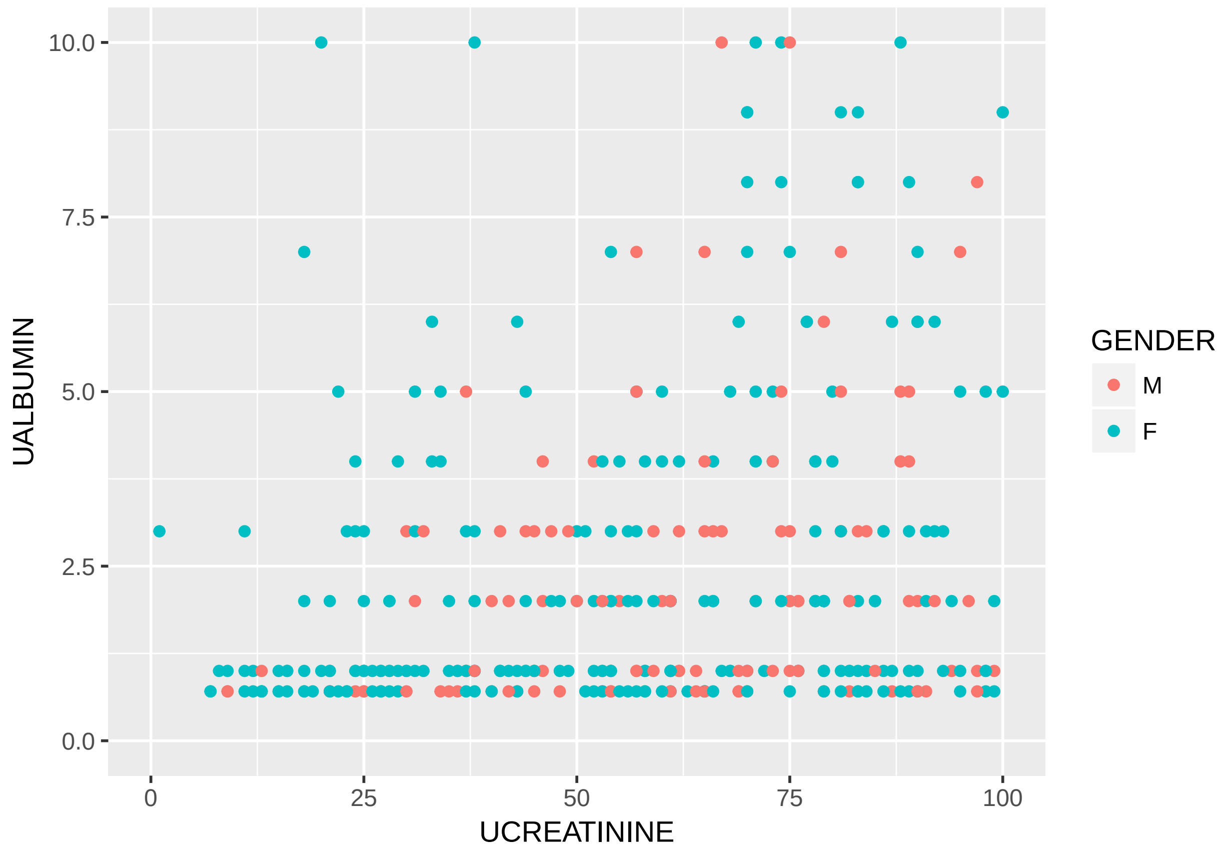
```r
# Make a ggplot with facets
ggplot(data = HANES) +
geom_point(mapping = aes(x = log(A1C), y = log(UACR))) +
facet_wrap(~ DX_DBTS, nrow = 2)
```

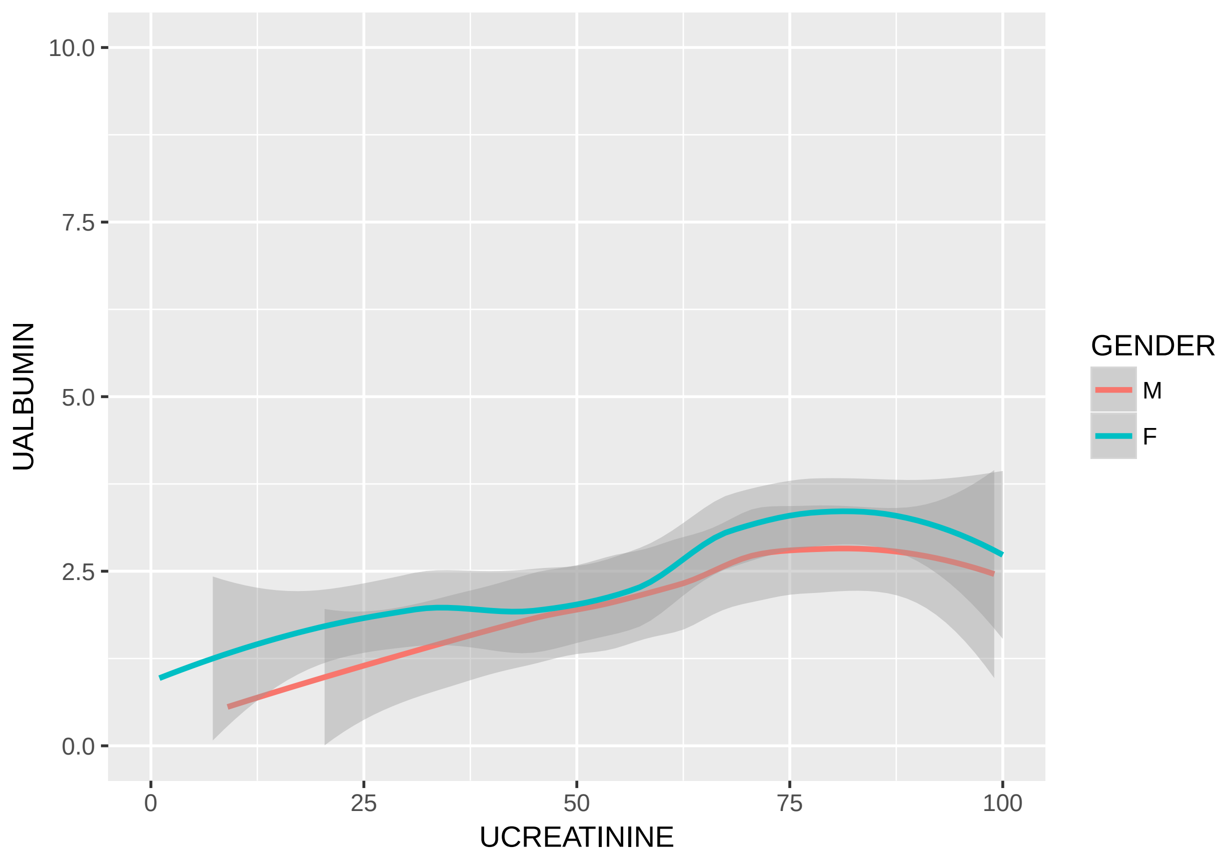
What are the advantages to using faceting instead of the colour aesthetic? What are the disadvantages? How might the balance change if you had a larger dataset?

5. Read `?facet_wrap`. What does nrow do? What does ncol do? What other options control the layout of the individual panels? Why doesn't `facet_grid()` have `nrow` and `ncol` argument?
6. When using `facet_grid()` you should usually put the variable with more unique levels in the columns. Why?

Geometric objects

How are these two plots similar?





Both plots contain the same x variable, the same y variable, and both describe the same data. But the plots are not identical. Each plot uses a different visual object to represent the data. In ggplot2 syntax, they're called **geoms**.

A **geom** is the geometrical object that a plot uses to represent data.

Plots are often described by the type of geom that the plot uses. For example, bar charts use bar geoms, line charts use line geoms, boxplots use boxplot geoms, and so on. Scatterplots break the trend; they use the point geom. We can use different geoms to plot the same data. The plot on top uses the point geom, and the plot in the bottom uses the smooth geom, a smooth line fitted to the data.

Here is the point geom code:

```
# Plot point geom (scatter plot) UCREATININE vs UALBUMIN, coloring GENDER
ggplot(data = HANES) +
geom_point(mapping = aes(x = UCREATININE, y = UALBUMIN, color=GENDER)) +
xlim(c(0,100)) + ylim(c(0,10))
```

And here is the line geom code:

```
# Plot smooth geom UCREATININE vs UALBUMIN, coloring GENDER
ggplot(data = HANES) +
geom_smooth(mapping = aes(x = UCREATININE, y = UALBUMIN, color=GENDER)) +
xlim(c(0,100)) + ylim(c(0,10))
```

Classwork/Homework: Try `geom_smmoth` on different variables in HANES and MIMIC3 data frames.

Every geom function in ggplot2 takes a mapping argument.

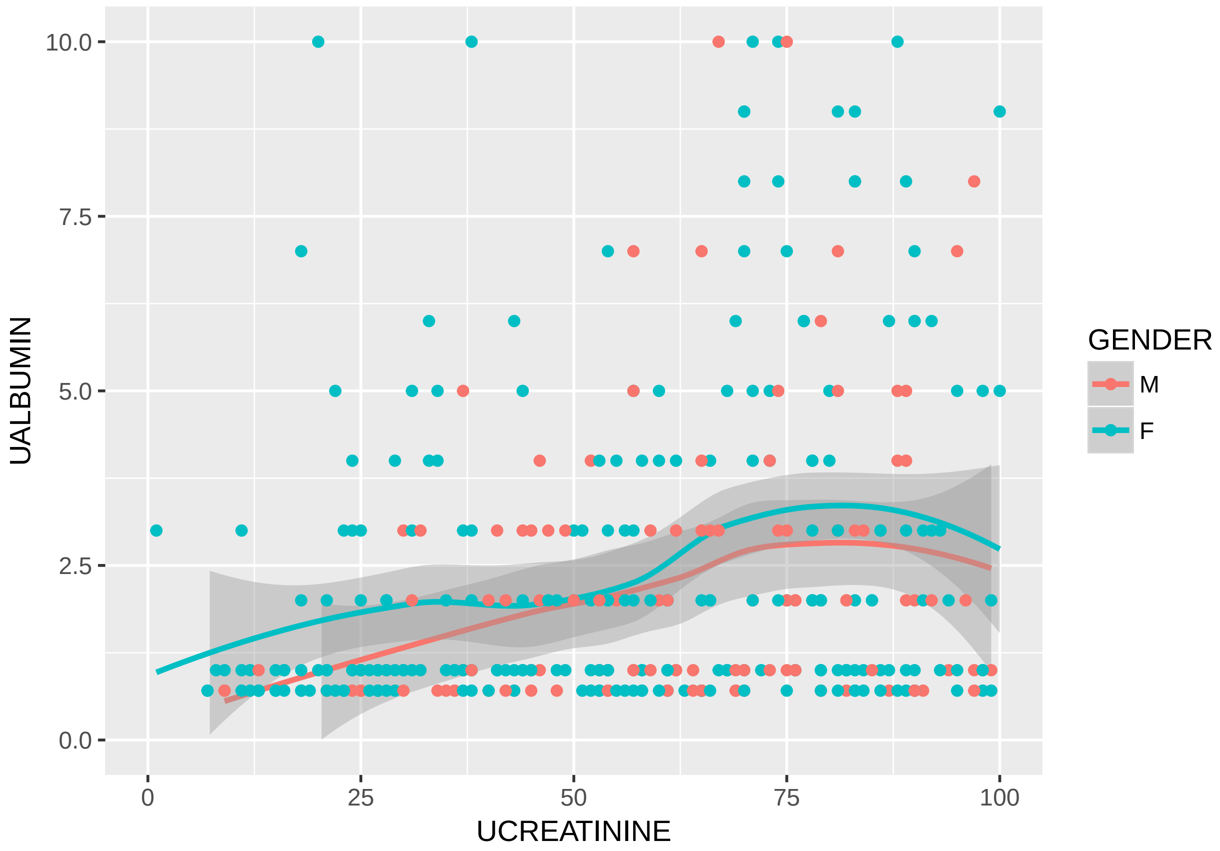
However, not every aesthetic works with every geom. We could set the shape of a point, but we cannot set the "shape" of a line.

On the other hand, we could set the color/linetype of a line.

`geom_smooth()` will draw a different line, with a different color for male and female, for each unique value of the variable that we map to color. One can also specify `linetype` instead of color.

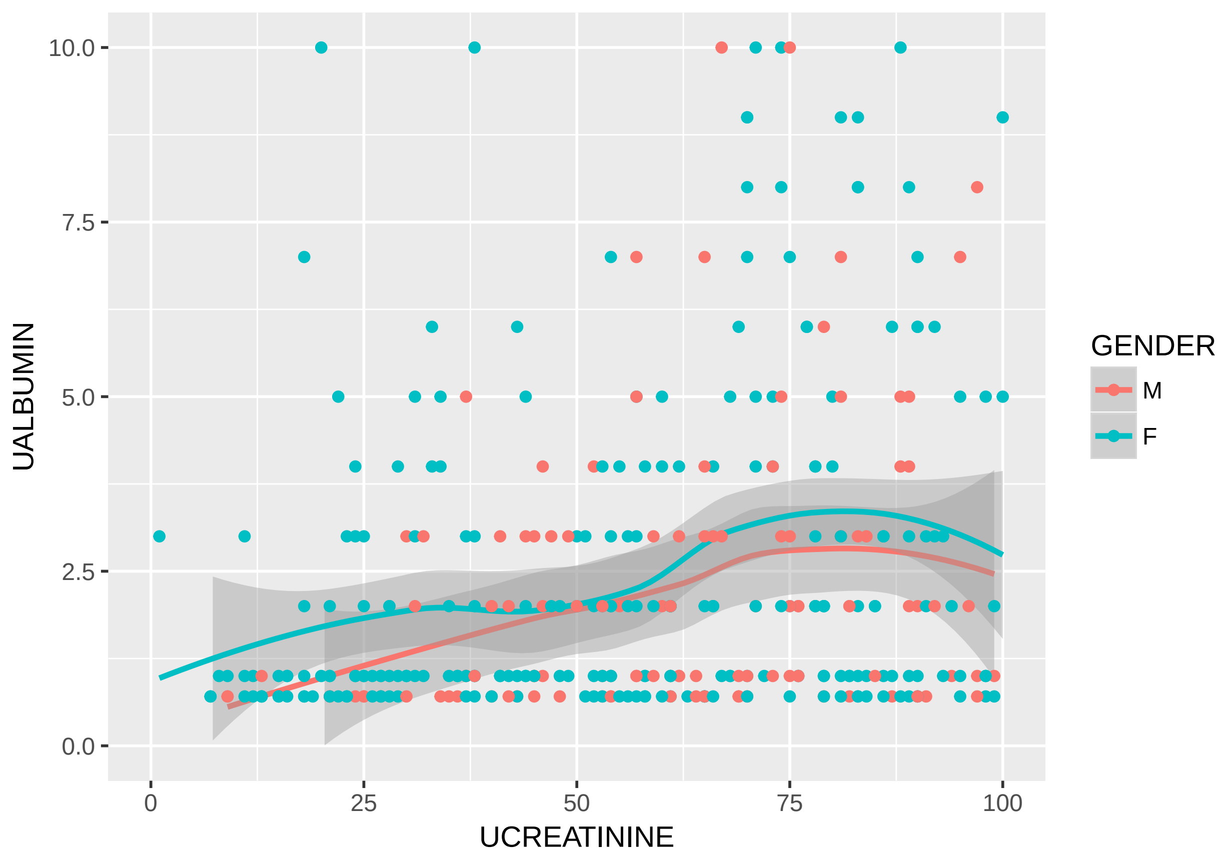
In fact, we can over lay two geoms on the **same** plot - super cool, isn't it (compare it with `abline()` function):

```r
# Plot smooth and point geom in the same plot
ggplot(data = HANES) +
geom_smooth(mapping = aes(x = UCREATININE, y = UALBUMIN, color=GENDER)) +
xlim(c(0,100)) + ylim(c(0,10)) +
geom_point(mapping = aes(x = UCREATININE, y = UALBUMIN, color=GENDER))
```



Note: This introduces some duplication in our code. Imagine if we wanted to change the y-axis to display UACR values instead of UALBUMIN. We need to change the variable in two places, and we might forget to update one. We can avoid this type of repetition by passing a set of mappings to the `ggplot()` function. ggplot2 will treat these mappings as global mappings that apply to each geom in the graph. In other words, this code will produce the same plot as the previous code:

```r
# Plot smooth and point geom in the same plot
ggplot(data = HANES, mapping = aes(x = UCREATININE, y = UALBUMIN, color=GENDER)) +
xlim(c(0,100)) + ylim(c(0,10)) +
geom_smooth() +
geom_point()
```

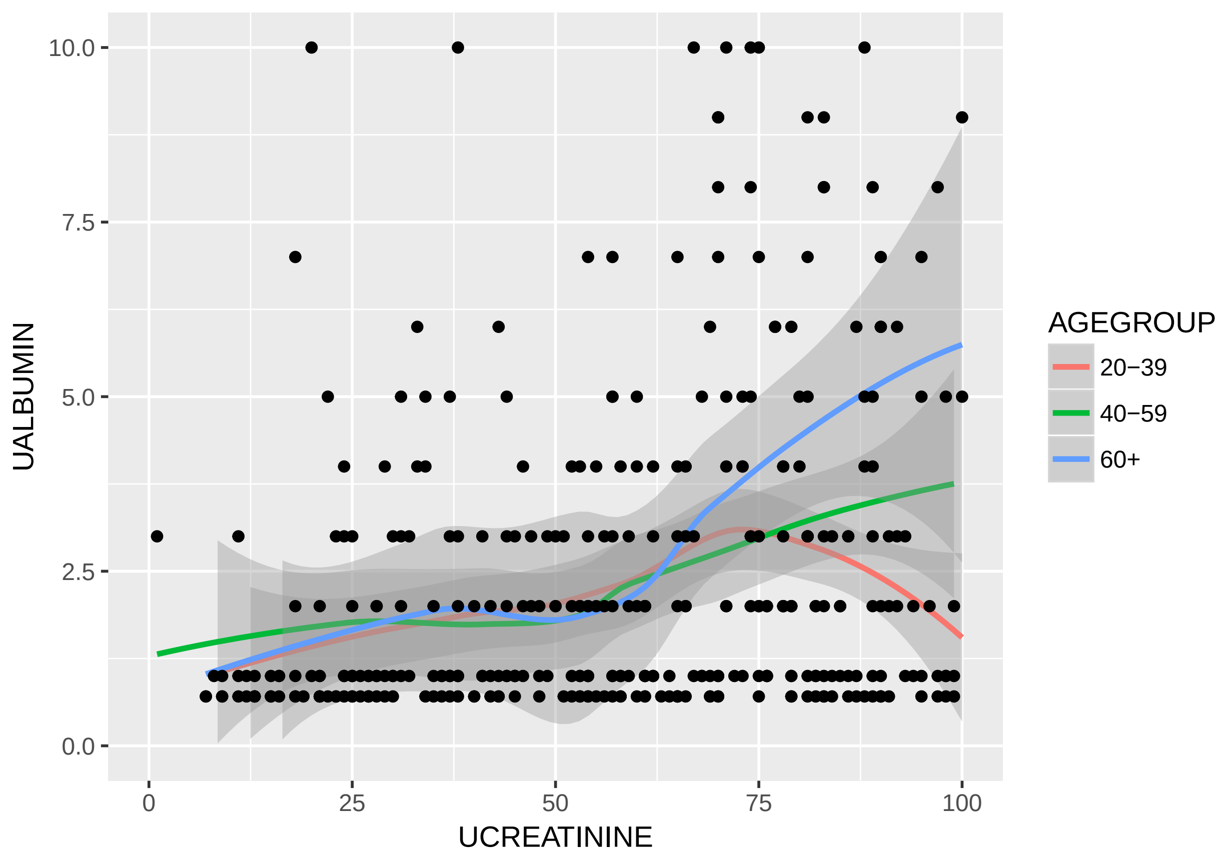


If we place mappings in a geom function, ggplot2 will treat them as local mappings for the layer.

It will use these mappings to extend or overwrite the global mappings for that layer only.

This makes it possible to display different aesthetics in different layers.

```
# Plot smooth and point geom in the same plot
ggplot(data = HANES, mapping = aes(x = UCREATININE, y = UALBUMIN)) +
xlim(c(0,100)) + ylim(c(0,10)) +
geom_smooth(mapping = aes(color = AGEGROUP)) +
geom_point()
```



Note: One can use this idea for plotting *different* data to different layers. For example, the geom `geom_point` can be specified to select DX_DBTS variable consisting only diabetic people, for instance.

ggplot2 provides over 30 geoms, and extension packages provide even more (see https://www.ggplot2-exts.org for a sampling). The best way to get a comprehensive overview is the ggplot2 cheatsheet, which you can find at http://rstudio.com/cheatsheets. To learn more about any single geom, use help: ?geom_smooth.

Classwork/Homework:

1. What does the `group` argument in aesthetics in the line geom do? Try it in an example and report the result.
2. What geom would you use to draw a line chart? A boxplot? A histogram? An area chart?
3. What does `show.legend = FALSE` do? What happens if you remove it?
4. What does the `se` argument to `geom_smooth()` do?
5. Read sections 3.7, 3.8 in the book R for Data Science. For each code example (ignore code in exercises) given in these sections, you need to provide similar examples using HANES/MIMIC3 data sets. Also, change the documentation to reflect your demo.

Selected materials and references

R for Data Science - Explore Part