Creating Publication-Quality Graphics with ggplot2

- To be able to use ggplot2 to generate publication-quality graphics.
- To apply geometry, aesthetic, and statistics layers to a ggplot plot.
- To manipulate the aesthetics of a plot using different colors, shapes, and lines.
- To improve data visualization through transforming scales and paneling by group.
- To save a plot created with ggplot to disk.

- How can I create publication-quality graphics in R?

This code is cribbed from our earlier discussion about dplyr / data munging (13-dplyr.pdf).

Plotting our data is one of the best ways to quickly explore it and the various relationships between variables.

There are three main plotting systems in R, the base plotting system, the lattice package, and the ggplot2 package.

Today we'll be learning about the ggplot2 package, because it is the most effective for creating publication-quality graphics.

ggplot2 is built on the grammar of graphics, the idea that any plot can be built from the same set of components: a **data set**, **mapping aesthetics**, and graphical **layers**:

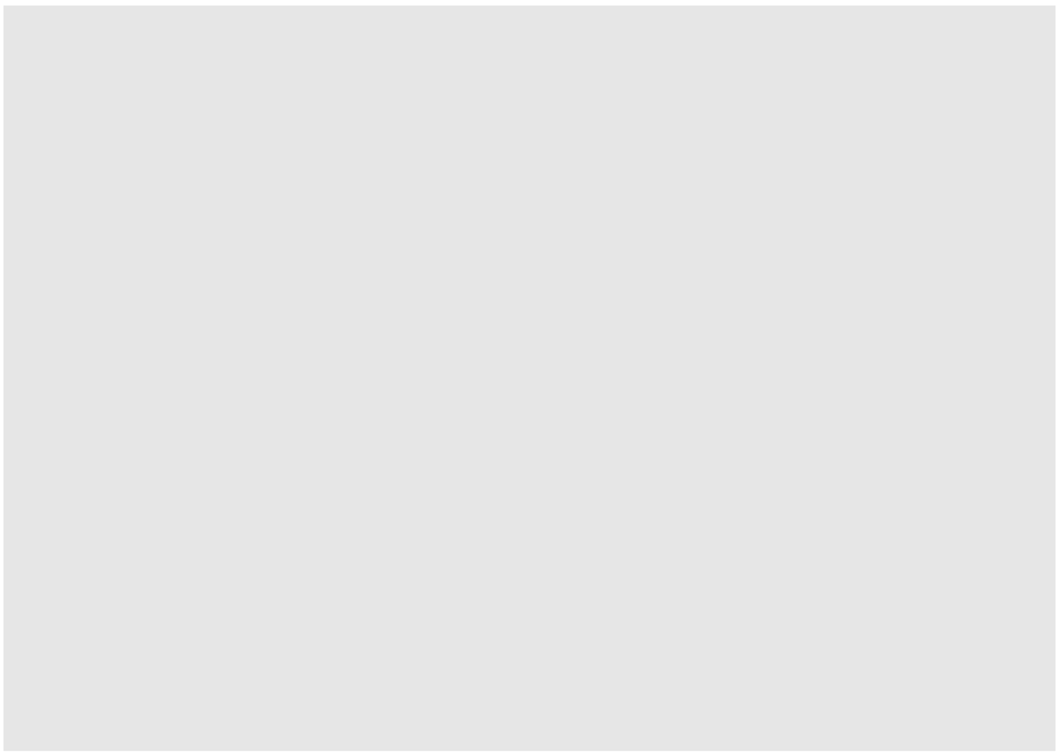
- **Data sets** are the data that you, the user, provide.

- **Mapping aesthetics** are what connect the data to the graphics. They tell ggplot2 how to use your data to affect how the graph looks, such as changing what is plotted on the X or Y axis, or the size or color of different data points.

- **Layers** are the actual graphical output from ggplot2. Layers determine what kinds of plot are shown (scatterplot, histogram, etc.), the coordinate system used (rectangular, polar, others), and other important aspects of the plot. The idea of layers of graphics may be familiar to you if you have used image editing programs like Photoshop, Illustrator, or Inkscape.

Let's start off building an example using the gapminder data from earlier. The most basic function is `ggplot`, which lets R know that we're creating a new plot. Any of the arguments we give the `ggplot` function are the *global* options for the plot: they apply to all layers on the plot.

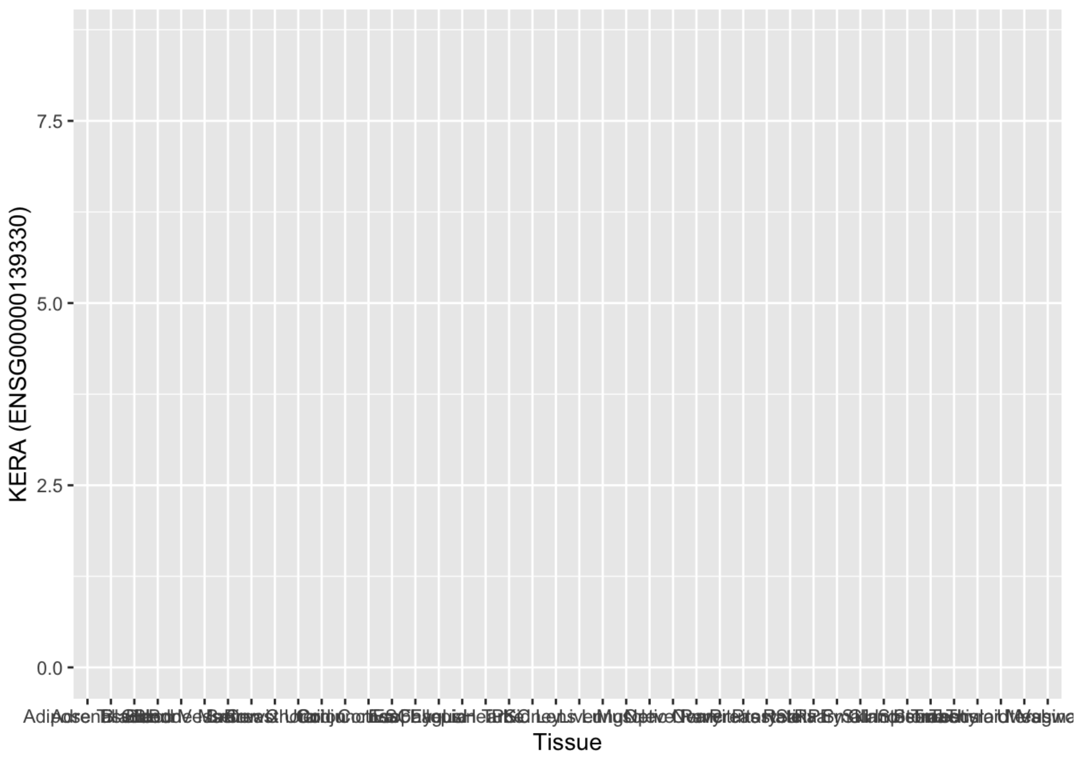
```
library("ggplot2")
ggplot(data = eiad_meta_exp)
```



Here we called `ggplot` and told it what data we want to show on our figure. This is not enough information for `ggplot` to actually draw anything. It only creates a blank slate for other elements to be added to.

Now we're going to add in the **mapping aesthetics** using the `aes` function. `aes` tells `ggplot` how variables in the **data** map to *aesthetic* properties of the figure, such as which columns of the data should be used for the **x** and **y** locations.

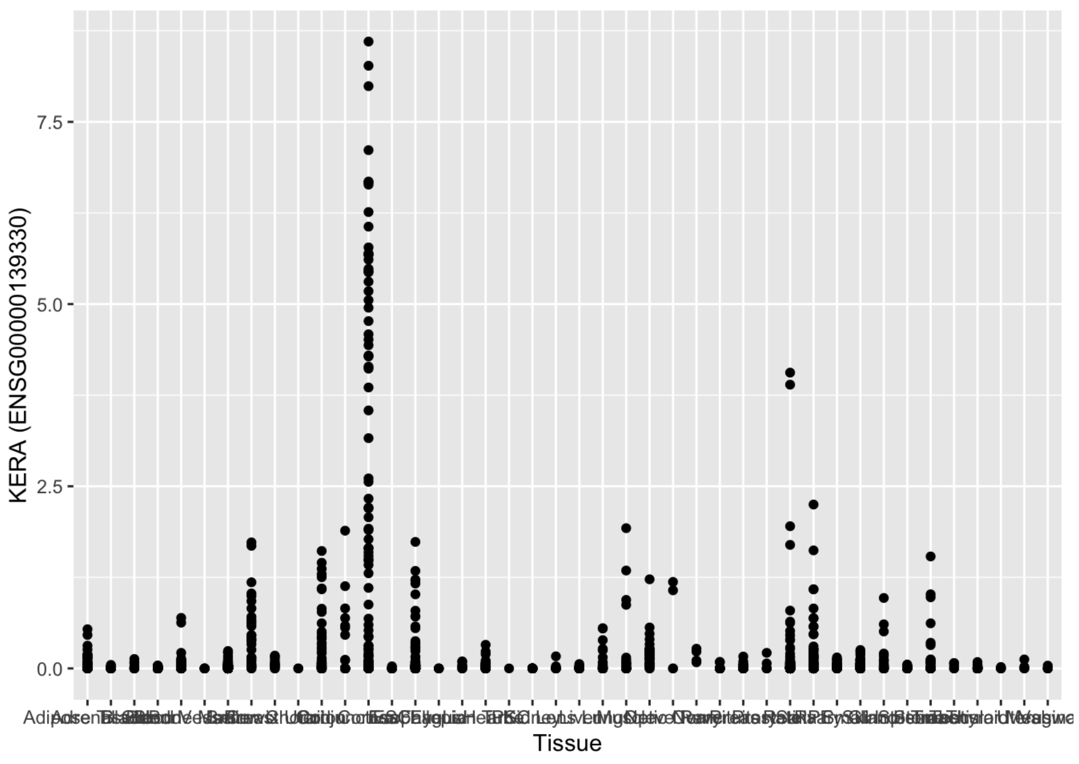
```
ggplot(data = eiad_meta_exp, mapping = aes(x = Tissue, y = `KERA (ENSG00000139330)`))
```



Here we told `ggplot` we want to plot the “Tissue” column of the eiad_meta_exp data frame on the x-axis, and the “KERA (ENSG00000139330)” column on the y-axis. Notice that we didn’t need to explicitly pass `aes` these columns (e.g. `x = eiad_meta_exp[, "KERA (ENSG00000139330)"]`), this is because `ggplot` is smart enough to know to look in the **data** for that column!

The final part of making our plot is to tell `ggplot` how we want to visually represent the data. We do this by adding a new **layer** to the plot using one of the **geom** functions.

```
ggplot(data = eiad_meta_exp, mapping = aes(x = Tissue, y = `KERA (ENSG00000139330)`)) + geom_point()
```



Here we used `geom_point`, which tells `ggplot` we want to visually represent the relationship between **x** and **y** as a scatterplot of points.

:: challenge

Challenge 1

Modify the example so that the figure shows how Rho expression changes within sub_tissues of only eye tissues

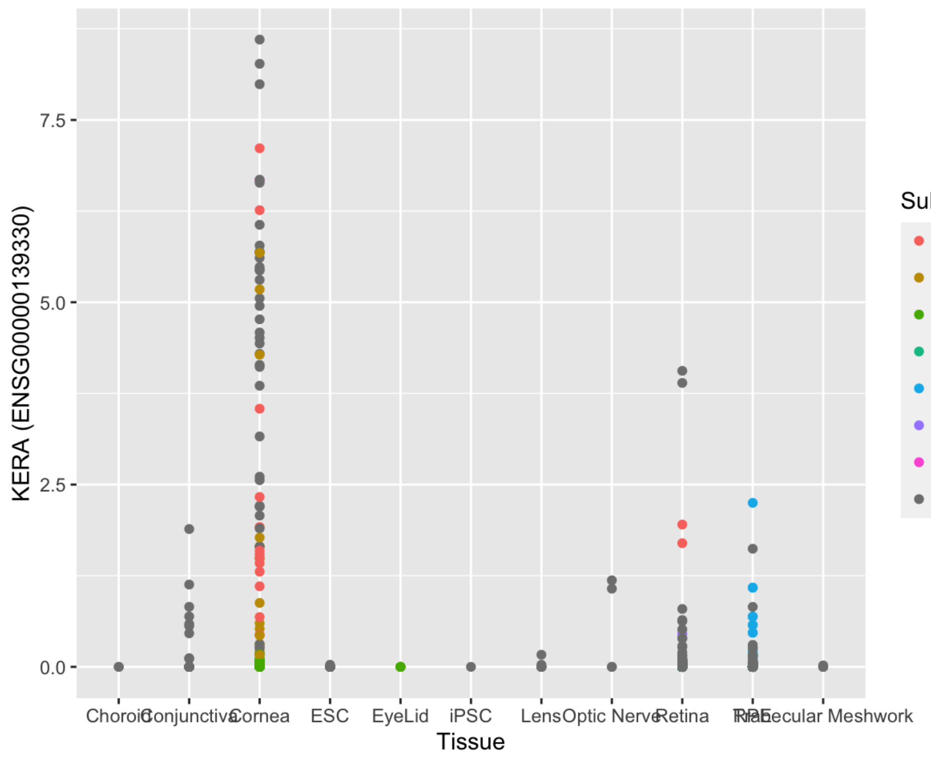
```
ggplot(data = eiad_meta_exp, mapping = aes(x = Tissue, y = `KERA (ENSG00000139330)`)) + geom_point()
```

Hint: You can pipe data into ggplot like so: `your_data %>% ggplot(aes())`

Solution to challenge 1

Here is one possible solution:

```
eiad_meta_exp %>% filter(Cohort == 'Eye') %>%
  ggplot(mapping = aes(x = Tissue, y = `KERA (ENSG00000139330)`, color = Sub_Tissue)) +
  geom_point()
```



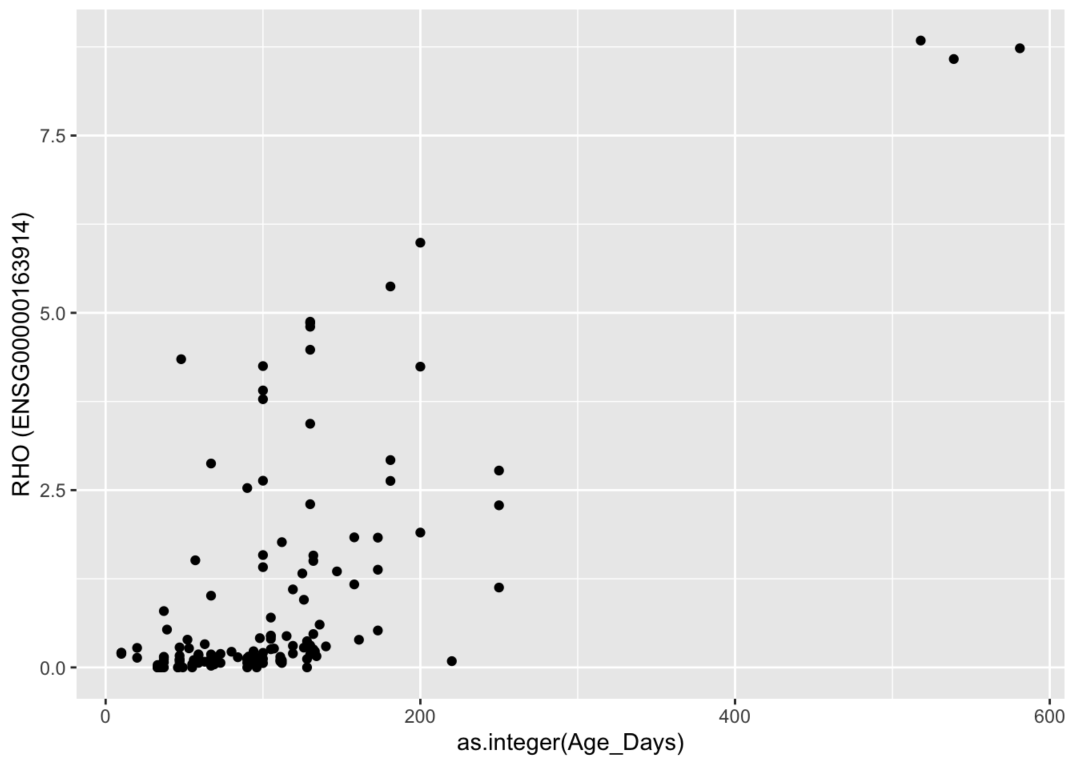
Binned scatterplot of life expectancy versus year showing how life expectancy has increased over time

New Plot

Let's now use our dplyr knowledge and look at gene expression changes over time in a small number of early development tissues.

```
eiad_meta_exp %>%
  mutate(Age_Days = as.integer(Age_Days)) %>%
  filter(Tissue == 'Retina', !is.na(Age_Days)) %>%
  ggplot(aes(x=as.integer(Age_Days),
             y=`RHO (ENSG00000163914)`)) +
  geom_point()
```

```
## Warning: There was 1 warning in `mutate()`.
## ℹ In argument: `Age_Days = as.integer(Age_Days)`.
## Caused by warning:
## ! NAs introduced by coercion
```

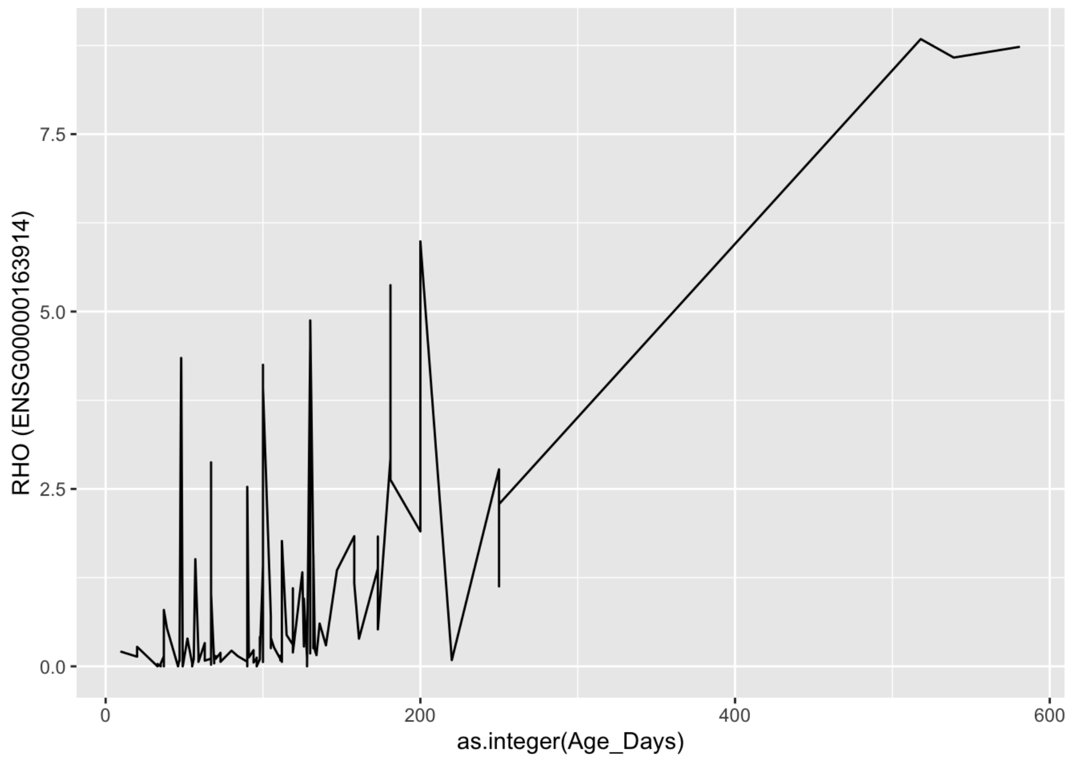


## Layers

Perhaps a scatterplot probably isn't the best for visualizing change over time. Instead, let's tell `ggplot` to visualize the data as a line plot:

```
eiad_meta_exp %>%
  mutate(Age_Days = as.integer(Age_Days)) %>%
  filter(Tissue == 'Retina', !is.na(Age_Days)) %>%
  ggplot(aes(x=as.integer(Age_Days),
             y=`RHO (ENSG00000163914)`)) +
  geom_line()
```

```
## Warning: There was 1 warning in `mutate()`.
## ℹ In argument: `Age_Days = as.integer(Age_Days)`.
## Caused by warning:
## ! NAs introduced by coercion
```



Instead of adding a `geom_point` layer, we've added a `geom_line` layer.

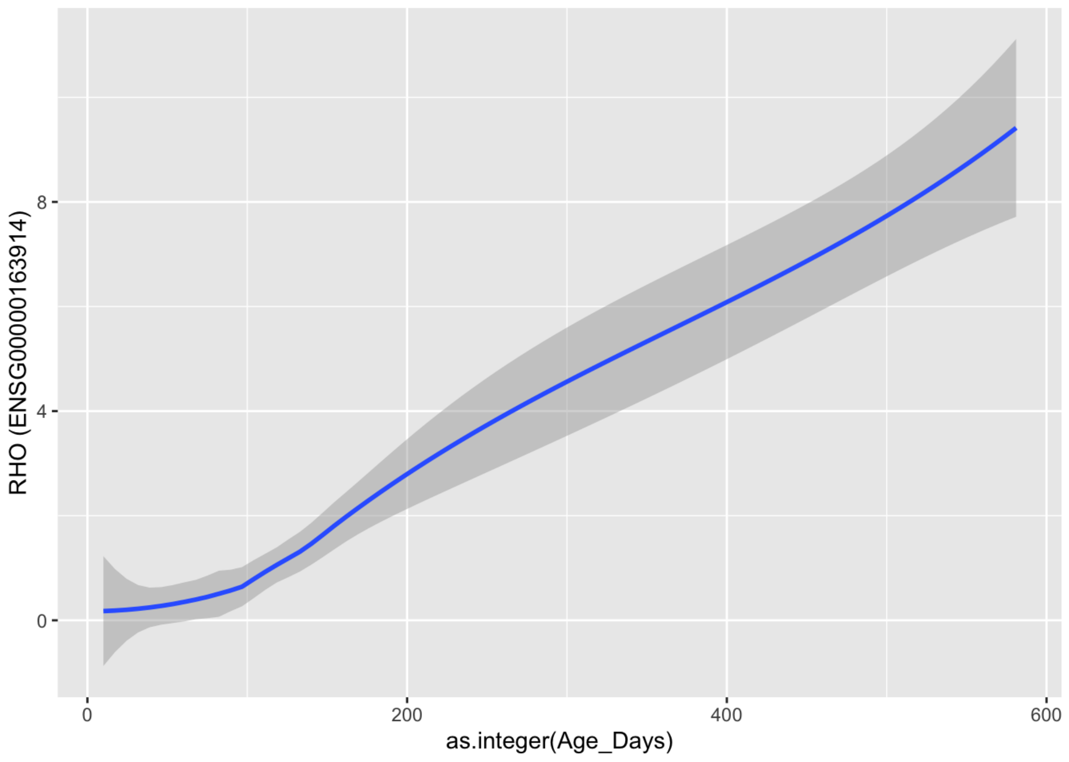
However, the result doesn't look quite as we might have expected: it seems to be jumping around a lot. What we actually want is a line that summarizes (or smooths) the trends.

Statistical Functions

```
eiad_meta_exp %>%
  mutate(Age_Days = as.integer(Age_Days)) %>%
  filter(Tissue == 'Retina', !is.na(Age_Days)) %>%
  ggplot(aes(x=as.integer(Age_Days),
             y=`RHO (ENSG00000163914)`)) +
  geom_smooth()
```

```
## Warning: There was 1 warning in `mutate()`.
## ℹ In argument: `Age_Days = as.integer(Age_Days)`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

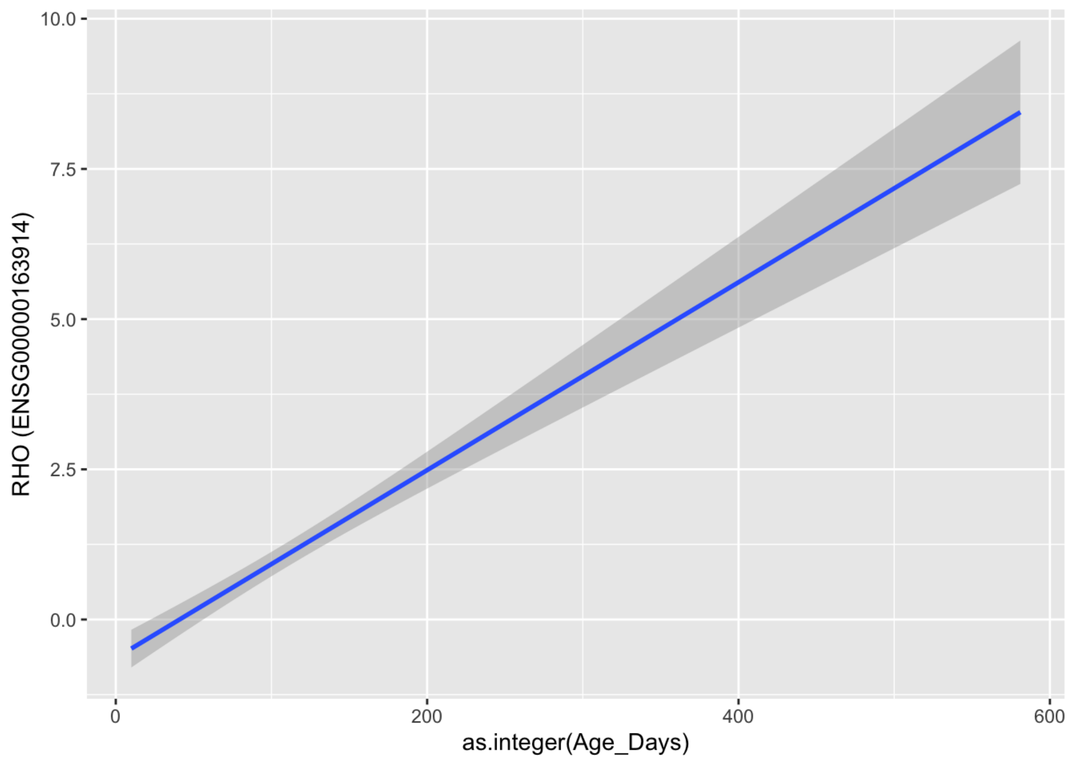


Now perhaps we would like to see the linear trend. What is plotted in a “loess” which is a smoothed curve across the data.

```
eiad_meta_exp %>%
  mutate(Age_Days = as.integer(Age_Days)) %>%
  filter(Tissue == 'Retina', !is.na(Age_Days)) %>%
  ggplot(aes(x=as.integer(Age_Days),
             y=`RHO (ENSG00000163914)`)) +
  geom_smooth(method = 'lm')
```

```
## Warning: There was 1 warning in `mutate()`.
## ℹ In argument: `Age_Days = as.integer(Age_Days)`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



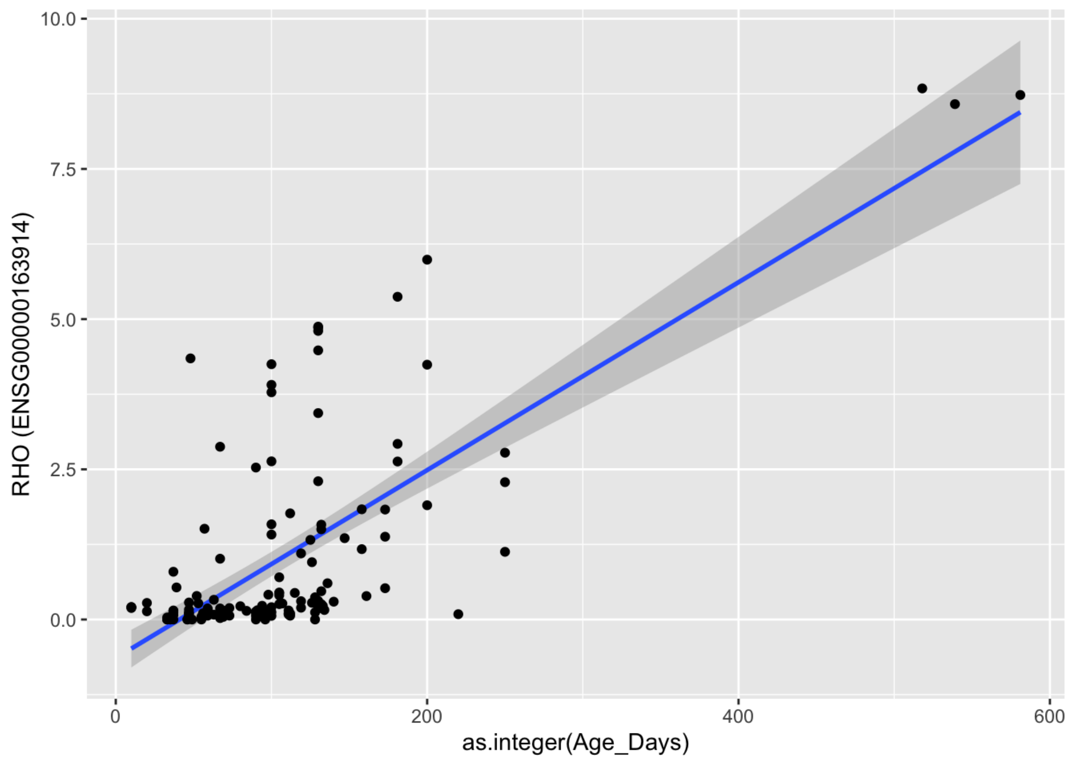
Multiple Layers

How do we make the dots come back….any ideas?

```
eiad_meta_exp %>%
  mutate(Age_Days = as.integer(Age_Days)) %>%
  filter(Tissue == 'Retina', !is.na(Age_Days)) %>%
  ggplot(aes(x=as.integer(Age_Days),
             y=`RHO (ENSG00000163914)`)) +
  geom_smooth(method = 'lm') +
  geom_point()
```

```
## Warning: There was 1 warning in `mutate()`.
## ℹ In argument: `Age_Days = as.integer(Age_Days)`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



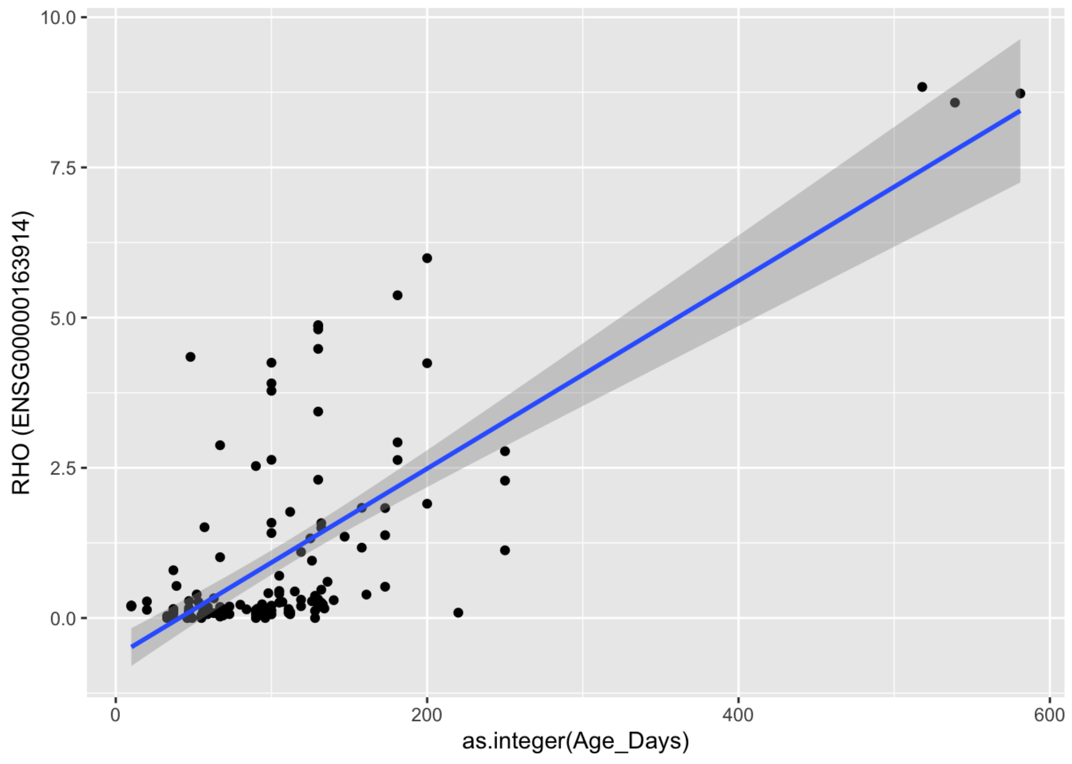
## Order of operations

How do we get the line in FRONT of the dots?????

```
eiad_meta_exp %>%
  mutate(Age_Days = as.integer(Age_Days)) %>%
  filter(Tissue == 'Retina', !is.na(Age_Days)) %>%
  ggplot(aes(x=as.integer(Age_Days),
             y=`RHO (ENSG00000163914)`)) +
  geom_point() +
  geom_smooth(method = 'lm')
```

```
## Warning: There was 1 warning in `mutate()`.
## ℹ In argument: `Age_Days = as.integer(Age_Days)`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



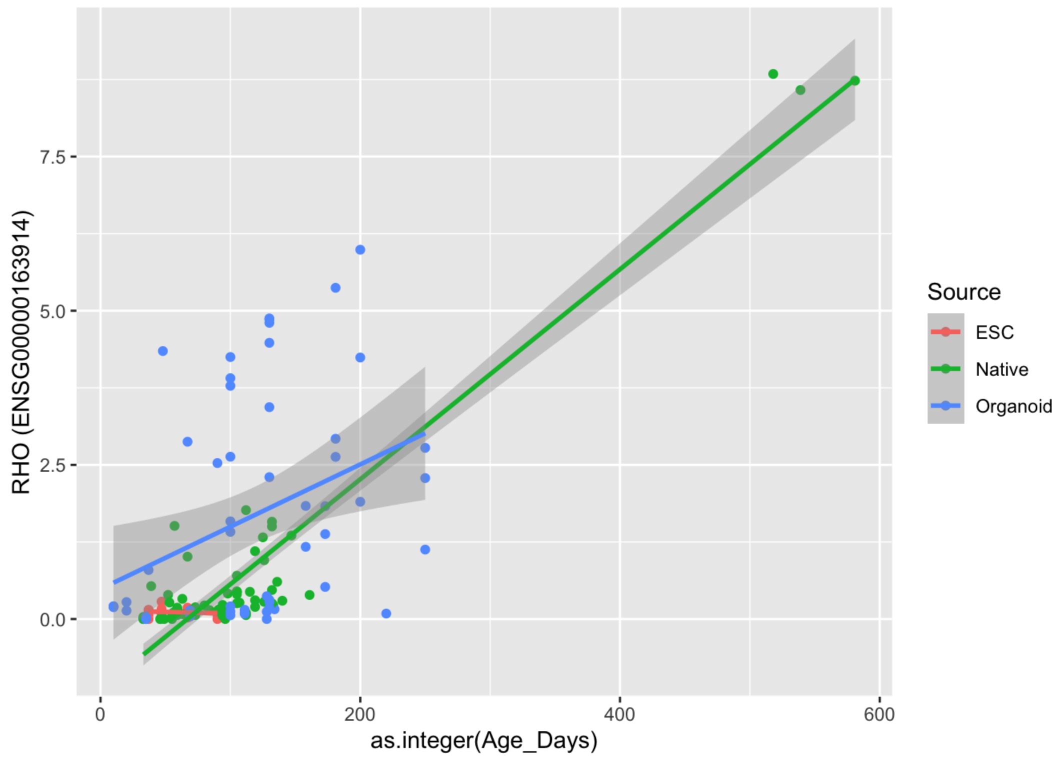
Color aesthetics

Now we have different KINDS of retina here. More specifically we have three sources of retina (ESC, organoid, primary tissue). How do we separate the lines for each group?

```
eiad_meta_exp %>%
  mutate(Age_Days = as.integer(Age_Days)) %>%
  filter(Tissue == 'Retina', !is.na(Age_Days)) %>%
  ggplot(aes(x=as.integer(Age_Days),
             y=`RHO (ENSG00000163914)`,
             color = Source)) +
  geom_point() + geom_smooth(method = 'lm')
```

```
## Warning: There was 1 warning in `mutate()`.
## ℹ In argument: `Age_Days = as.integer(Age_Days)`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



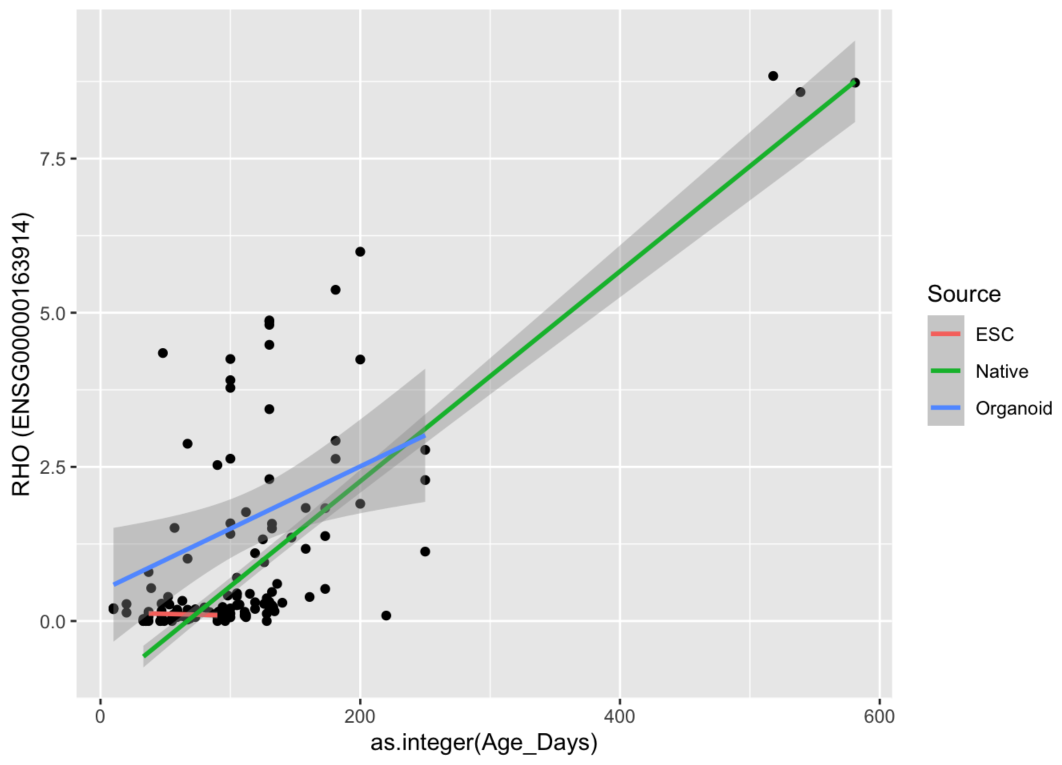
# Where the aes goes is very important

This is visually a lot to look at…what if you want to just color the lines?

```
eiad_meta_exp %>%
  mutate(Age_Days = as.integer(Age_Days)) %>%
  filter(Tissue == 'Retina', !is.na(Age_Days)) %>%
  ggplot(aes(x=as.integer(Age_Days),
             y=`RHO (ENSG00000163914)`)) +
  geom_point() +
  geom_smooth(aes(color = Source), method = 'lm')
```

```
## Warning: There was 1 warning in `mutate()`.
## ℹ In argument: `Age_Days = as.integer(Age_Days)`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



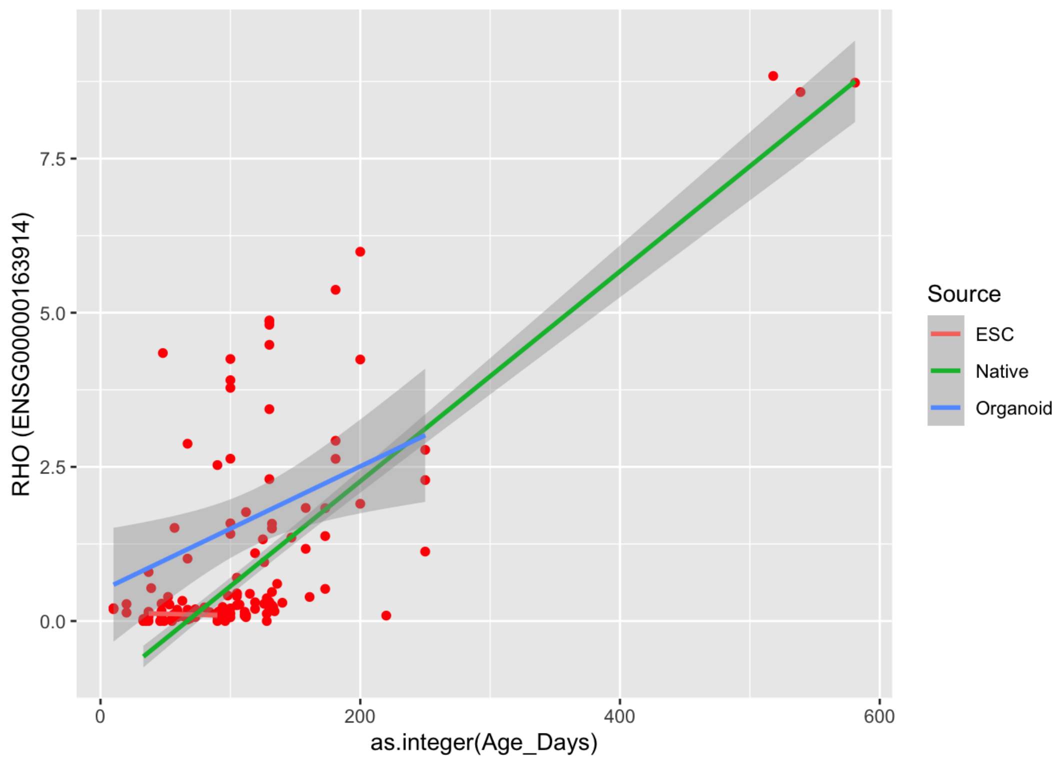
Tip: Setting an aesthetic to a value instead of a mapping

So far, we've seen how to use an aesthetic (such as **color**) as a *mapping* to a variable in the data. For example, when we use `geom_line(mapping = aes(color=Age_Source))`, ggplot will give a different color to each continent. But what if we want to change the color of all lines to blue? You may think that `geom_line(mapping = aes(color="blue"))` should work, but it doesn't. Since we don't want to create a mapping to a specific variable, we can move the color specification outside of the `aes()` function, like this: `geom_line(color="blue")`.

```
eiad_meta_exp %>%
  mutate(Age_Days = as.integer(Age_Days)) %>%
  filter(Tissue == 'Retina', !is.na(Age_Days)) %>%
  ggplot(aes(x=as.integer(Age_Days),
             y=`RHO (ENSG00000163914)`)) +
  geom_point(color = 'red') +
  geom_smooth(aes(color = Source), method = 'lm')
```

```
## Warning: There was 1 warning in `mutate()`.
## ℹ In argument: `Age_Days = as.integer(Age_Days)`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

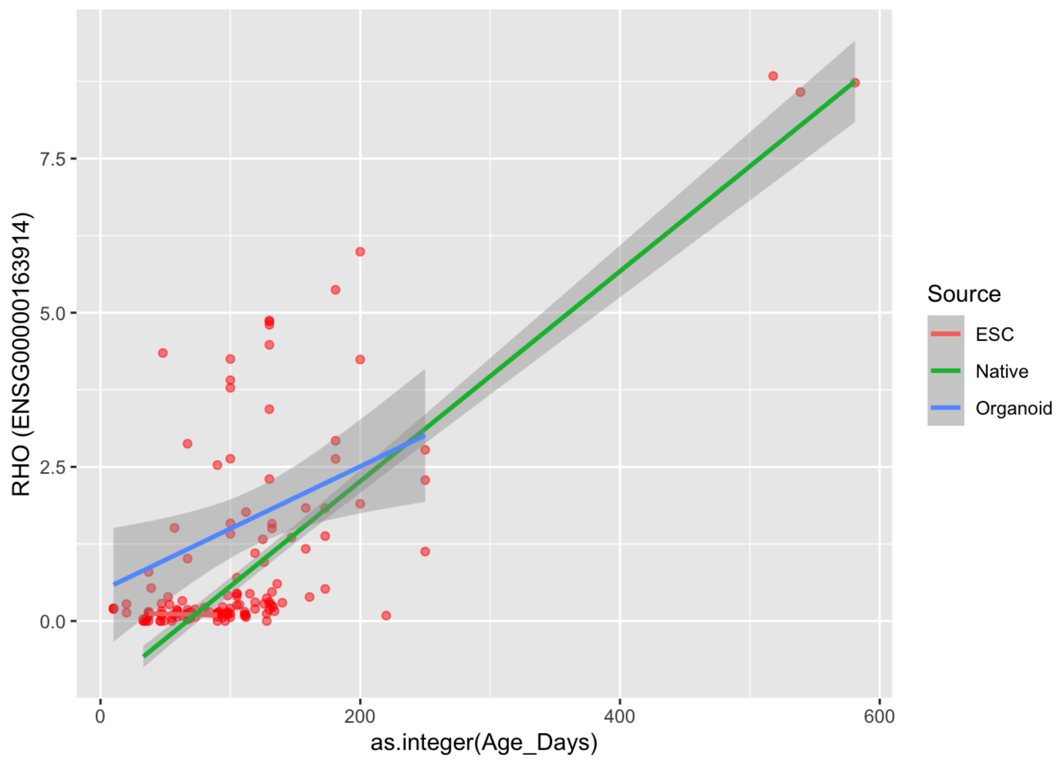


The red dots are REALLY distracting though. We can also modify the transparency of the points, using the *alpha* function, which is especially helpful when you have a large amount of data which is very clustered.

```
eiad_meta_exp %>%
  mutate(Age_Days = as.integer(Age_Days)) %>%
  filter(Tissue == 'Retina', !is.na(Age_Days)) %>%
  ggplot(aes(x=as.integer(Age_Days),
             y=`RHO (ENSG00000163914)`)) +
  geom_point(color = 'red', alpha =0.5) +
  geom_smooth(aes(color = Source), method = 'lm')
```

```
## Warning: There was 1 warning in `mutate()`.
## ℹ In argument: `Age_Days = as.integer(Age_Days)`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Scatterplot of GDP vs life expectancy showing logarithmic x-axis data spread

Tip Reminder: Setting an aesthetic to a value instead of a mapping

Notice that we used `geom_point(alpha = 0.5)`. As the previous tip mentioned, using a setting outside of the `aes()` function will cause this value to be used for all points, which is what we want in this case. But just like any other aesthetic setting, *alpha* can also be mapped to a variable in the data. For example, we can give a different transparency to each continent with `geom_point(mapping = aes(alpha = continent))`.

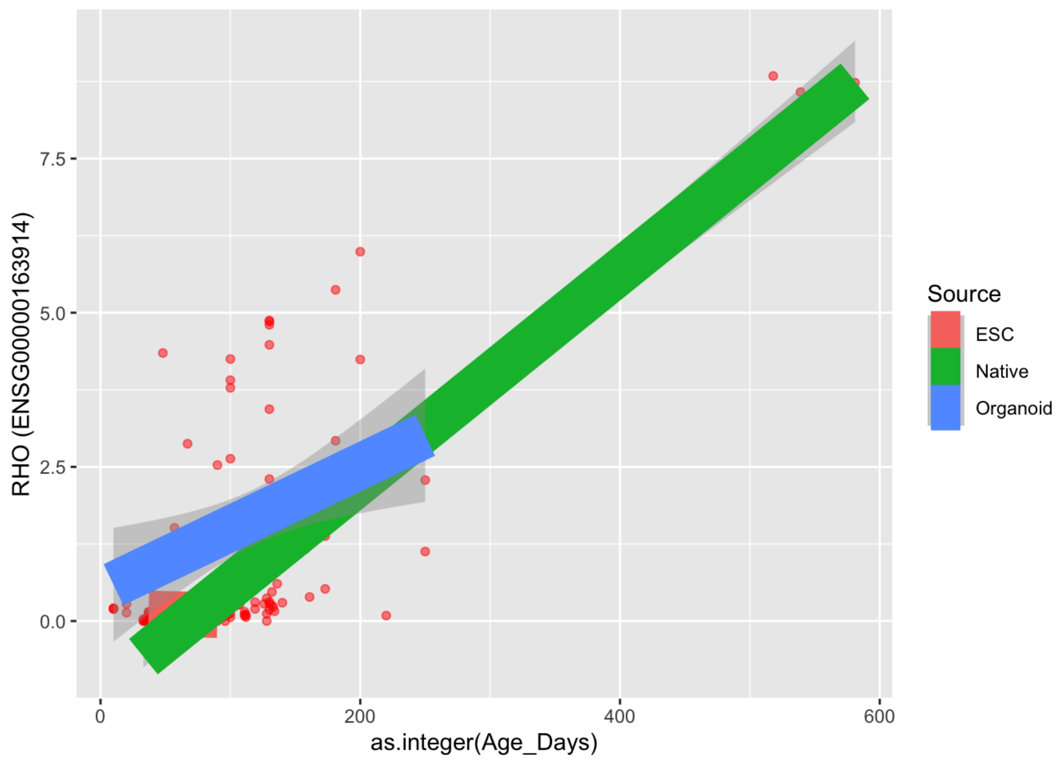
We can make the line thicker by *setting* the **size** aesthetic in the `geom_smooth` layer:

```
eiad_meta_exp %>%
  mutate(Age_Days = as.integer(Age_Days)) %>%
  filter(Tissue == 'Retina', !is.na(Age_Days)) %>%
  ggplot(aes(x=as.integer(Age_Days),
             y=`RHO (ENSG00000163914)`)) +
  geom_point(color = 'red', alpha =0.5) +
  geom_smooth(aes(color = Source), method = 'lm', size = 10)
```

```
## Warning: There was 1 warning in `mutate()`.
## ℹ In argument: `Age_Days = as.integer(Age_Days)`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



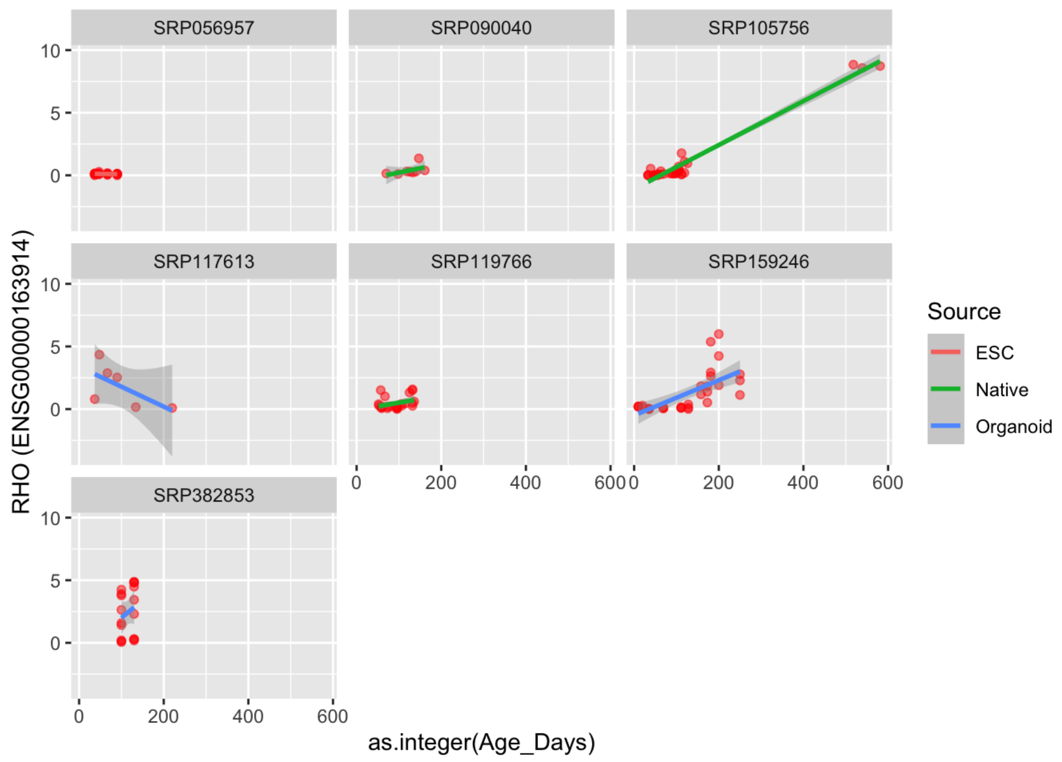
Multi-panel figures

Earlier we visualized the change in life expectancy over time across all countries in one plot. Alternatively, we can split this out over multiple panels by adding a layer of **facet** panels.

```
eiad_meta_exp %>%
  mutate(Age_Days = as.integer(Age_Days)) %>%
  filter(Tissue == 'Retina', !is.na(Age_Days)) %>%
  ggplot(aes(x=as.integer(Age_Days),
             y=`RHO (ENSG00000163914)`)) +
  geom_point(color = 'red', alpha =0.5) +
  geom_smooth(aes(color = Source), method = 'lm') +
  facet_wrap(~study_accession)
```

```
## Warning: There was 1 warning in `mutate()`.
## ℹ In argument: `Age_Days = as.integer(Age_Days)`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



The `facet_wrap` layer took a "formula" as its argument, denoted by the tilde (~).

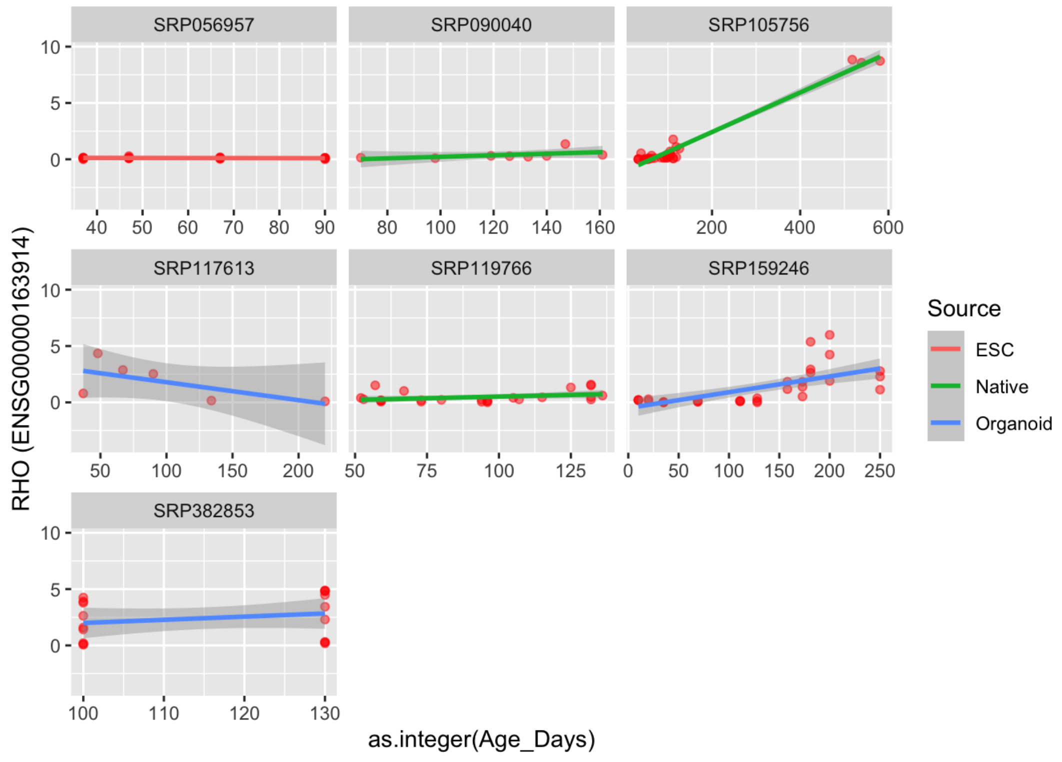
Modify x-axis range

Facet wrap, by default, keeps the x axis range (in this case 0 to 600) across all subsets. You can modify this behavior if you want to better see the trend in all studies

```
eiad_meta_exp %>%
  mutate(Age_Days = as.integer(Age_Days)) %>%
  filter(Tissue == 'Retina', !is.na(Age_Days)) %>%
  ggplot(aes(x=as.integer(Age_Days),
             y=`RHO (ENSG00000163914)`)) +
  geom_point(color = 'red', alpha =0.5) +
  geom_smooth(aes(color = Source), method = 'lm') +
  facet_wrap(~study_accession, scales = 'free_x')
```

```
## Warning: There was 1 warning in `mutate()`.
## ℹ In argument: `Age_Days = as.integer(Age_Days)`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

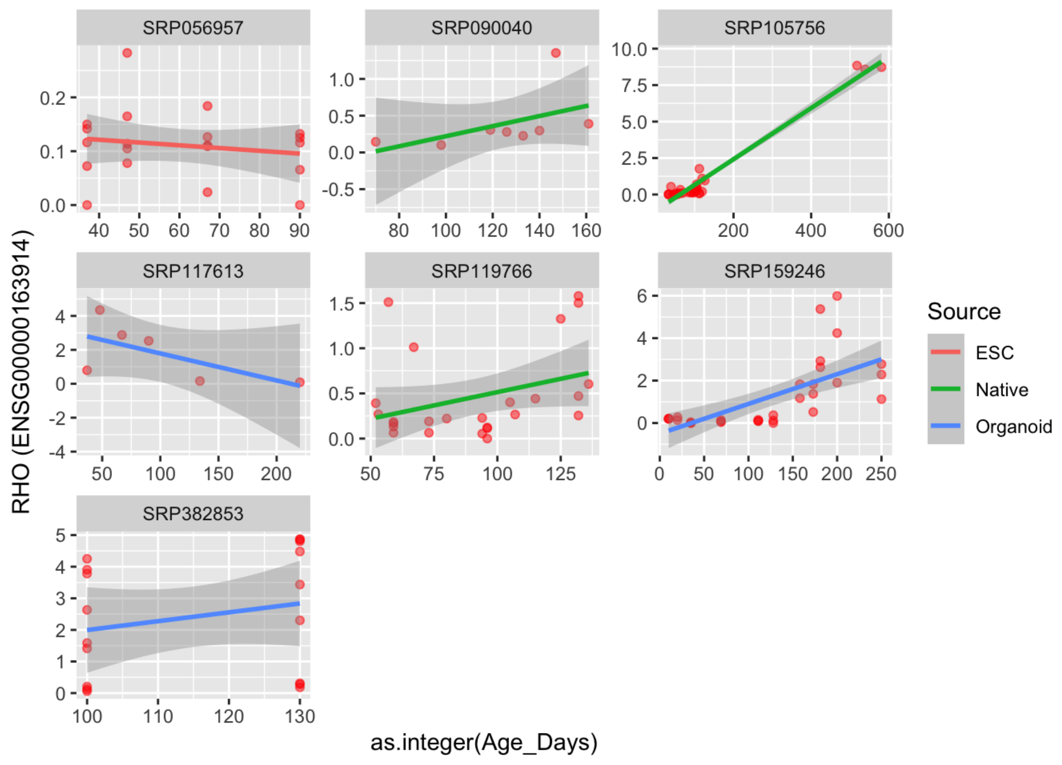


Hmm, we may want to apply this to both axes

```
eiad_meta_exp %>%
  mutate(Age_Days = as.integer(Age_Days)) %>%
  filter(Tissue == 'Retina', !is.na(Age_Days)) %>%
  ggplot(aes(x=as.integer(Age_Days),
             y=`RHO (ENSG00000163914)`)) +
  geom_point(color = 'red', alpha =0.5) +
  geom_smooth(aes(color = Source), method = 'lm') +
  facet_wrap(~study_accession, scales = 'free')
```

```
## Warning: There was 1 warning in `mutate()`.
## ℹ In argument: `Age_Days = as.integer(Age_Days)`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



## Modifying text

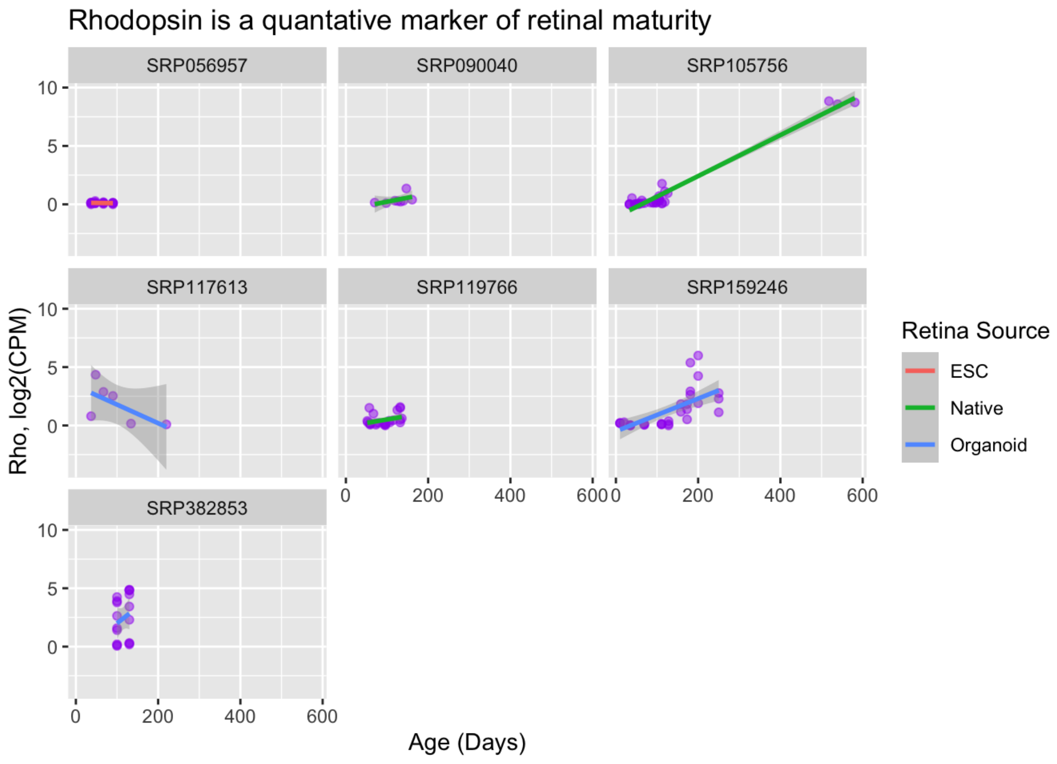
To clean this figure up for a publication we need to change some of the text elements. The x-axis is too cluttered, and the y axis should read “Life expectancy”, rather than the column name in the data frame.

We can do this by adding a couple of different layers. The **theme** layer controls the axis text, and overall text size. Labels for the axes, plot title and any legend can be set using the `labs` function. Legend titles are set using the same names we used in the `aes` specification. Thus below the color legend title is set using `color = "Continent"`, while the title of a fill legend would be set using `fill = "MyTitle"`.

```
eiad_meta_exp %>%
  mutate(Age_Days = as.integer(Age_Days)) %>%
  filter(Tissue == 'Retina', !is.na(Age_Days)) %>%
  ggplot(aes(x=as.integer(Age_Days),
             y=`RHO (ENSG00000163914)`)) +
  geom_point(color = 'purple', alpha =0.5) +
  geom_smooth(aes(color = Source), method = 'lm') +
  facet_wrap(~study_accession) +
  labs(
    x = "Age (Days)",               # x axis title
    y = "Rho, log2(CPM)",    # y axis title
    title = "Rhodopsin is a quantative marker of retinal maturity",      # main title of figure
    color = "Retina Source"       # title of legend
  )
```

```
## Warning: There was 1 warning in `mutate()`.
## ℹ In argument: `Age_Days = as.integer(Age_Days)`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

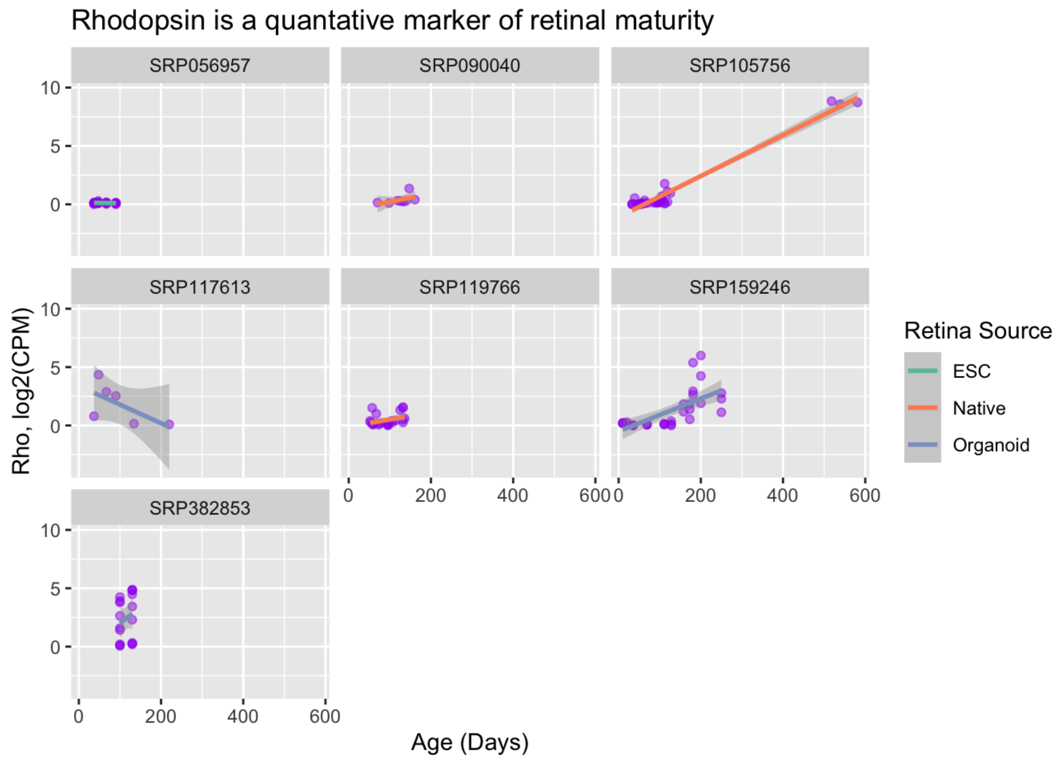


Modify color choices

```
eiad_meta_exp %>%
  mutate(Age_Days = as.integer(Age_Days)) %>%
  filter(Tissue == 'Retina', !is.na(Age_Days)) %>%
  ggplot(aes(x=as.integer(Age_Days),
             y=`RHO (ENSG00000163914)`)) +
  geom_point(color = 'purple', alpha =0.5) +
  geom_smooth(aes(color = Source), method = 'lm') +
  facet_wrap(~study_accession) +
  labs(
    x = "Age (Days)",                # x axis title
    y = "Rho, log2(CPM)",    # y axis title
    title = "Rhodopsin is a quantative marker of retinal maturity",      # main title of figure
    color = "Retina Source"       # title of legend
  ) +
  scale_colour_brewer(palette = 'Set2')
```

```
## Warning: There was 1 warning in `mutate()`.
## ℹ In argument: `Age_Days = as.integer(Age_Days)`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

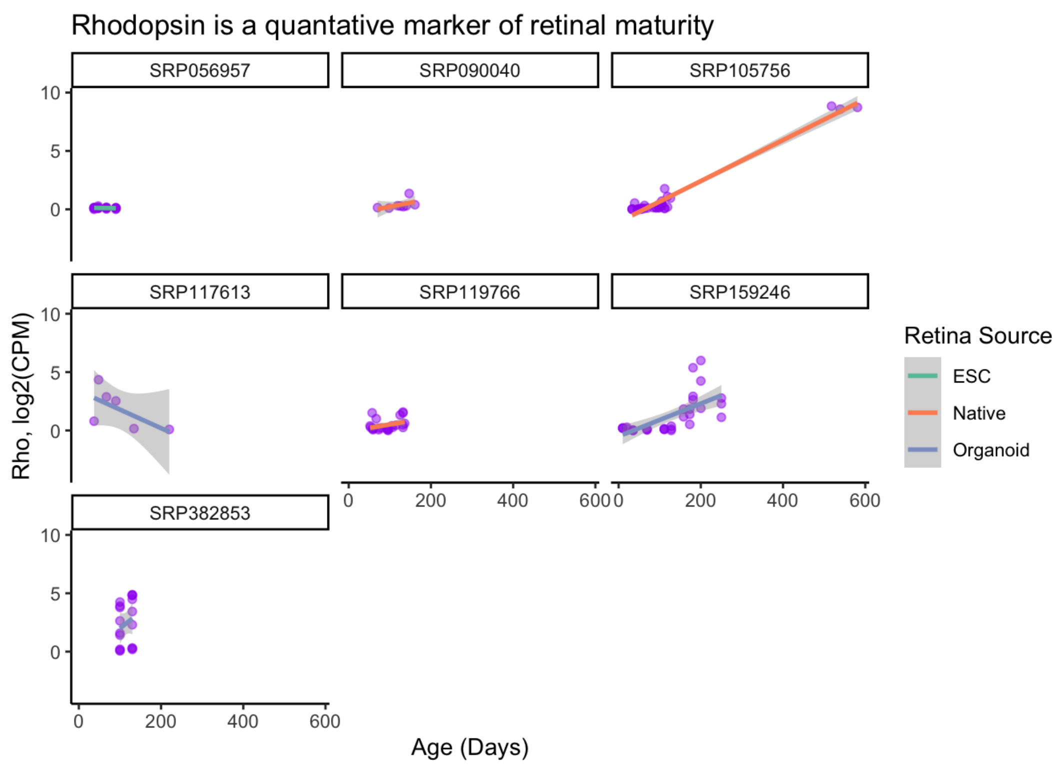


Modify theme

```
eiad_meta_exp %>%
  mutate(Age_Days = as.integer(Age_Days)) %>%
  filter(Tissue == 'Retina', !is.na(Age_Days)) %>%
  ggplot(aes(x=as.integer(Age_Days),
             y=`RHO (ENSG00000163914)`)) +
  geom_point(color = 'purple', alpha =0.5) +
  geom_smooth(aes(color = Source), method = 'lm') +
  facet_wrap(~study_accession) +
  labs(
    x = "Age (Days)",                # x axis title
    y = "Rho, log2(CPM)",    # y axis title
    title = "Rhodopsin is a quantative marker of retinal maturity",       # main title of figure
    color = "Retina Source"       # title of legend
  ) +
  scale_colour_brewer(palette = 'Set2') +
  theme_classic()
```

```
## Warning: There was 1 warning in `mutate()`.
## ℹ In argument: `Age_Days = as.integer(Age_Days)`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Exporting the plot

The `ggsave()` function allows you to export a plot created with ggplot. You can specify the dimension and resolution of your plot by adjusting the appropriate arguments (`width`, `height` and `dpi`) to create high quality graphics for publication. In order to save the plot from above, we first assign it to a variable `lifeExp_plot`, then tell `ggsave` to save that plot in `png` format to a directory called `results`. (Make sure you have a `results/` folder in your working directory.)

```
awesome_plot <- eiad_meta_exp %>%
  mutate(Age_Days = as.integer(Age_Days)) %>%
  filter(Tissue == 'Retina', !is.na(Age_Days)) %>%
  ggplot(aes(x=as.integer(Age_Days),
             y=`RHO (ENSG00000163914)`)) +
  geom_point(color = 'purple', alpha =0.5) +
  geom_smooth(aes(color = Source), method = 'lm') +
  facet_wrap(~study_accession) +
  labs(
    x = "Age (Days)",                # x axis title
    y = "Rho, log2(CPM)",    # y axis title
    title = "Rhodopsin is a quantative marker of retinal maturity",      # main title of figure
    color = "Retina Source"      # title of legend
  ) +
  scale_colour_brewer(palette = 'Set2') +
  theme_classic()
```

```
## Warning: There was 1 warning in `mutate()`.
## ℹ In argument: `Age_Days = as.integer(Age_Days)`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
ggsave(filename = "awesome_plots/rho_across_time.svg", plot = awesome_plot, width = 12, height = 10, dpi =
300, units = "cm")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

There are two nice things about `ggsave`. First, it defaults to the last plot, so if you omit the `plot` argument it will automatically save the last plot you created with `ggplot`. Secondly, it tries to determine the format you want to save your plot in from the file extension you provide for the filename (for example `.png` or `.pdf`). If you need to, you can specify the format explicitly in the `device` argument.

This is a taste of what you can do with ggplot2. RStudio provides a really useful cheat sheet of the different layers available, and more extensive documentation is available on the ggplot2 website. All RStudio cheat sheets are available from the RStudio website. Finally, if you have no idea how to change something, a quick Google search will usually send you to a relevant question and answer on Stack

Overflow with reusable code to modify!

Challenge 5

Generate boxplots to compare RPE65 expression across ocular tissues

Advanced:

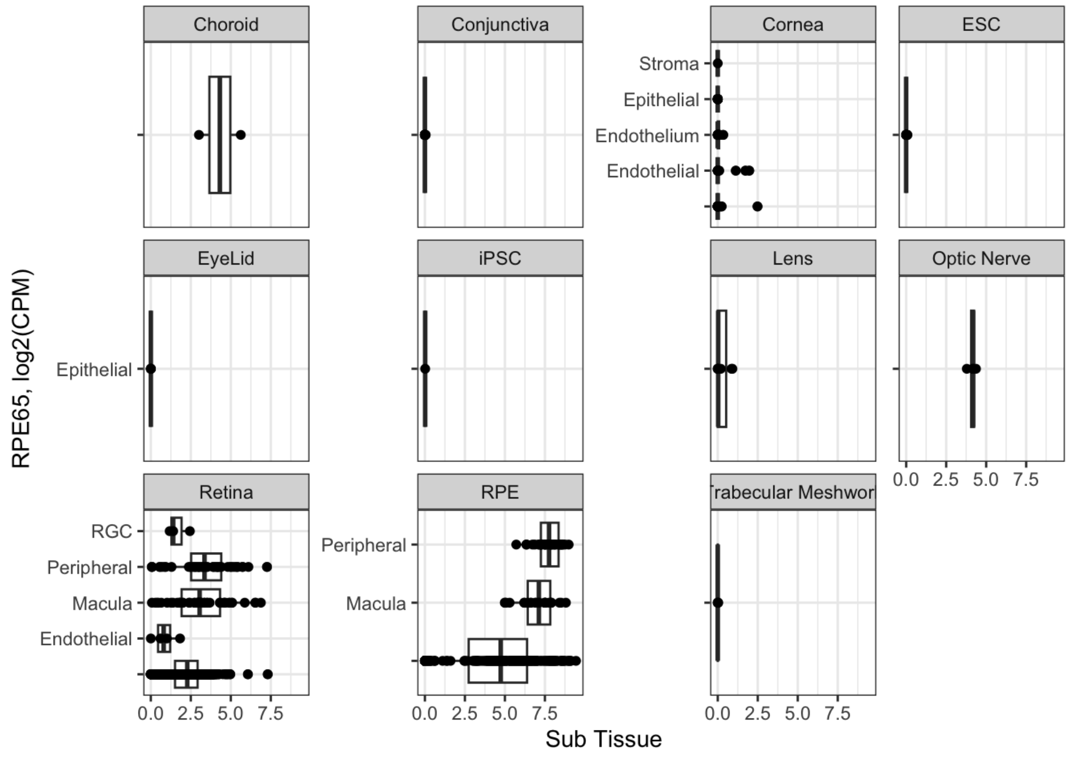
- Rename axes
- Account for Sub_Tissue
- Fix the “NA” issue for Sub_Tissue
- Customize the appearance for max awesome

Solution to Challenge 5

Here a possible solution:

```
eiad_meta_exp %>%
  filter(Cohort == 'Eye') %>%
  mutate(Sub_Tissue = case_when(is.na(Sub_Tissue) ~ '',
                                TRUE ~ Sub_Tissue)) %>%
  ggplot(aes(x=Sub_Tissue, y=`RPE65 (ENSG00000116745)`)) +

  geom_boxplot() +
  geom_point() +
  facet_wrap(~Tissue, scale = 'free_y') +
  coord_flip() +
  labs(
    x='RPE65, log2(CPM)',
    y = 'Sub Tissue'
  ) +
  theme_bw()
```



- Use `ggplot2` to create plots.
- Think about graphics in layers: aesthetics, geometry, statistics, scale transformation, and grouping.