A practical introduction to ggplot2

gscontras

6/8/2017

Preface

This tutorial borrows heavily from Judith Degen's Using `ggplot2` to visualize data tutorial and the Introduction to R Graphics with `ggplot2` from the Data Science Services at Harvard's IQSS.

Introduction

`ggplot2` is a langauge for creating graphics in R. Its author, Hadley Wickham, based his language on "The Grammar of Graphics" (Wikinson, 2005).

Getting started

You should have R installed. If you do not have R installed, then:

- Go to http://cran.r-project.org and download and install R
- Go to http://rstudio.com and download and install RStudio

Download the tutorial materials:

- Go to https://github.com/gscontras/tutorials/ggplot2
- Download `ggplot2.zip`
- Extract the zip file; tutorial notes are available in .R, .Rmd, and .html formats
- Open `ggplot2.R` in RStudio

Install the `ggplot2` package by typing `install.packages("ggplot2")` into your RStudio console. Then, load the `ggplot2` package:

```
library(ggplot2)
```

Why `ggplot2`?

There are some clear advantages of `ggplot2` over the base graphics system included with R:

- consistent underlying grammar of graphics (Wilkinson, 2005)
- mature and complete graphics system
- many users and lots of documentation
- ability to specify plot detail at a high level of abstraction
- amazingly flexible
- comprehensive `theme` system for polishing plot appearance

Still, the IQSS tutorial points to some clear limitations:

- 3D graphics (see the `rgl` package instead)
- Graph-theory type graphs with nodes and edges (see the `igraph` package instead)
- Interactive graphics (see the `ggvis` package instead)

The Grammar of Graphics

Graphics are data visualizations: mappings from data to aesthetic attributes of geometric objects. Building blocks of a graphic include:

- data
- aesthetic mappings from data to attributes
- geometric objects
- statistical transformations
- scales
- coordinate system
- positional adjustments
- facets for geting the same plot for different subsets of the data

For a handy guide to specifying details of your graphics, consult the `ggplot2` reference.

A sample dataset

The `languageR` package contains lots of handy information, including the `lexdec` dataset: lexical decsion latencies elicited from 21 subjects for 79 English nouns, with information tied to the participants and nouns.

Load the `languageR` package and the `lexdec` dataset:

```
library(languageR)
data(lexdec)
head(lexdec)
```

```
##   Subject       RT Trial Sex NativeLanguage Correct PrevType PrevCorrect
## 1      A1 6.340359    23   F        English correct     word     correct
## 2      A1 6.308098    27   F        English correct  nonword     correct
## 3      A1 6.349139    29   F        English correct  nonword     correct
## 4      A1 6.186209    30   F        English correct     word     correct
## 5      A1 6.025866    32   F        English correct  nonword     correct
## 6      A1 6.180017    33   F        English correct     word     correct
##         Word Frequency FamilySize SynsetCount Length  Class FreqSingular
## 1        owl  4.859812  1.3862944   0.6931472      3 animal           54
## 2       mole  4.605170  1.0986123   1.9459101      4 animal           69
## 3     cherry  4.997212  0.6931472   1.6094379      6  plant           83
## 4       pear  4.727388  0.0000000   1.0986123      4  plant           44
## 5        dog  7.667626  3.1354942   2.0794415      3 animal         1233
## 6 blackberry  4.060443  0.6931472   1.3862944     10  plant           26
##   FreqPlural DerivEntropy Complex      rInfl meanRT SubjFreq meanSize
## 1         74       0.7912 simplex -0.3101549 6.3582     3.12   3.4758
## 2         30       0.6968 simplex  0.8145080 6.4150     2.40   2.9999
## 3         49       0.4754 simplex  0.5187938 6.3426     3.88   1.6278
## 4         68       0.0000 simplex -0.4274440 6.3353     4.52   1.9908
## 5        828       1.2129 simplex  0.3977961 6.2956     6.04   4.6429
## 6         31       0.3492 complex -0.1698990 6.3959     3.28   1.5831
##   meanWeight      BNCw      BNCc       BNCd BNCcRatio BNCdRatio
## 1     3.1806 12.057065  0.000000   6.175602  0.000000  0.512198
## 2     2.6112  5.738806  4.062251   2.850278  0.707856  0.496667
## 3     1.2081  5.716520  3.249801  12.588727  0.568493  2.202166
## 4     1.6114  2.050370  1.462410   7.363218  0.713242  3.591166
## 5     4.5167 74.838494 50.859385 241.561040  0.679589  3.227765
## 6     1.1365  1.270338  0.162490   1.187616  0.127911  0.934882
```

A simple histogram

Pro-tip: begin by setting the background to white instead of gray.

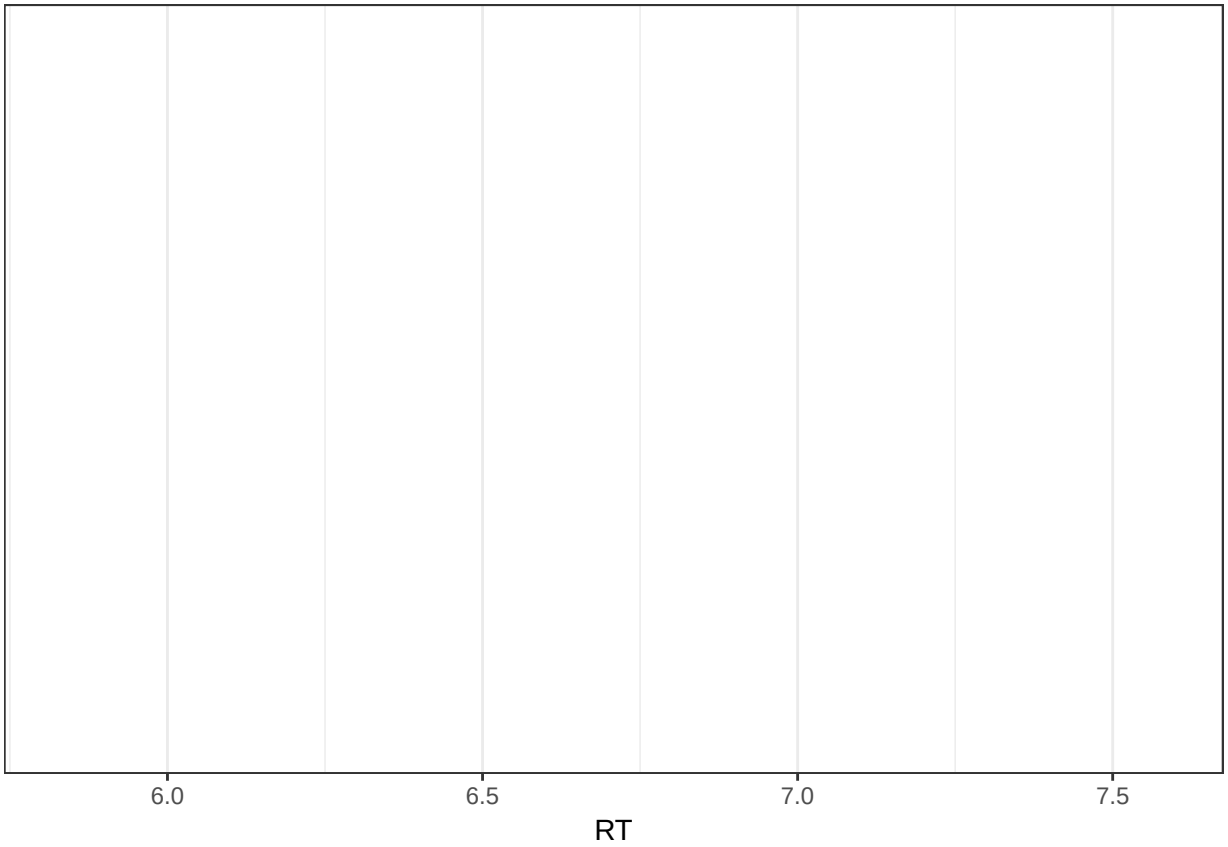
```
theme_set(theme_bw())
```

You'll also want to make sure you've set your working directory to the folder these files are in:

```
setwd("~/git/tutorials/ggplot2/")
```

Suppose you want to visualize the data in `lexdec`. We can start with a histogram of the response time distribution. We start by calling the `ggplot` function, which looks for a dataframe argument and an aesthetics argument. In the `aes` argument, we specify the x-axis.

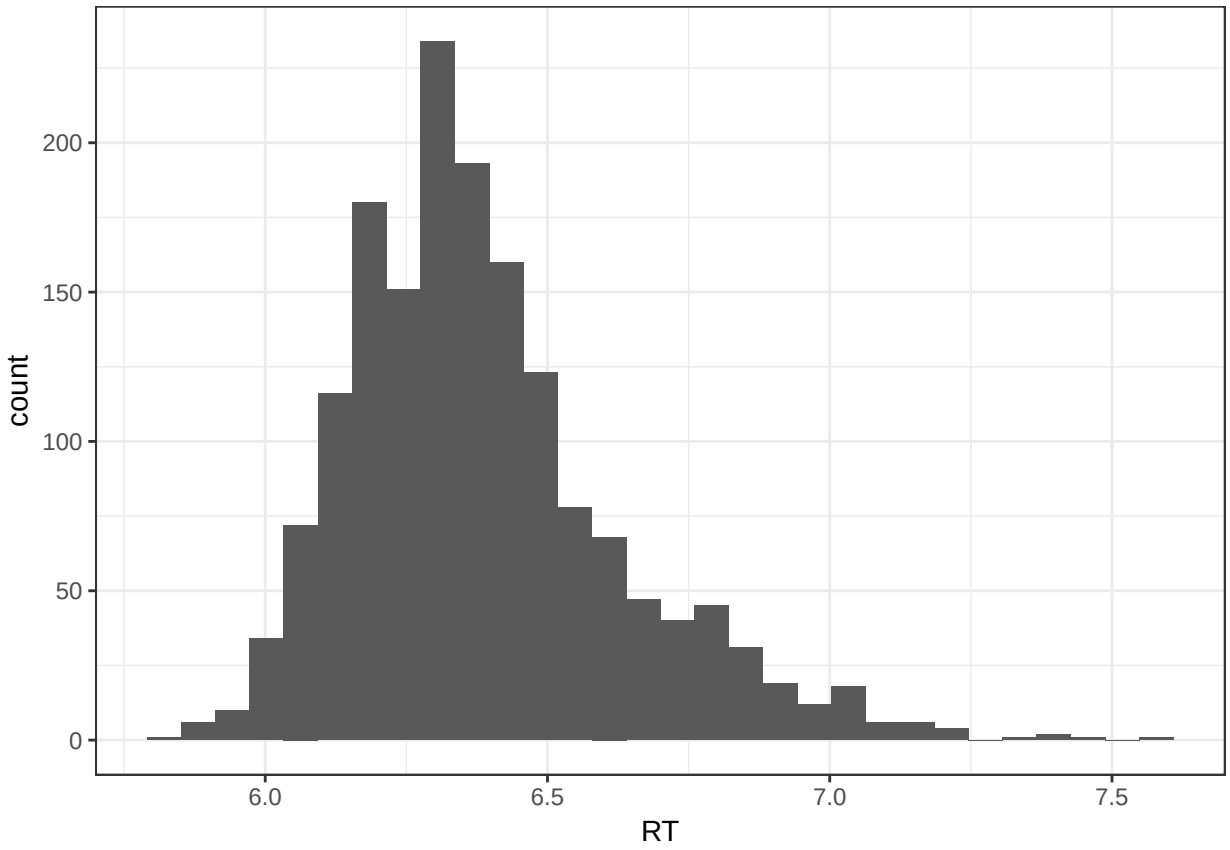
```
ggplot(lexdec, aes(x=RT))
```



Now, we want to add a `geom_histogram` layer to our base plot:

```
ggplot(lexdec, aes(x=RT)) +
  geom_histogram()
```

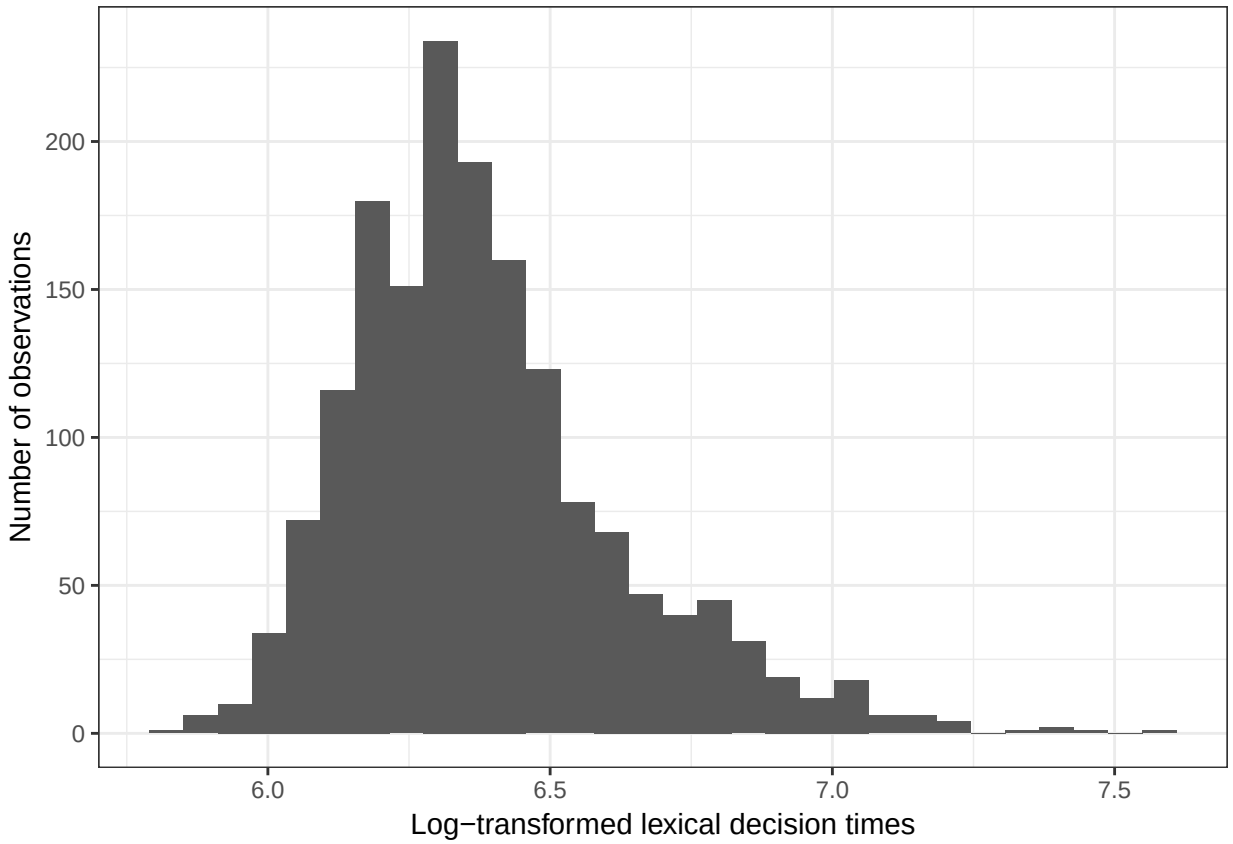
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



It's always helpful to add informative axis labels:

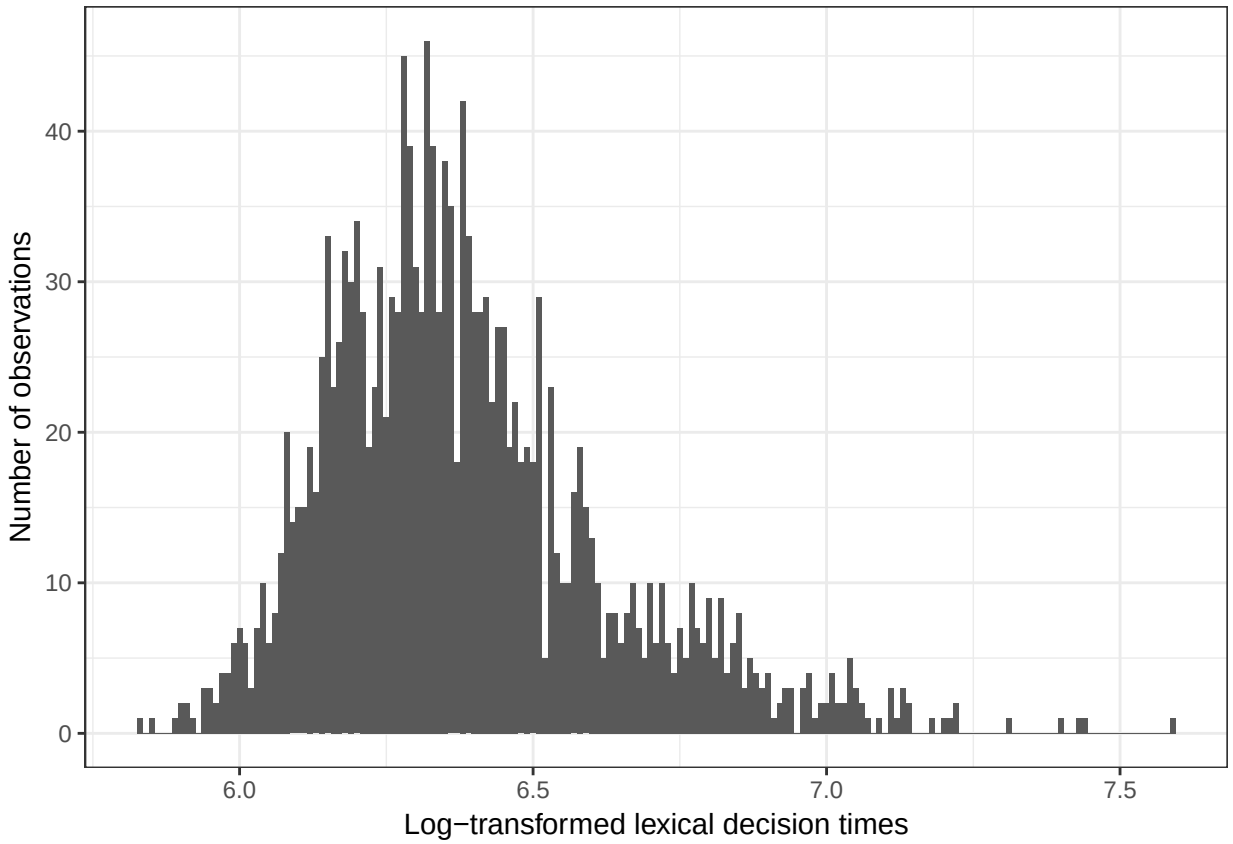
```
ggplot(lexdec, aes(x=RT)) +
  geom_histogram() +
  xlab("Log-transformed lexical decision times") +
  ylab("Number of observations")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Each of the `geom` layers has attributions of its own that you can specify. Here, we specify the `binwidth` of `geom_histogram`:

```
ggplot(lexdec, aes(x=RT)) +
  geom_histogram(binwidth=0.01) +
  xlab("Log-transformed lexical decision times") +
  ylab("Number of observations")
```



To save a plot you've created, use `ggsave`. Be sure to fine-tune the `width` and `height` arguments to suit your needs.

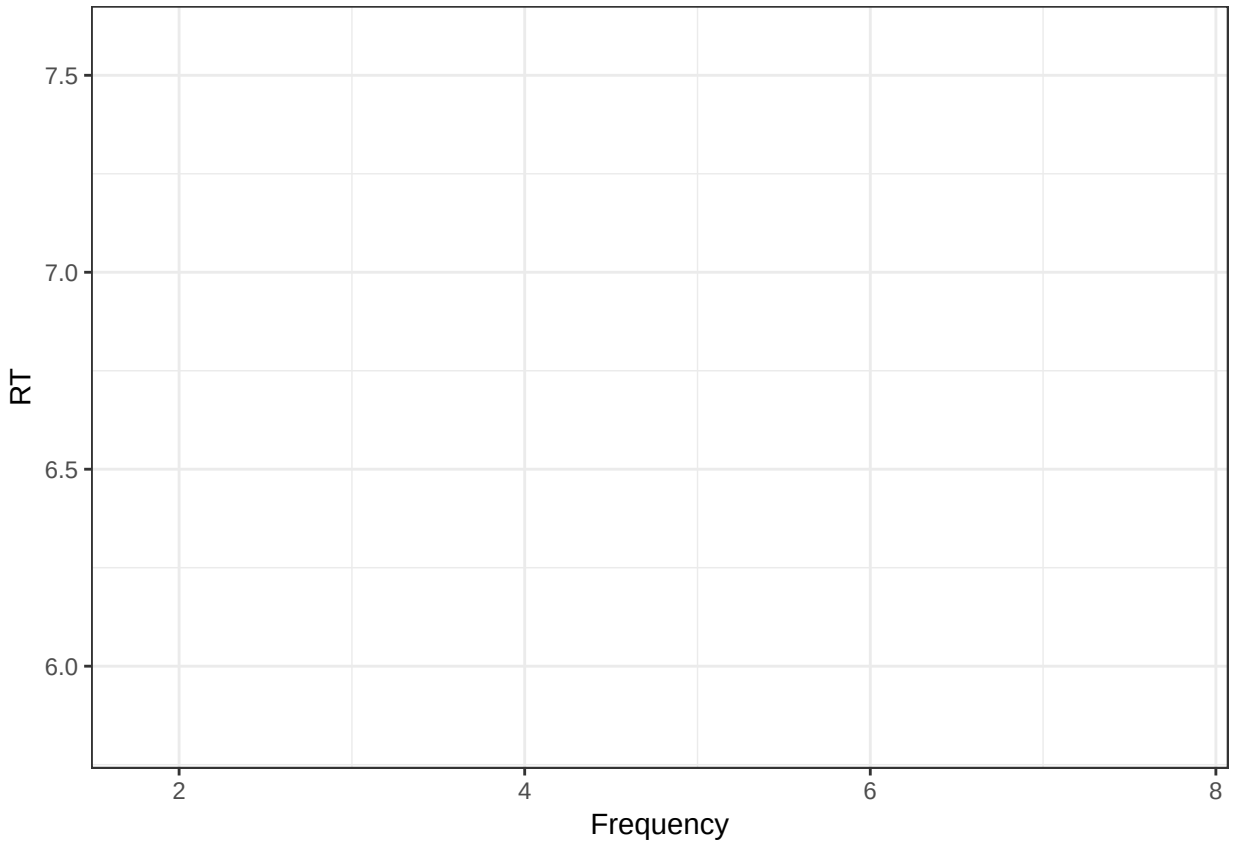
```
ggsave("plots/rt_histogram.png",width=5,height=4)
```

A scatterplot

Often, histograms won't suffice to visualize the patterns of data we're interested in. To see how lexical decision times pattern as a function of word frequency, we can use the same dataset to create a scatterplot.

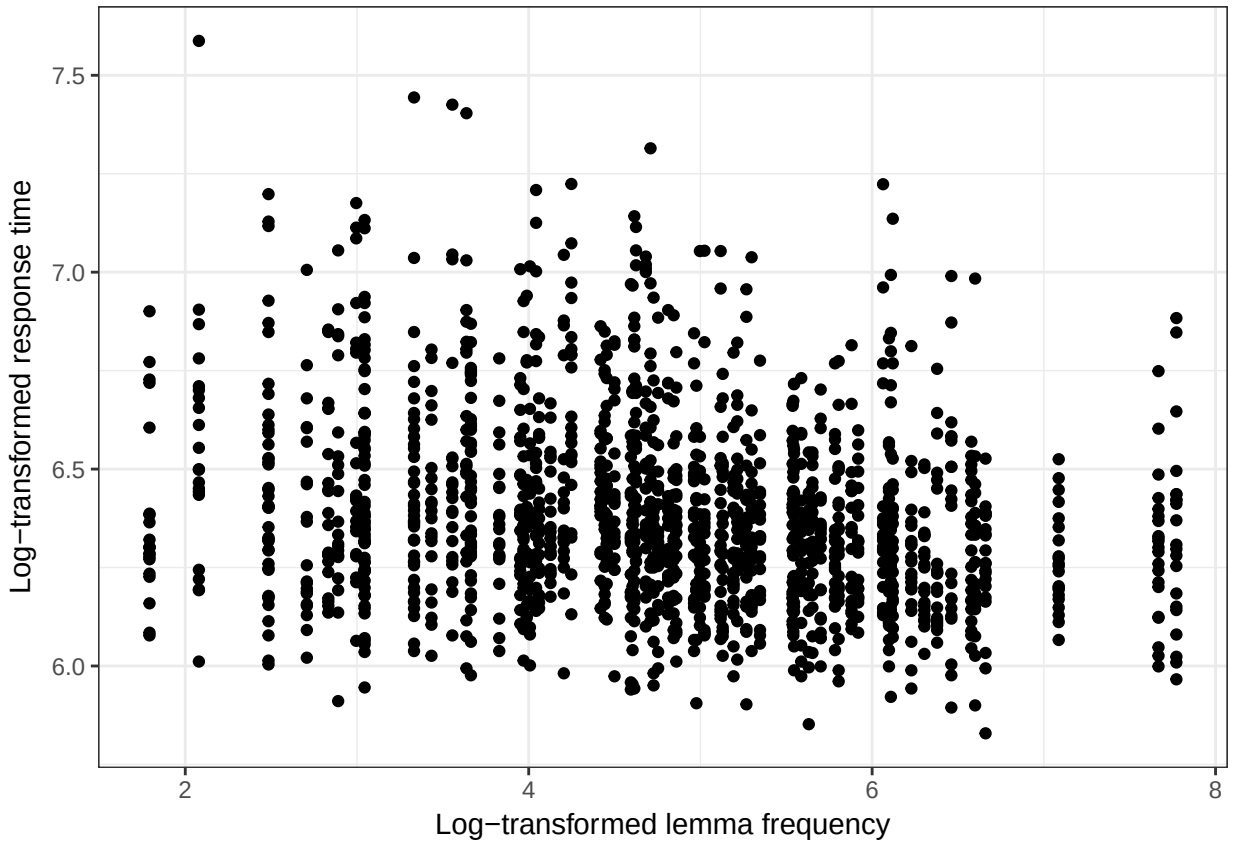
We begin as before, by suppling a dataframe and some aesthetics to `ggplot2`; now, we'll want our x-axis to plot word frequency, while our y-axis plot lexical decision times.

```
ggplot(lexdec, aes(x=Frequency, y=RT))
```



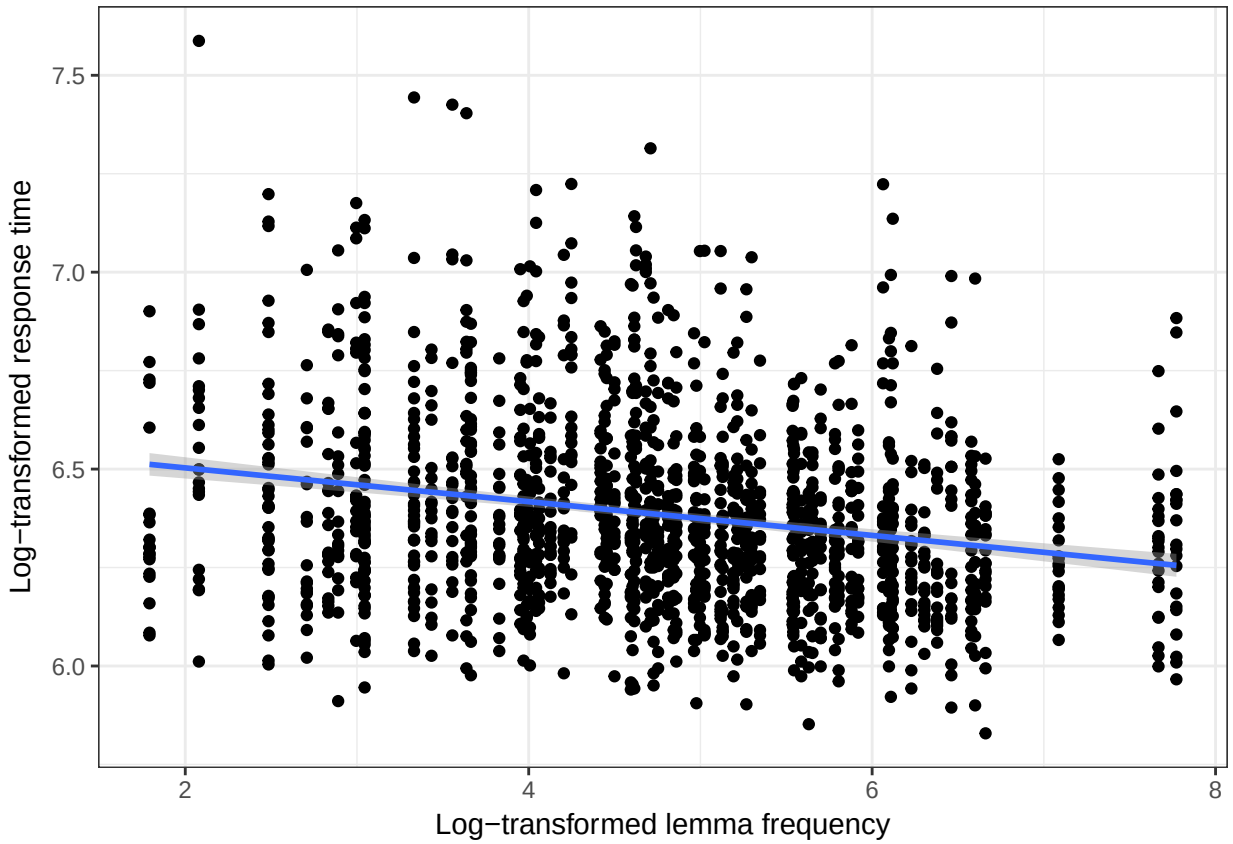
To this base we add our `geom_point` layer and some reasonable axis labels:

```
ggplot(lexdec, aes(x=Frequency, y=RT)) +
  geom_point() +
  xlab("Log-transformed lemma frequency") +
  ylab("Log-transformed response time")
```



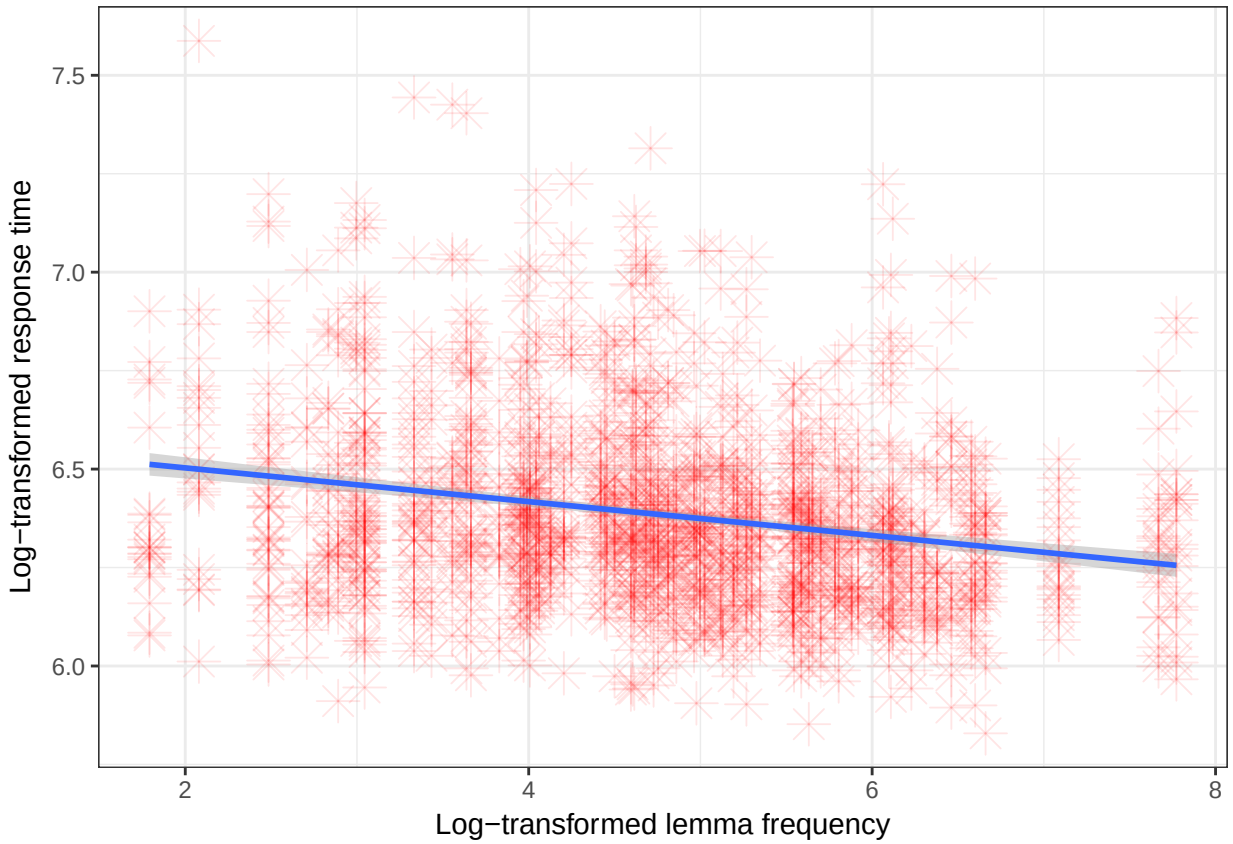
We can get a better sense of what's going on in our data by adding a smoothing layer with `geom_smooth`. Here, we specify the `method` to `lm` (short for *linear model*) so that it draws the line of best fit:

```
ggplot(lexdec, aes(x=Frequency, y=RT)) +
  geom_point() +
  geom_smooth(method="lm") +
  xlab("Log-transformed lemma frequency") +
  ylab("Log-transformed response time")
```



We can customise attributes of the plot, for example the `color` or `size` or `shape` or the opacity (i.e., `alpha`) of our points:

```
ggplot(lexdec, aes(x=Frequency, y=RT)) +
  geom_point(color="red",size=5,shape=8,alpha=0.1) +
  geom_smooth(method="lm") +
  xlab("Log-transformed lemma frequency") +
  ylab("Log-transformed response time")
```

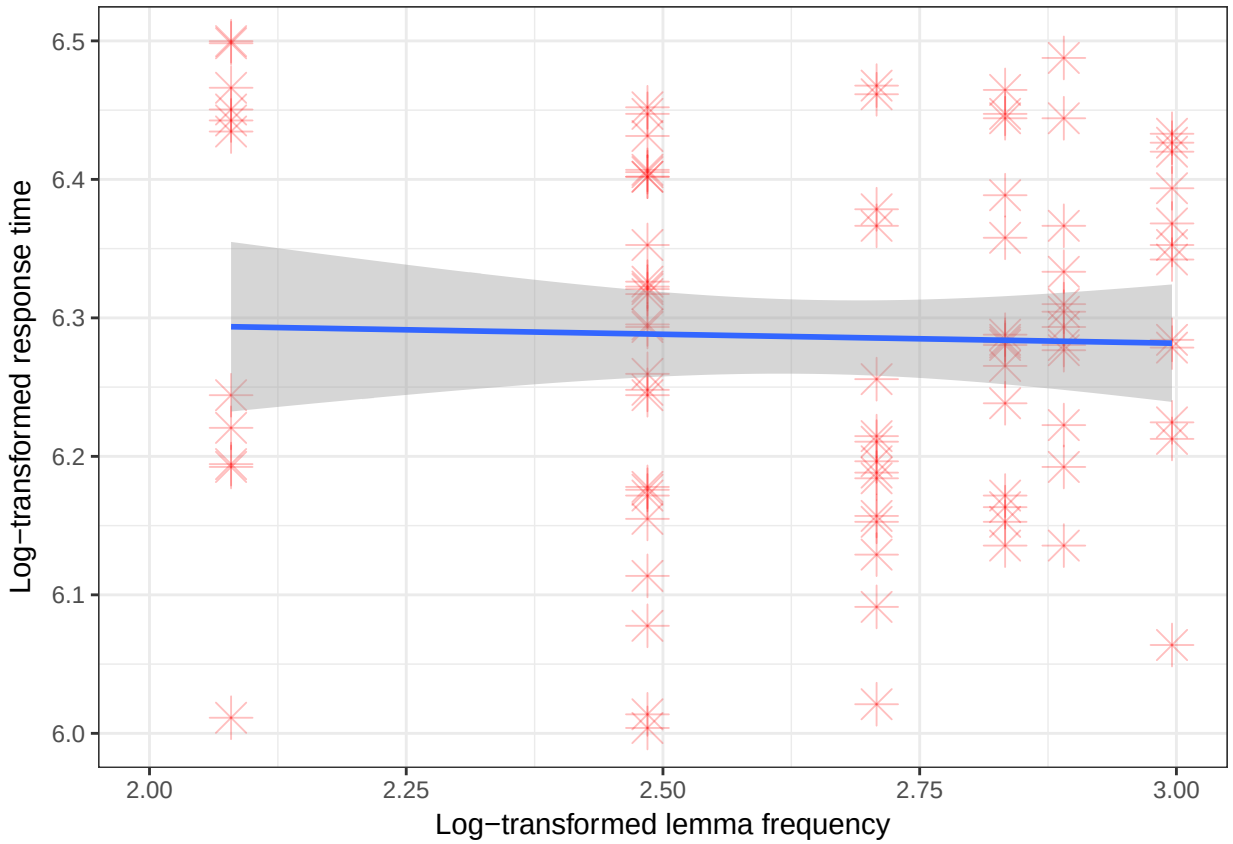


We can also adjust our x- and y-axis limits, which will zoom in on the plot while deleting points outside the specified range.

```
ggplot(lexdec, aes(x=Frequency, y=RT)) +
  geom_point(color="red",size=5,shape=8,alpha=0.25) +
  geom_smooth(method="lm") +
  xlim(c(2,3)) +
  ylim(c(6,6.5)) +
  xlab("Log-transformed lemma frequency") +
  ylab("Log-transformed response time")
```

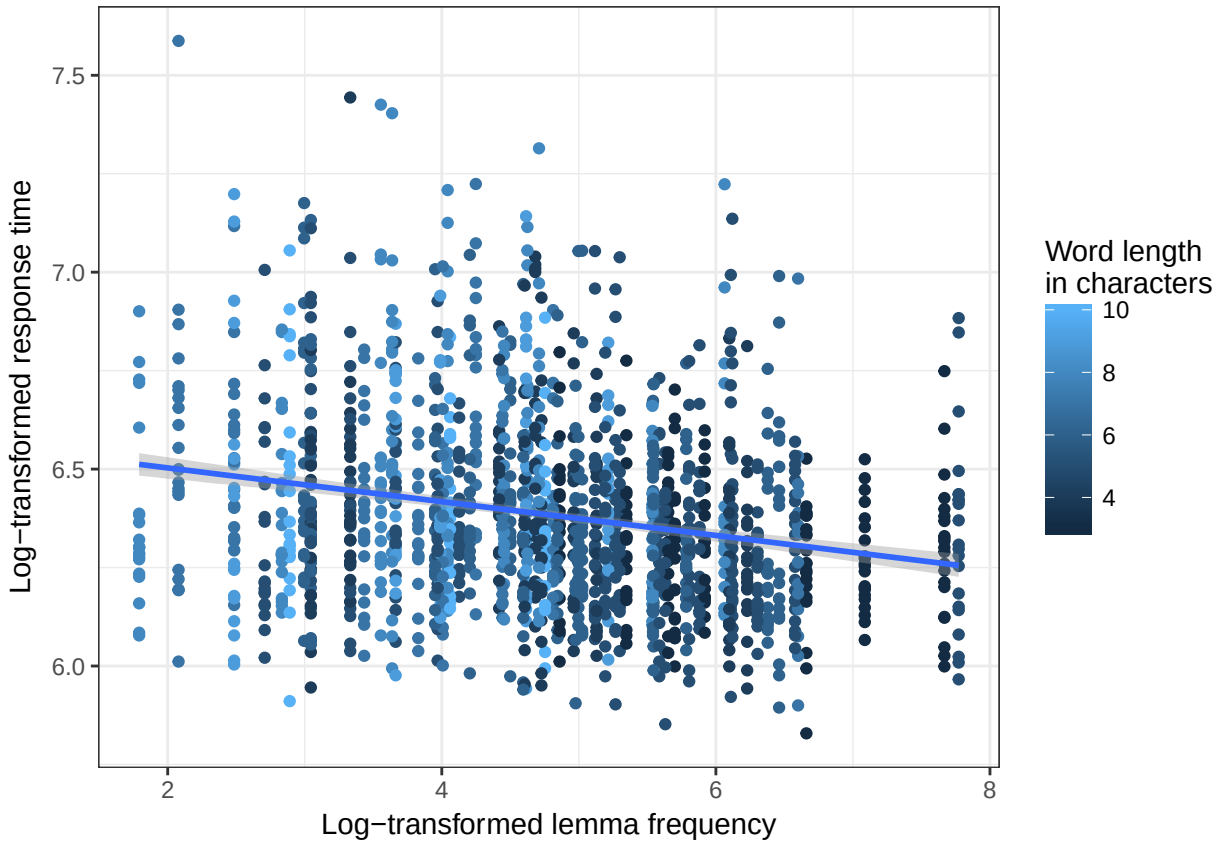
```
## Warning: Removed 1569 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 1569 rows containing missing values (geom_point).
```



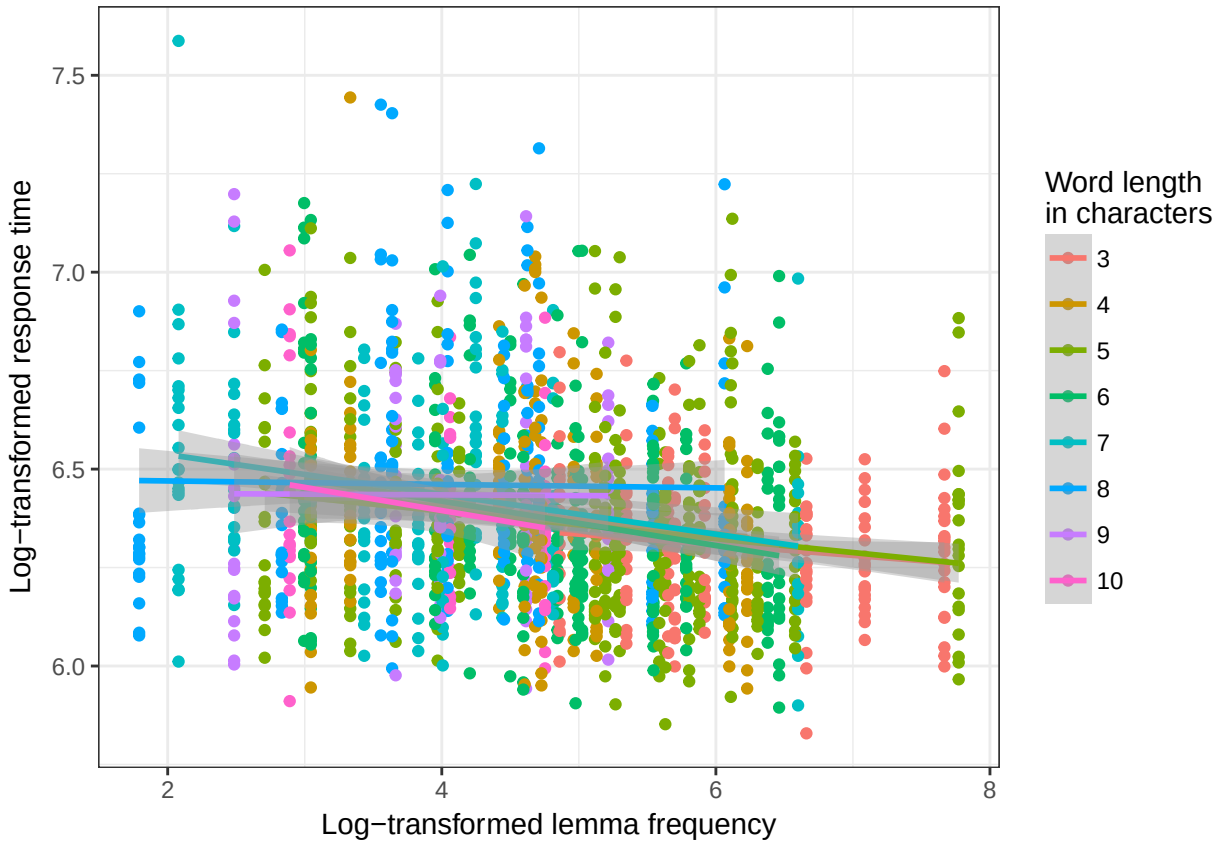
We can include a lot more information in our scatter plots by specifying various aesthetics. For example, we can `color` our points by the 'Length' of the words in question.

```
ggplot(lexdec, aes(x=Frequency, y=RT, color=Length)) +
  geom_point() +
  geom_smooth(method="lm") +
  xlab("Log-transformed lemma frequency") +
  ylab("Log-transformed response time") +
  labs(color="Word length\nin characters")
```



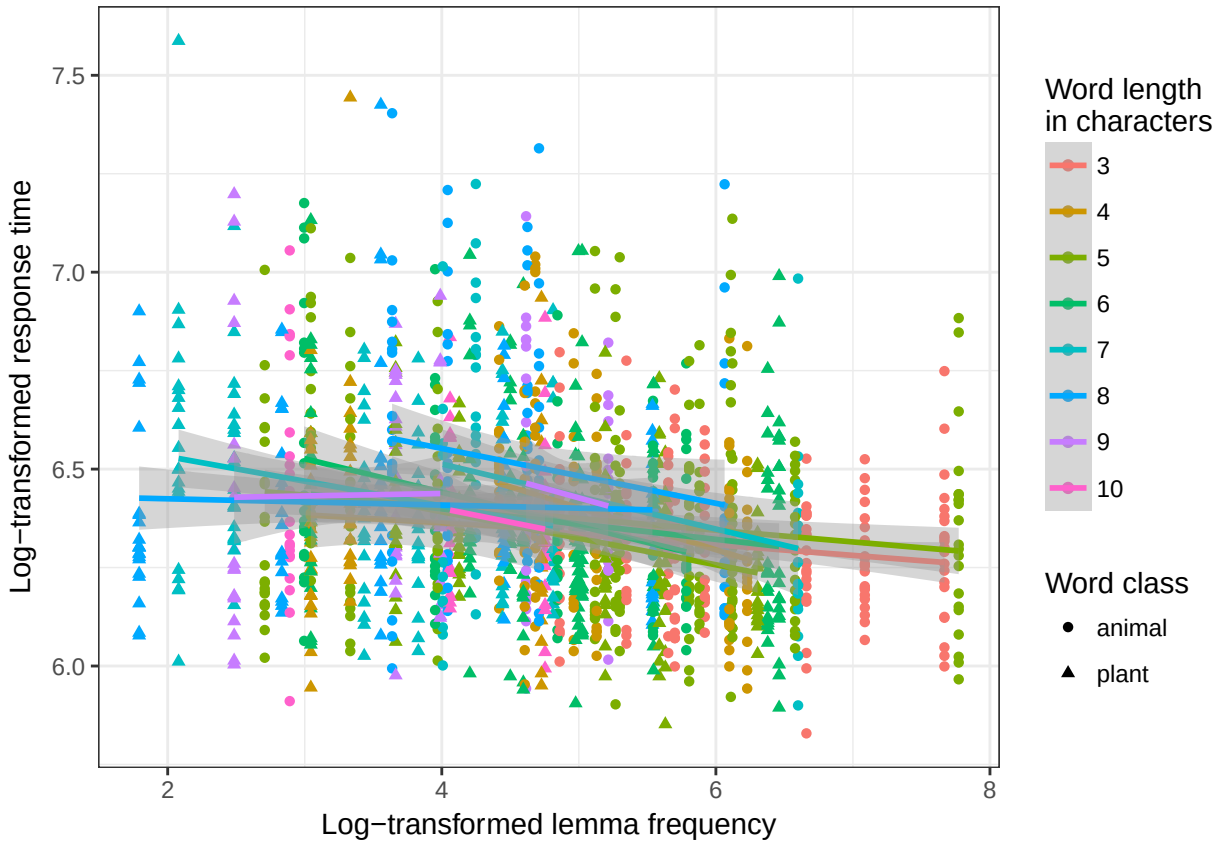
Rather than treating `Length` as a continuous collection of intergers, we can treat it as a `factor` with discrete levels. Doing so will now draw a different smoothing line for each possible word `Length`.

```
ggplot(lexdec, aes(x=Frequency, y=RT, color=as.factor(Length))) +
  geom_point() +
  geom_smooth(method="lm") +
  xlab("Log-transformed lemma frequency") +
  ylab("Log-transformed response time") +
  labs(color="Word length\nin characters")
```



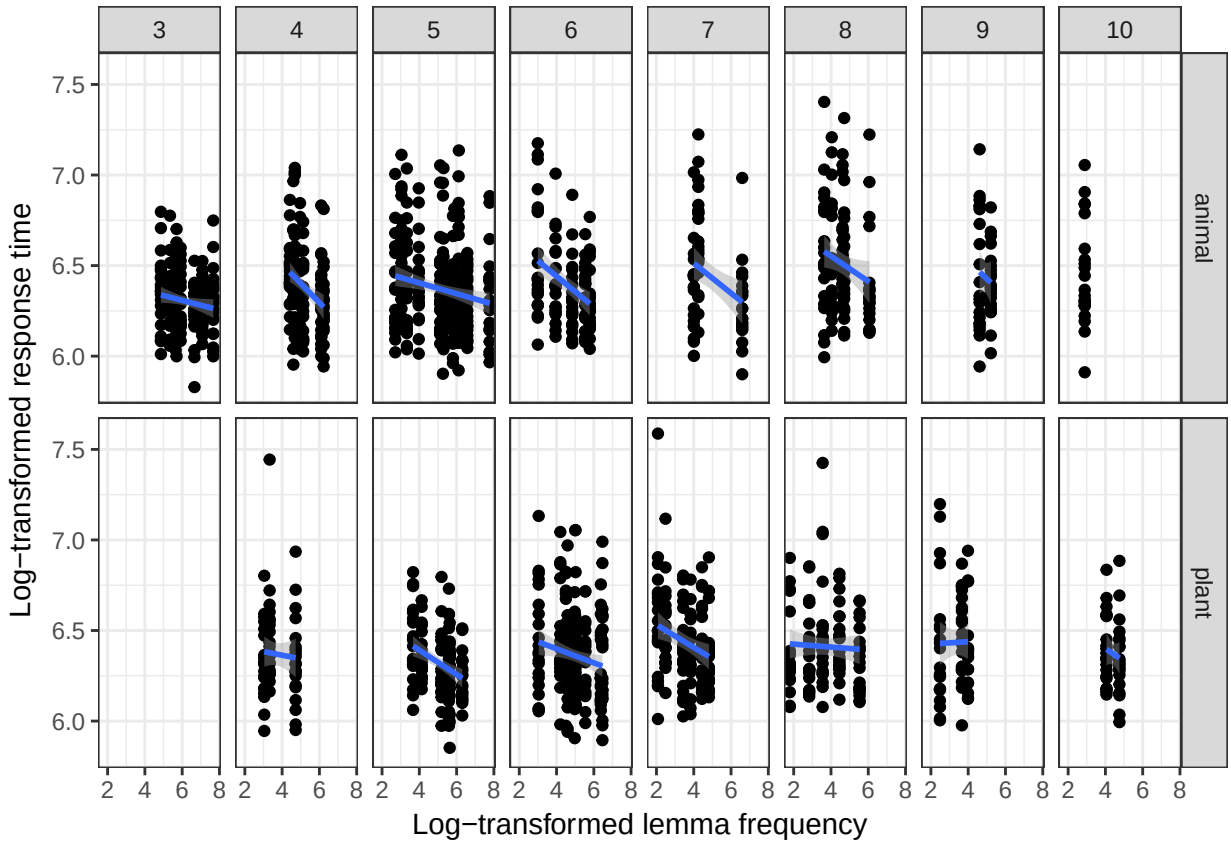
We can also specify the `shape` aesthetic to communcation informatio, say the `Class` to which a given word belongs (*animal* vs. *plant*).

```
ggplot(lexdec, aes(x=Frequency, y=RT, color=as.factor(Length), shape=Class)) +
  geom_point() +
  geom_smooth(method="lm") +
  xlab("Log-transformed lemma frequency") +
  ylab("Log-transformed response time") +
  labs(color="Word length\nin characters", shape="Word class")
```



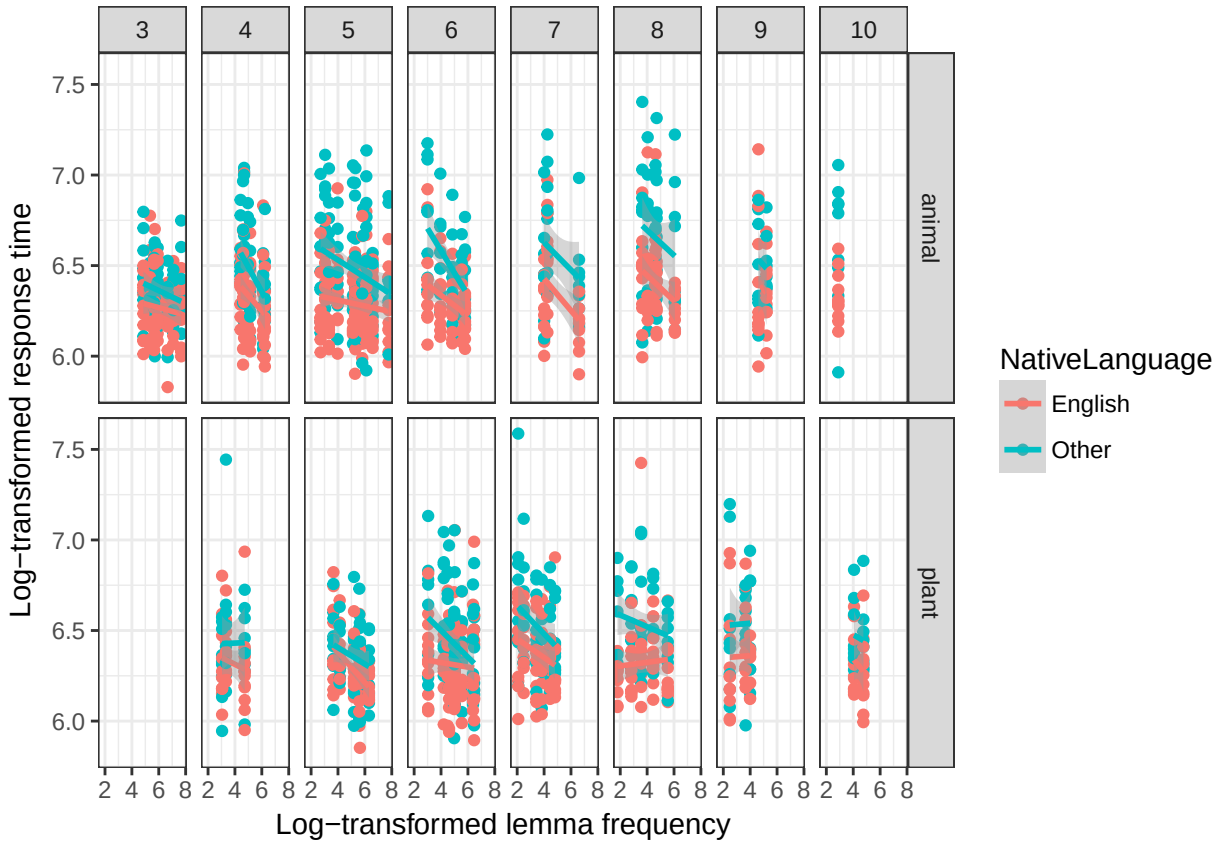
This plot has gotten awfully busy with all this information. We can clean things up a bit by faceting the graphic by `Length` and `Class`, rather than including all of the information on a single plot.

```
ggplot(lexdec, aes(x=Frequency, y=RT)) +
  geom_point() +
  geom_smooth(method="lm") +
  xlab("Log-transformed lemma frequency") +
  ylab("Log-transformed response time") +
  facet_grid(Class~Length)
```



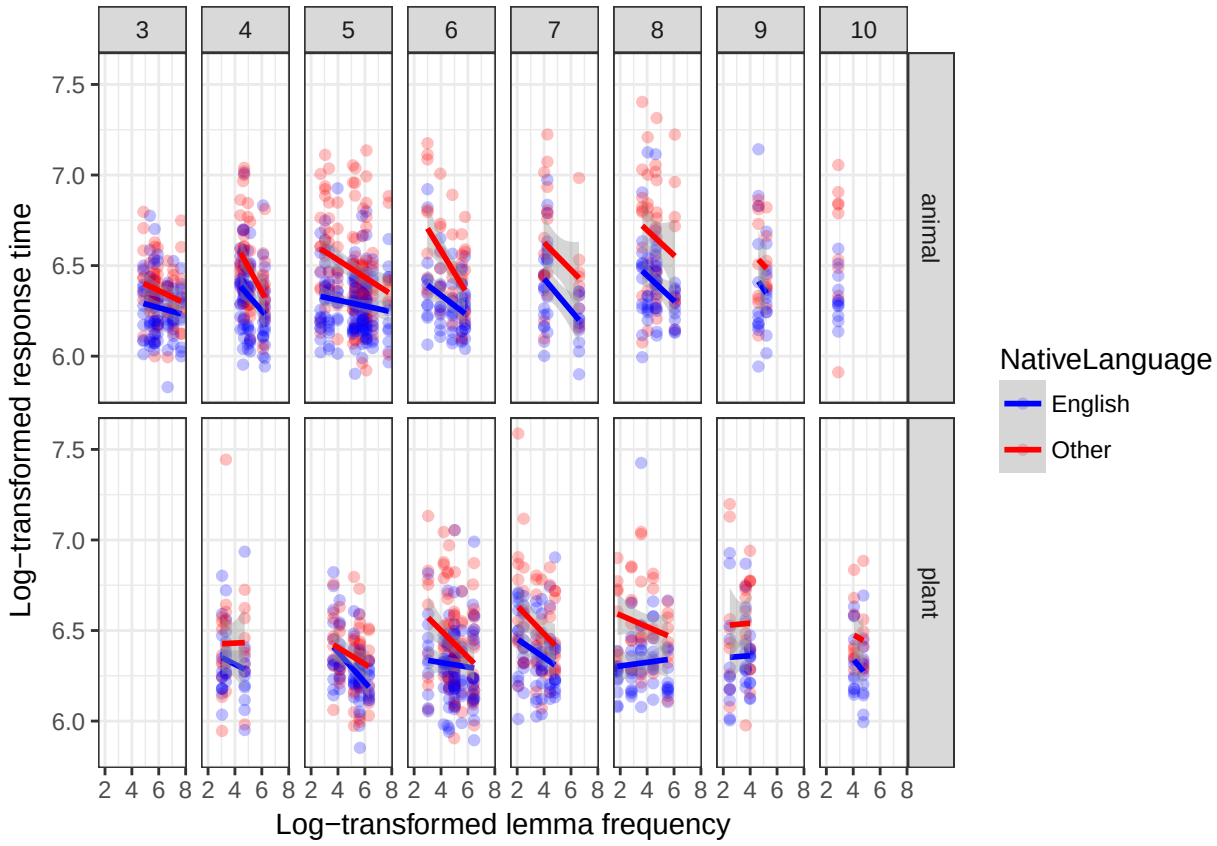
As before, we can add more information to our plots with aesthetics, for example `color` by the `NativeLanguage` of the participants.

```
ggplot(lexdec, aes(x=Frequency, y=RT, color=NativeLanguage)) +
  geom_point() +
  geom_smooth(method="lm") +
  xlab("Log-transformed lemma frequency") +
  ylab("Log-transformed response time") +
  facet_grid(Class~Length)
```



If you don't like the default colors, feel free to choose your own.

```
ggplot(lexdec, aes(x=Frequency, y=RT, color=NativeLanguage)) +
  geom_point(alpha=0.25) +
  geom_smooth(method="lm") +
  scale_color_manual(values=c("blue","red")) +
  xlab("Log-transformed lemma frequency") +
  ylab("Log-transformed response time") +
  facet_grid(Class~Length)
```



A barplot

Scatterplots are useful when it comes to visualizing lots of data, but sometimes you want to focus in on summary statistics like mean values. In that case, you're more likely to want a barplot to visualize your results.

It's important to remember that the point estimate of a mean is useless information without some sense of the variation in the data that led to that point estimate. We can communicate that information with error bars representing confidence intervals.

The following helper file will load the `bootsSummary` function, which calculates these confidence intervals for you.

```
source("helpers.R")
```

Now, suppose we want to visualize the average response time for each possible word length. First we'll use `bootsSummary` to calculate those values for us.

```
d_s = bootsSummary(data=lexdec, measurevar="RT", groupvars=c("Length"))
```

```
## Loading required package: plyr
```

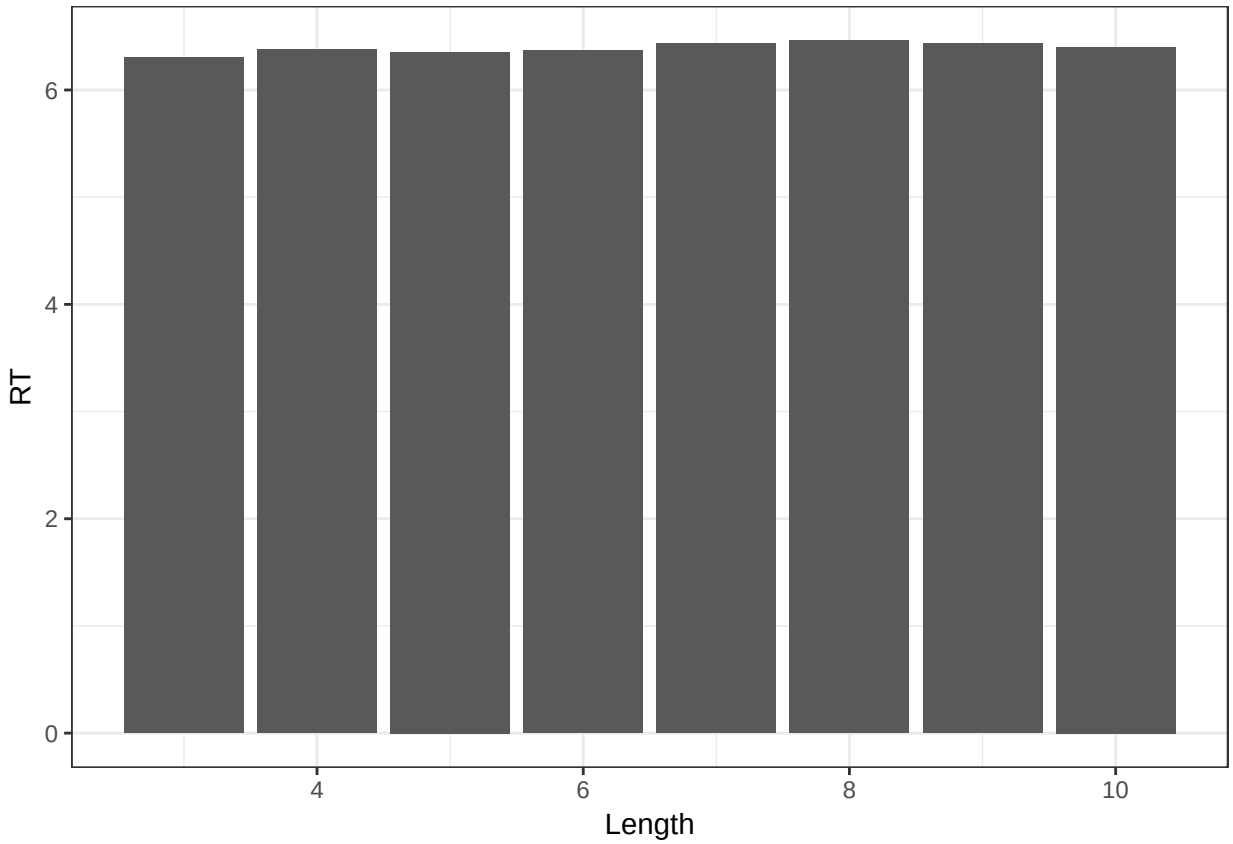
```
d_s
```

```
##   Length   N       RT bootsci_high bootsci_low
## 1      3 168 6.304878     6.330185    6.279797
## 2      4 210 6.380140     6.411993    6.348741
## 3      5 399 6.353711     6.375780    6.332576
## 4      6 315 6.370368     6.396564    6.344508
```

```
## 5      7 189 6.436170     6.473356    6.400170
## 6      8 210 6.460569     6.497492    6.424323
## 7      9 105 6.434549     6.487418    6.384208
## 8     10  63 6.400557     6.459426    6.344304
```

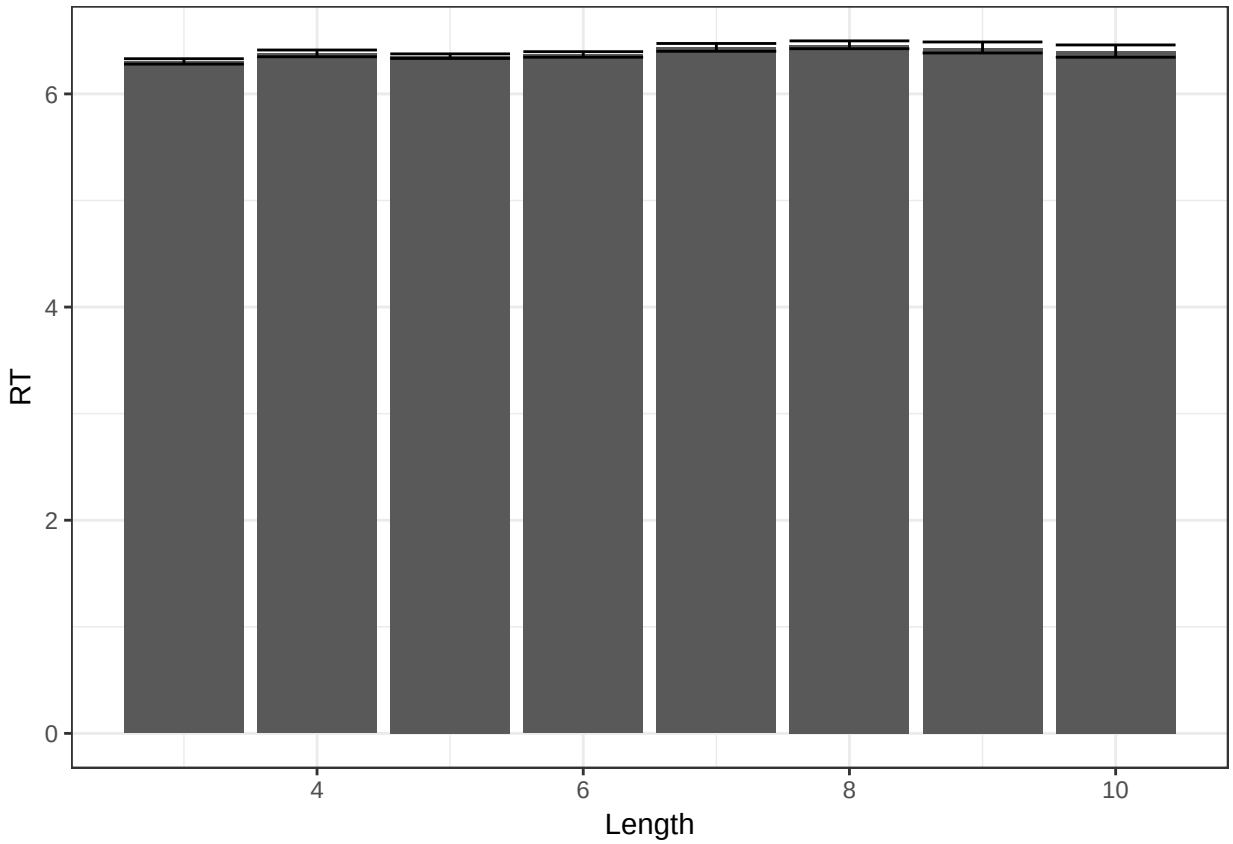
We'll feed this information into a new `ggplot`, this time using `geom_bar`.

```
ggplot(d_s,aes(x=Length,y=RT))+
  geom_bar(stat="identity")
```



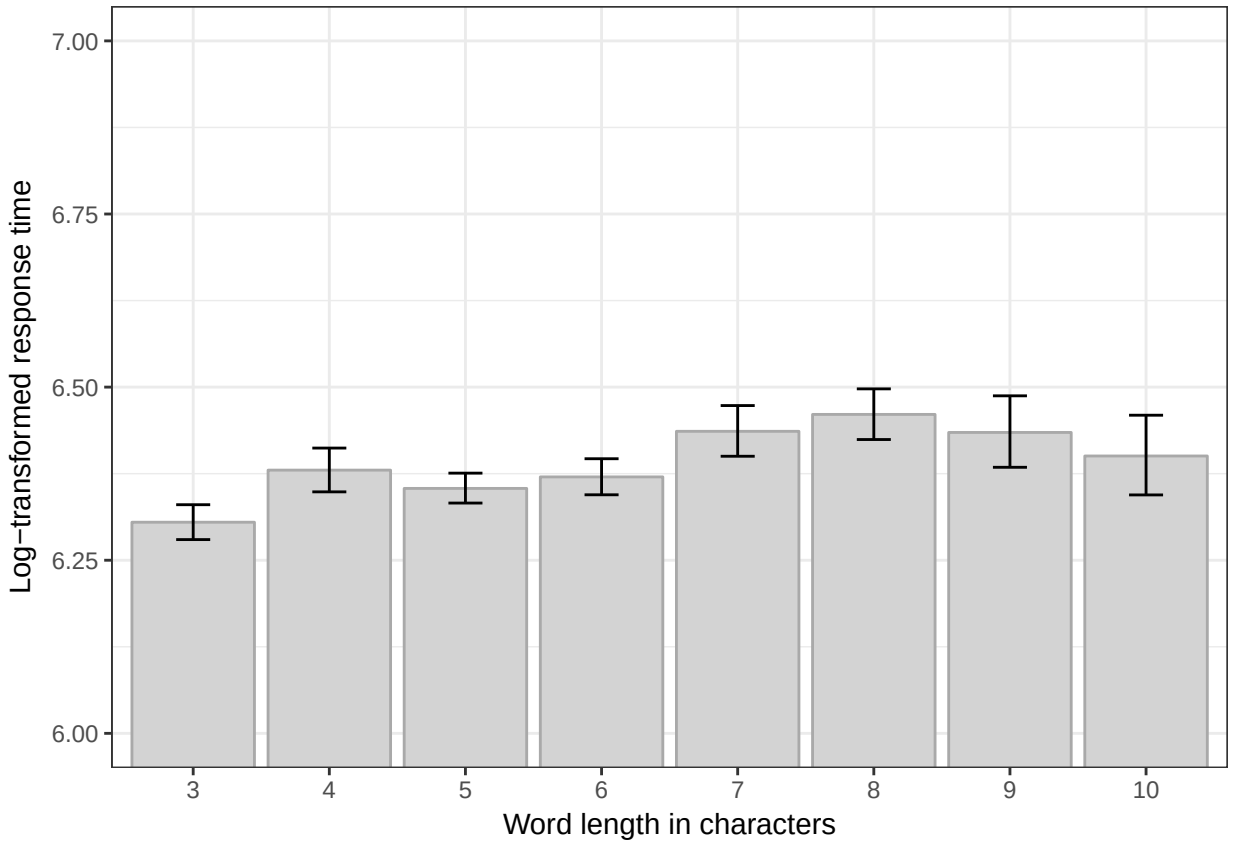
To add error bars that represent our confidence intervals, we use `geom_errorbar`.

```
ggplot(d_s,aes(x=Length,y=RT))+
  geom_bar(stat="identity") +
  geom_errorbar(aes(ymin=bootsci_low, ymax=bootsci_high, x=Length))
```



We can pretty the plot up some by changing colors and attributes.

```
ggplot(d_s,aes(x=as.factor(Length),y=RT))+
  geom_bar(stat="identity",fill="lightgray",color="darkgray") +
  geom_errorbar(aes(ymin=bootsci_low, ymax=bootsci_high, x=as.factor(Length), width=0.25)) +
  coord_cartesian(ylim=c(6,7)) +
  xlab("Word length in characters") +
  ylab("Log-transformed response time")
```



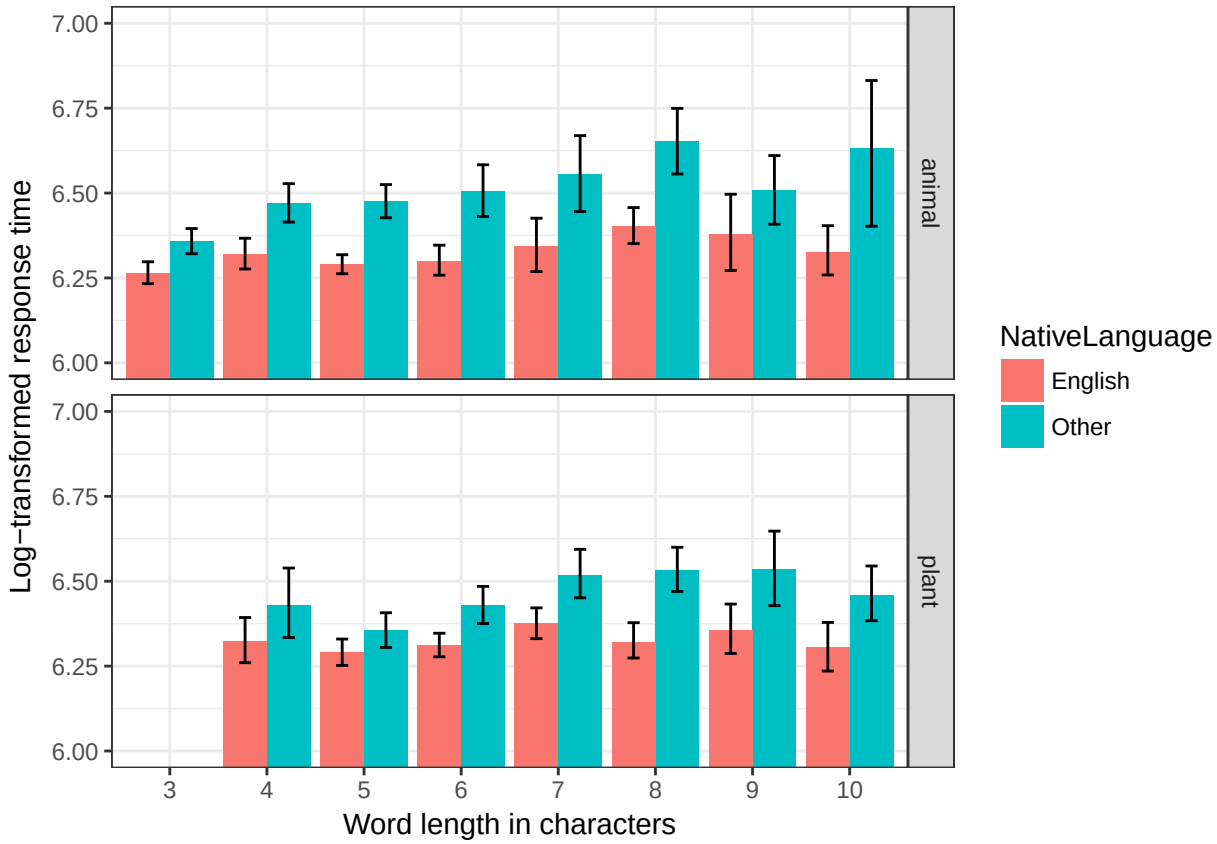
As before, we can add lots of information to this plot by specifying additional attributes. Suppose we want to keep track of the noun `Class` and `NativeLanguage`. We'll need to compute new means that take these factors into account.

```
d_s2 = bootsSummary(data=lexdec, measurevar="RT", groupvars=c("Length","Class","NativeLanguage"))
head(d_s2)
```

```
##   Length  Class NativeLanguage  N       RT bootsci_high bootsci_low
## 1      3 animal        English 96 6.265256     6.297642    6.233121
## 2      3 animal          Other 72 6.357708     6.395491    6.321298
## 3      4 animal        English 84 6.320975     6.366893    6.276386
## 4      4 animal          Other 63 6.469473     6.527673    6.414407
## 5      4  plant        English 36 6.324715     6.393062    6.260009
## 6      4  plant          Other 27 6.429661     6.538772    6.334252
```

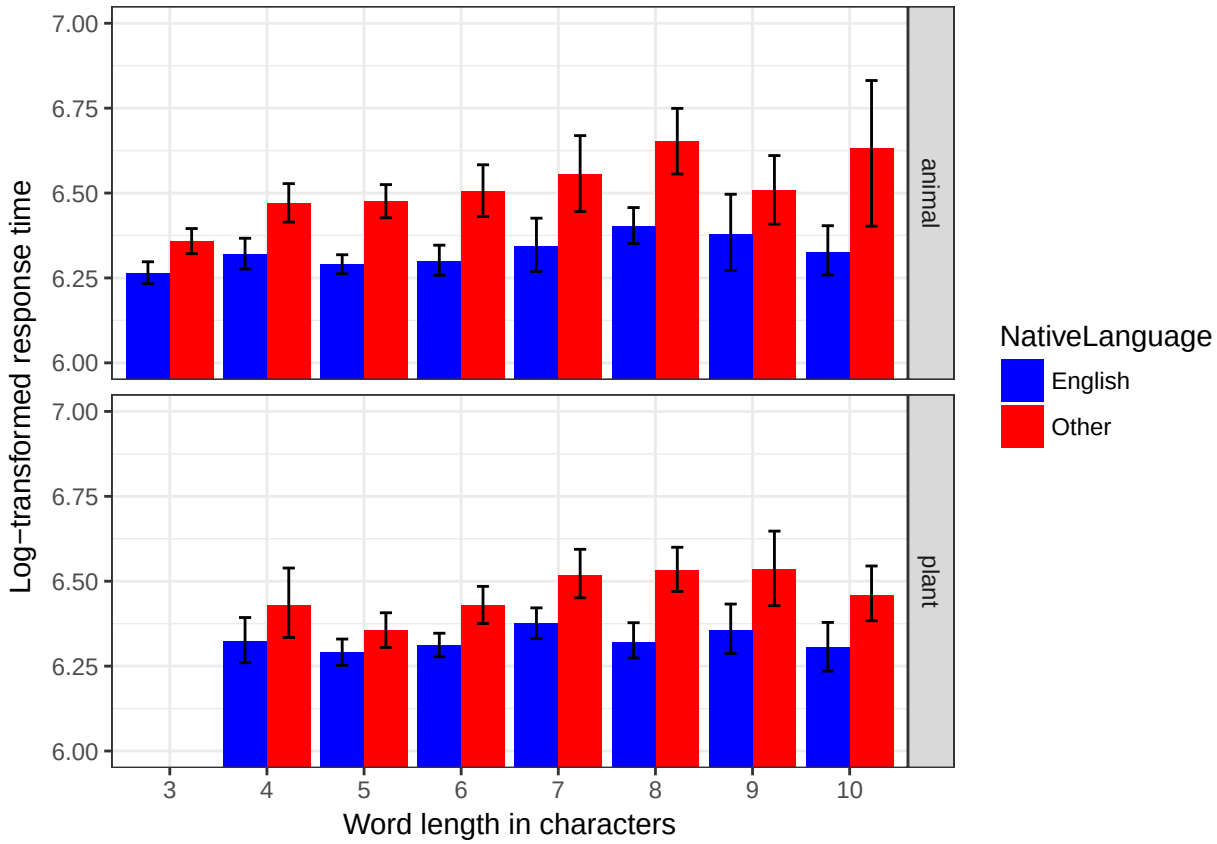
We can then use these values in a new barplot.

```
ggplot(d_s2,aes(x=as.factor(Length),y=RT,fill=NativeLanguage))+
  geom_bar(stat="identity", position=position_dodge()) +
  geom_errorbar(aes(ymin=bootsci_low, ymax=bootsci_high, x=as.factor(Length), width=0.25), position=pos
  coord_cartesian(ylim=c(6,7)) +
  xlab("Word length in characters") +
  ylab("Log-transformed response time") +
  facet_grid(Class~.)
```



You might want to further customize the plot attributes, for example with custom colors.

```
ggplot(d_s2,aes(x=as.factor(Length),y=RT,fill=NativeLanguage))+
  geom_bar(stat="identity", position=position_dodge()) +
  geom_errorbar(aes(ymin=bootsci_low, ymax=bootsci_high, x=as.factor(Length), width=0.25), position=pos
  coord_cartesian(ylim=c(6,7)) +
  scale_fill_manual(values=c("blue","red")) +
  xlab("Word length in characters") +
  ylab("Log-transformed response time") +
  facet_grid(Class~.)
```



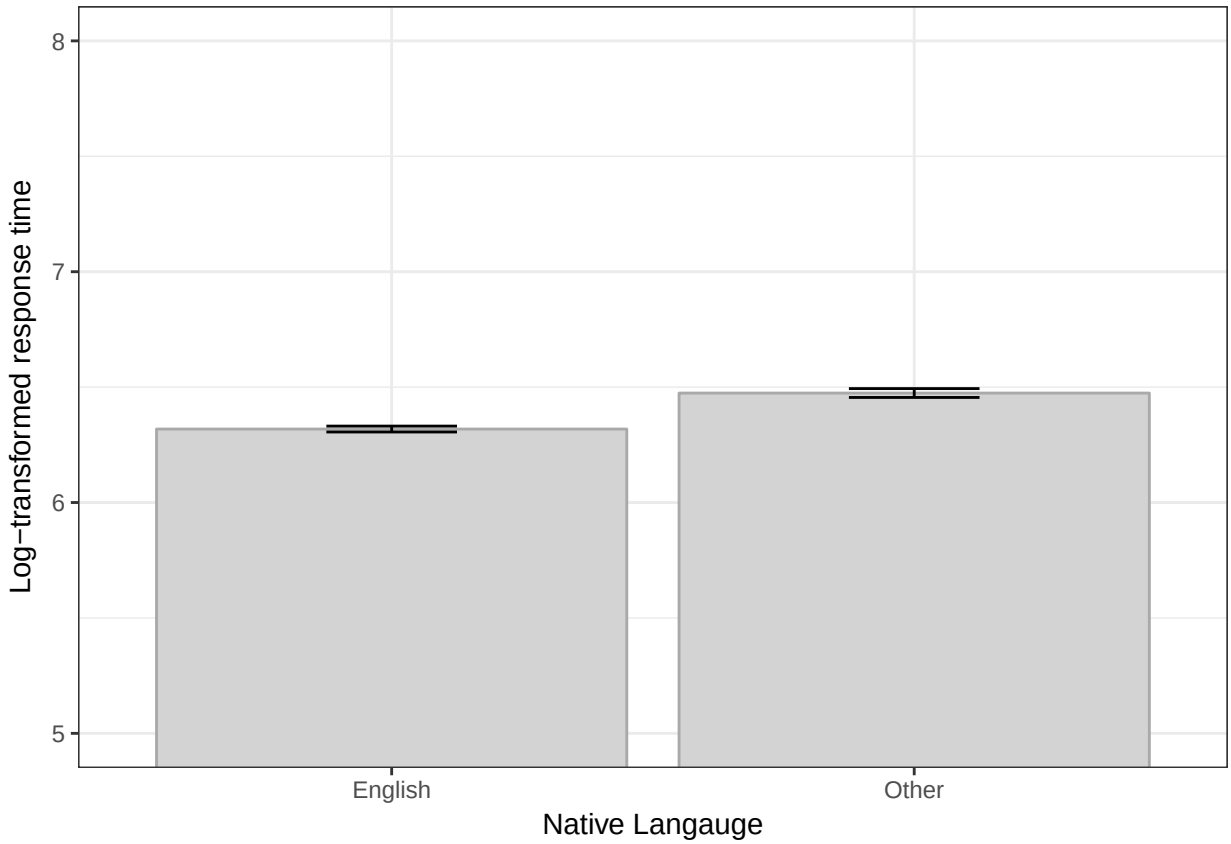
Let's simplify things a bit and focus in on the effect of `NativeLanguage` on `RT`.

```
d_s3 = bootsSummary(data=lexdec, measurevar="RT", groupvars=c("NativeLanguage"))
head(d_s3)
```

```
##   NativeLanguage   N       RT bootsci_high bootsci_low
## 1        English 948 6.318309     6.331196    6.305599
## 2          Other 711 6.474130     6.493613    6.455215
```

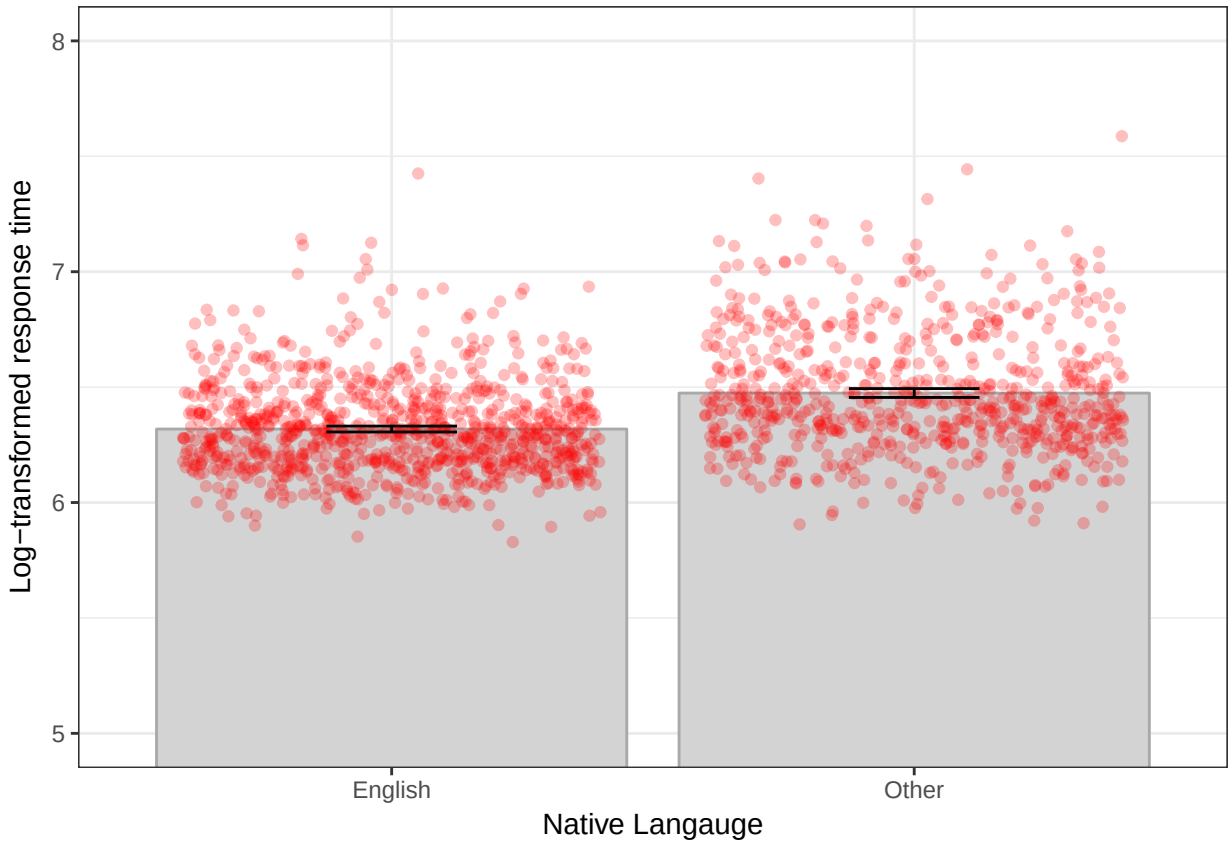
We can then visualize these values.

```
ggplot(d_s3,aes(x=NativeLanguage,y=RT))+
  geom_bar(stat="identity", position=position_dodge(), fill="lightgray",color="darkgray") +
  geom_errorbar(aes(ymin=bootsci_low, ymax=bootsci_high, x=NativeLanguage, width=0.25), position=positi
  coord_cartesian(ylim=c(5,8)) +
  xlab("Native Langauge") +
  ylab("Log-transformed response time")
```



But suppose in addition to the means and the confidence intervals, we want to get a sense of the actual observations that led to these values. We can plot these observations by adding a `geom_jitter` layer using the original `RT` values from the `lexdec` dataframe.

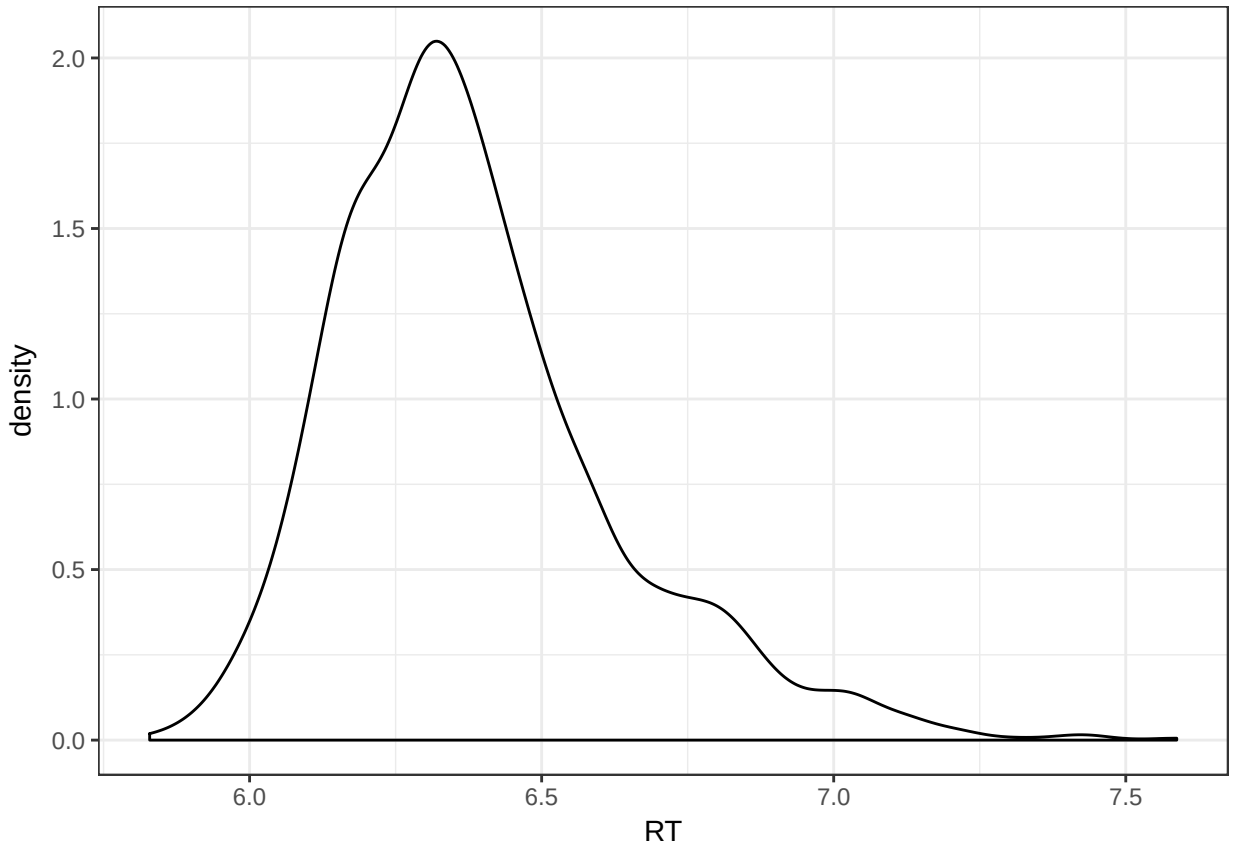
```
ggplot(d_s3,aes(x=NativeLanguage,y=RT))+
  geom_bar(stat="identity", position=position_dodge(), fill="lightgray",color="darkgray") +
  geom_jitter(data=lexdec,aes(y=RT),alpha=.25,color="red") +
  geom_errorbar(aes(ymin=bootsci_low, ymax=bootsci_high, x=NativeLanguage, width=0.25), position=positi
  coord_cartesian(ylim=c(5,8)) +
  xlab("Native Langauge") +
  ylab("Log-transformed response time")
```



A violin plot

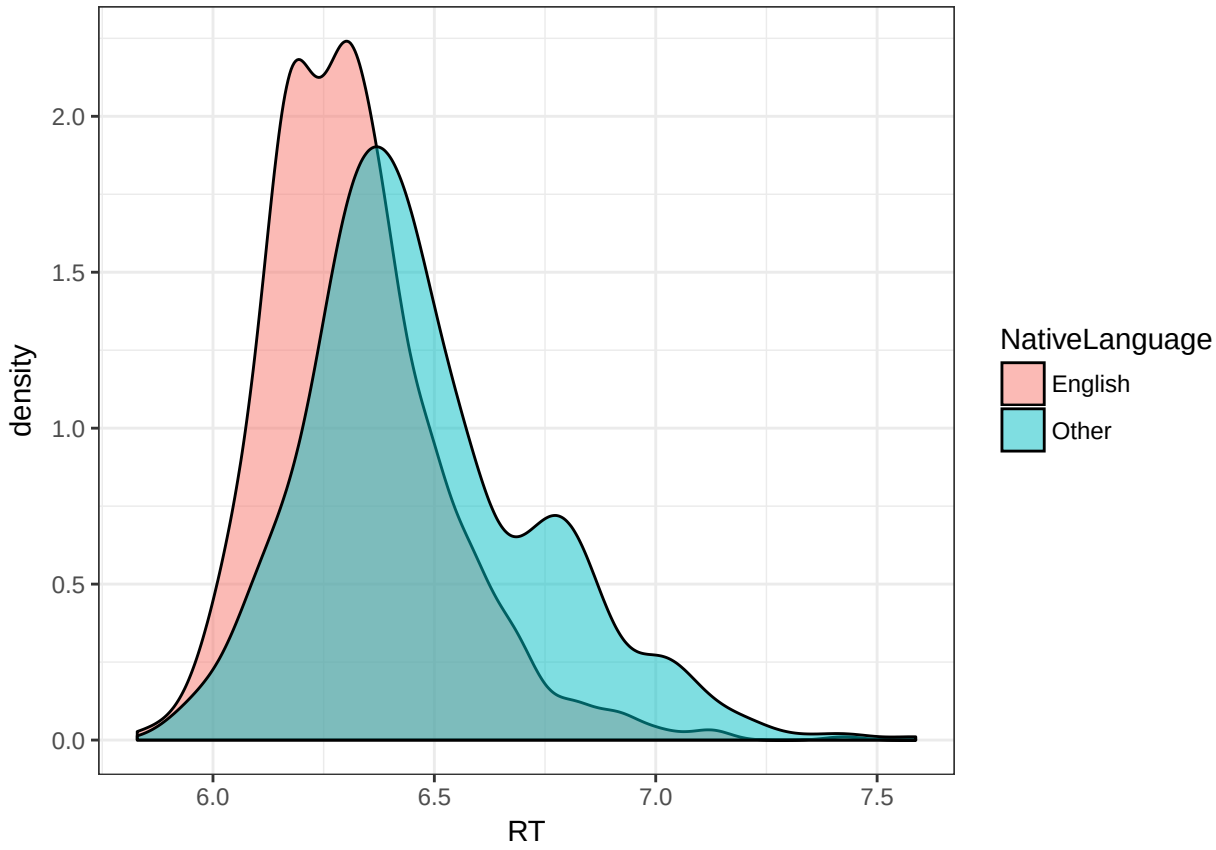
Violin plots provide another means of visualizing the distribution of responses. To understand violin plots, we start by visualizing the density of a distribution, in this case the distribution of RTs.

```
ggplot(lexdec,aes(x=RT))+
  geom_density()
```



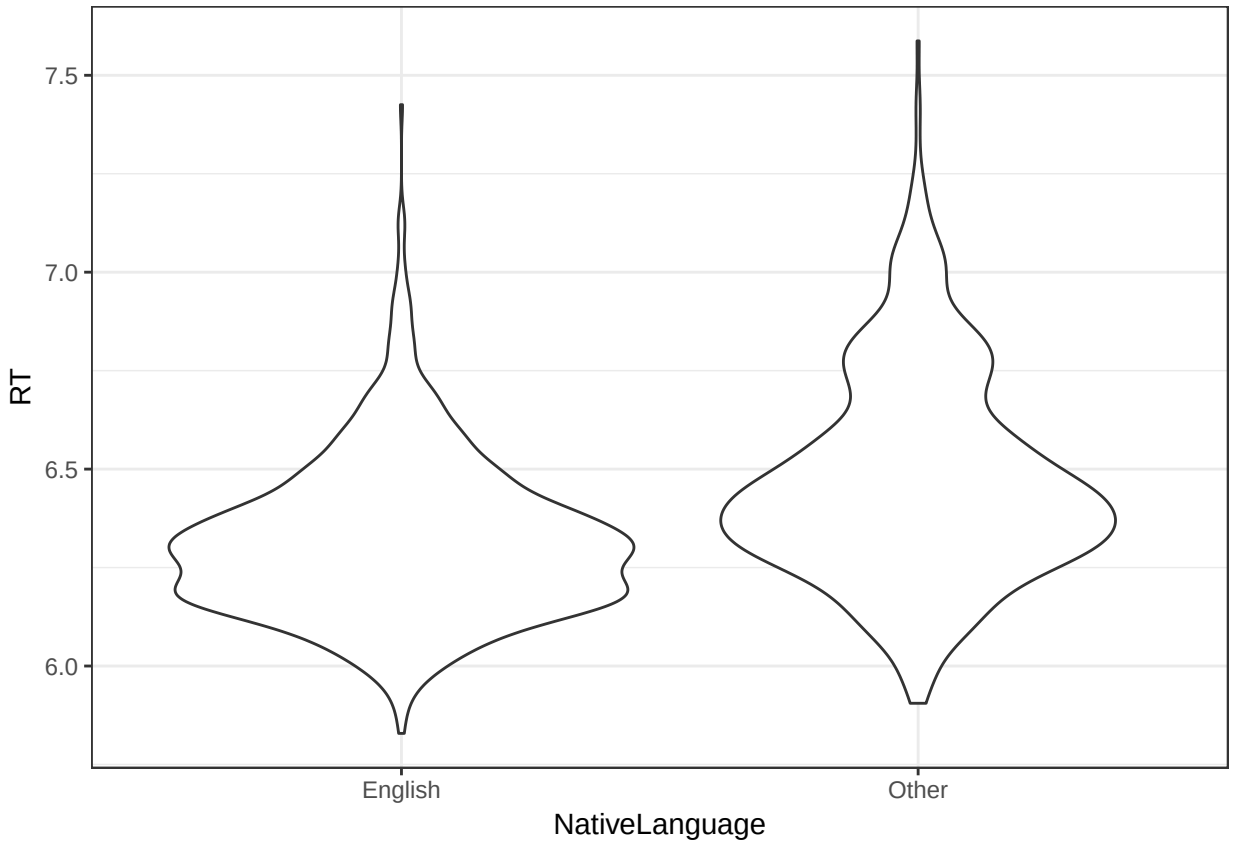
We can add information to this plot with additonal aesthetics.

```
ggplot(lexdec,aes(x=RT,fill=NativeLanguage))+
  geom_density(alpha=0.5)
```



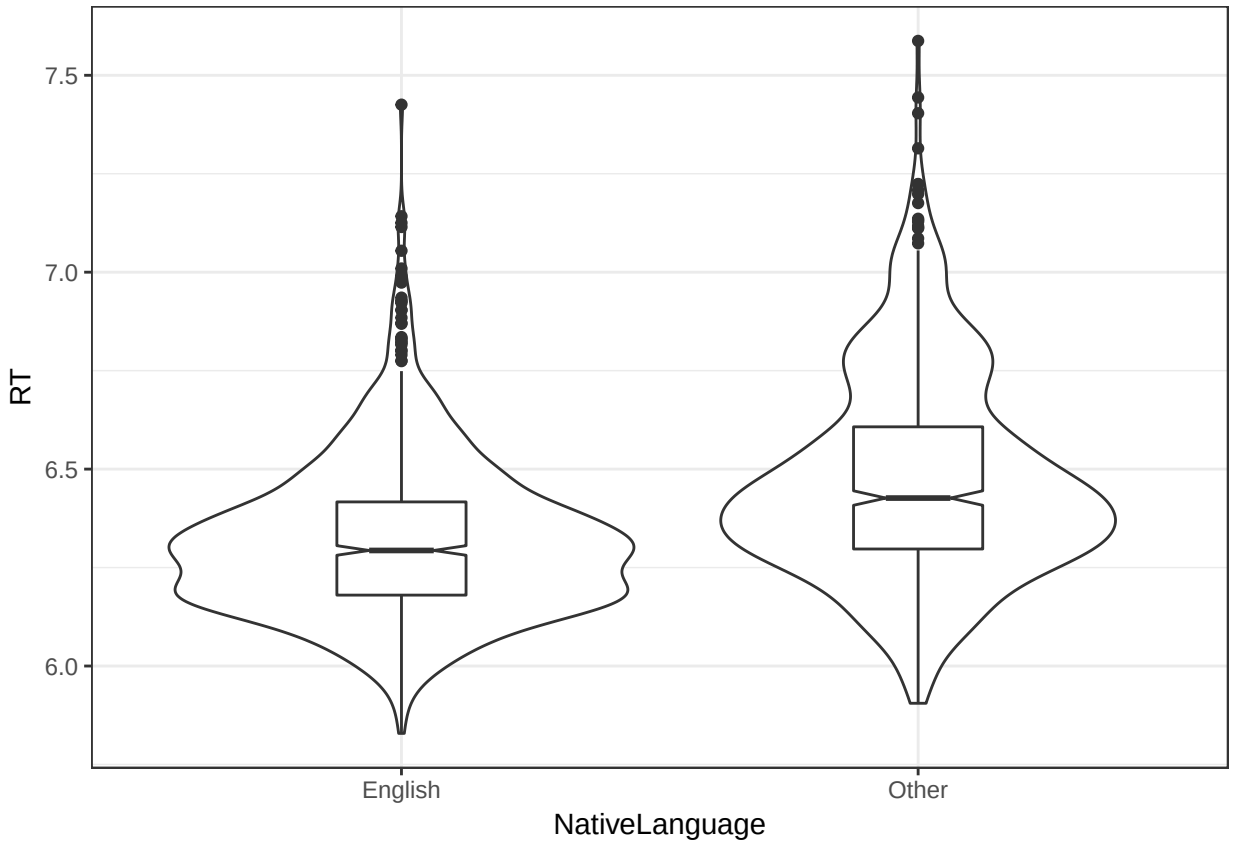
A violin plot effectively flips this information on its side:

```
ggplot(lexdec,aes(y=RT,x=NativeLanguage))+
  geom_violin()
```



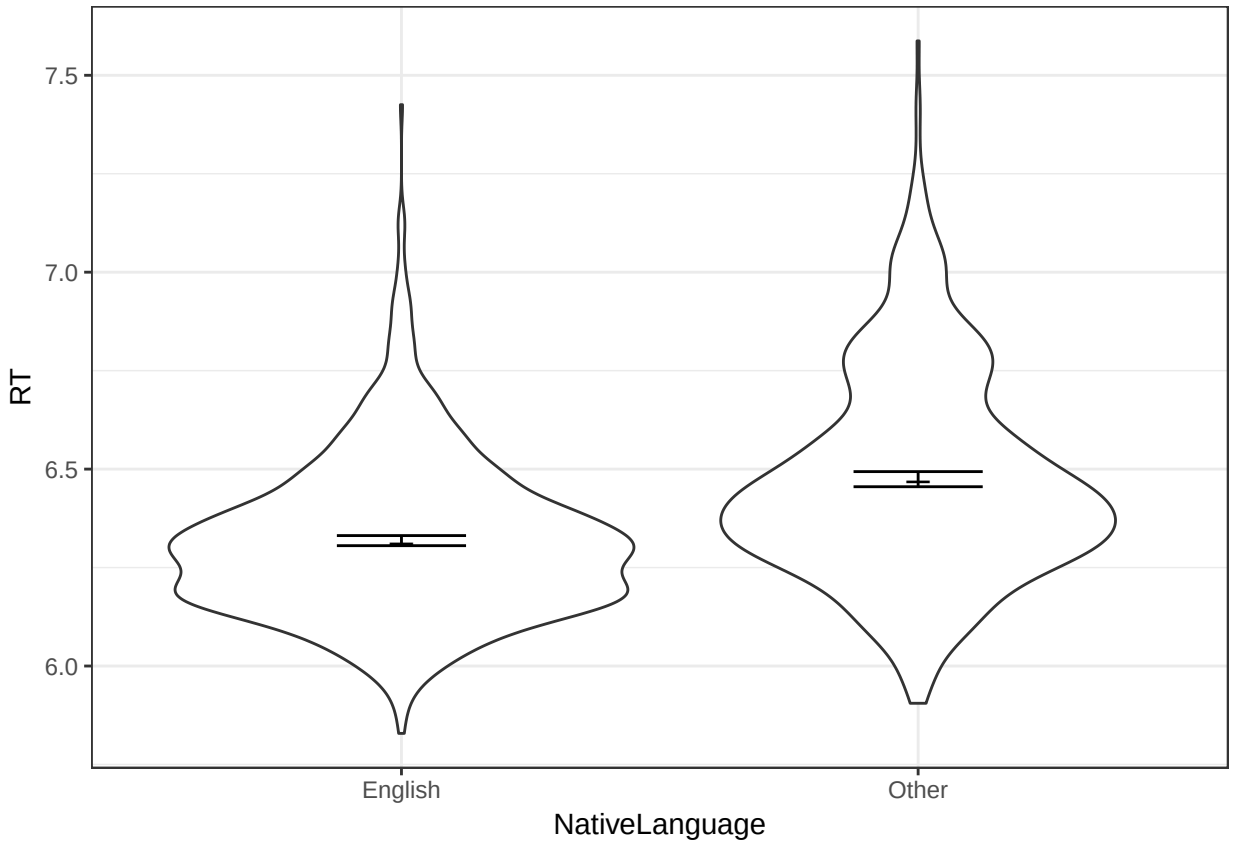
To add in information about quartiles and outliers, we can include a `geom_boxplot` layer.

```
ggplot(lexdec,aes(y=RT,x=NativeLanguage))+
  geom_violin() +
  geom_boxplot(notch=T,width=.25)
```



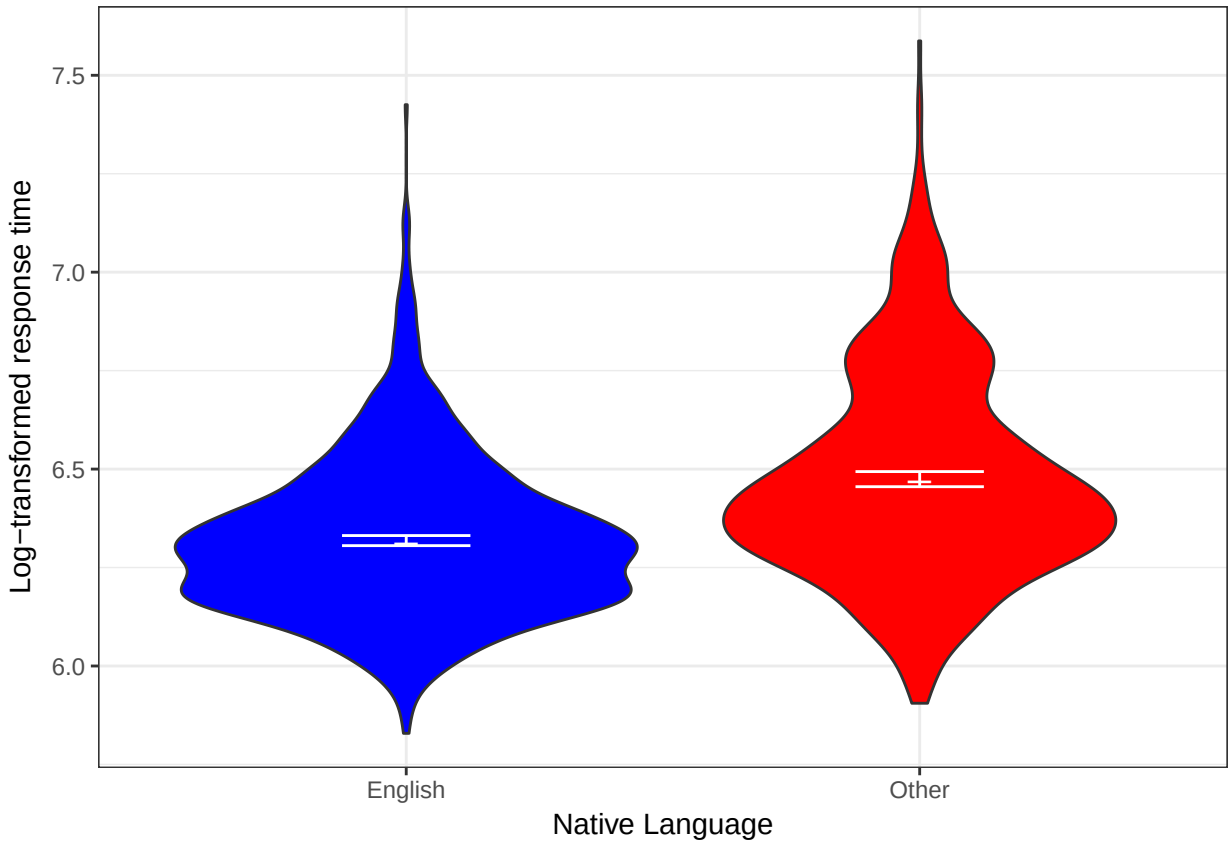
We could also add in information about the mean and confidence intervals; we've computed those values in `d_s3` above.

```
ggplot(lexdec,aes(y=RT,x=NativeLanguage))+
  geom_violin() +
  geom_point(data=d_s3,aes(y=RT,x=NativeLanguage),shape=95,size=5) +
  geom_errorbar(data=d_s3,aes(ymin=bootsci_low, ymax=bootsci_high, x=NativeLanguage, width=0.25))
```



And of course, we can customize the `color` information and other attributes.

```
ggplot(lexdec,aes(y=RT,x=NativeLanguage,fill=NativeLanguage))+
  geom_violin() +
  geom_point(data=d_s3,aes(y=RT,x=NativeLanguage),shape=95,size=5,color="white") +
  geom_errorbar(data=d_s3,aes(ymin=bootsci_low, ymax=bootsci_high, x=NativeLanguage, width=0.25),color=
  scale_fill_manual(values=c("blue","red")) +
  xlab("Native Language") +
  ylab("Log-transformed response time") +
  guides(fill=FALSE)
```


7.5
7.0
6.5
6.0
Log−transformed response time
English
Other
Native Language