Exploratory Data Analysis (EDA)

John Tukey wrote "Exploratory Data Analysis"

The process of analyzing data to uncover its key features

- Quality control
- Distribution of data } today
- Relationships among variables }
- Dimension reduction
- Model formulation } later
- Hypothesis generation }

Two things to think about:

① What is the structure of my variables?

② What kind of variables do I have?

Structures of variables:

① A single variable: observed data

$x_1, x_2, \dots, x_n$ n datapoints

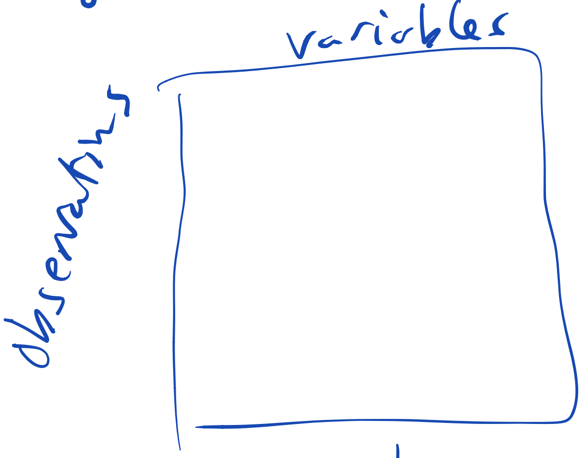
② n-tuple variables

Ex: 2-tuple

$(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$

n pairs of data points

in general, you have a data frame:



↳ n columns

⟹ n variables
n-tuples

③ High-dimensional data

m variables of the same type measered simutaneously for n observations

Kind of variables:

1. Quantitative variables
   - continuous — Ex microarray gene expression
   - discrete — Ex. RNA-seq gene expression

2. Categorical variables
   - ordered AA, AT, TT
   - unordered

Three topics:

1. Quantitative summaries
2. Dimension Reduction
3. Visualization

Quantitative Summaries

quantitative data

- Center
- Quantiles
- Spread
- Outliers
- Shape

single variables

- Concordance

2-tuples

Single variable, n observations

center

mean: $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$

median: $x_{(1)} \le x_{(2)} \le \dots \le x_{(n)}$

identify the middle

mode: the most frequent obs. value

distribution

percentiles: the number such that p% of the data is ≤ that value

quantiles: q-quantiles are cut points that divide the into q approximately equal sizes

five number summary!

min, Q1, median, Q3, max

↳ 25%-tile ↳ 75%-tile

→ 50%-tile

<u>spread</u>

sample variance

$$s^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}$$

sample standard deviation:

$s = \sqrt{s^2}$

Interquartile Range (IQR):

$Q3 - Q1$

robust to outliers

median + five number summary + IQR

vs

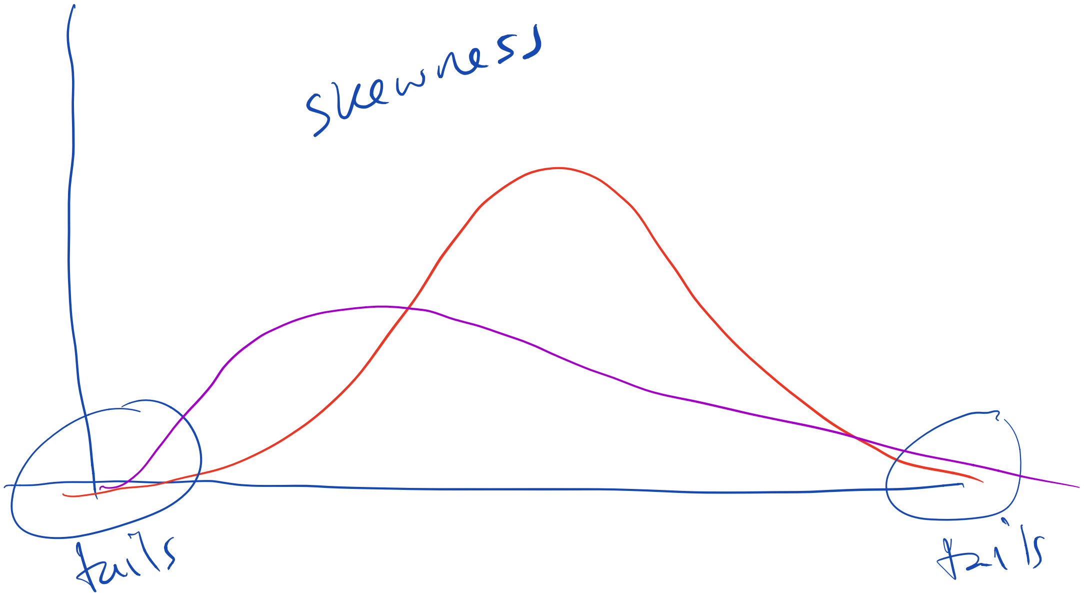
are not robust to outliers

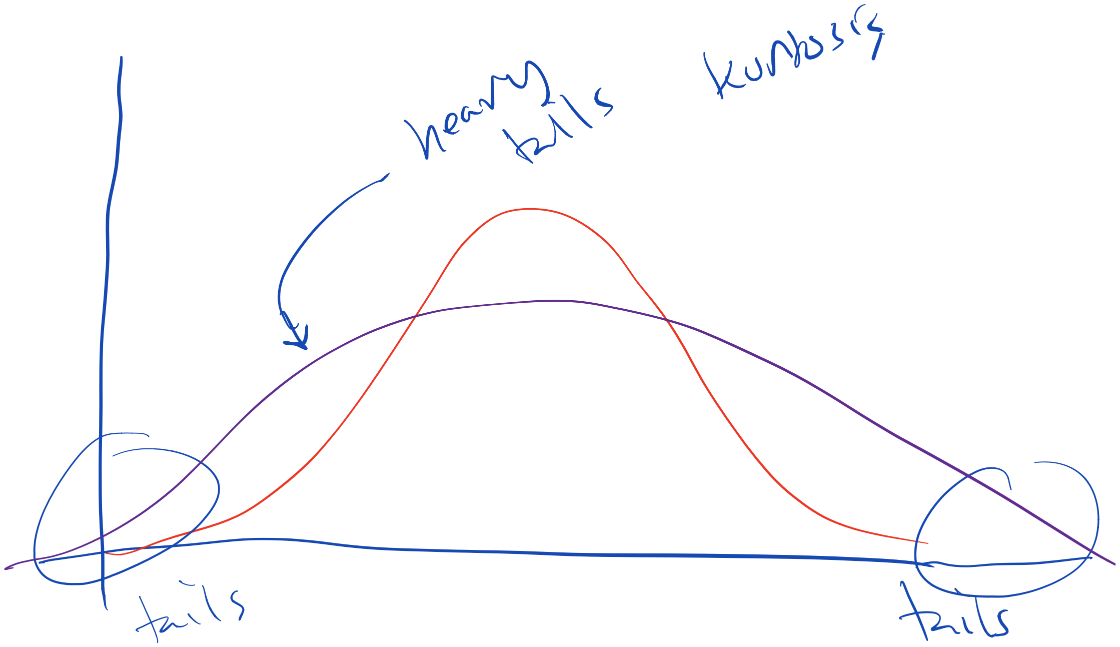
mean, std dev

Outliers:

data points outside of Q1 and Q3 by 1.5 IQR

Read skewness and kurtosis





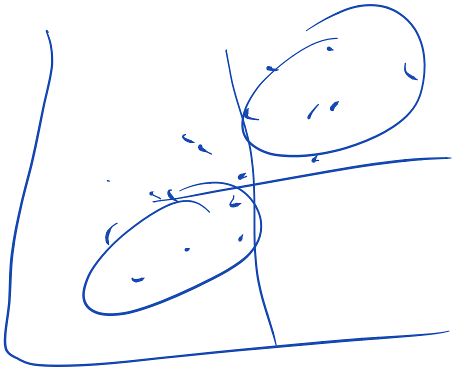
Covariance and correlation?

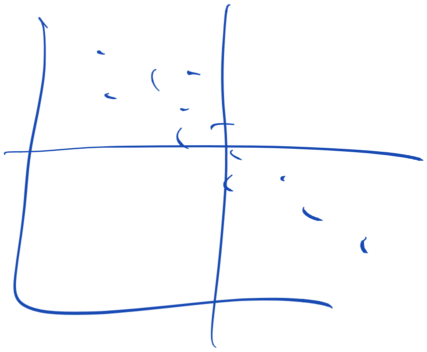
$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$

	x	y
obs 1		
obs 2		
⋮		
obs n		

$$\mathrm{cov}_{xy} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{n-1}$$

$$\mathrm{cov}_{xx} = s^2$$





Pearson correlation

$$r_{xy} = \frac{\operatorname{cov}_{xy}}{S_x S_y}$$

$$-1 \leq r_{xy} \leq 1$$

Spearman correlation

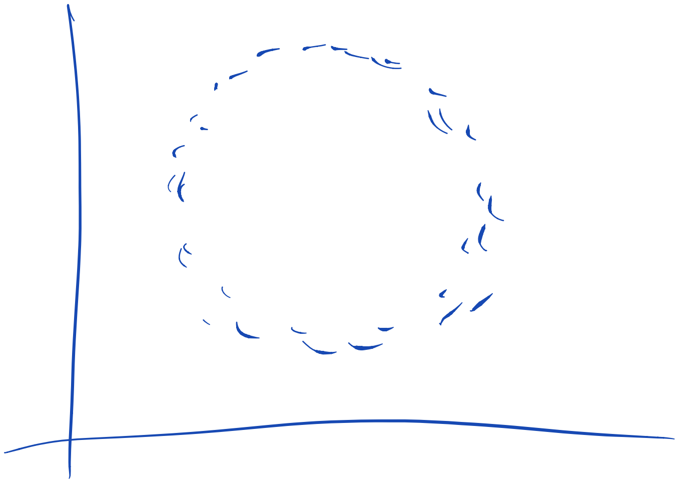
$$\begin{tikzcd}
x_{(1)} \arrow[d] & \leq & x_{(2)} \arrow[d] & \leq \cdots \leq & x_{(n)} \arrow[d] \\
1 & & 2 & & n
\end{tikzcd}$$

$$\begin{tikzcd}
y_{(1)} \arrow[d] & \leq & y_{(2)} \arrow[d] & \leq \cdots \leq & y_{(n)} \arrow[d] \\
1 & & 2 & & n
\end{tikzcd}$$

$$(x_1^*, y_1^*), \ldots, (x_n^*, y_n^*)$$

Calculate Pearson correlation

Check out "distance correlation"



EDA of High-Dimensional Data

Many variables of the same type (usually) measured simultaneously on the same individuals

$$\left.\begin{aligned} \mathbf{x}_1 &= (x_{11}, x_{12}, \dots, x_{1n}) \\ \mathbf{x}_2 &= (x_{21}, x_{22}, \dots, x_{2n}) \\ \vdots \quad & \quad \vdots \\ \mathbf{x}_m &= (x_{m1}, \dots, x_{mn}) \end{aligned}\right\} \mathbf{X}_{m \times n}$$

m variables

n observations

dimensionality reduction:

reducing m variables to smaller number to be able to analyze them

Principal Components Analysis: (PCA)

PCA finds (constrained) weighted sums of variables to produce new variables (called "principal components") that capture consecutively maximal levels of variation in the data

$$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$$

Find $\mathbf{u} = (u_1, u_2, \ldots, u_m)^T$

$$\boxed{\tilde{\mathbf{x}} = \sum_{i=1}^{m} u_i \mathbf{x}_i}$$

s.t. $\|u\|_2^2 = \sum_{i=1}^{m} u_i^2 = 1$

where $S^2_{\tilde{\mathbf{x}}}$ is maximal

This is PC1, principal component 1.

$\tilde{x}$ is PC1

u is the vector of loadings

$x_1 - \boxed{u_1 \tilde{x}} \rightarrow x_1$ w/ PC1 removed

There are $\min(m, n-1)$ PCs

$$x_{ij}^* = x_{ij} - \frac{1}{n}\sum_{k=1}^{n} x_{ik}$$

variable-wise mean centered data

$$\boxed{\tilde{x}^* = \sum_{i=1}^{m} u_i x^*_i} \quad - \; s^2_{\tilde{x}^*}$$

$$s^2_{\tilde{x}} = s^2_{\tilde{x}^*}$$

Actually $\tilde{x}^*$ is PC1

$X^*_{m\times n}$ mean centered data

Calculate covariance matrix

$$S_{m\times m} = \frac{1}{n-1} X^* X^{*T}$$

(i,j) entry is

$$s_{ij} = \frac{\sum_{k=1}^{n} (x_{ik} - \bar{x}_i)(x_{jk} - \bar{x}_j)}{n-1}$$

$s^2_{\tilde{x}^*} = u^T S u$ quadratic form

Find u that maximizes $u^T S u$

Use Lagrange multiplier and maximize:

$$u^T S u + \lambda(u^T u - 1)$$

Solution s.t. $Su = \lambda u$

Eigendecomposition of S finds the solutions to

$$Su = \lambda u$$

$$S_{m \times m} = U_{m \times q} \Lambda_{q \times q} U^T_{q \times m}$$

Λ diagonal

$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_q$

$q = \min(m, n-1)$

PC1: $s^2_{\tilde{x}*} = u_1^T S u_1 = \lambda_1$

- $UU^T = I$
- column j of U, u_j Is such that

  $Su_j = \lambda_j u_j$
- $\|u_j\|_2^2 = 1$, $u_j^T u_k = 0$ for $\lambda_j \neq \lambda_k$

- sample variance of

  $u_j^T x^* = \lambda_j$

- sample variance of

  $u_1^T x^* \geq u_2^T x^* \geq \dots \geq u_q^T x^*$

- $S = \sum_{j=1}^{m} \lambda_j u_j u_j^T \swarrow$

- Covariance of

  $\underline{u_j^T x^*}$ and $\underline{u_k^T x^*}$ is 0

PC$_j$ is $u_j^T x^*$

Variance is λ_j

$$PVE_j = \frac{\lambda_j}{\sum \lambda_k}$$

Say m is huge

1) X^* (mean centered)

diagonal D

2) SVD on $\frac{1}{\sqrt{n-1}} X^* = U\overset{\downarrow}{D}V^T$

- U is the matrix of loadings where each column j is the loadings for PC j.
- Row j of $DV^T_{q \times n}$ is PC j

$$u_1^T X^* = u_1^T U D V^T$$

$$= \begin{pmatrix} 1 & & 0 \\ & 0 & \\ 0 & 0 & 0 \; 0 \end{pmatrix} D V^T$$

Jolliffe

$$= \text{row } 1 \text{ of } DV^T$$

$$D = \begin{pmatrix} d_1 & & & 0 \\ & d_2 & & \\ & & \ddots & \\ 0 & & & d_q \end{pmatrix} \qquad d_i = \sqrt{\lambda_i}$$

$$PVE_i = \frac{d_i^2}{\sum d_k^2}$$

A Simple PCA Function

```
> pca <- function(x, space=c("rows", "columns"),
+                 center=TRUE, scale=FALSE) {
+   space <- match.arg(space)
+   if(space=="columns") {x <- t(x)}
+   x <- t(scale(t(x), center=center, scale=scale))
+   x <- x/sqrt(nrow(x)-1)
+   s <- svd(x)
+   loading <- s$u
+   colnames(loading) <- paste0("Loading", 1:ncol(loading))
+   rownames(loading) <- rownames(x)
+   pc <- diag(s$d) %*% t(s$v)
+   rownames(pc) <- paste0("PC", 1:nrow(pc))
+   colnames(pc) <- colnames(x)
+   pve <- s$d^2 / sum(s$d^2)
+   if(space=="columns") {pc <- t(pc); loading <- t(loading)}
+   return(list(pc=pc, loading=loading, pve=pve))
+ }
```

The input is as follows:

- x: a matrix of numerical values
- space: either "rows" or "columns", denoting which dimension contains the variables
- center: if TRUE then the variables are mean centered before calculating PCs
- scale: if TRUE then the variables are std dev scaled before calculating PCs

The output is a list with the following items:

- pc: a matrix of all possible PCs
- loading: the weights or "loadings" that determined each PC
- pve: the proportion of variation explained by each PC

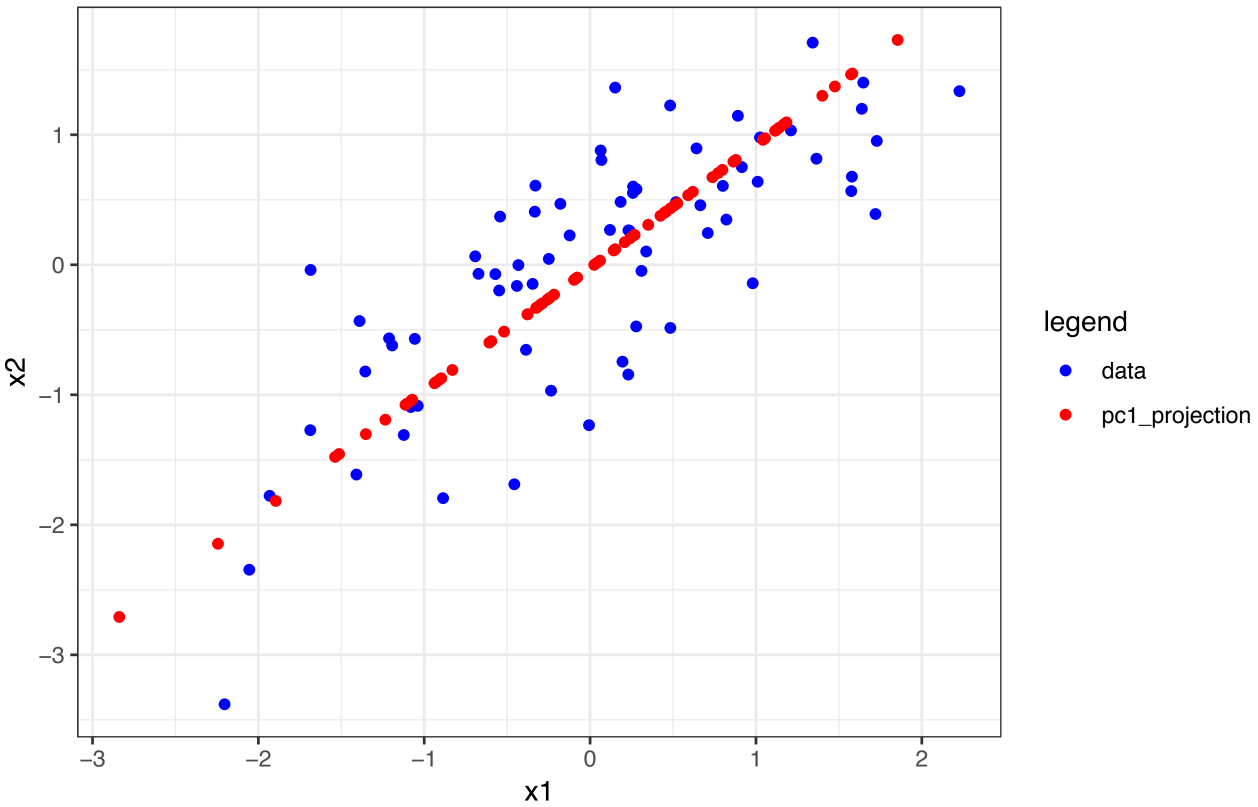
Note that the rows or columns of pc and loading have names to let you know on which dimension the values are organized.

The Ubiquitous PCA Example

Here's an example very frequently encountered to explain PCA, but it's slightly complicated and conflates several ideas in PCA. I think it's not a great example to motivate PCA, but it's so common I want to carefully clarify what it's displaying.

```
> set.seed(508)
> n <- 70
> z <- sqrt(0.8) * rnorm(n)
> x1 <- z + sqrt(0.2) * rnorm(n)
> x2 <- z + sqrt(0.2) * rnorm(n)
> X <- rbind(x1, x2)
> p <- pca(x=X, space="rows")
```

PCS is often explained by showing the following plot and stating, "The first PC finds the direction of maximal variance in the data…"



The above figure was made with the following code:

```
> a1 <- p$loading[1,1] * p$pc[1,]
> a2 <- p$loading[2,2] * p$pc[2,]
> df <- data.frame(x1=c(x1, a1),
+                  x2=c(x2, a2),
+                  legend=c(rep("data",n),rep("pc1_projection",n)))
```

```
> ggplot(df) + geom_point(aes(x=x1,y=x2,color=legend)) +
+   scale_color_manual(values=c("blue", "red"))
```

The red dots are therefore the projection of `x1` and `x2` onto the first PC, so they are neither the loadings nor the PC. This is rather complicated to understand before loadings and PCs are full understood.

Note that there are several ways to calculate these projections.

```
# all equivalent ways to get a1
p$loading[1,1] * p$pc[1,]
outer(p$loading[,1], p$pc[1,])[1,] + mean(x1)
lm(x1 ~ p$pc[1,])$fit # and

# all equivalent ways to get a2
p$loading[2,2] * p$pc[2,]
outer(p$loading[,1], p$pc[1,])[2,] + mean(x2)
lm(x2 ~ p$pc[1,])$fit
```

We haven't seen the `lm()` function yet, but once we do this example will be useful to revisit to understand what is meant by "projection".

Weather Data

These daily temperature data (in tenths of degrees C) come from meteorological observations for weather stations in the US for the year 2012 provided by NOAA (National Oceanic and Atmospheric Administration).:

```
> load("./data/weather_data.RData")
> dim(weather_data)
[1] 2811   50
>
> weather_data[1:5, 1:7]
                  11       16  18       19  27  30       31
AG000060611 138.0000 175.0000 173 164.0000 218 160 163.0000
AGM00060369 158.0000 162.0000 154 159.0000 165 125 171.0000
AGM00060425 272.7619 272.7619 152 163.0000 163 108 158.0000
AGM00060444 128.0000 102.0000 100 111.0000 125  33 125.0000
AGM00060468 105.0000 122.0000  97 263.5714 155  52 263.5714
```

This matrix contains temperature data on 50 days and 2811 stations that were randomly selected.

First, we will convert temperatures to Fahrenheit:

```
> weather_data <- 0.18*weather_data + 32
> weather_data[1:5, 1:6]
                  11       16    18       19    27    30
AG000060611 56.84000 63.50000 63.14 61.52000 71.24 60.80
AGM00060369 60.44000 61.16000 59.72 60.62000 61.70 54.50
AGM00060425 81.09714 81.09714 59.36 61.34000 61.34 51.44
AGM00060444 55.04000 50.36000 50.00 51.98000 54.50 37.94
AGM00060468 50.90000 53.96000 49.46 79.44286 59.90 41.36
>
> apply(weather_data, 1, median) %>%
+   quantile(probs=seq(0,1,0.1))
        0%        10%        20%        30%        40%        50%
  8.886744  49.010000  54.500000  58.460000  62.150000  65.930000
       60%        70%        80%        90%       100%
 69.679318  73.490000  77.990000  82.940000 140.000000
```

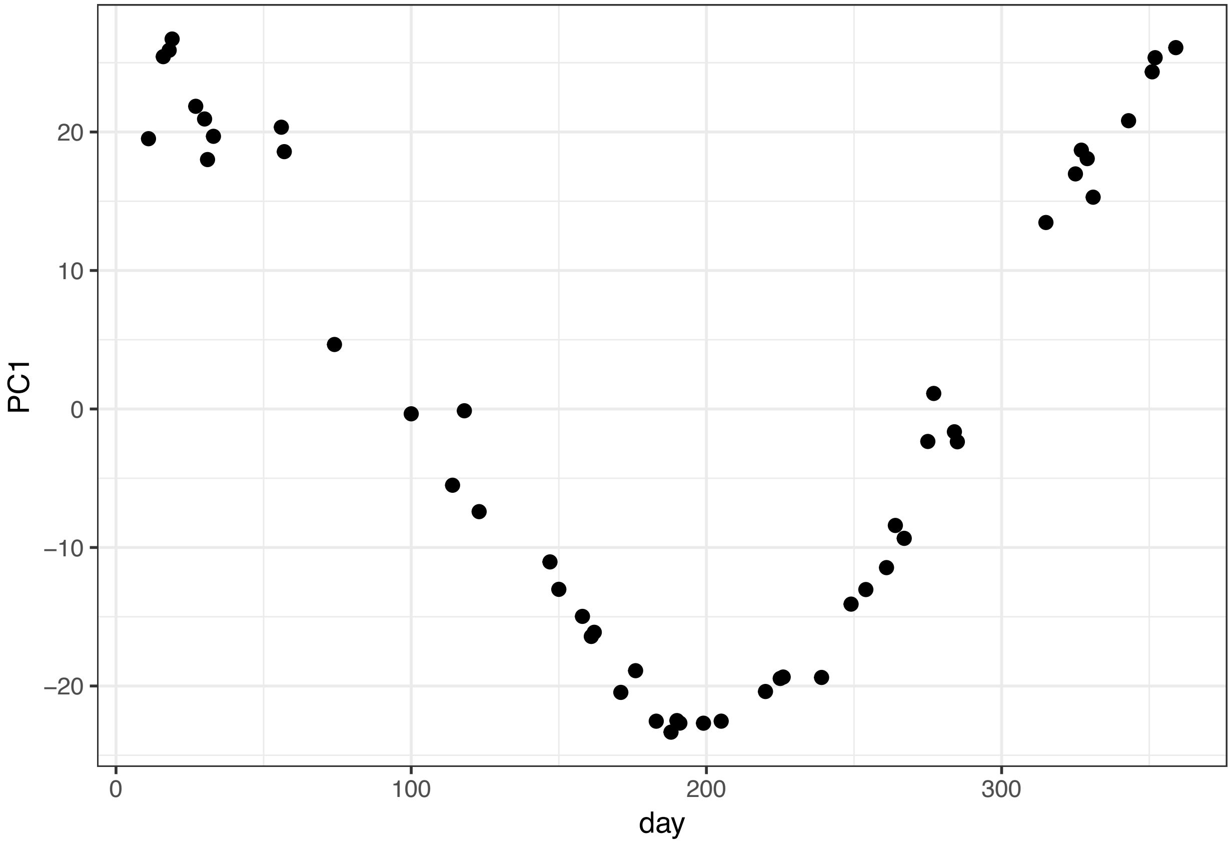
Let's perform PCA on these data.

```
> mypca <- pca(weather_data, space="rows")
>
> names(mypca)
[1] "pc"      "loading" "pve"
> dim(mypca$pc)
[1] 50 50
> dim(mypca$loading)
[1] 2811   50
```

```
> mypca$pc[1:3, 1:3]
            11        16         18
PC1 19.5166741 25.441401 25.9023874
PC2 -2.6025225 -4.310673  0.9707207
PC3 -0.6681223 -1.240748 -3.7276658
> mypca$loading[1:3, 1:3]
                Loading1    Loading2     Loading3
AG000060611 -0.015172744 0.013033849 -0.011273121
AGM00060369 -0.009439176 0.016884418 -0.004611284
AGM00060425 -0.015779138 0.007026312 -0.009907972
```

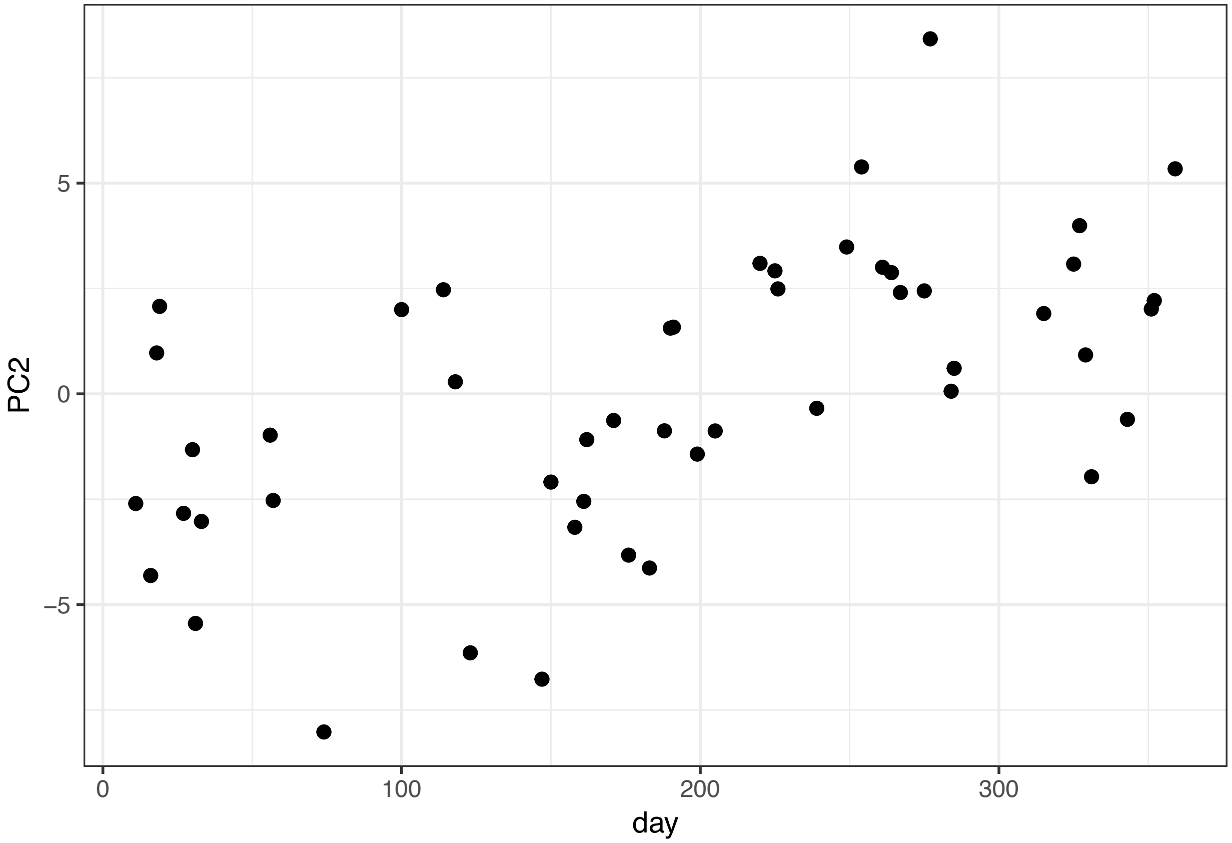
PC1 vs Time:

```
> day_of_the_year <- as.numeric(colnames(weather_data))
> data.frame(day=day_of_the_year, PC1=mypca$pc[1,]) %>%
+   ggplot() + geom_point(aes(x=day, y=PC1), size=2)
```



PC2 vs Time:

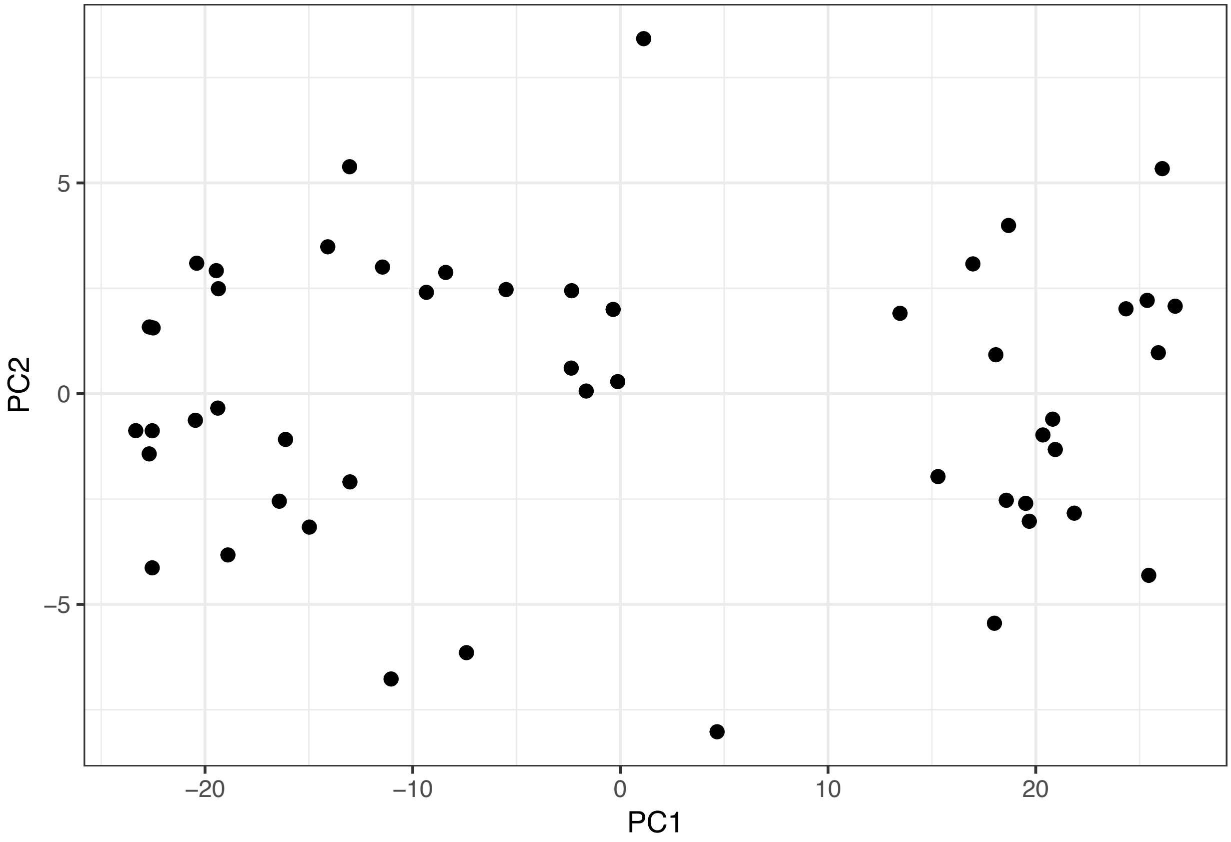
```
> data.frame(day=day_of_the_year, PC2=mypca$pc[2,]) %>%
+   ggplot() + geom_point(aes(x=day, y=PC2), size=2)
```



PC1 vs PC2 Biplot:

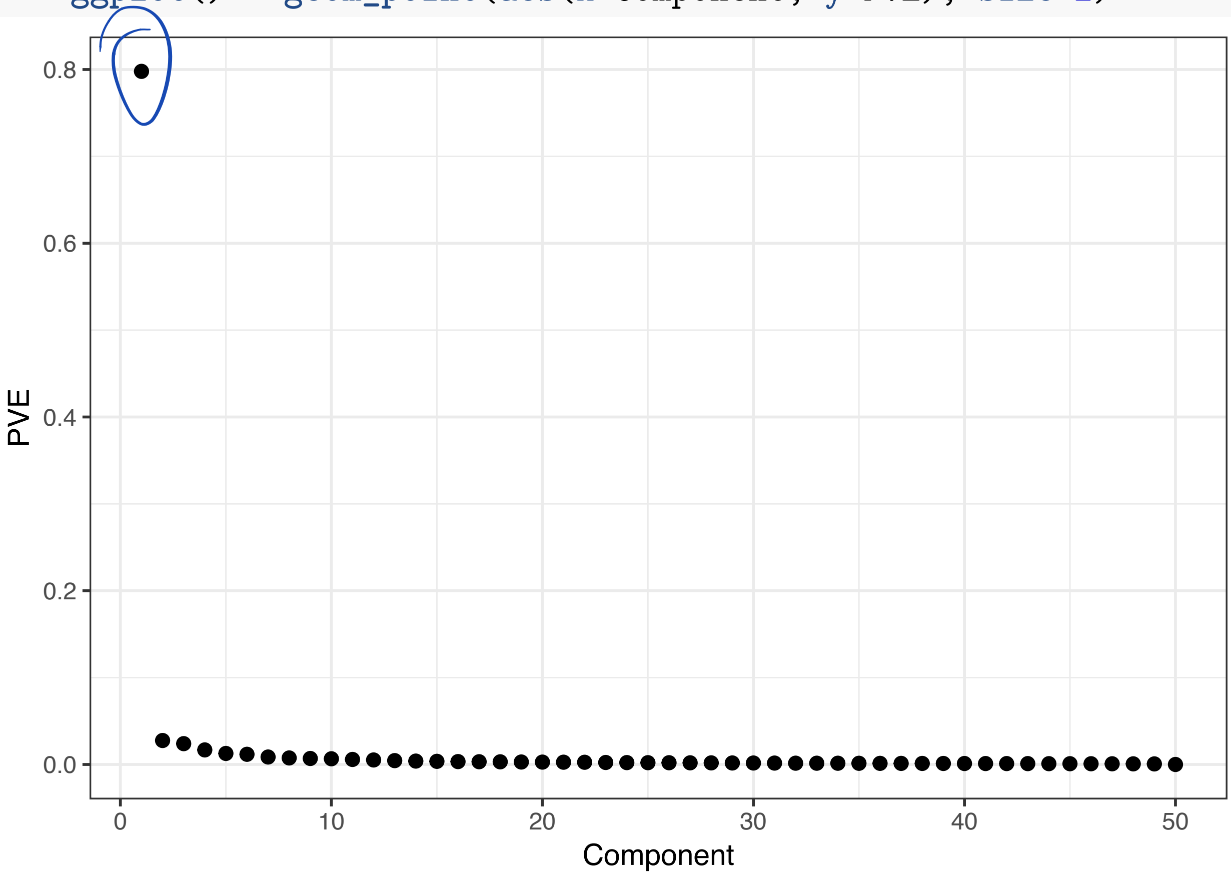
This does not appear to be subgroups or clusters in the weather data set biplot of PC1 vs PC2.

```
> data.frame(PC1=mypca$pc[1,], PC2=mypca$pc[2,]) %>%
+   ggplot() + geom_point(aes(x=PC1, y=PC2), size=2)
```



Proportion of Variance Explained:

```
> data.frame(Component=1:length(mypca$pve), PVE=mypca$pve) %>%
+   ggplot() + geom_point(aes(x=Component, y=PVE), size=2)
```



We can multiple the loadings matrix by the PCs matrix to reproduce the data:

```
> # mean centered weather data
> weather_data_mc <- weather_data - rowMeans(weather_data)
>
> # difference between the PC projections and the data
> # the small sum is just machine imprecision
> sum(abs(weather_data_mc/sqrt(nrow(weather_data_mc)-1) -
+           mypca$loading %*% mypca$pc))
[1] 1.329755e-10
```

The sum of squared weights – i.e., loadings – equals one for each component:

```
> sum(mypca$loading[,1]^2)
[1] 1
>
> apply(mypca$loading, 2, function(x) {sum(x^2)})
 Loading1  Loading2  Loading3  Loading4  Loading5  Loading6  Loading7
        1         1         1         1         1         1         1
 Loading8  Loading9 Loading10 Loading11 Loading12 Loading13 Loading14
```

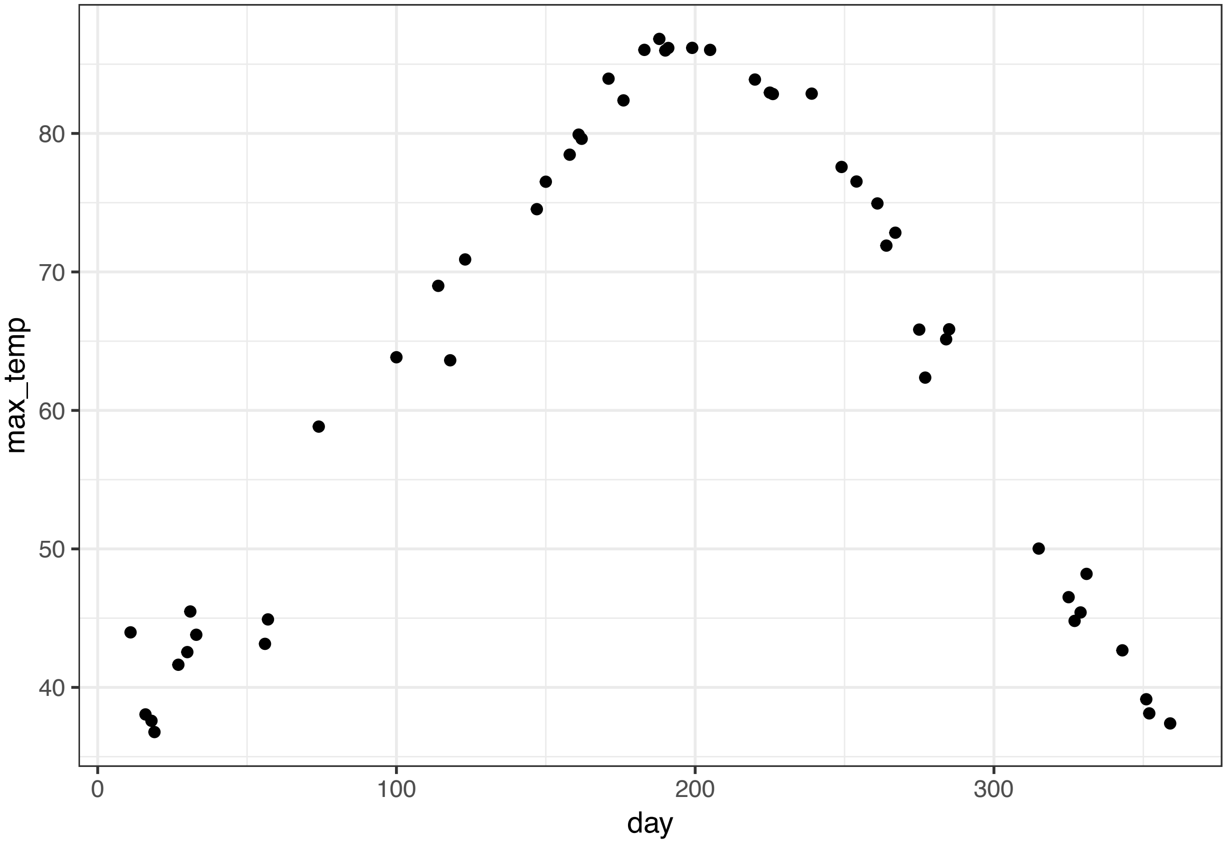
```
        1         1         1         1         1         1         1 
Loading15 Loading16 Loading17 Loading18 Loading19 Loading20 Loading21 
        1         1         1         1         1         1         1 
Loading22 Loading23 Loading24 Loading25 Loading26 Loading27 Loading28 
        1         1         1         1         1         1         1 
Loading29 Loading30 Loading31 Loading32 Loading33 Loading34 Loading35 
        1         1         1         1         1         1         1 
Loading36 Loading37 Loading38 Loading39 Loading40 Loading41 Loading42 
        1         1         1         1         1         1         1 
Loading43 Loading44 Loading45 Loading46 Loading47 Loading48 Loading49 
        1         1         1         1         1         1         1 
Loading50 
        1 
```

PCs by contruction have sample correlation equal to zero:

```
> cor(mypca$pc[1,], mypca$pc[2,])
[1] 3.135149e-17
> cor(mypca$pc[1,], mypca$pc[3,])
[1] 2.273613e-16
> cor(mypca$pc[1,], mypca$pc[12,])
[1] -1.231339e-16
> cor(mypca$pc[5,], mypca$pc[27,])
[1] -2.099516e-17
> # etc...
```

I can transform the top PC back to the original units to display it at a scale that has a more direct interpretation.

```
> day_of_the_year <- as.numeric(colnames(weather_data))
> y <- -mypca$pc[1,] + mean(weather_data)
> data.frame(day=day_of_the_year, max_temp=y) %>%
+   ggplot() + geom_point(aes(x=day, y=max_temp))
```



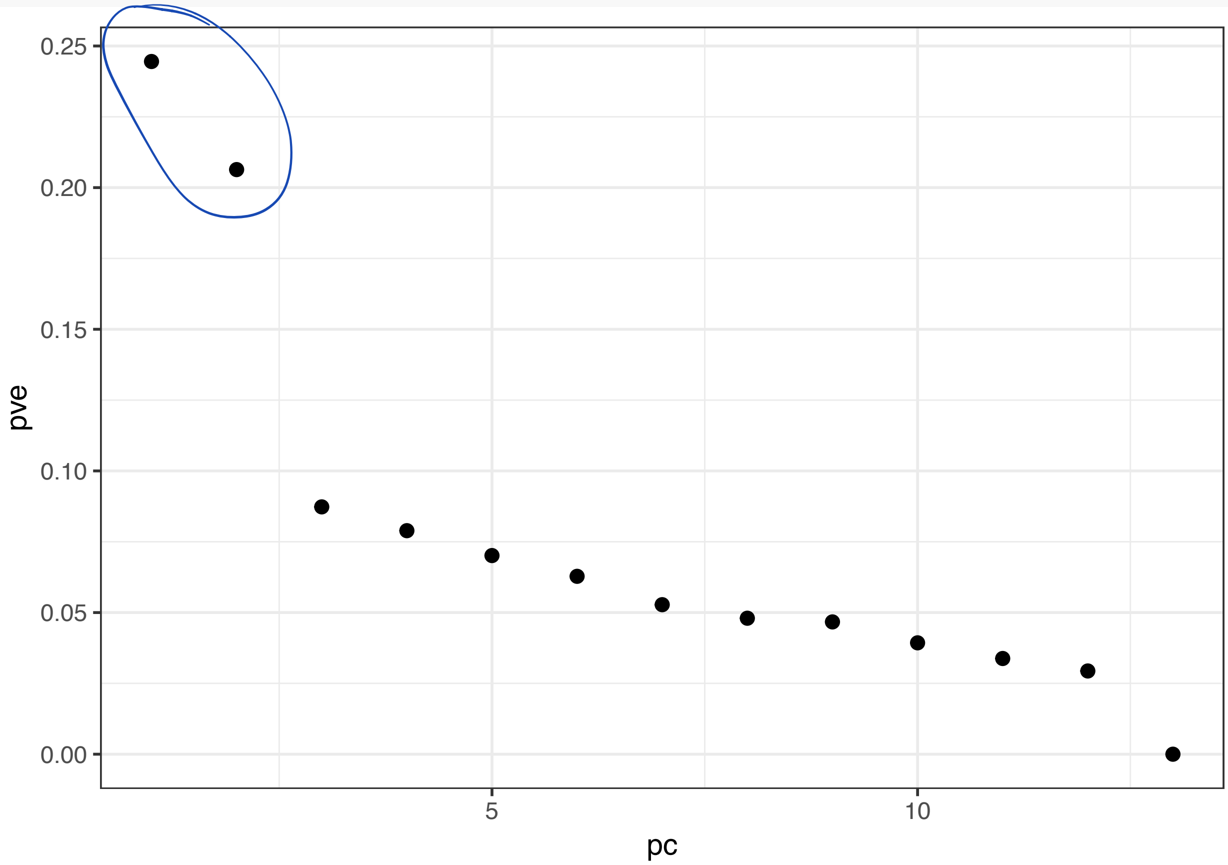
Yeast Gene Expression

Yeast cells were synchronized so that they were on the same approximate cell cycle timing in Spellman et al. (1998). The goal was to understand how gene expression varies over the cell cycle from a genome-wide perspective.

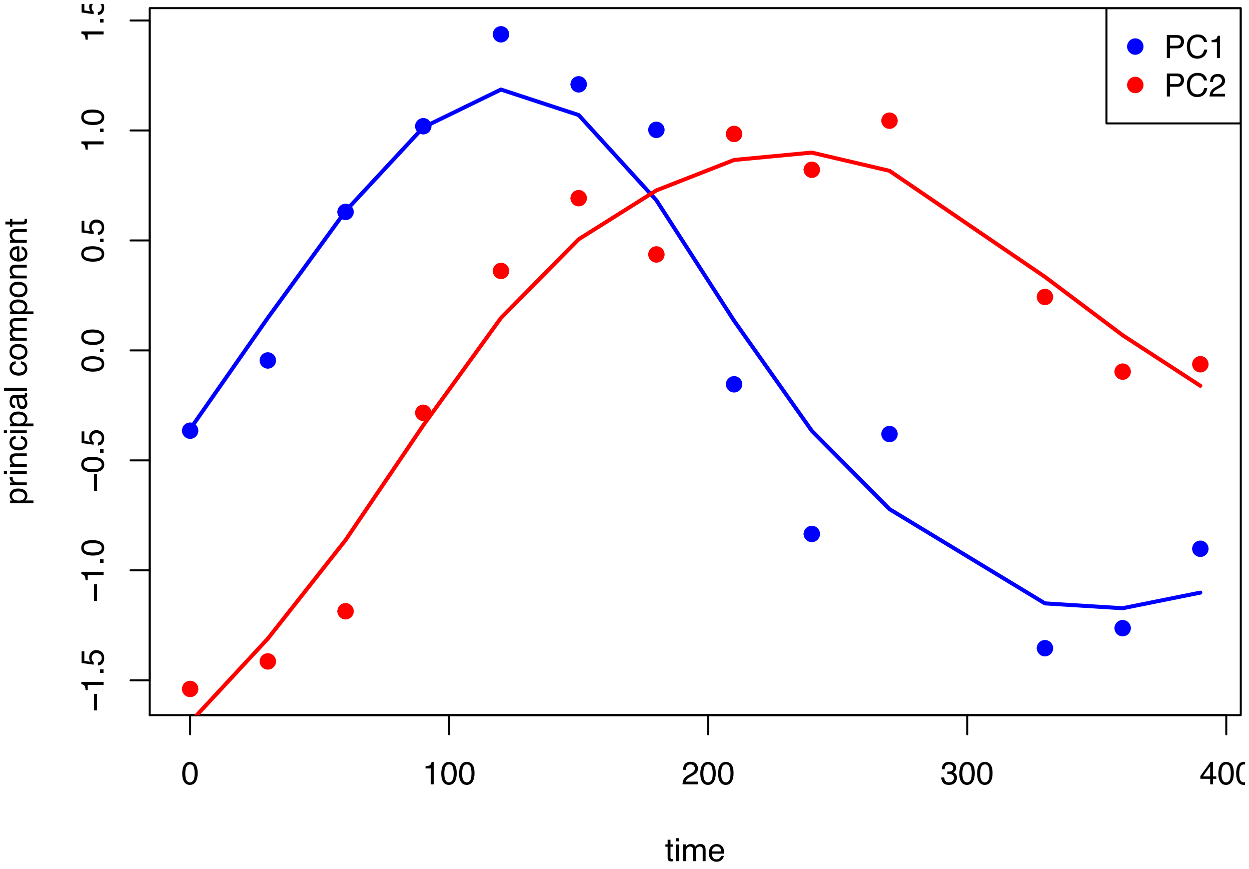
```
> load("./data/spellman.RData")
> time
 [1]   0  30  60  90 120 150 180 210 240 270 330 360 390
> dim(gene_expression)
[1] 5981   13
> gene_expression[1:6,1:5]
                  0         30         60        90        120
YAL001C  0.69542786 -0.4143538  3.2350520 1.6323737 -2.1091820
YAL002W -0.01210662  3.0465649  1.1062193 4.0591467 -0.1166399
YAL003W -2.78570526 -1.0156981 -2.1387564 1.9299681  0.7797033
YAL004W  0.55165887  0.6590093  0.5857847 0.3890409 -1.0009777
YAL005C -0.53191556  0.1577985 -1.2401448 0.8170350 -1.3520947
YAL007C -0.86693416 -1.1642322 -0.6359588 1.1179131  1.9587021
```

Proportion Variance Explained:

```
> p <- pca(gene_expression, space="rows")
> ggplot(data.frame(pc=1:13, pve=p$pve)) +
+   geom_point(aes(x=pc,y=pve), size=2)
```



PCs vs Time (with Smoothers):



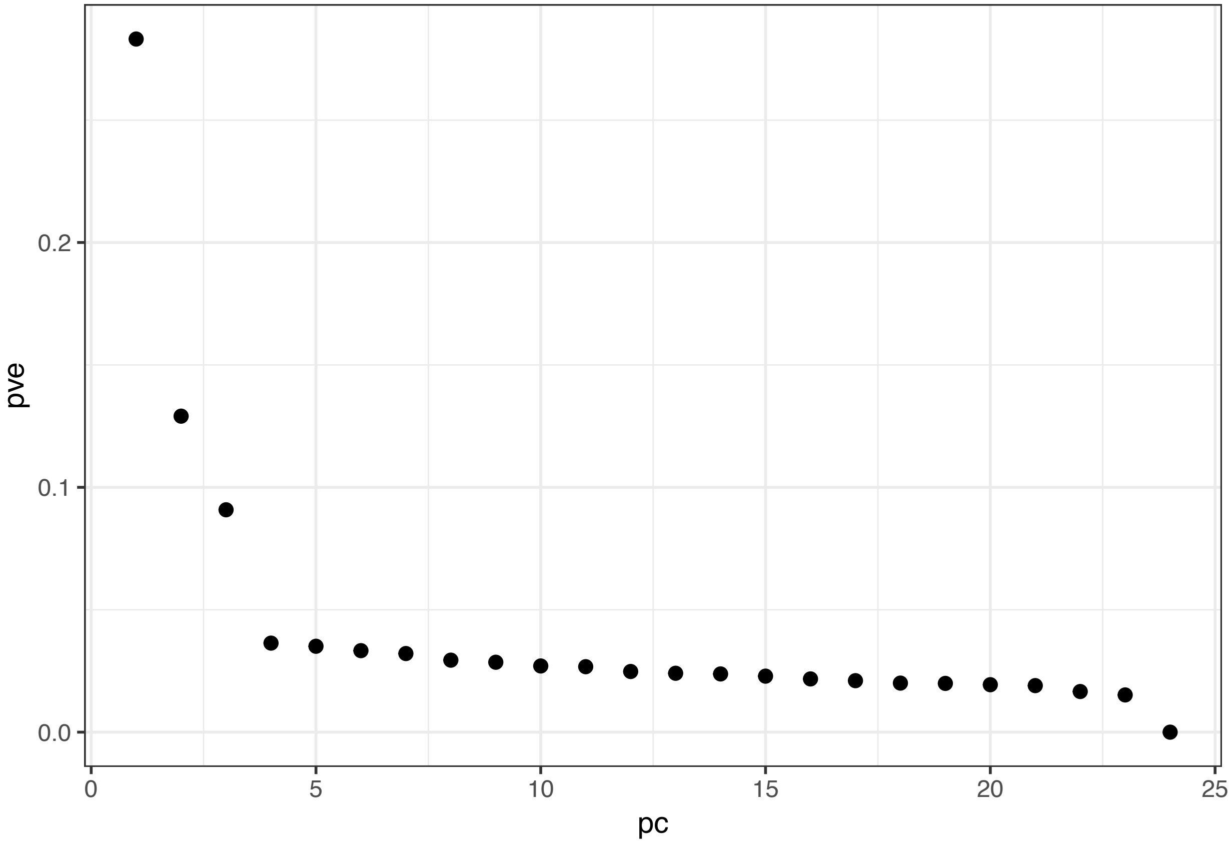
HapMap Genotypes

I curated a small data set that cleanly separates human subpopulations from the HapMap data. These include unrelated individuals from Yoruba people from Ibadan, Nigeria (YRI), Utah residents of northern and western European ancestry (CEU), Japanese individuals from Tokyo, Japan (JPT), and Han Chinese individuals from Beijing, China (CHB).

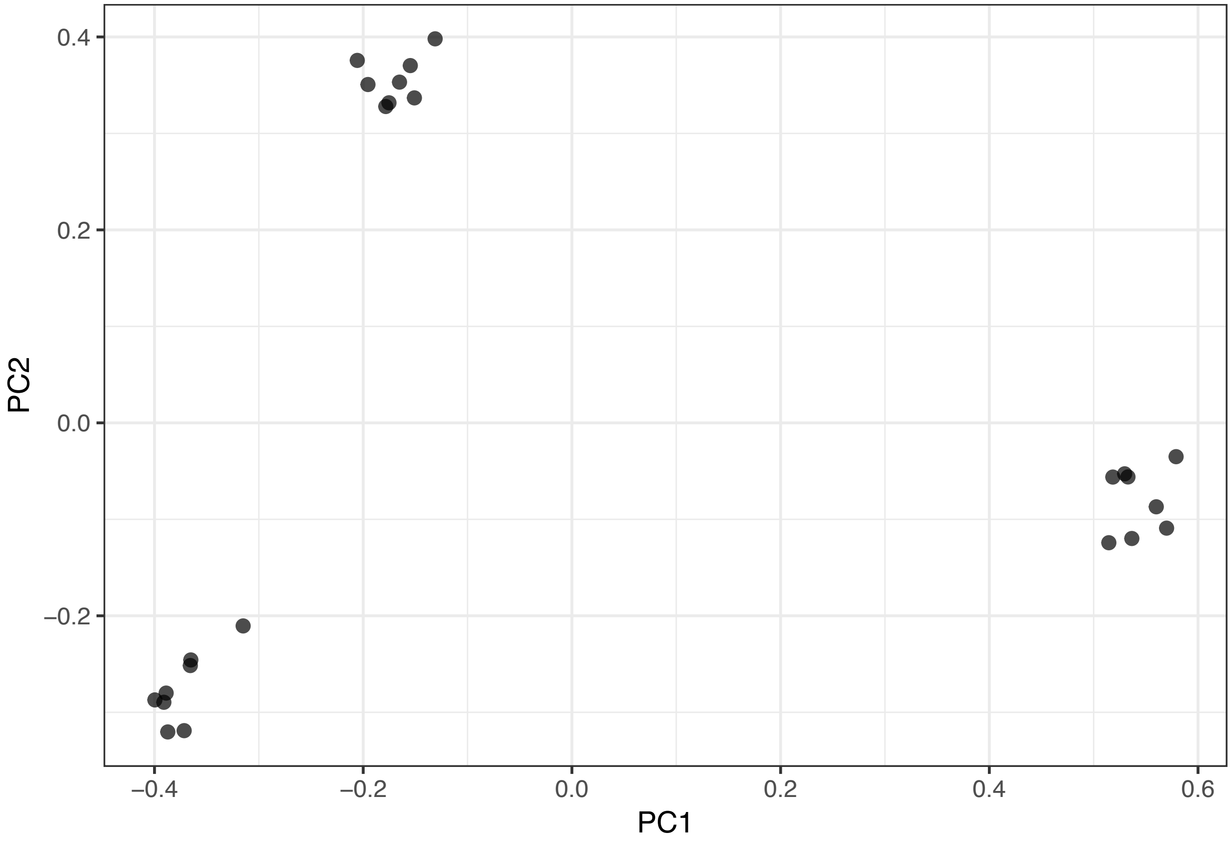
```
> hapmap <- read.table("./data/hapmap_sample.txt")
> dim(hapmap)
[1] 400  24
> hapmap[1:6,1:6]
           NA18516 NA19138 NA19137 NA19223 NA19200 NA19131
rs2051075        0       1       2       1       1       1
rs765546         2       2       0       0       0       0
rs10019399       2       2       2       1       1       2
rs7055827        2       2       1       2       0       2
rs6943479        0       0       2       0       1       0
rs2095381        1       2       1       2       1       1
```

Proportion Variance Explained:

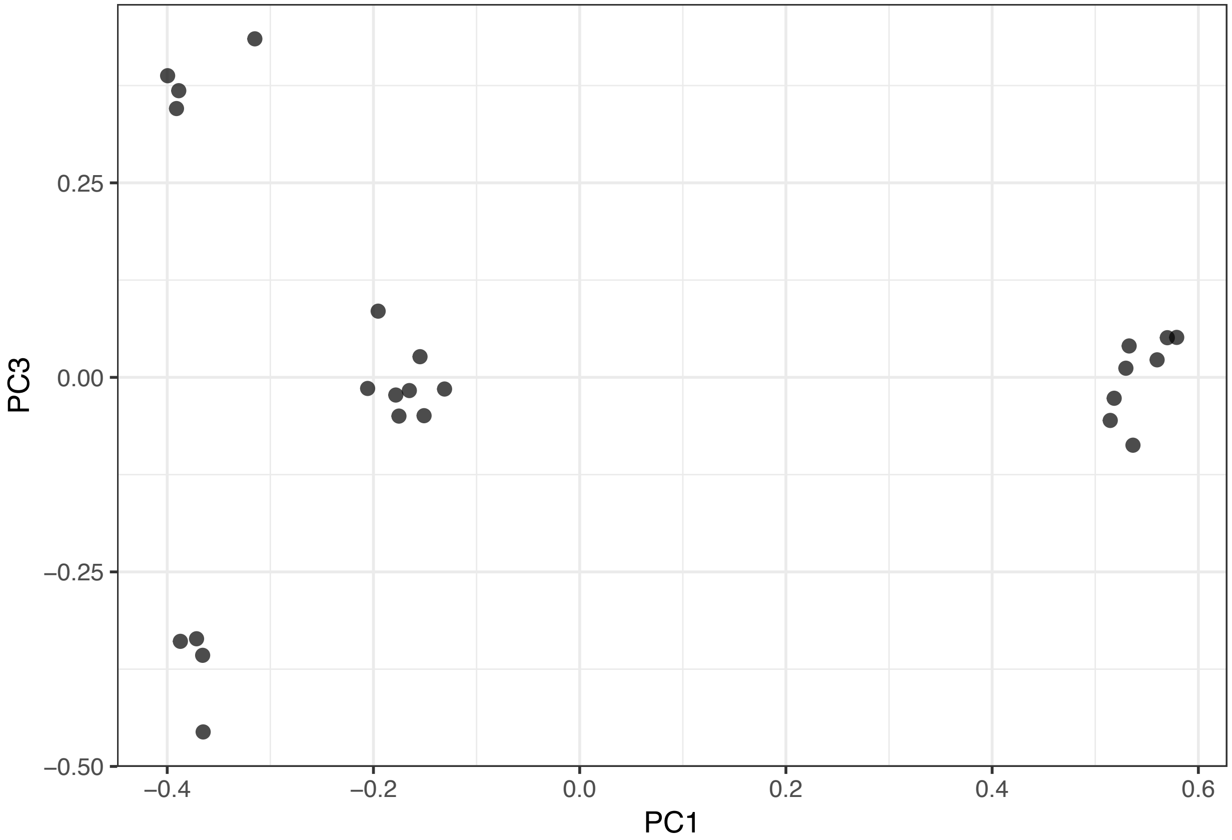
```
> p <- pca(hapmap, space="rows")
> ggplot(data.frame(pc=(1:ncol(hapmap)), pve=p$pve)) +
+   geom_point(aes(x=pc,y=pve), size=2)
```



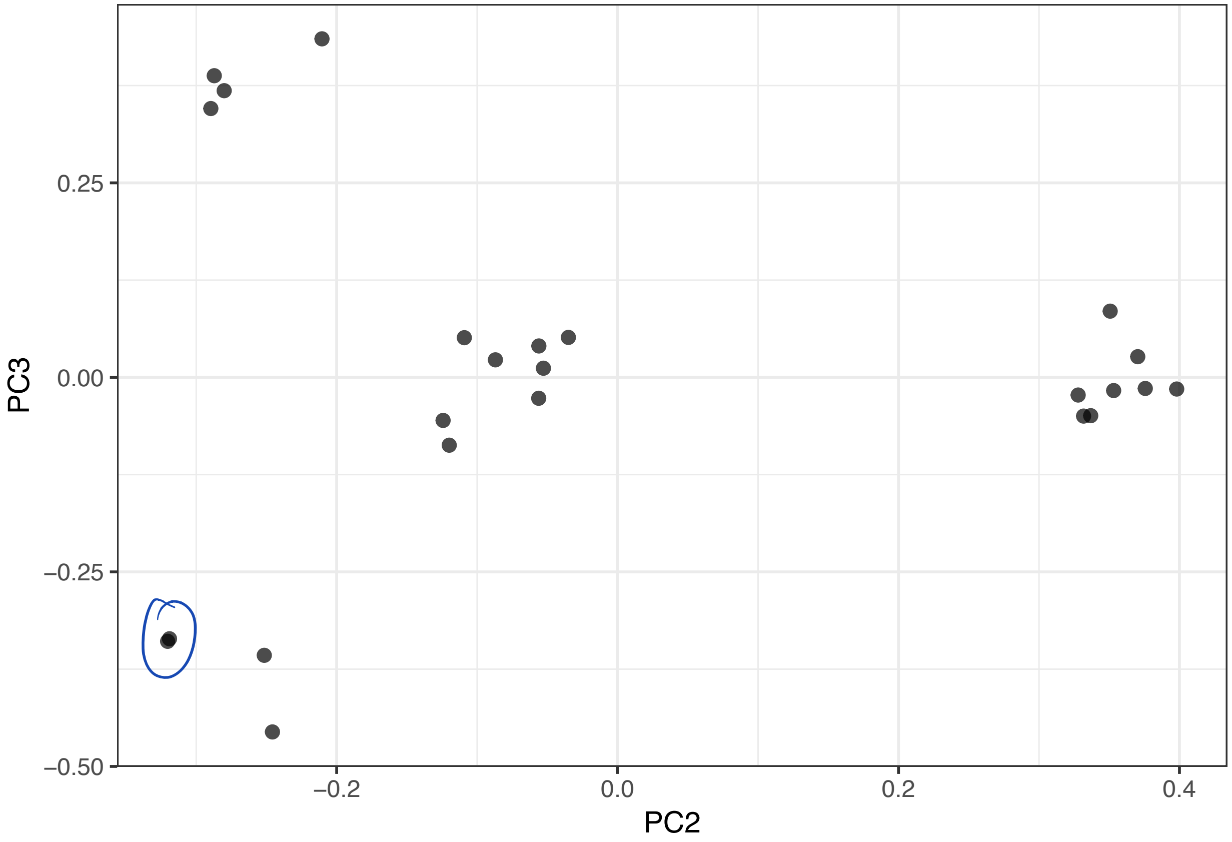
PC1 vs PC2 Biplot:



PC1 vs PC3 Biplot:



PC2 vs PC3 Biplot:



Data Sets

For the majority of this chapter, we will use some simple data sets to demonstrate the ideas.

Data `mtcars`

Load the `mtcars` data set:

```
> library("tidyverse") # why load tidyverse?
> data("mtcars", package="datasets")
> mtcars <- as_tibble(mtcars)
> head(mtcars)
# A tibble: 6 x 11
    mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  21       6   160   110  3.9   2.62  16.5     0     1     4     4
2  21       6   160   110  3.9   2.88  17.0     0     1     4     4
3  22.8     4   108    93  3.85  2.32  18.6     1     1     4     1
4  21.4     6   258   110  3.08  3.22  19.4     1     0     3     1
5  18.7     8   360   175  3.15  3.44  17.0     0     0     3     2
6  18.1     6   225   105  2.76  3.46  20.2     1     0     3     1
```

Data `mpg`

Load the `mpg` data set:

```
> data("mpg", package="ggplot2")
> head(mpg)
# A tibble: 6 x 11
  manufacturer model displ  year   cyl trans  drv     cty   hwy fl    class
  <chr>        <chr> <dbl> <int> <int> <chr>  <chr> <int> <int> <chr> <chr>
1 audi         a4      1.8  1999     4 auto(~ f        18    29 p     comp~
2 audi         a4      1.8  1999     4 manua~ f        21    29 p     comp~
3 audi         a4      2    2008     4 manua~ f        20    31 p     comp~
4 audi         a4      2    2008     4 auto(~ f        21    30 p     comp~
5 audi         a4      2.8  1999     6 auto(~ f        16    26 p     comp~
6 audi         a4      2.8  1999     6 manua~ f        18    26 p     comp~
```

Data `diamonds`

Load the `diamonds` data set:

```
> data("diamonds", package="ggplot2")
> head(diamonds)
```

```
# A tibble: 6 x 10
  carat cut       color clarity depth table price     x     y     z
  <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
1 0.23  Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
2 0.21  Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
3 0.23  Good      E     VS1      56.9    65   327  4.05  4.07  2.31
4 0.290 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
5 0.31  Good      J     SI2      63.3    58   335  4.34  4.35  2.75
6 0.24  Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
```

Data gapminder

Load the gapminder data set:

```
> library("gapminder")
> data("gapminder", package="gapminder")
> gapminder <- as_tibble(gapminder)
> head(gapminder)
# A tibble: 6 x 6
  country     continent  year lifeExp      pop gdpPercap
  <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
1 Afghanistan Asia       1952    28.8  8425333      779.
2 Afghanistan Asia       1957    30.3  9240934      821.
3 Afghanistan Asia       1962    32.0 10267083      853.
4 Afghanistan Asia       1967    34.0 11537966      836.
5 Afghanistan Asia       1972    36.1 13079460      740.
6 Afghanistan Asia       1977    38.4 14880372      786.
```

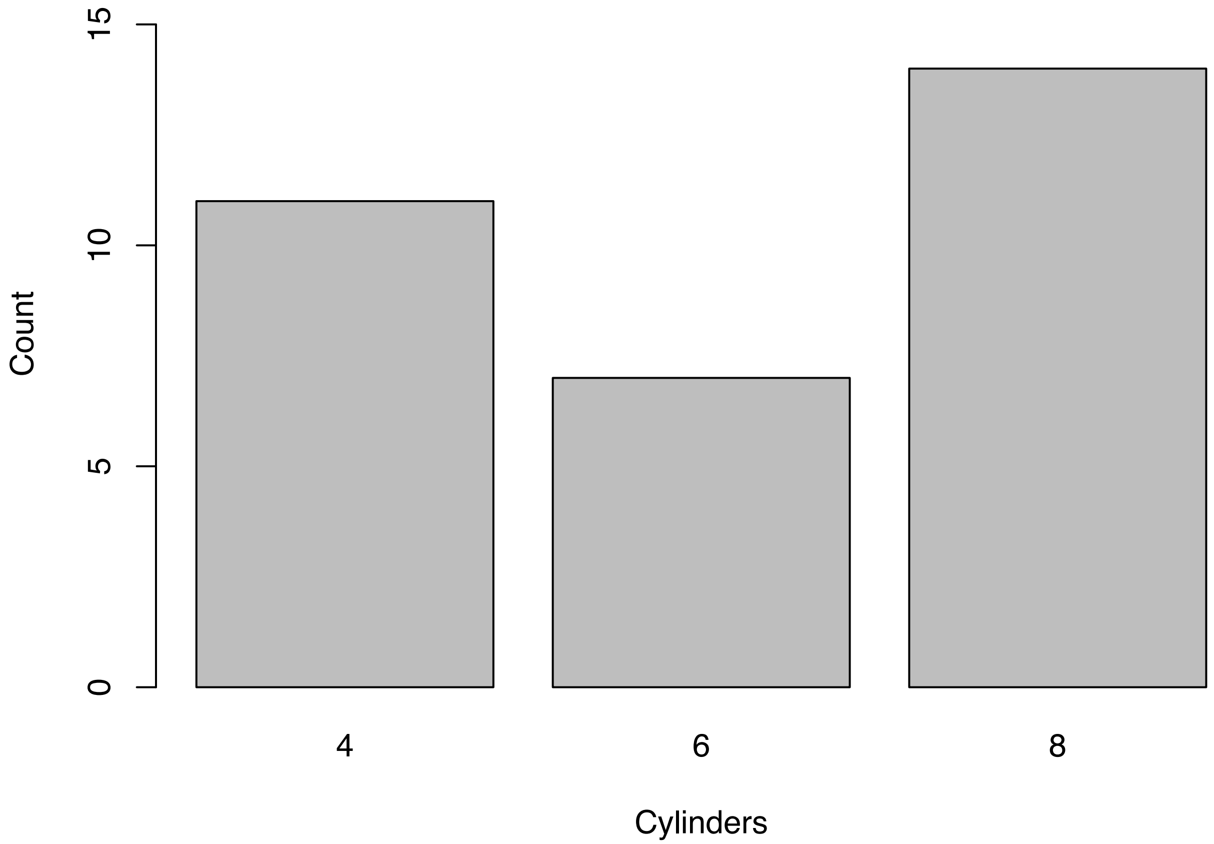
Read the Documentation

For all of the plotting functions covered below, read the help files.

```
> ?barplot
> ?boxplot
> ?hist
> ?density
> ?plot
> ?legend
> ?qqplot
```

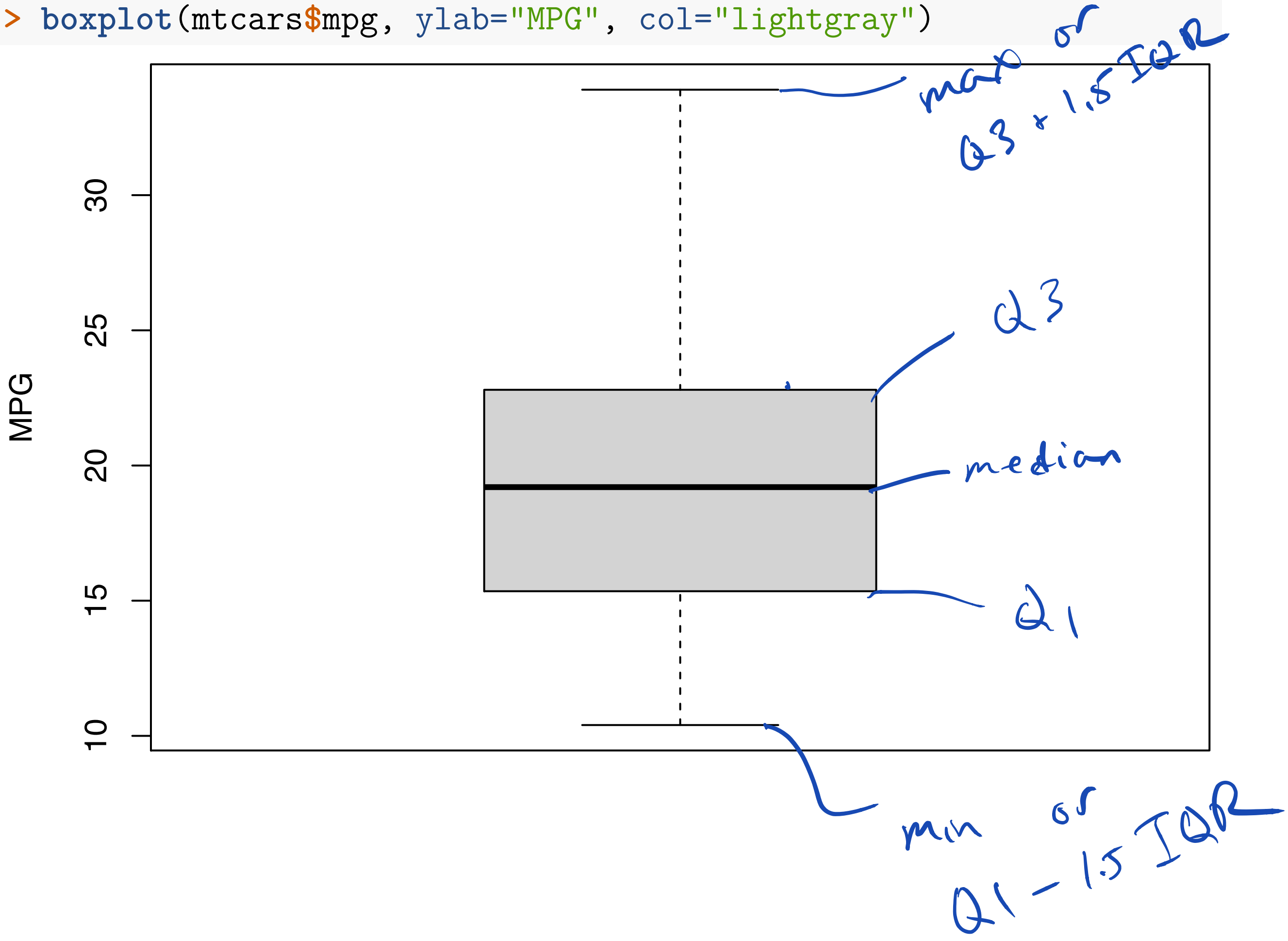
Barplot

```
> cyl_tbl <- table(mtcars$cyl)
> barplot(cyl_tbl, xlab="Cylinders", ylab="Count")
```



Boxplot

```
> boxplot(mtcars$mpg, ylab="MPG", col="lightgray")
```

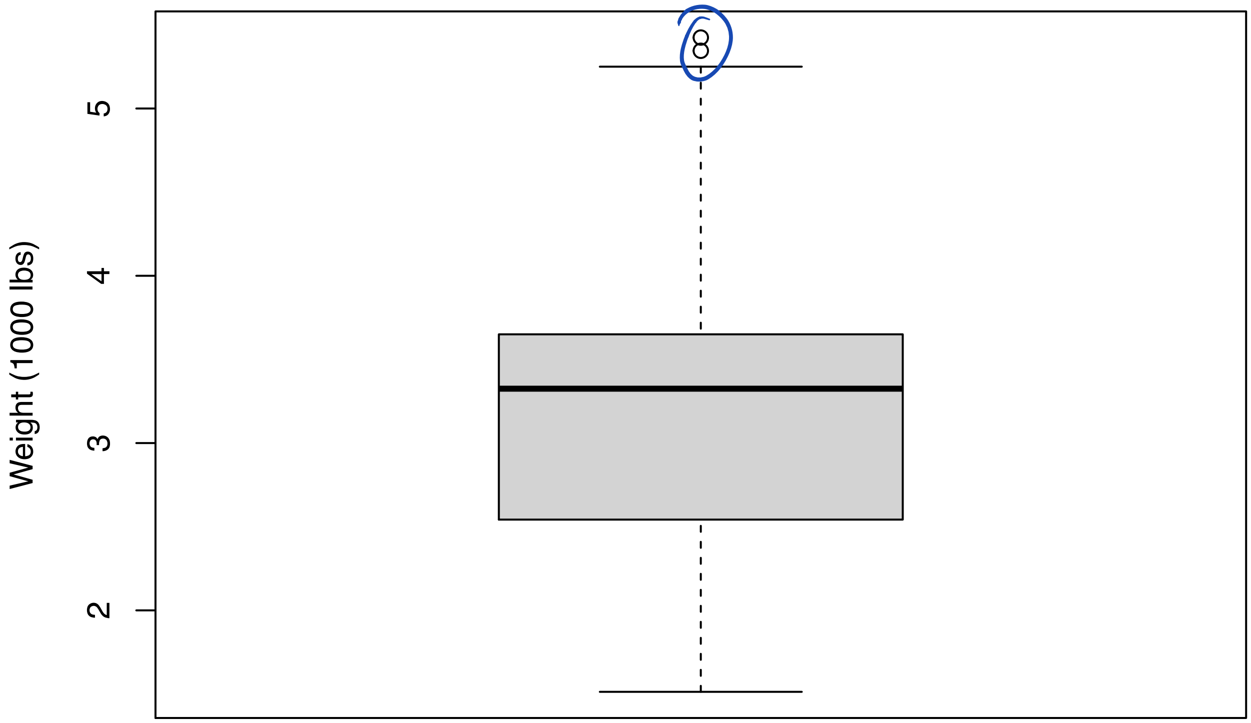


Constructing Boxplots

- The top of the box is Q3
- The line through the middle of the box is the median
- The bottom of the box is Q1
- The top whisker is the minimum of Q3 + 1.5 × IQR or the largest data point
- The bottom whisker is the maximum of Q1 - 1.5 × IQR or the smallest data point
- Outliers lie outside of (Q1 - 1.5 × IQR) or (Q3 + 1.5 × IQR), and they are shown as points
- Outliers are calculated using the `fivenum()` function

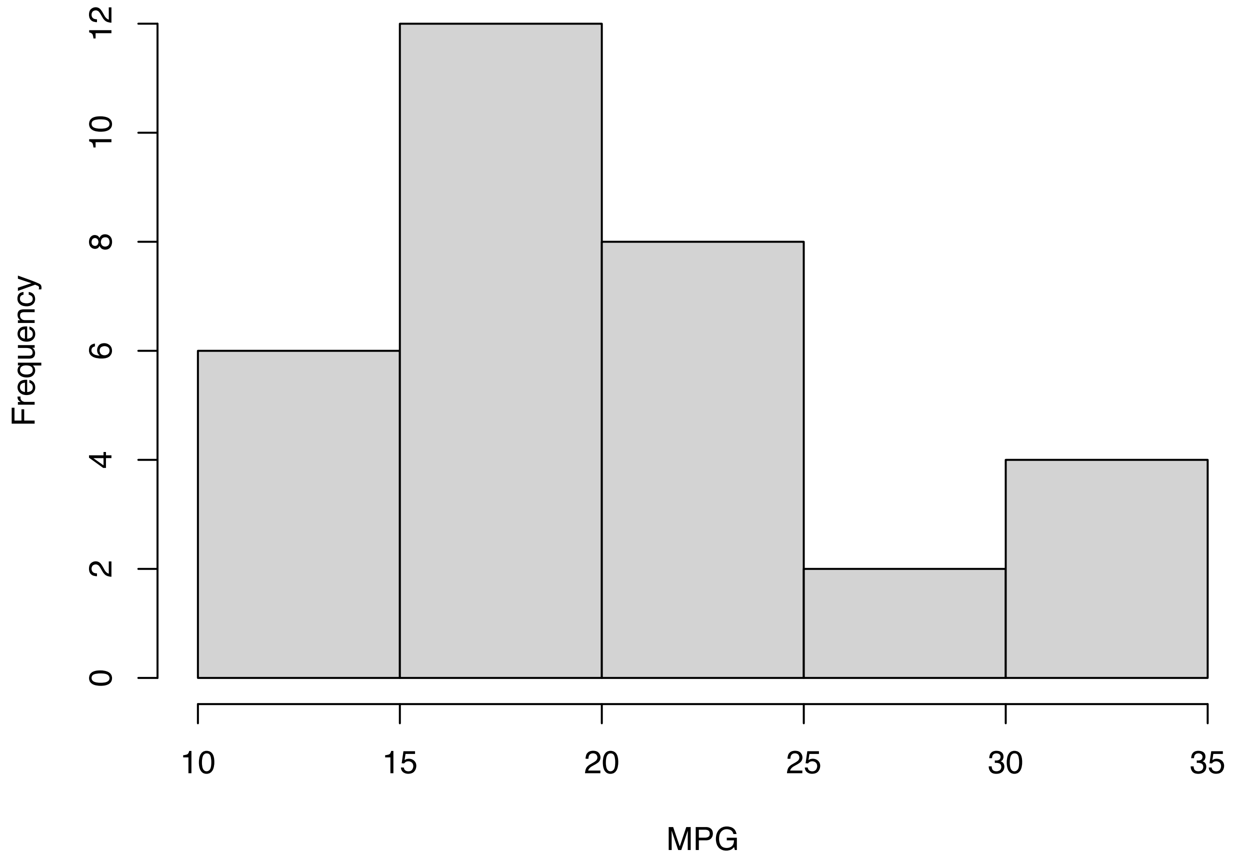
Boxplot with Outliers

```
> boxplot(mtcars$wt, ylab="Weight (1000 lbs)",
+         col="lightgray")
```



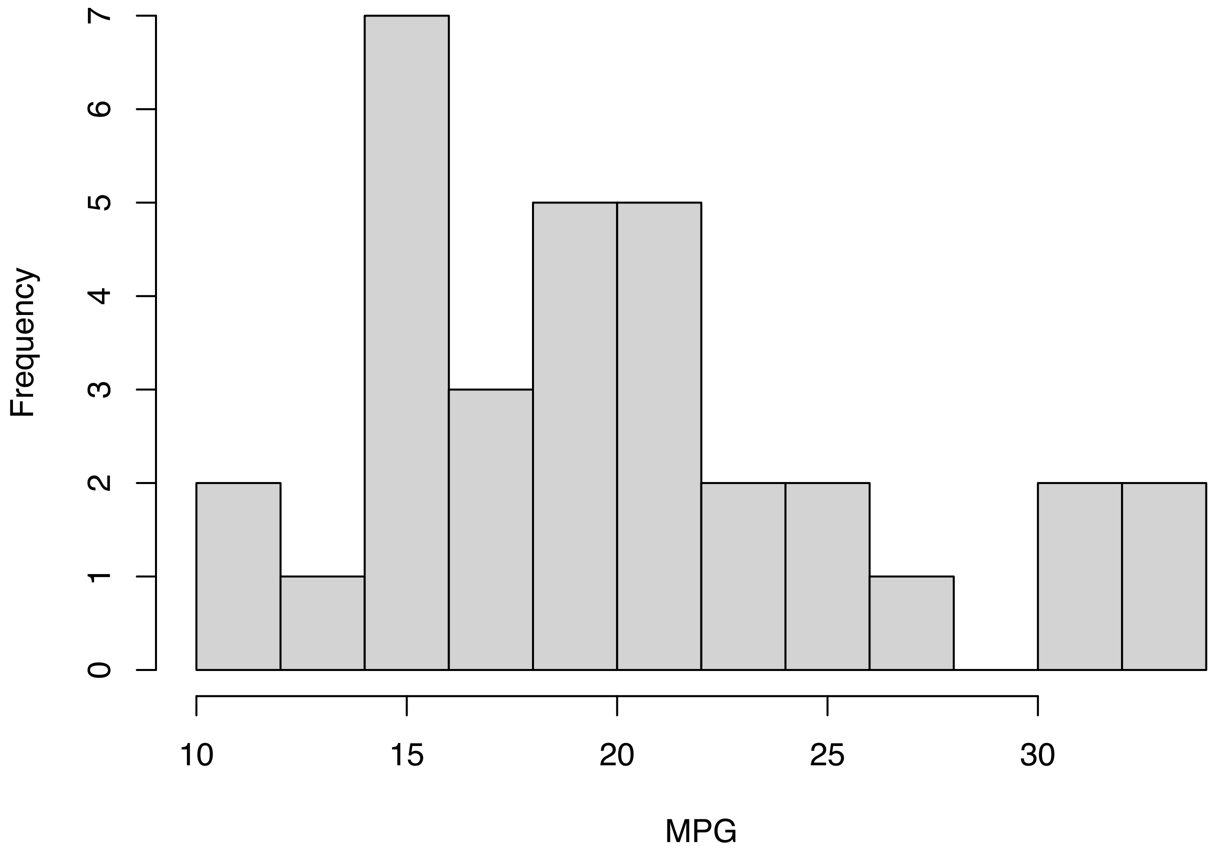
Histogram

```
> hist(mtcars$mpg, xlab="MPG", main="", col="lightgray")
```



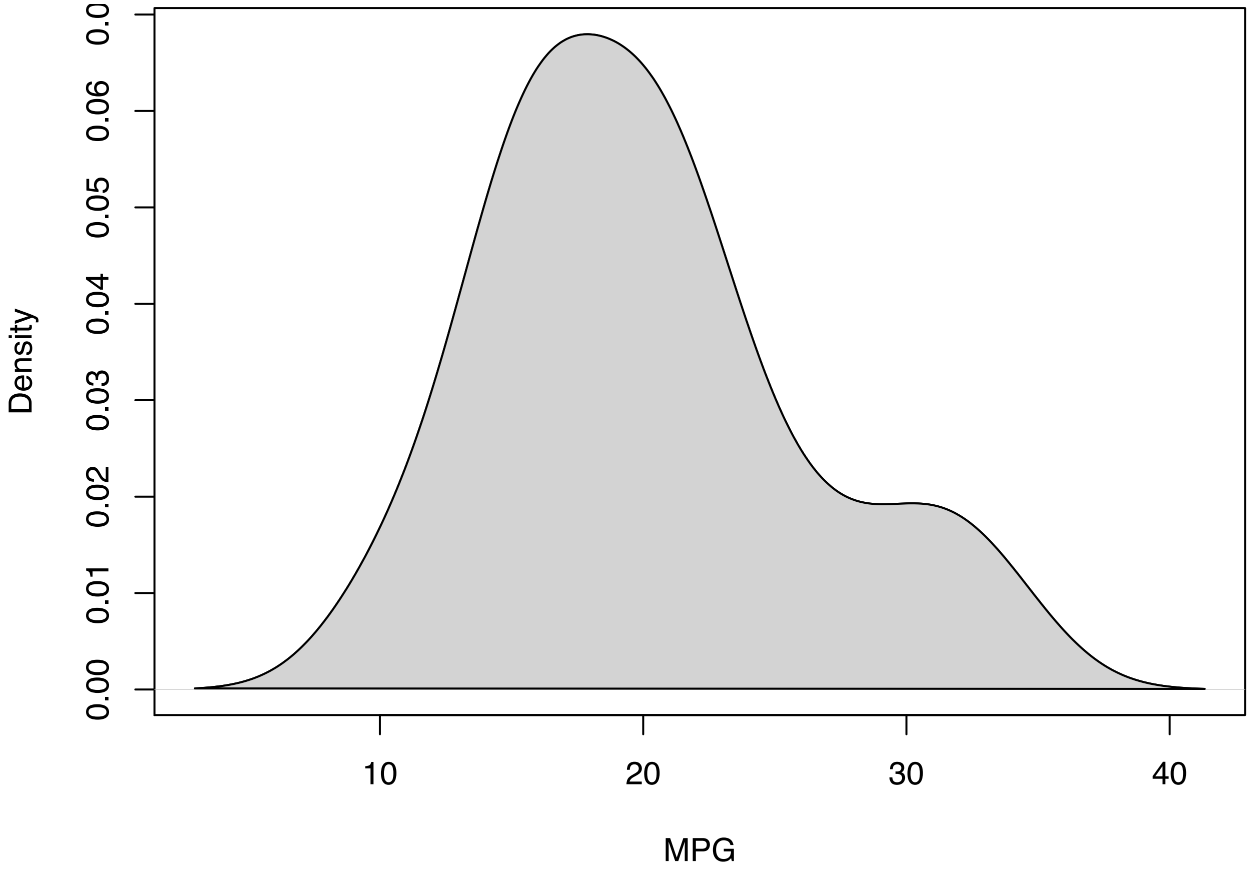
Histogram with More Breaks

```
> hist(mtcars$mpg, breaks=12, xlab="MPG", main="", col="lightgray")
```



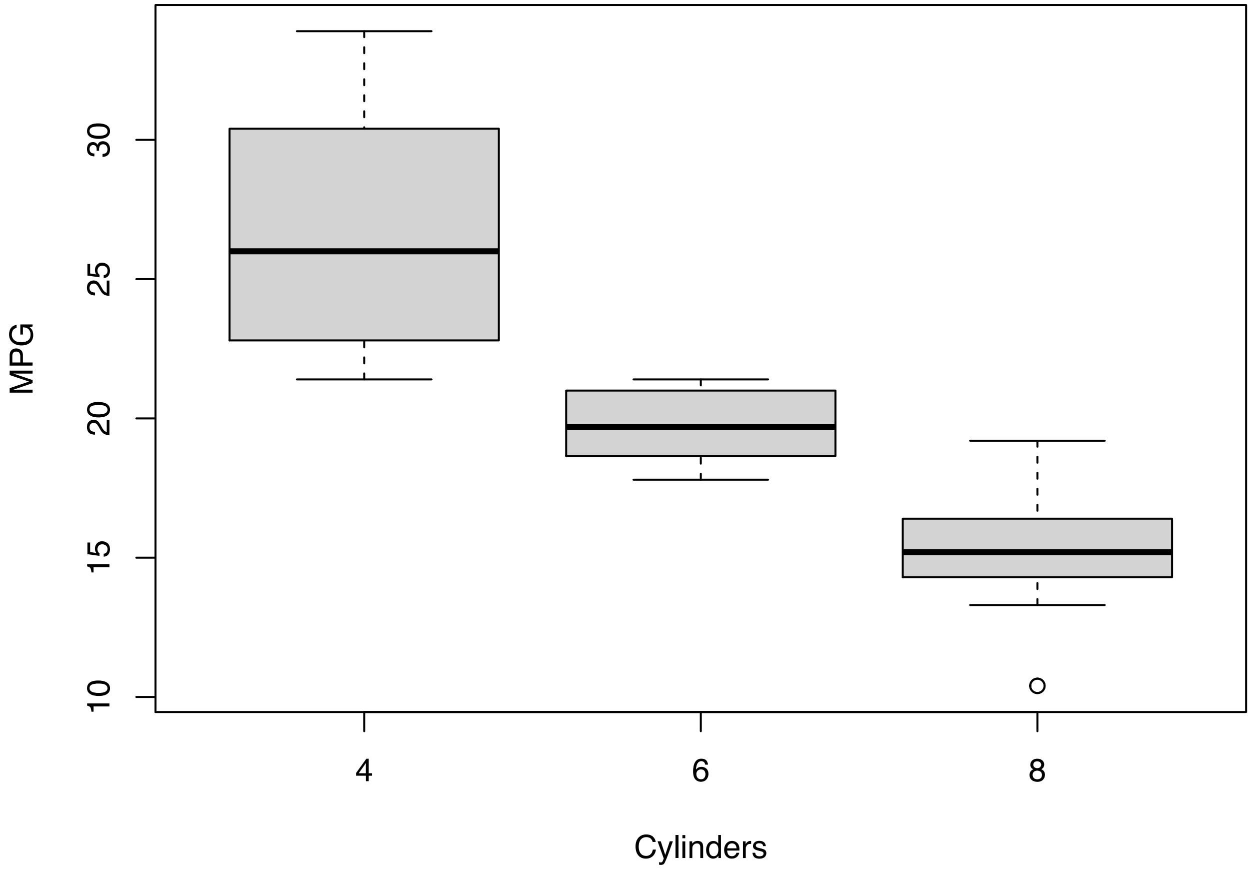
Density Plot

```
> plot(density(mtcars$mpg), xlab="MPG", main="")
> polygon(density(mtcars$mpg), col="lightgray", border="black")
```



Boxplot (Side-By-Side)

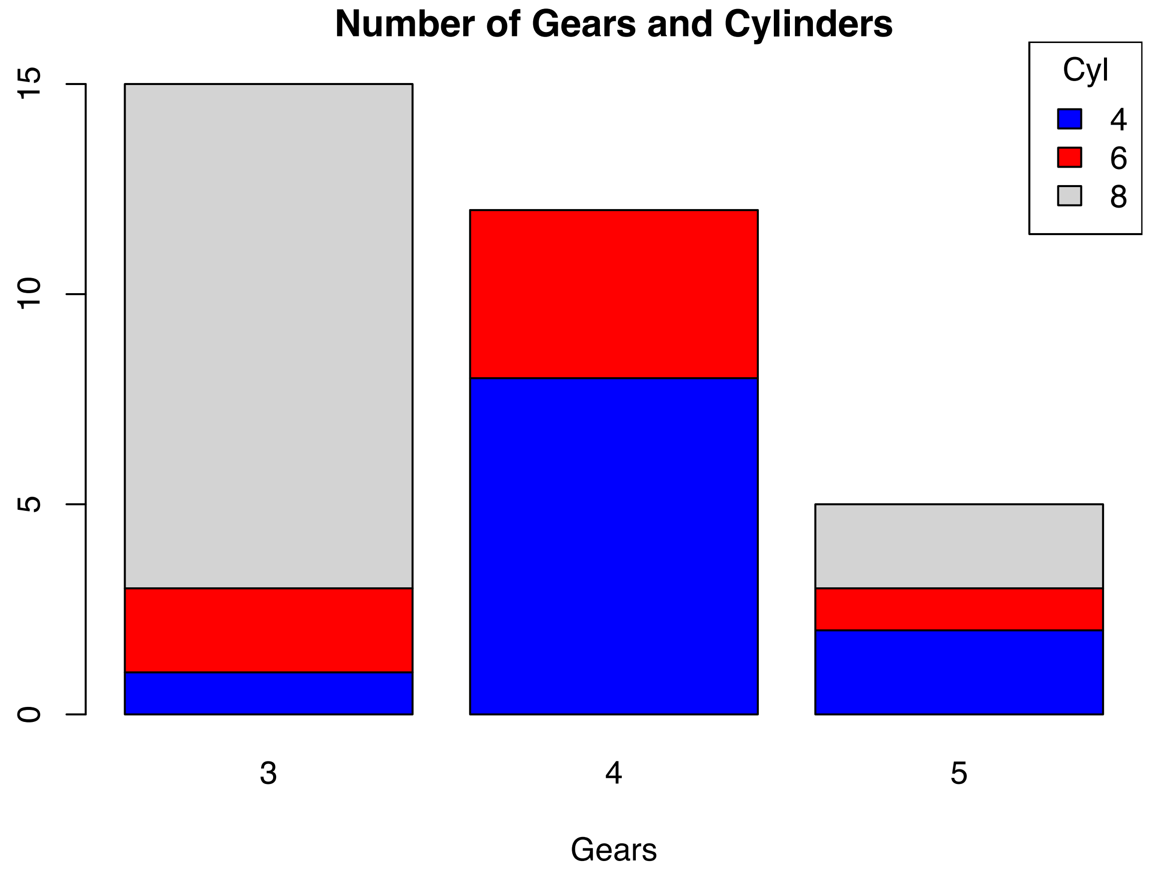
```
> boxplot(mpg ~ cyl, data=mtcars, xlab="Cylinders",
+         ylab="MPG", col="lightgray")
```



Stacked Barplot

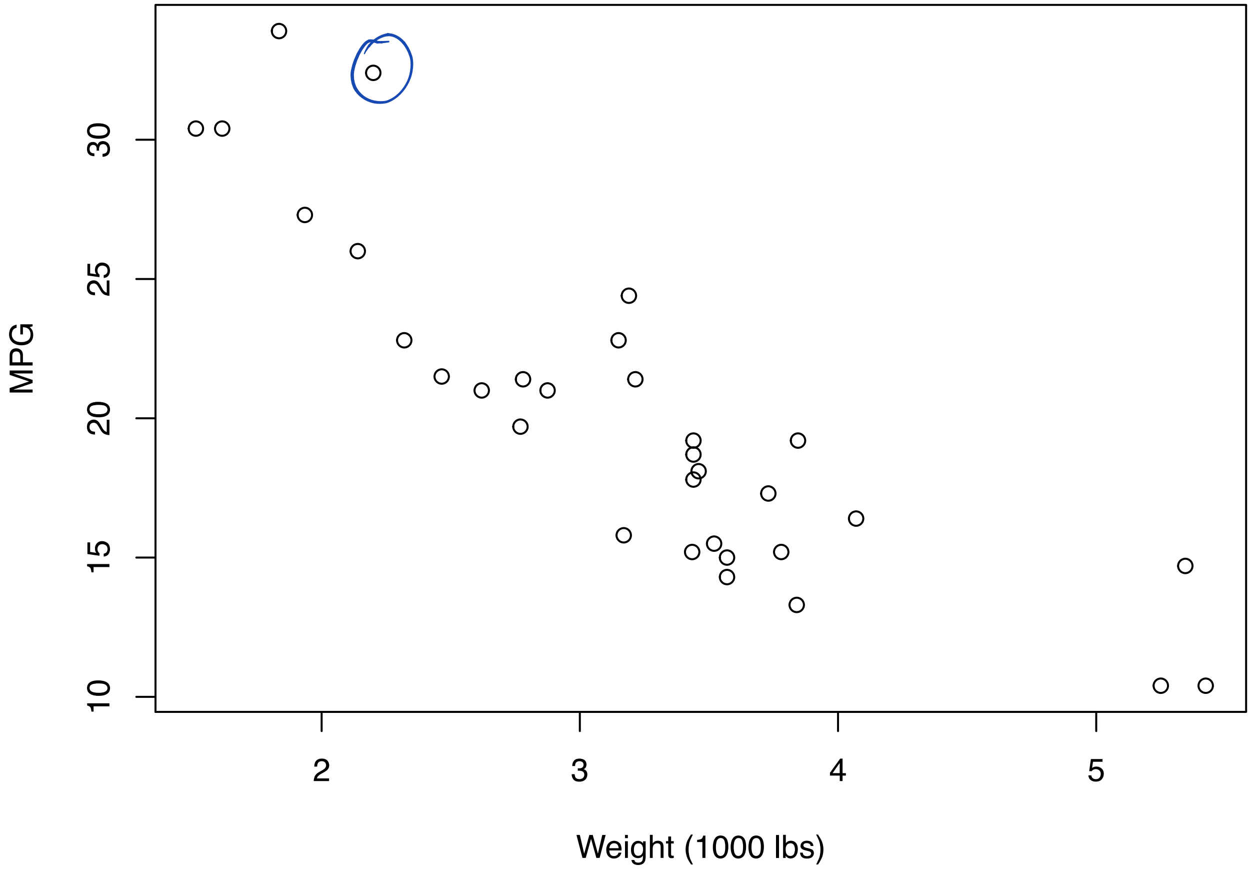
```
> counts <- table(mtcars$cyl, mtcars$gear)
> counts

     3  4  5
  4  1  8  2
  6  2  4  1
  8 12  0  2
```

```
> barplot(counts, main="Number of Gears and Cylinders",
+   xlab="Gears", col=c("blue","red", "lightgray"))
> legend(x="topright", title="Cyl",
+        legend = rownames(counts),
+        fill = c("blue","red", "lightgray"))
```



Scatterplot

```
> plot(mtcars$wt, mtcars$mpg, xlab="Weight (1000 lbs)",
+      ylab="MPG")
```



Quantile-Quantile Plots

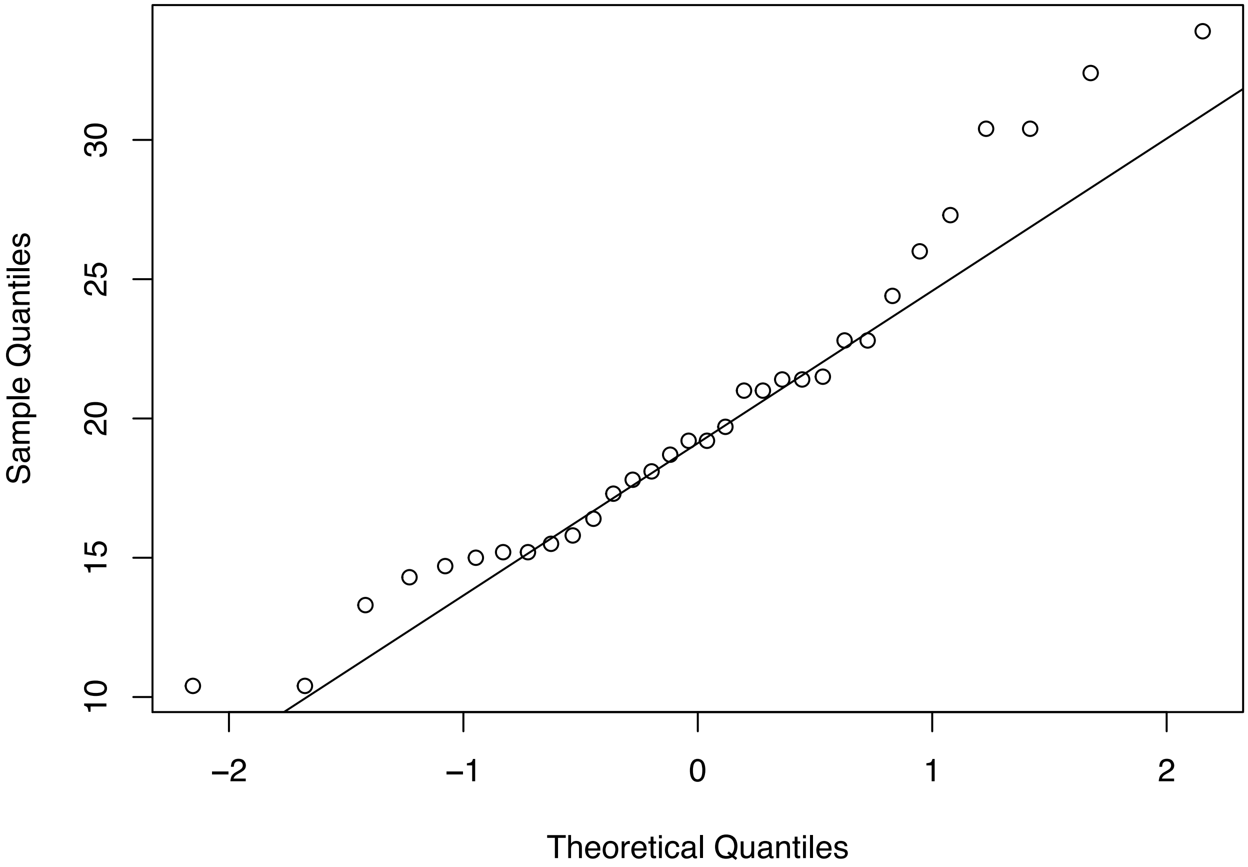
Quantile-quantile plots display the quantiles of:

1. two samples of data
2. a sample of data vs a theoretical distribution

The first type allows one to assess how similar the distributions are of two samples of data.

The second allows one to assess how similar a sample of data is to a theoretical distribution (often Normal with mean 0 and standard deviation 1).

```
> qqnorm(mtcars$mpg, main=" ")
> qqline(mtcars$mpg) # line through Q1 and Q3
```



A Grammar of Graphics

Rationale

A grammar for communicating data visualization:

- *Data*: the data set we are plotting
- *Aesthetics*: the variation or relationships in the data we want to visualize
- *Geometries*: the geometric object by which we render the aesthetics
- *Coordinates*: the coordinate system used (not covered here)
- *Facets*: the layout of plots required to visualize the data
- Other Options: any other customizations we wish to make, such as changing the color scheme or labels

These are strung together like words in a sentence.

Package ggplot2

The R package ggplot2 implements a grammar of graphics along these lines. First, let's load ggplot2:

```
> library(ggplot2)
```

Now let's set a theme (more on this later):

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

Pieces of the Grammar

- ggplot()
- aes()
- geom_*()
- facet_*()
- scale_*()
- theme()
- labs()

The * is a placeholder for a variety of terms that we will consider.

Geometries

Perhaps the most important aspect of ggplot2 is to understand the "geoms". We will cover the following:

- geom_bar()
- geom_boxplot()
- geom_violin()

- `geom_histogram()`
- `geom_density()`
- `geom_line()`
- `geom_point()`
- `geom_smooth()`
- `geom_hex()`

Call Format

The most basic `ggplot2` plot is made with something like:

```
ggplot(data = <DATA FRAME>) +
 geom_*(mapping = aes(x = <VAR X>, y = <VAR Y>))
```

where `<DATA FRAME>` is a data frame and `<VAR X>` and `<VAR Y>` are variables (i.e., columns) from this data frame. Recall `geom_*` is a placeholder for a geometry such as `geom_boxplot`.

Layers

There's a complex "layers" construct occurring in the `ggplot2` package. However, for our purposes, it suffices to note that the different parts of the plots are layered together through the `+` operator:

```
> ggplot(data = mpg) +
+   geom_point(mapping = aes(x = displ, y = hwy, color=drv)) +
+   ~~geom_smooth~~(mapping = aes(x = displ, y = hwy, color=drv)) +
+   scale_color_brewer(palette = "Set1", name = "Drivetrain") +
+   labs(title = "Highway MPG By Drivetrain and Displacement",
+        x = "Displacement", y = "Highway MPG")
```

Placement of the `aes()` Call

In the previous slide, we saw that the same `aes()` call was made for two `geom`'s. When this is the case, we may more simply call `aes()` from within `ggplot()`:

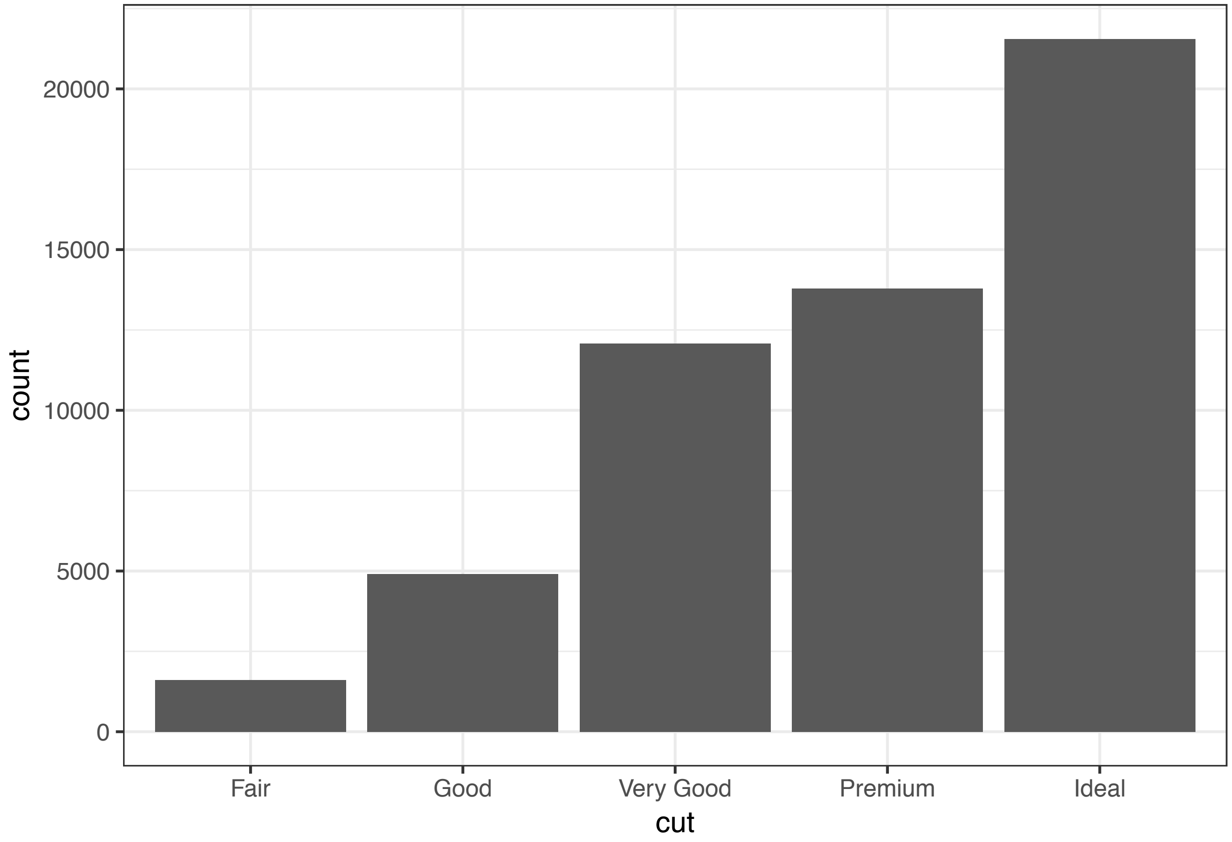
```
> ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color=drv)) +
+   geom_point() +
+   geom_smooth() +
+   scale_color_brewer(palette = "Set1", name = "Drivetrain") +
+   labs(title = "Highway MPG By Drivetrain and Displacement",
+        x = "Displacement", y = "Highway MPG")
```

There may be cases where different `geom`'s are layered and require different `aes()` calls. This is something to keep in mind as we go through the specifics of the `ggplot2` package.

Barplots

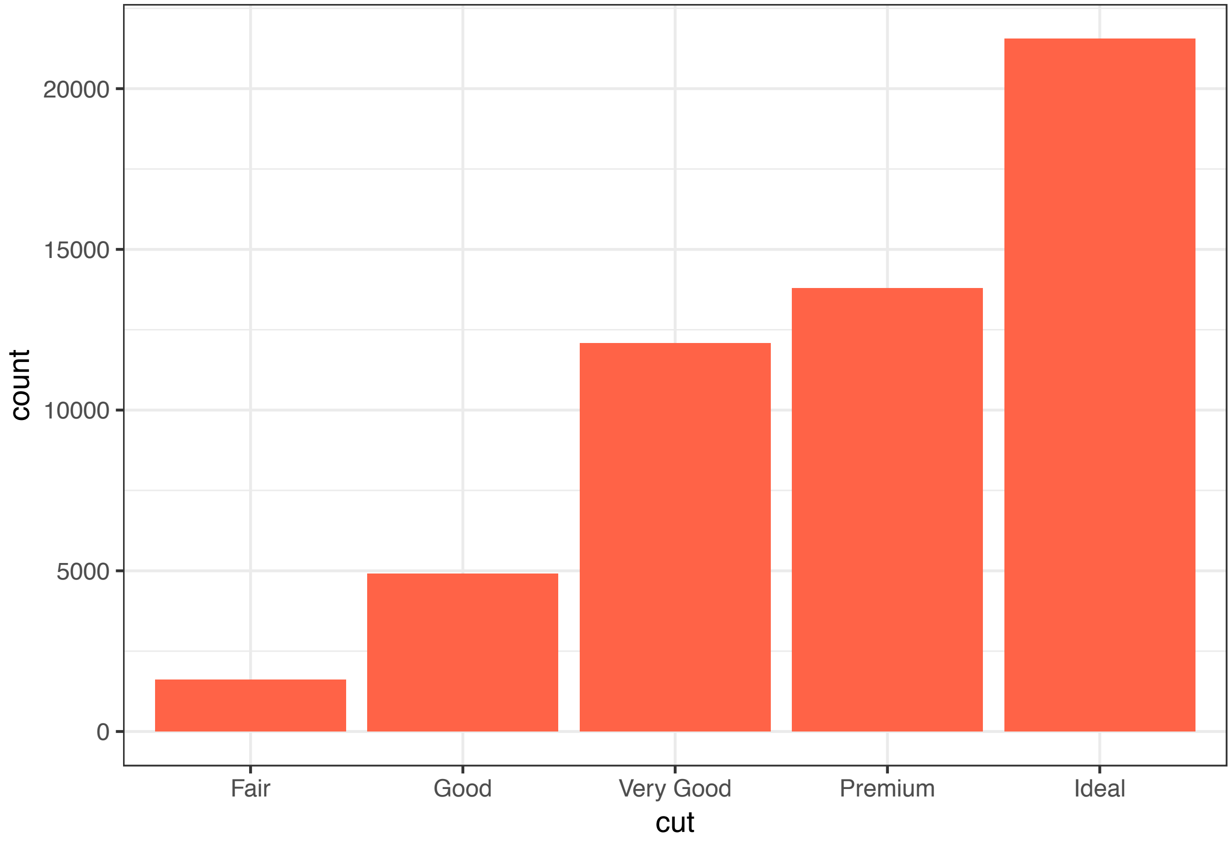
The `geom_bar()` layer forms a barplot and only requires an `x` assignment in the `aes()` call:

```
> ggplot(data = diamonds) +
+   geom_bar(mapping = aes(x = cut))
```



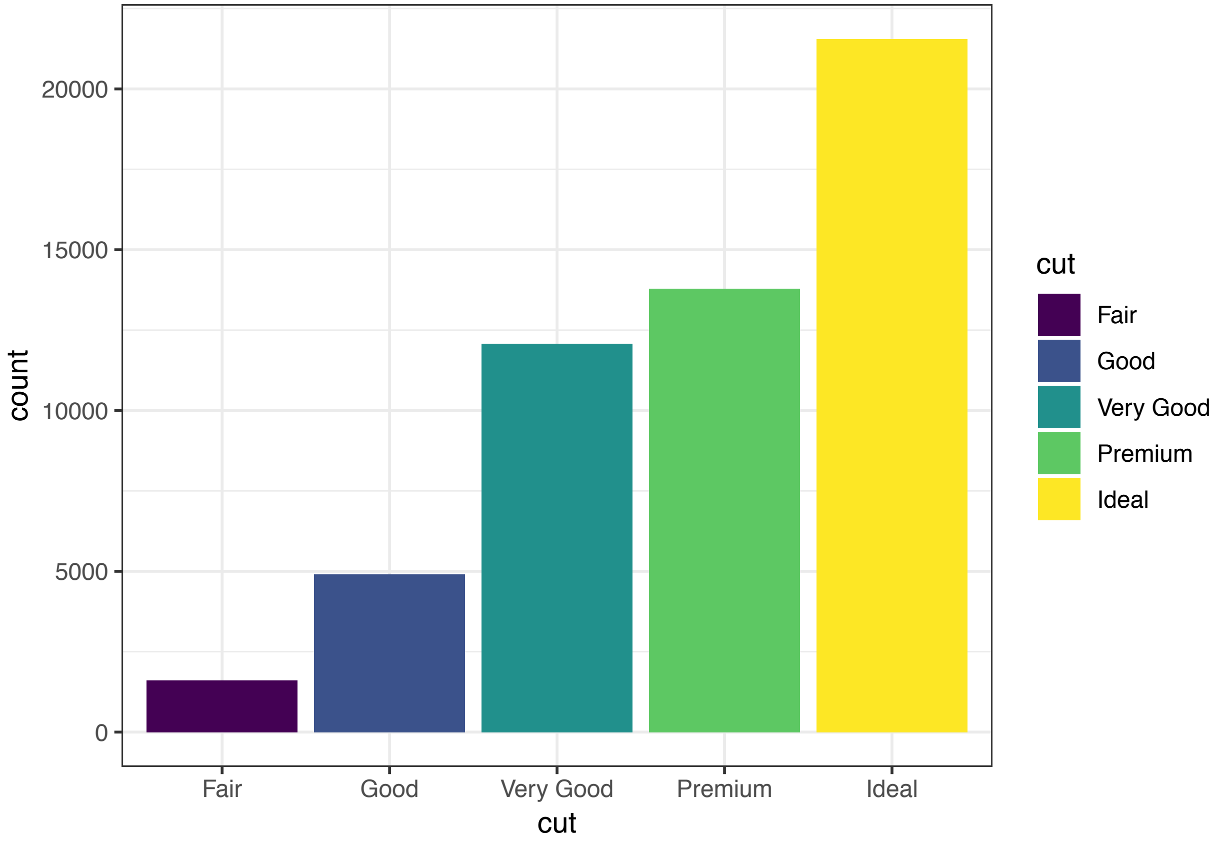
Color in the bars by assigning `fill` in `geom_bar()`, but outside of `aes()`:

```
> ggplot(data = diamonds) +
+   geom_bar(mapping = aes(x = cut), fill = "tomato")
```



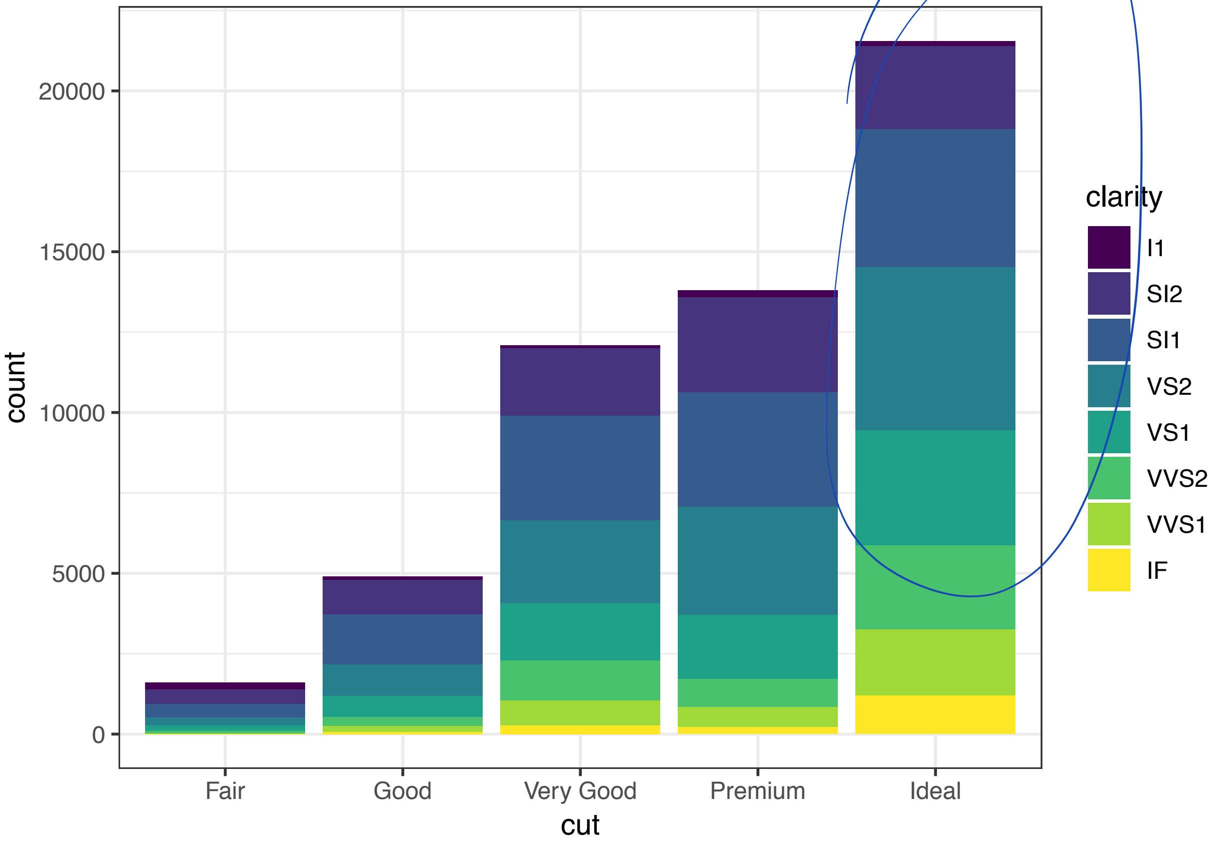
Color *within* the bars according to a variable by assigning `fill` in `geom_bar()` *inside* of `aes()`:

```
> ggplot(data = diamonds) +
+   geom_bar(mapping = aes(x = cut, fill = cut))
```



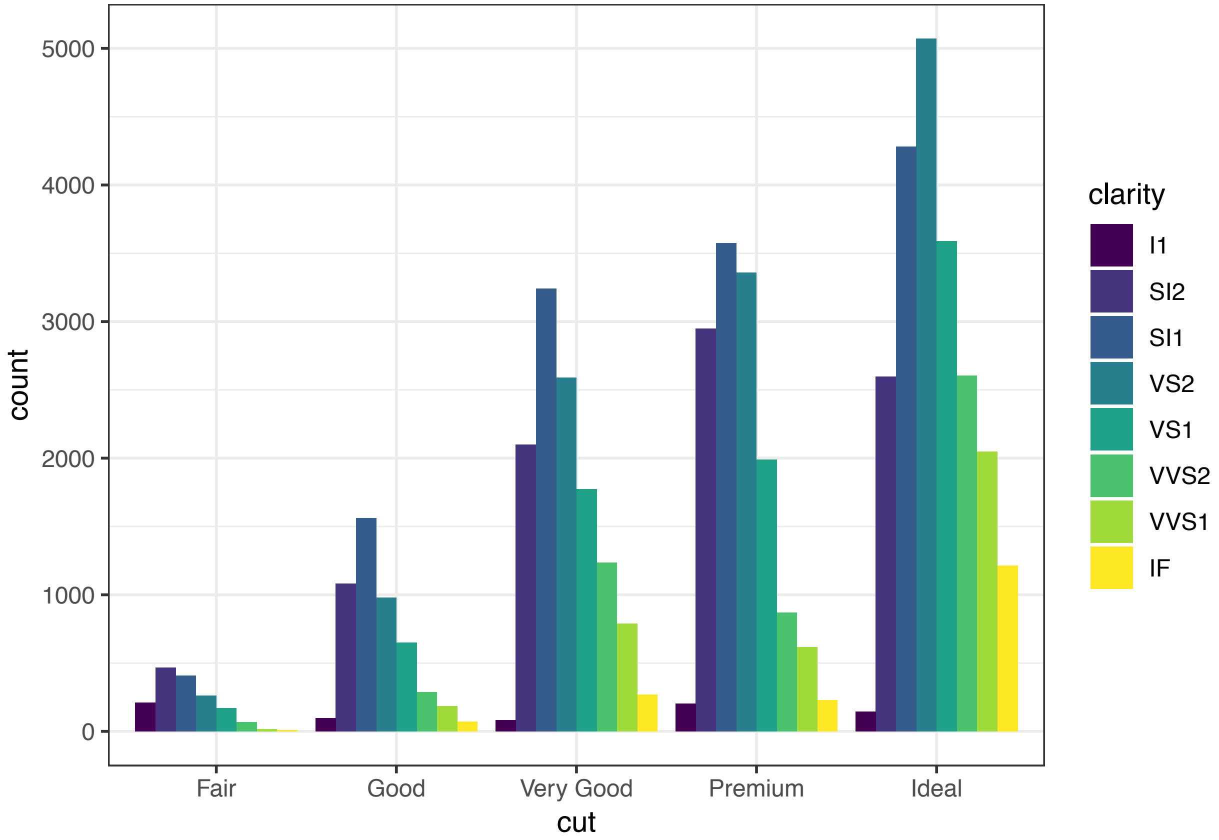
When we use `fill = clarity` within `aes()`, we see that it shows the proportion of each `clarity` value within each `cut` value:

```
> ggplot(data = diamonds) +
+   geom_bar(mapping = aes(x = cut, fill = clarity))
```



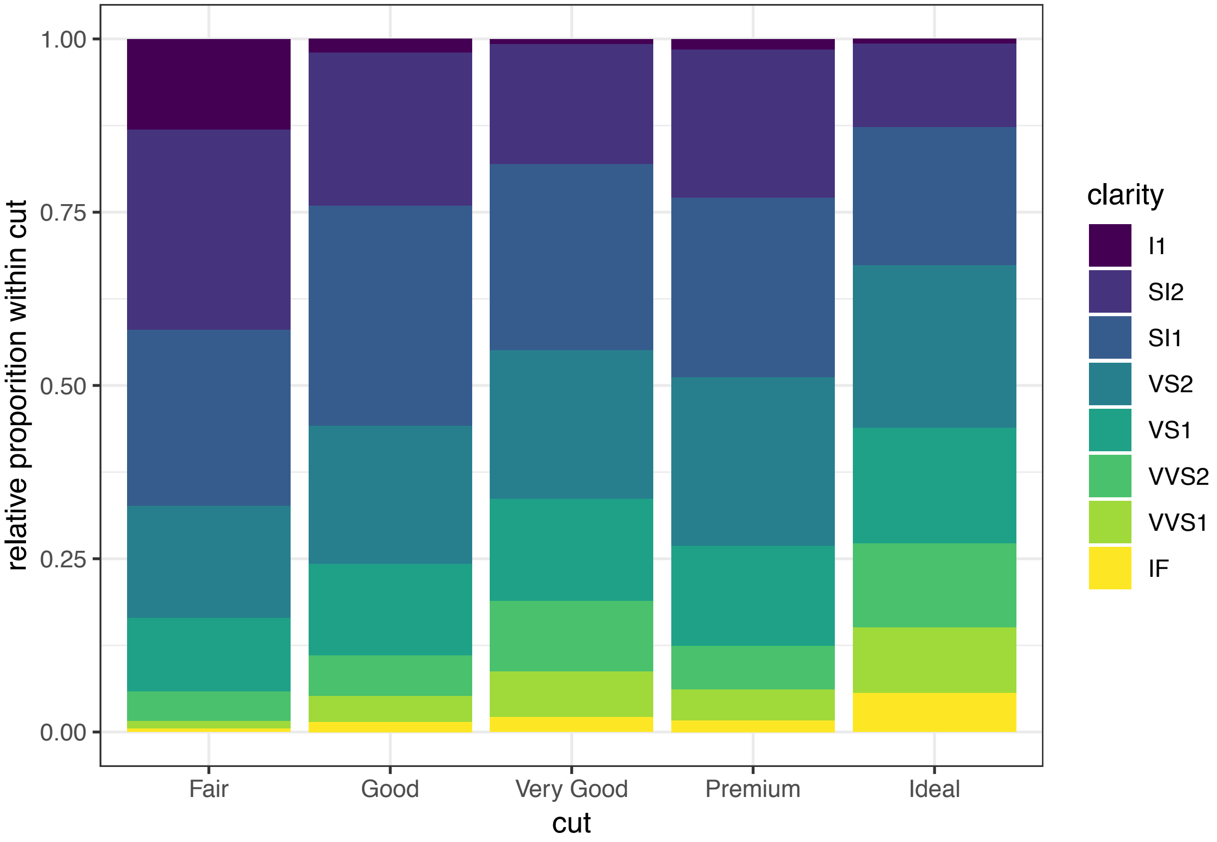
By setting `position = "dodge"` outside of `aes()`, it shows bar charts for the `clarity` values within each `cut` value:

```
> ggplot(data = diamonds) +
+   geom_bar(mapping= aes(x = cut, fill = clarity),
+            position = "dodge")
```



By setting `position = "fill"`, it shows the proportion of `clarity` values within each `cut` value and no longer shows the `cut` values:

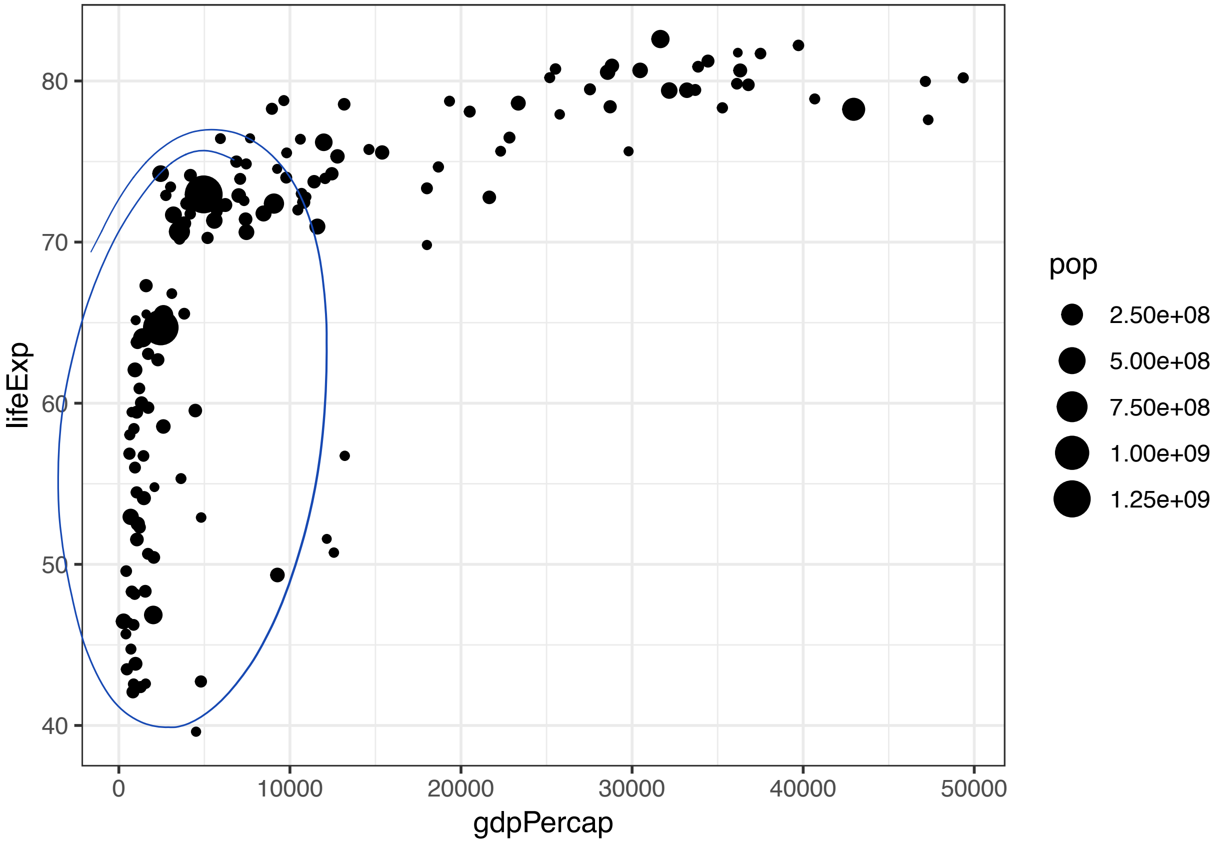
```
> ggplot(data = diamonds) +
+   geom_bar(mapping=aes(x = cut, fill = clarity),
+            position = "fill") +
+   labs(x = "cut", y = "relative proporition within cut")
```



Axis Scales

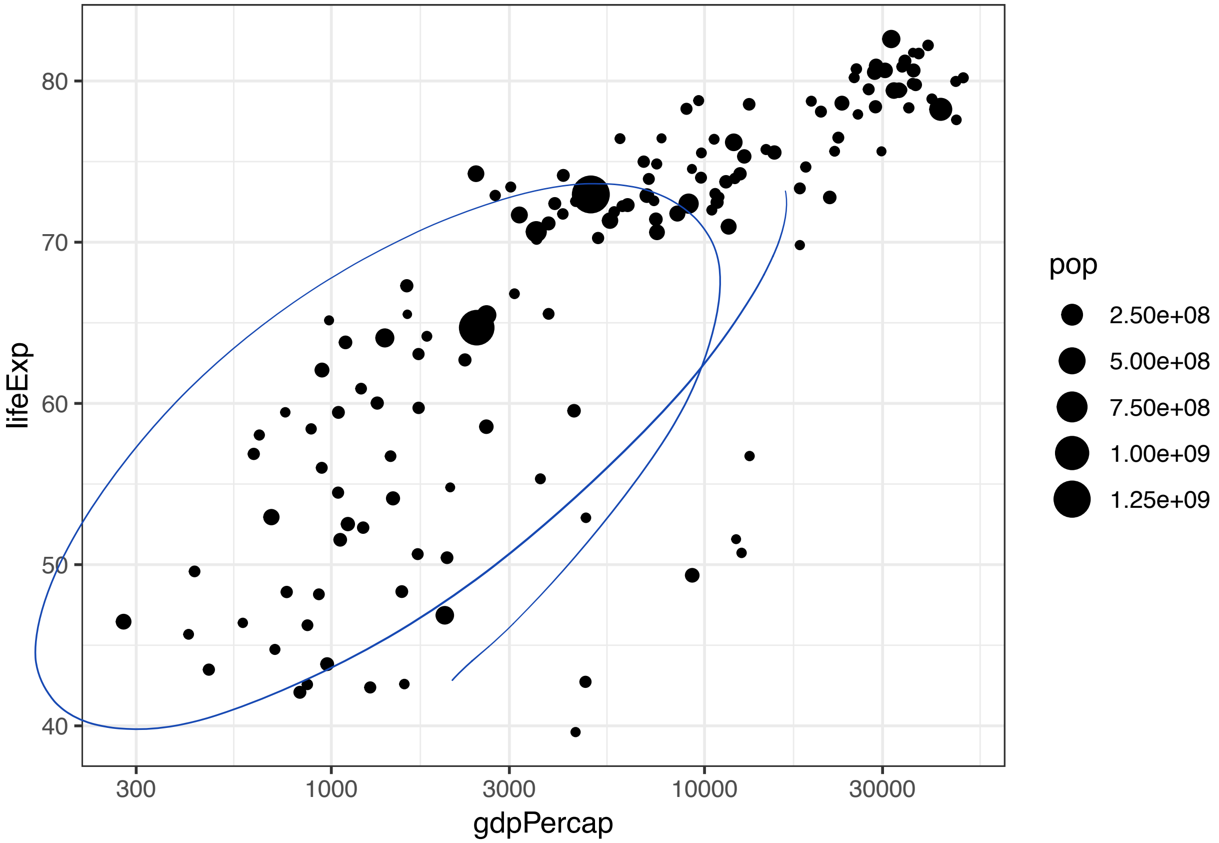
From the gapminder data set, we plot gdpPercap vs lifeExp in the year 2007.

```
> gapminder %>% filter(year==2007) %>% ggplot() +
+   geom_point(aes(x = gdpPercap, y = lifeExp,
+                  size = pop))
```



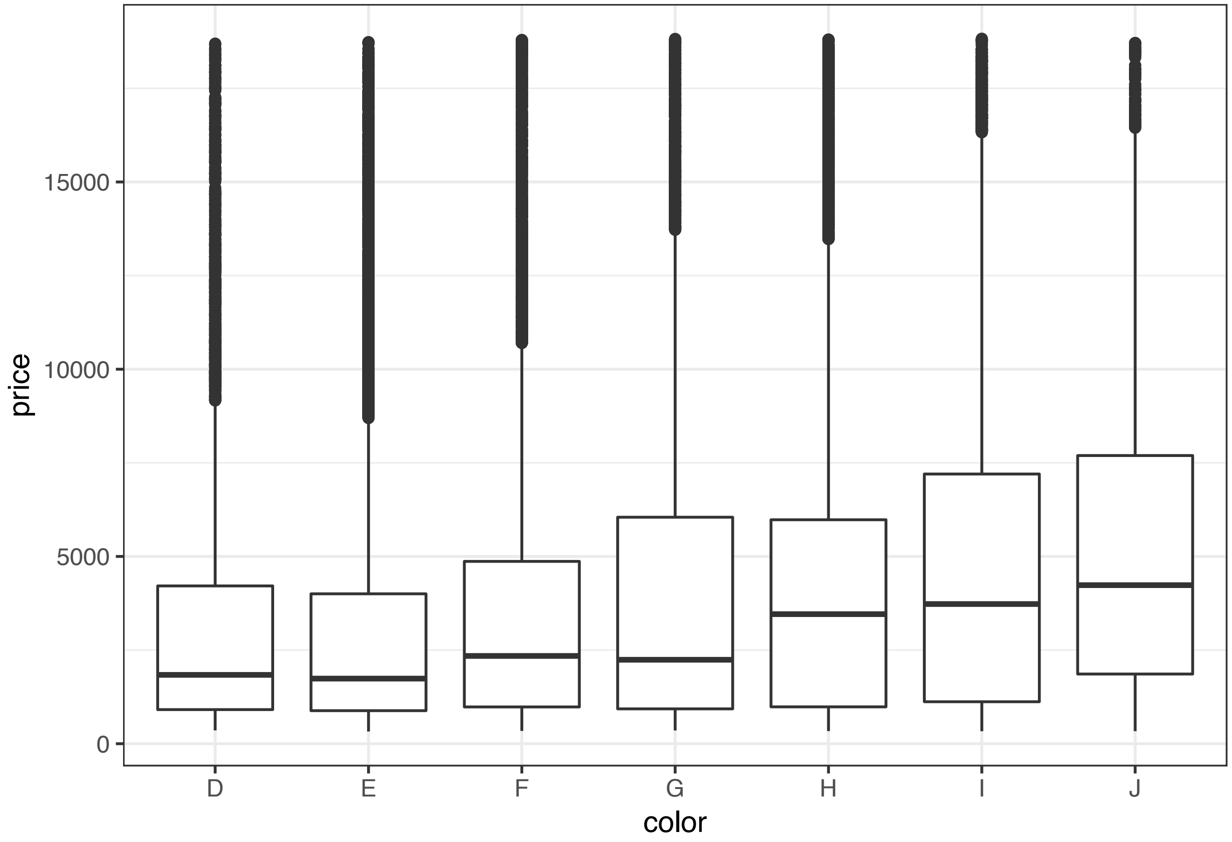
A different way to take the log of `gdpPercap`:

```
> gapminder %>% filter(year==2007) %>% ggplot() +
+   geom_point(aes(x = gdpPercap, y = lifeExp,
+                  size = pop)) +
+   scale_x_log10()
```



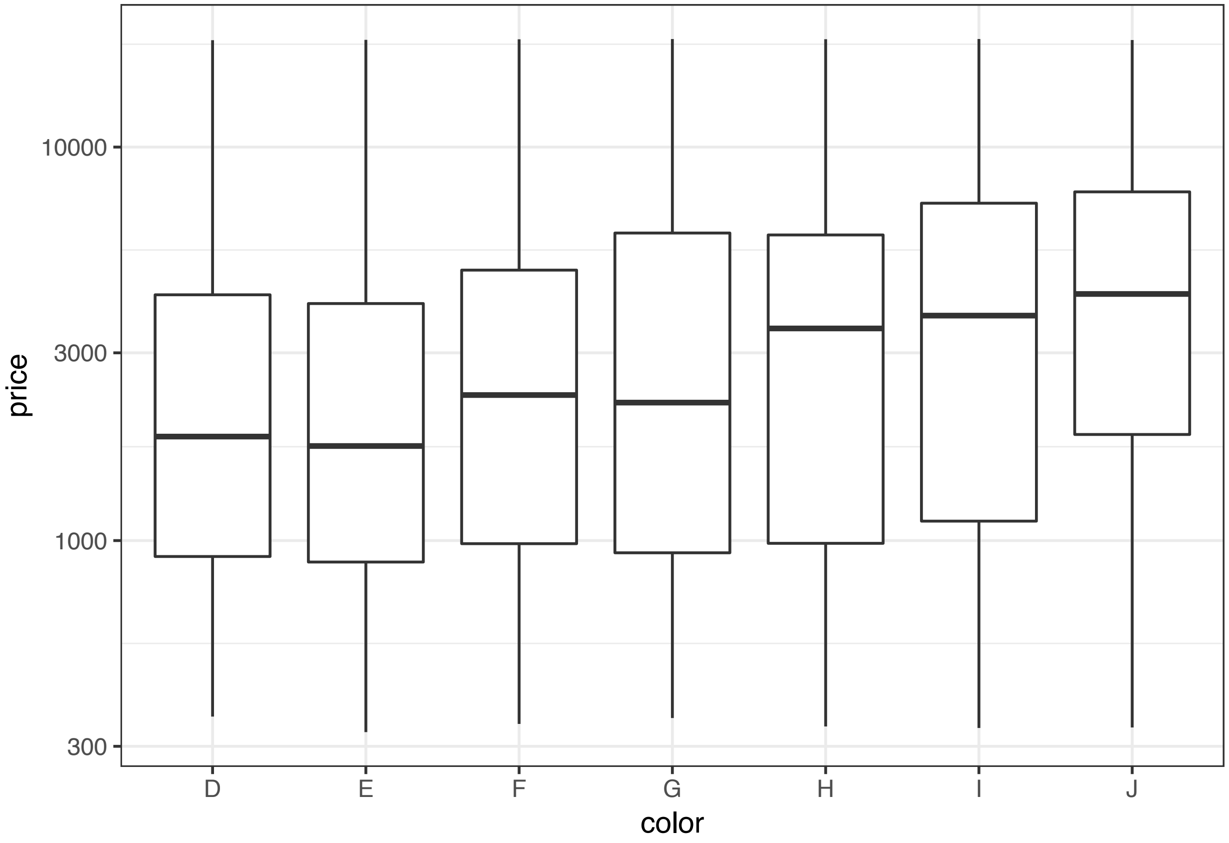
The `price` variable seems to be significantly right-skewed:

```
> ggplot(diamonds) +
+   geom_boxplot(aes(x=color, y=price))
```



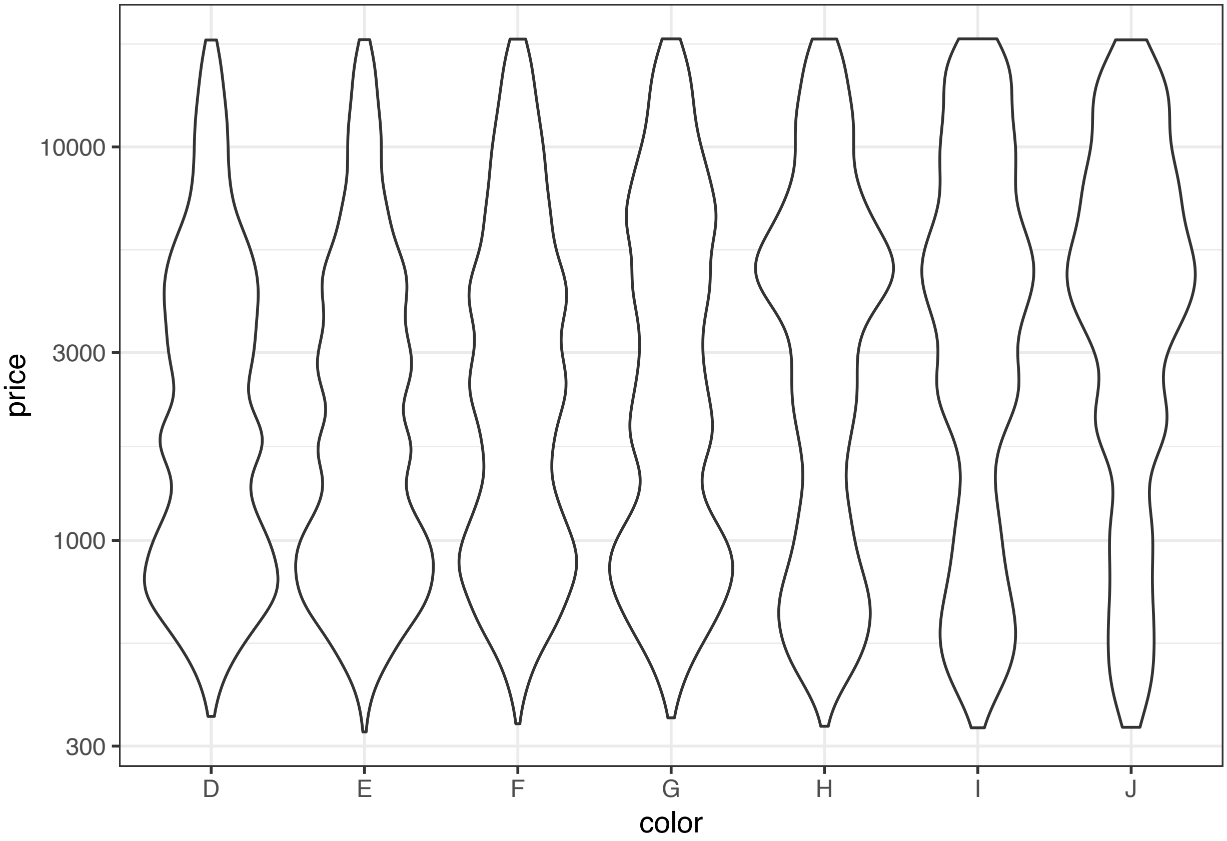
We can try to reduce this skewness by rescaling the variables. We first try to take the log(base=10) of the price variable via scale_y_log10():

```
> ggplot(diamonds) +
+   geom_boxplot(aes(x=color, y=price)) +
+   scale_y_log10()
```



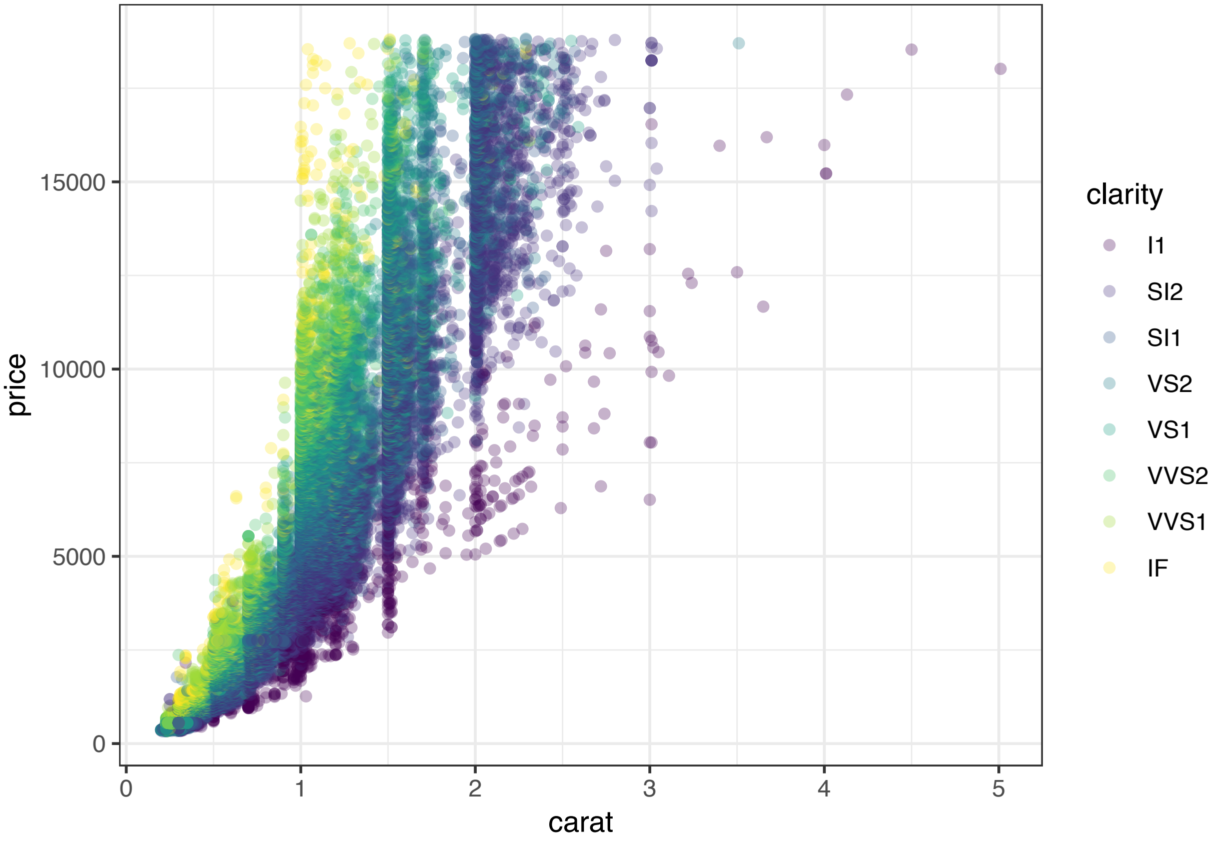
Let's repeat this on the analogous violing plots:

```
> ggplot(diamonds) +
+   geom_violin(aes(x=color, y=price)) +
+   scale_y_log10()
```



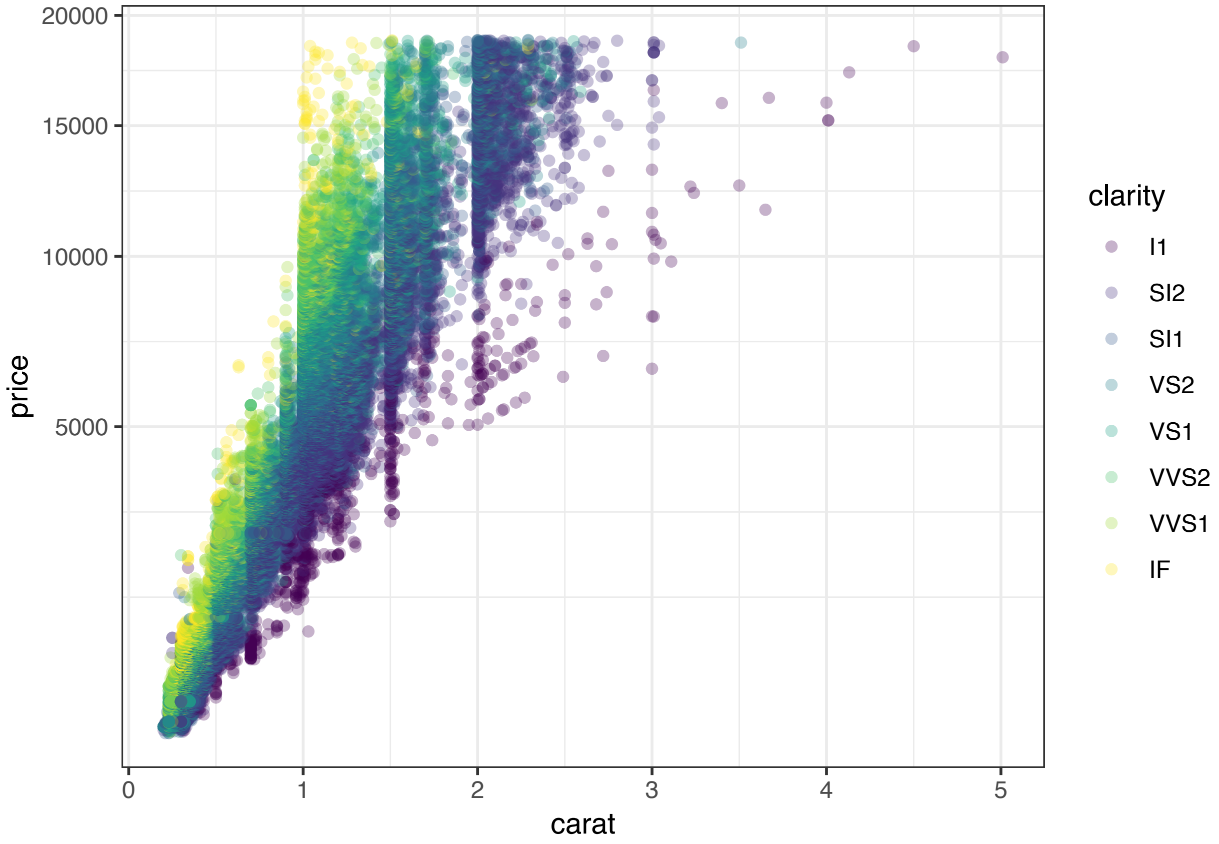
The relationship between `carat` and `price` is nonlinear. Let's explore different transformations to find an approximately linear relationship.

```
> ggplot(data = diamonds) +
+   geom_point(mapping=aes(x=carat, y=price, color=clarity),
+              alpha=0.3)
```



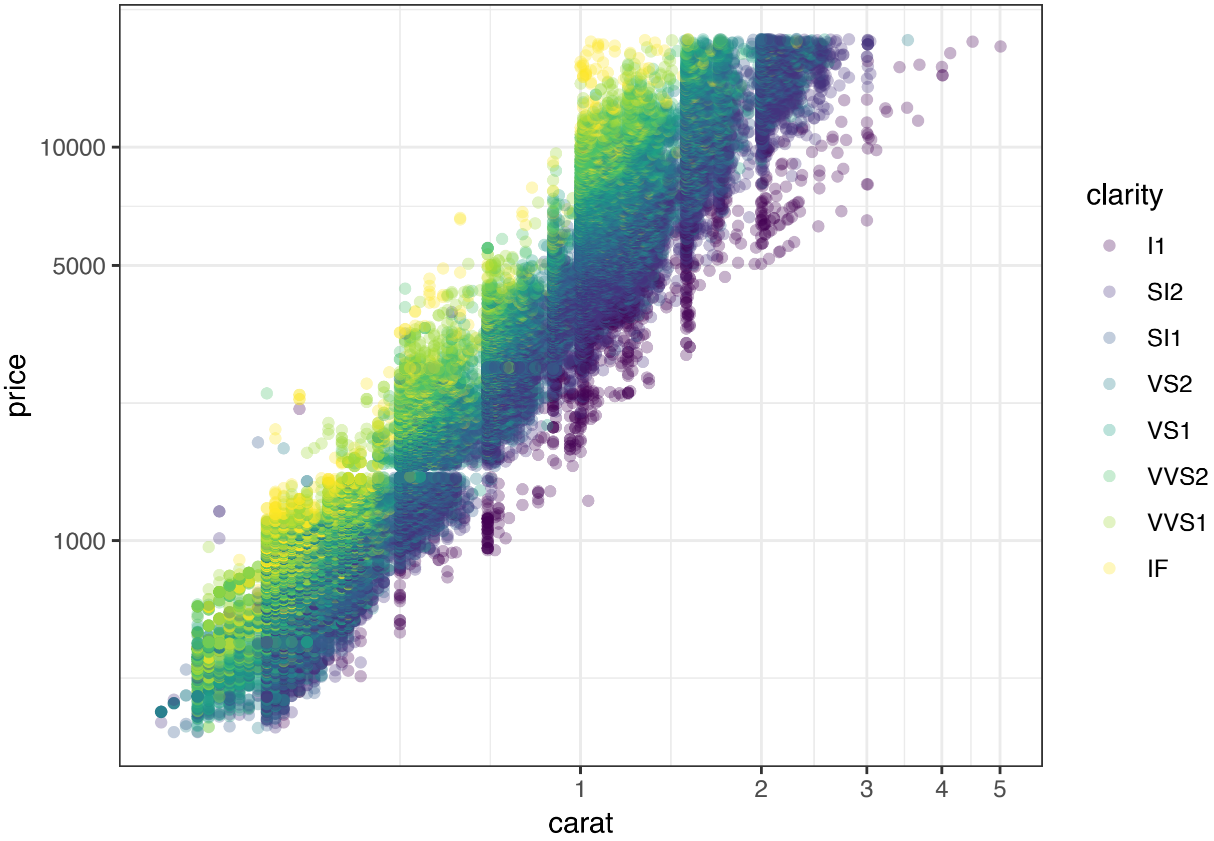
First try to take the squareroot of the the `price` variable:

```
> ggplot(data = diamonds) +
+   geom_point(aes(x=carat, y=price, color=clarity),
+              alpha=0.3) +
+   scale_y_sqrt()
```



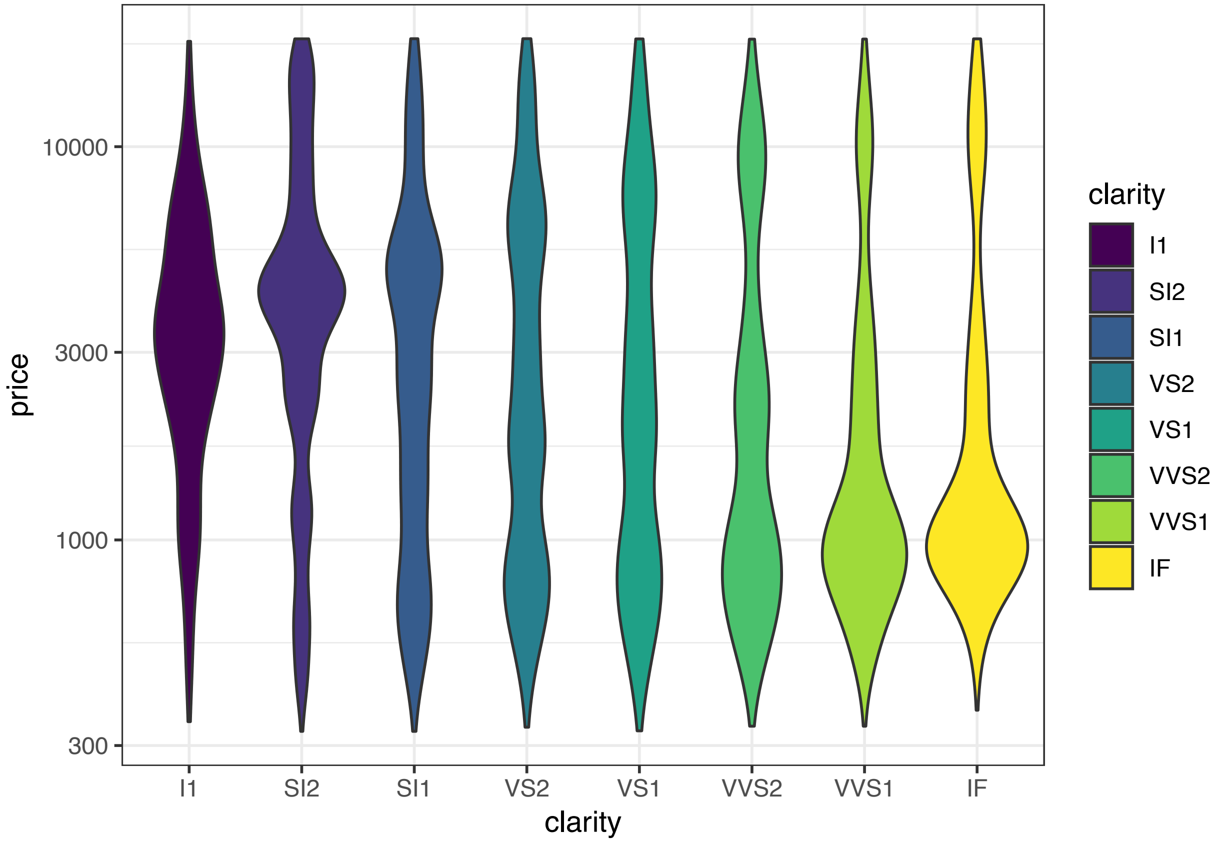
Now let's try to take `log(base=10)` on both the `carat` and `price` variables:

```
> ggplot(data = diamonds) +
+   geom_point(aes(x=carat, y=price, color=clarity), alpha=0.3) +
+   scale_y_log10(breaks=c(1000,5000,10000)) +
+   scale_x_log10(breaks=1:5)
```



Forming a violin plot of `price` stratified by `clarity` and transforming the `price` variable yields an interesting relationship in this data set:

```
> ggplot(diamonds) +
+   geom_violin(aes(x=clarity, y=price, fill=clarity),
+               adjust=1.5) +
+   scale_y_log10()
```


price
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF
clarity
clarity
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF