

# 5\_Scatter\_Plots

Gino Tesei

December 12, 2014

## 1. Making a Basic Scatter Plot

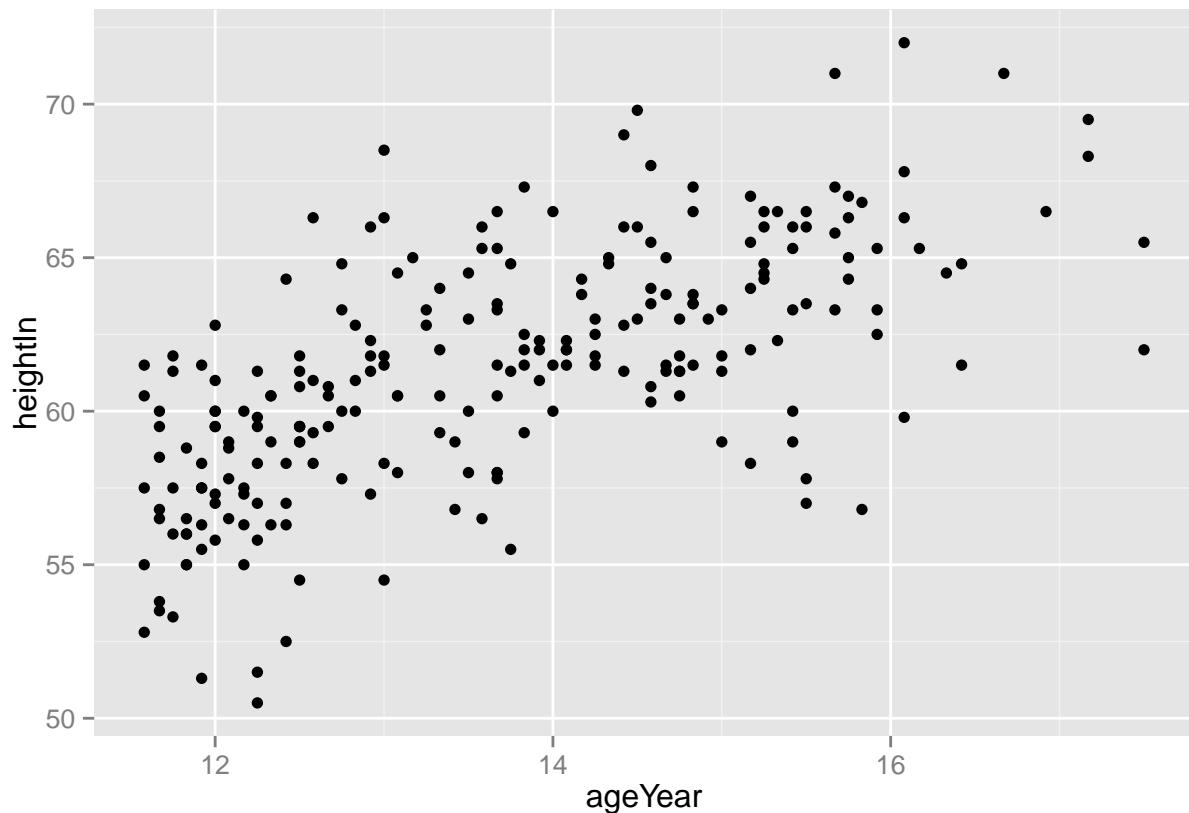
```
library(ggplot2)
library(gcookbook) # For the data set

library(plyr) ##

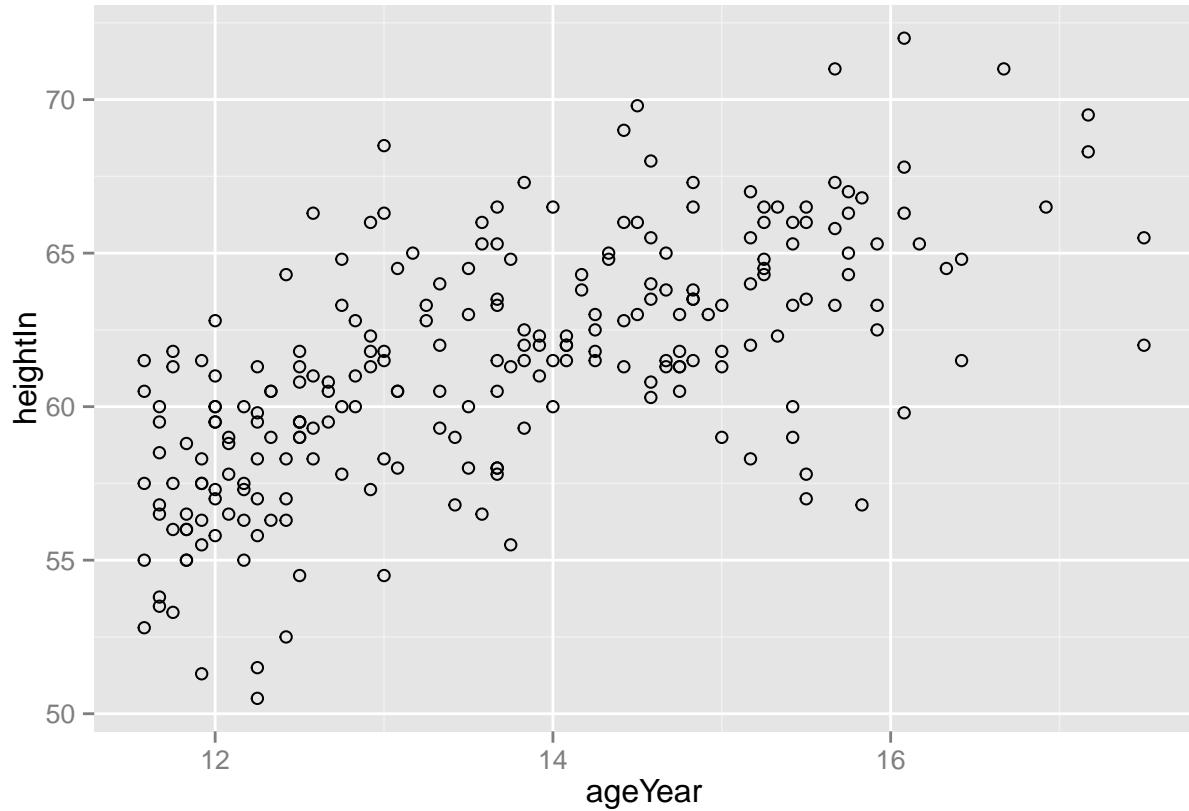
# List the two columns we'll use
head(heightweight[, c("ageYear", "heightIn")])

##   ageYear heightIn
## 1    11.92     56.3
## 2    12.92     62.3
## 3    12.75     63.3
## 4    13.42     59.0
## 5    15.92     62.5
## 6    14.25     62.5

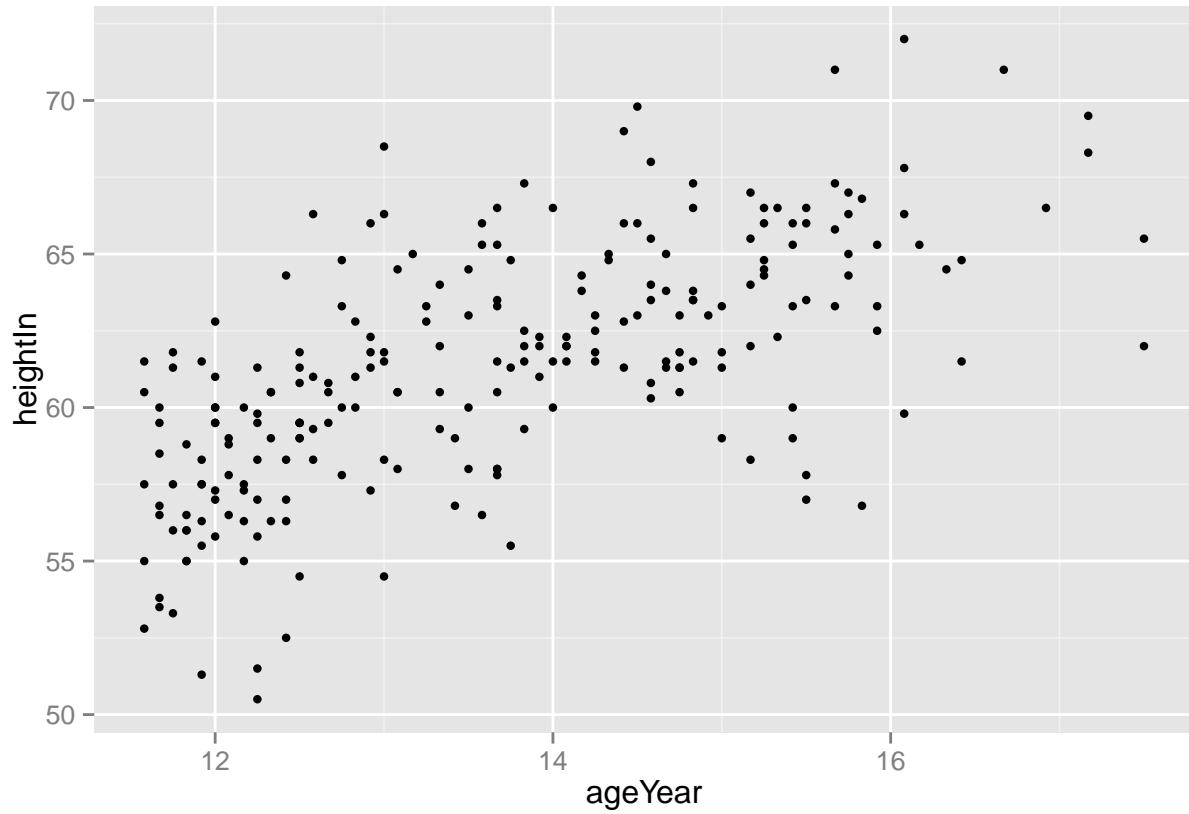
ggplot( heightweight, aes( x = ageYear, y = heightIn)) +
  geom_point()
```



```
## To use different shapes in a scatter plot, set shape.  
ggplot( heightweight, aes( x = ageYear, y = heightIn)) + geom_point( shape = 21)
```



```
## The size of the points can be controlled with size.  
ggplot( heightweight, aes( x = ageYear, y = heightIn)) + geom_point( size = 1.5)
```

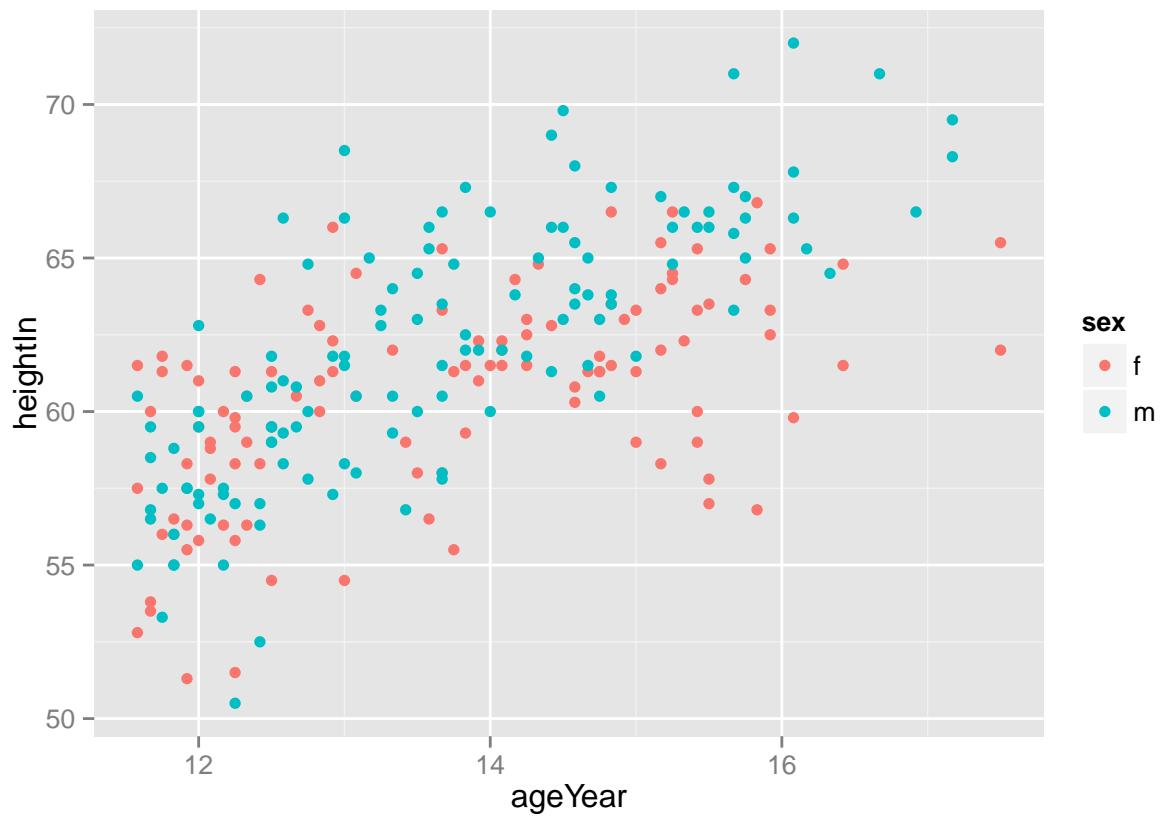


## 2. Grouping Data Points by a Variable Using Shape or Color

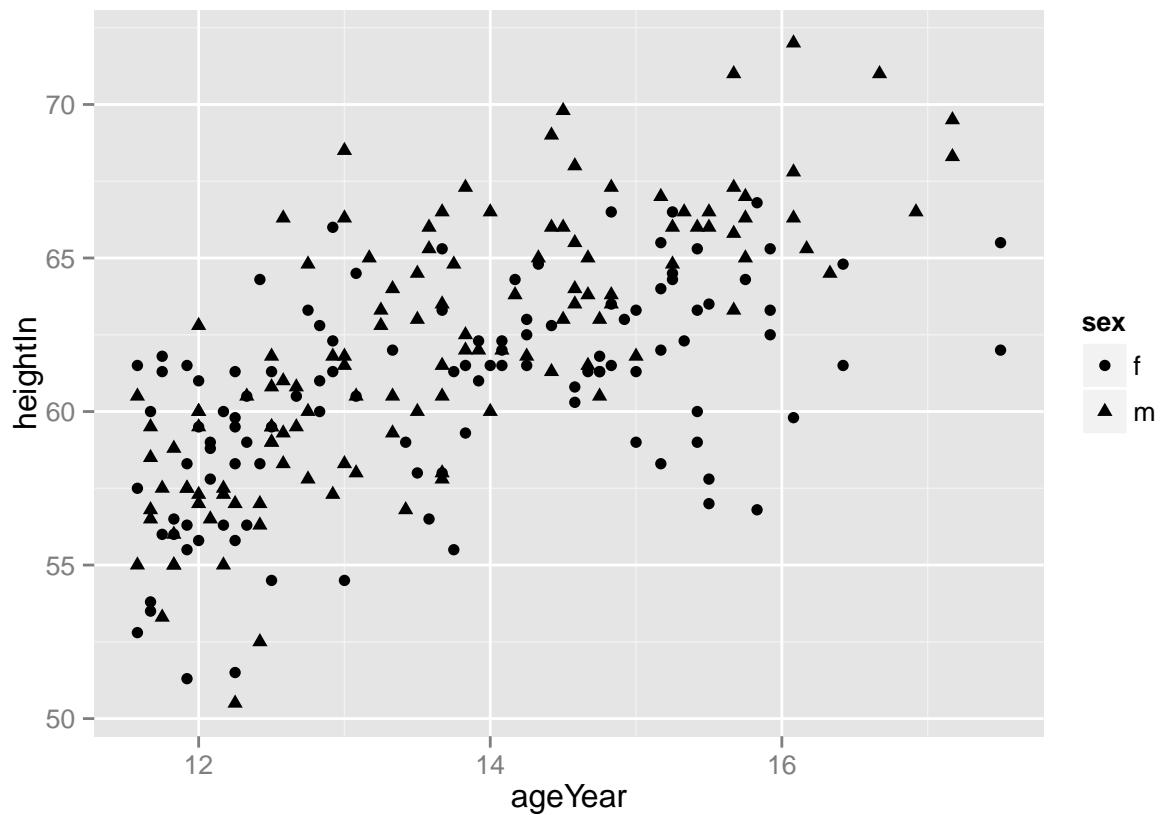
```
# Show the three columns we'll use
head(heightweight[, c("sex", "ageYear", "heightIn")])

##   sex ageYear heightIn
## 1   f    11.92    56.3
## 2   f    12.92    62.3
## 3   f    12.75    63.3
## 4   f    13.42    59.0
## 5   f    15.92    62.5
## 6   f    14.25    62.5

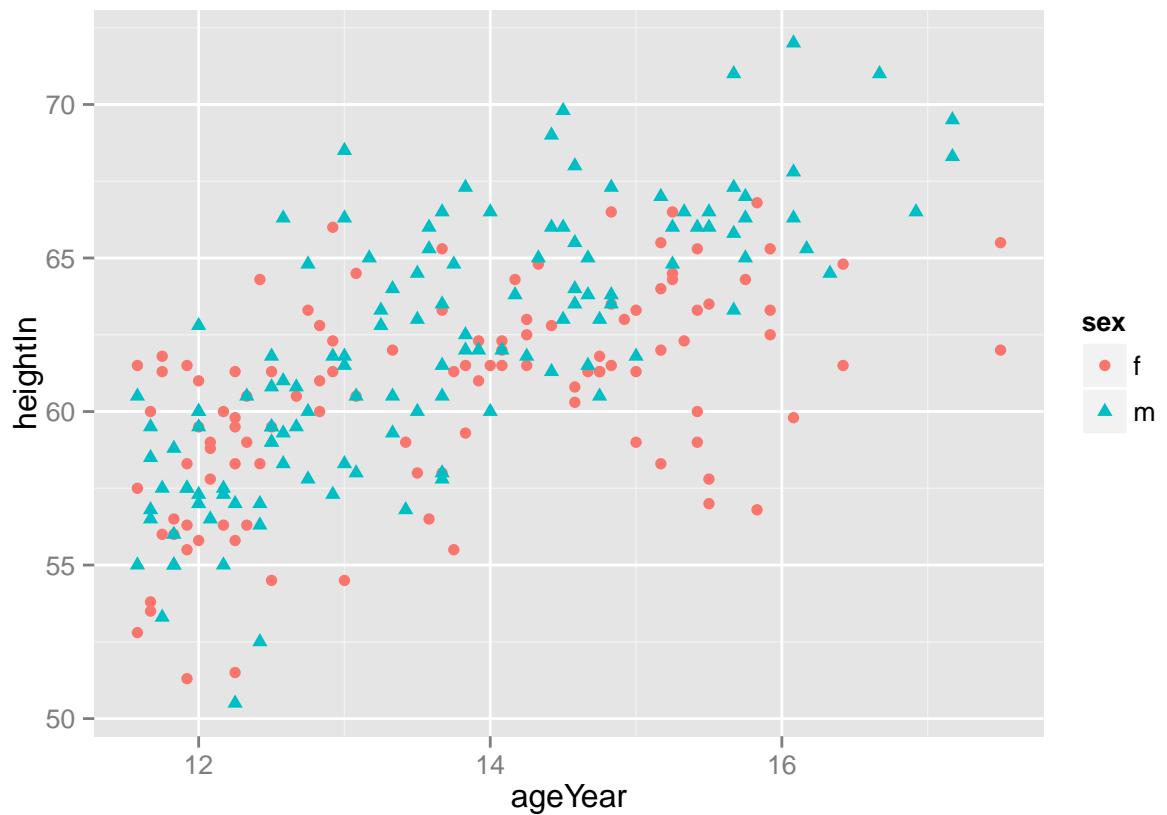
ggplot( heightweight, aes( x = ageYear, y = heightIn, colour = sex)) +
  geom_point()
```



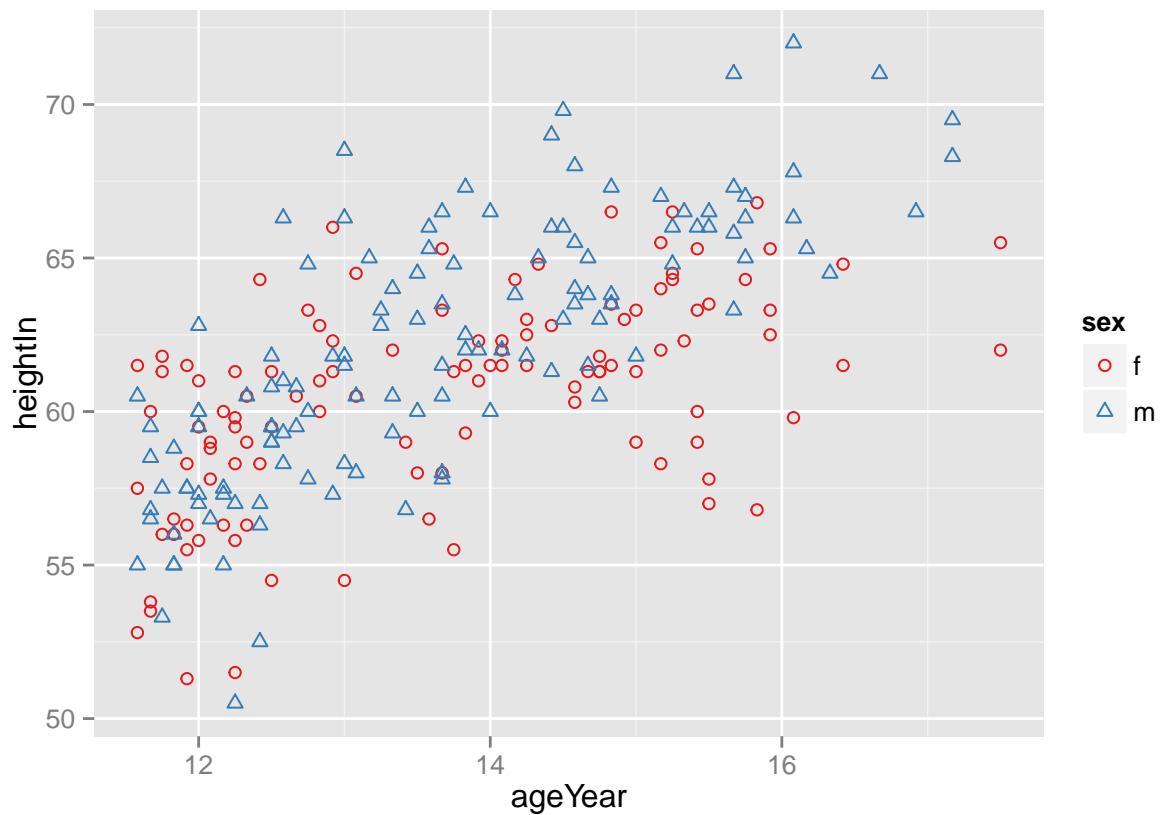
```
ggplot( heightweight, aes( x = ageYear, y = heightIn, shape = sex)) +  
  geom_point()
```



```
## It is possible to map a variable to both shape and colour,  
ggplot( heightweight, aes( x = ageYear, y = heightIn, shape = sex, colour = sex)) + geom_point()
```

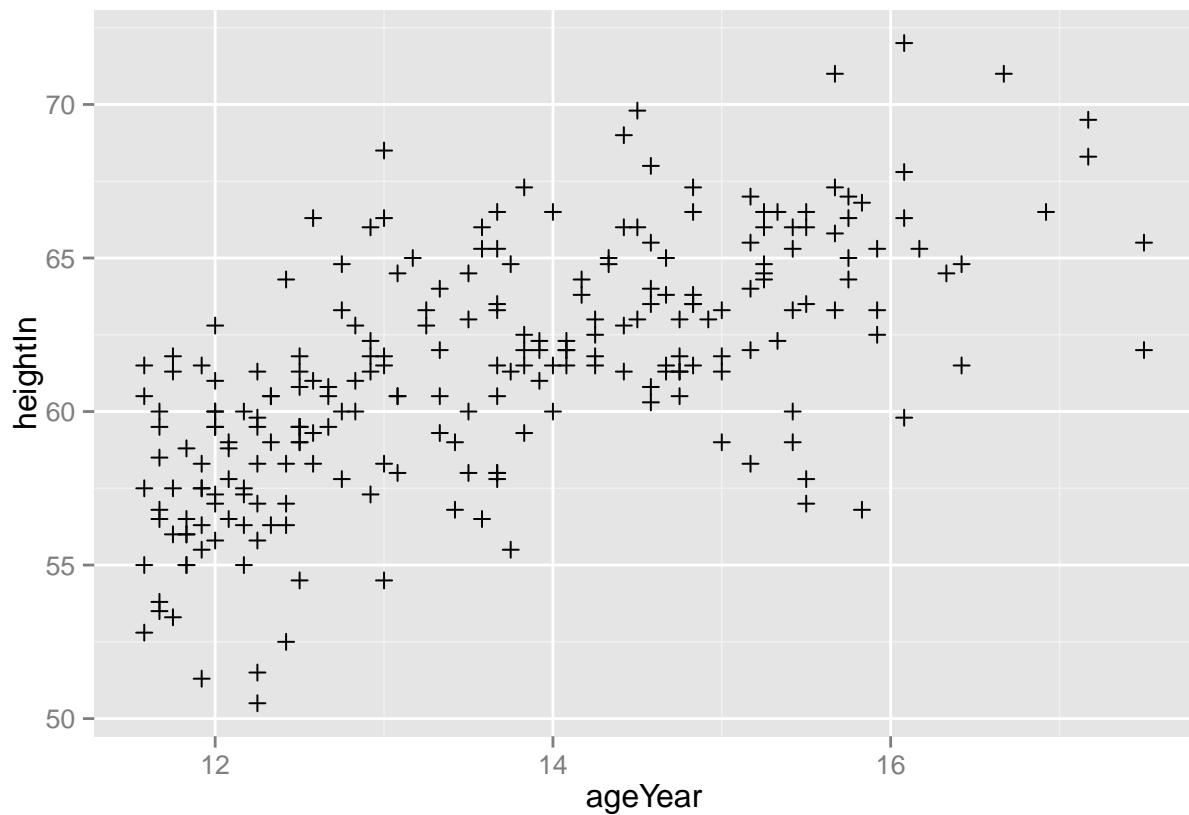


```
## set different shapes and colors for the grouping variables
ggplot( heightweight, aes( x = ageYear, y = heightIn, shape = sex, colour = sex)) +
  geom_point() +
  scale_shape_manual( values = c(1,2)) +
  scale_colour_brewer( palette ="Set1")
```

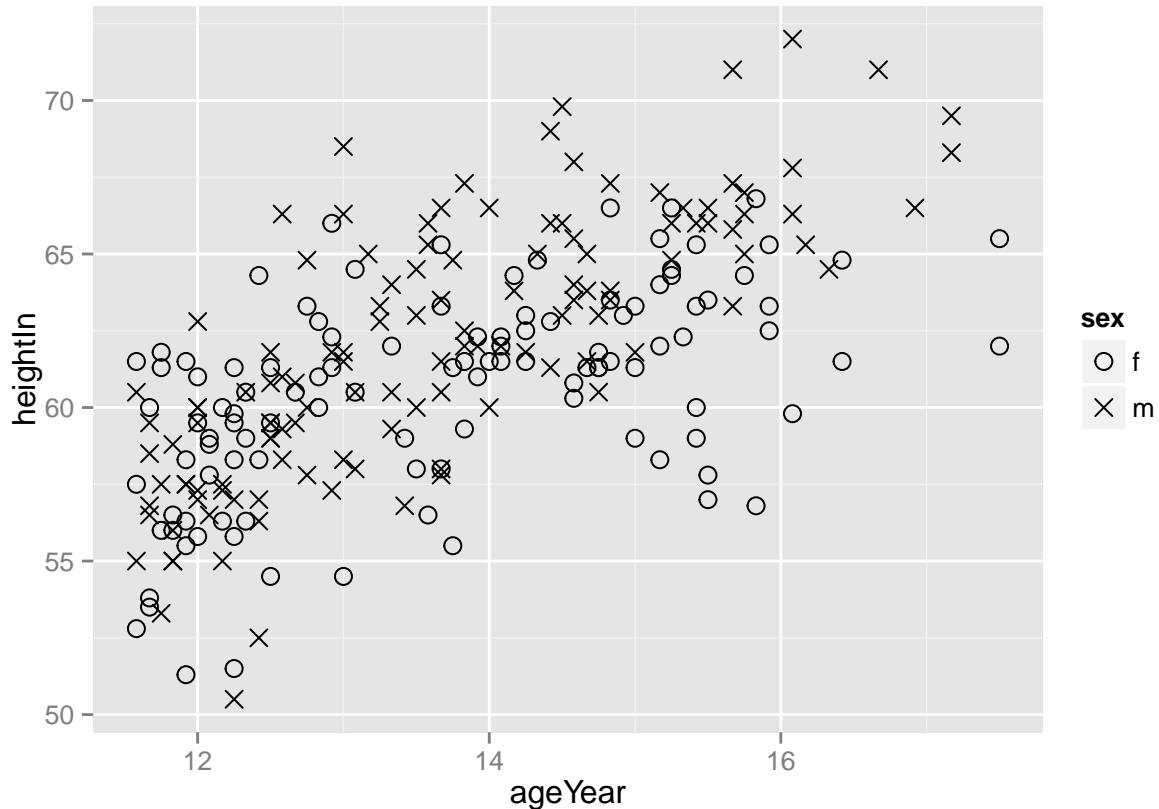


### 3. Using Different Point Shapes

```
ggplot( heightweight, aes( x = ageYear, y = heightIn)) +  
  geom_point( shape = 3)
```



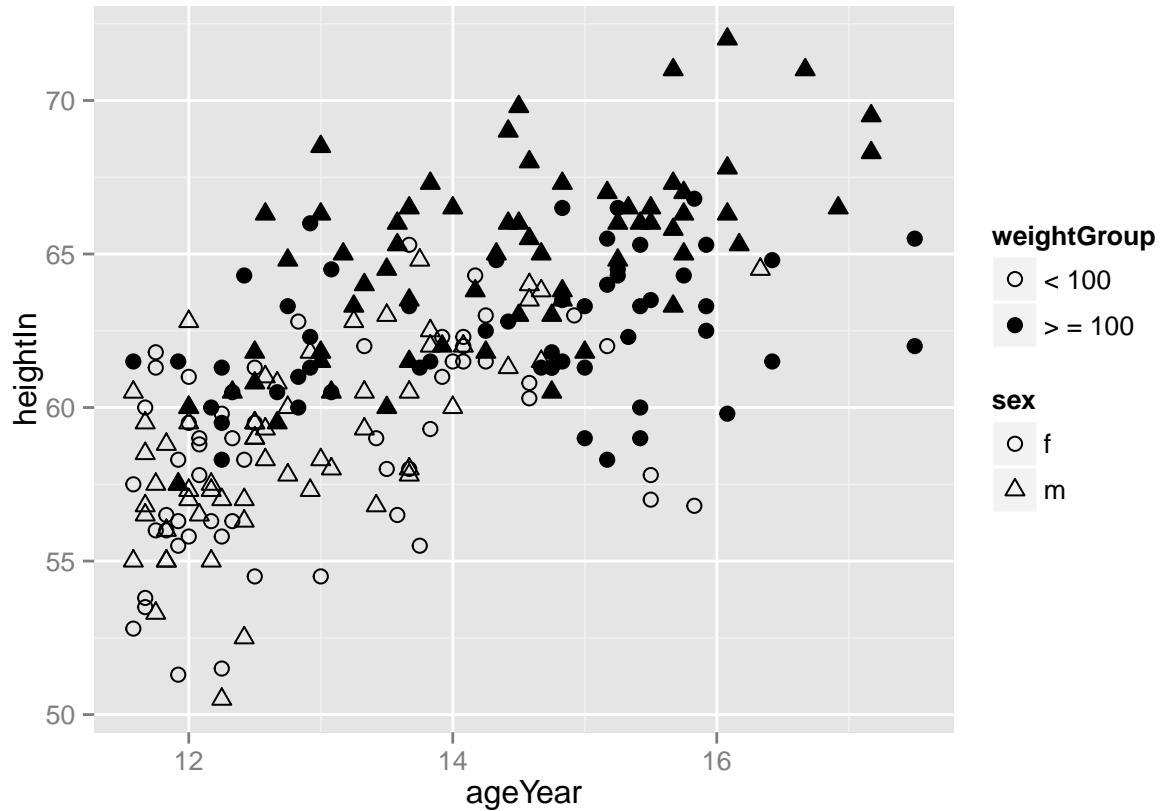
```
# Use slightly larger points and use a shape scale with custom values
ggplot( heightweight, aes( x = ageYear, y = heightIn, shape = sex)) +
  geom_point( size = 3) +
  scale_shape_manual( values = c( 1, 4))
```



```
## It's possible to have the shape represent one variable and the fill (empty or solid) represent another
# Make a copy of the data
hw <- heightweight

# Categorize into < 100 and > = 100 groups
hw$weightGroup <- cut( hw$weightLb, breaks = c(-Inf, 100, Inf),
                        labels = c("< 100", "> = 100"))

# Use shapes with fill and color, and use colors that are empty (NA) and # filled
ggplot( hw, aes( x = ageYear, y = heightIn, shape = sex, fill = weightGroup)) +
  geom_point( size = 2.5) + scale_shape_manual( values = c( 21, 24)) +
  scale_fill_manual( values = c( NA, "black"), guide = guide_legend( override.aes = list( shape = 21)))
```

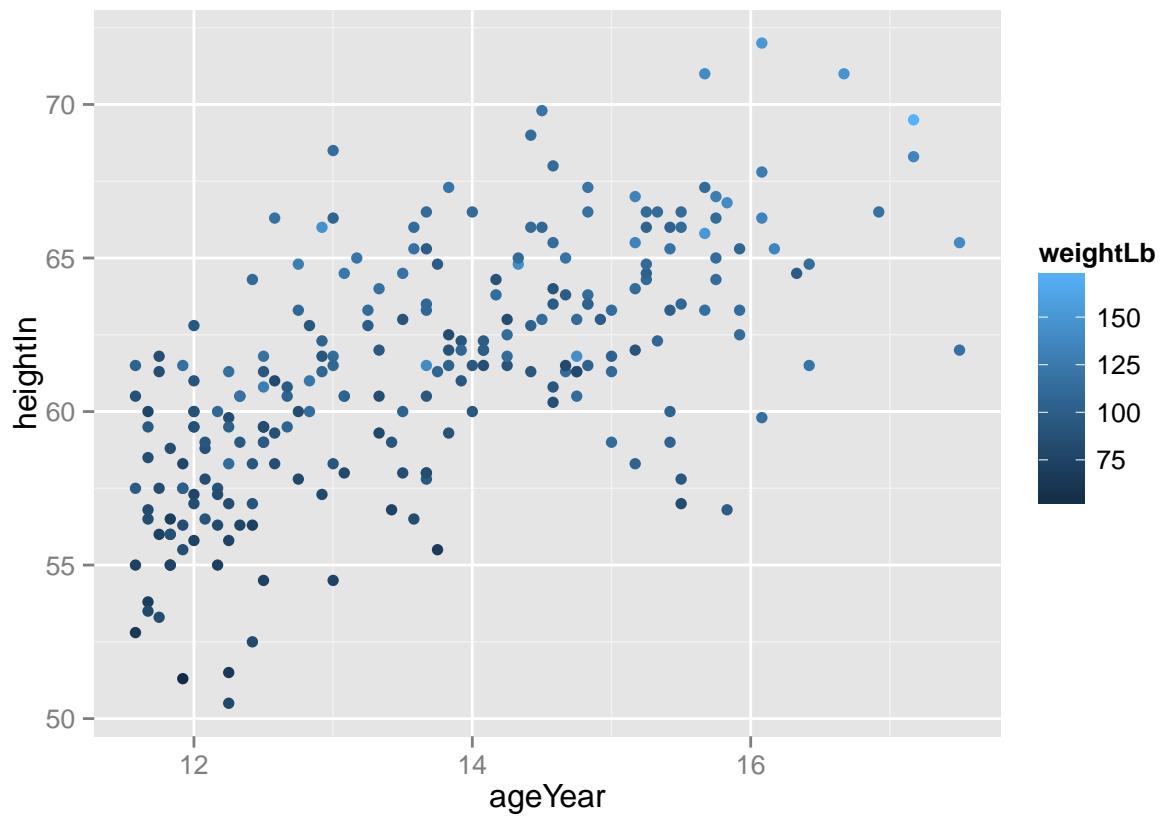


#### 4. Mapping a Continuous Variable to Color or Size

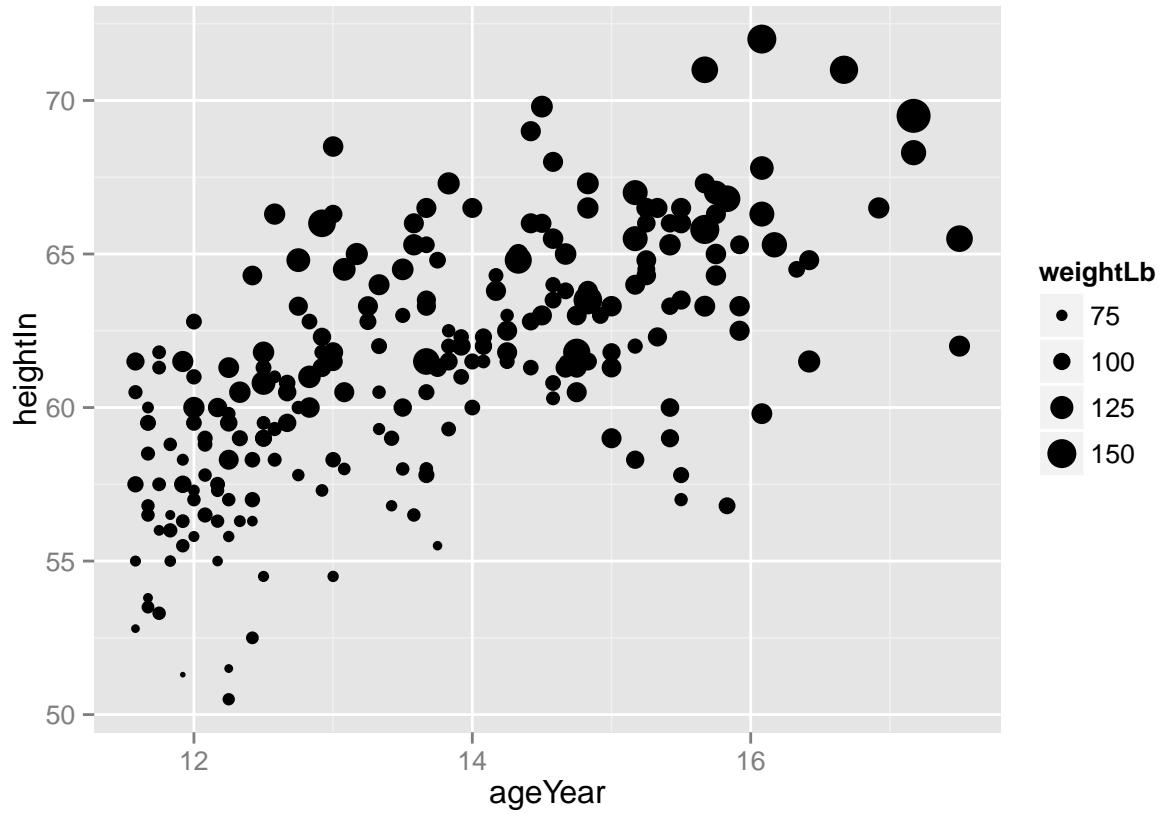
```
# List the four columns we'll use
head(heightweight[, c("sex", "ageYear", "heightIn", "weightLb")])
```

```
##   sex ageYear heightIn weightLb
## 1   f    11.92     56.3    85.0
## 2   f    12.92     62.3   105.0
## 3   f    12.75     63.3   108.0
## 4   f    13.42     59.0    92.0
## 5   f    15.92     62.5   112.5
## 6   f    14.25     62.5   112.0
```

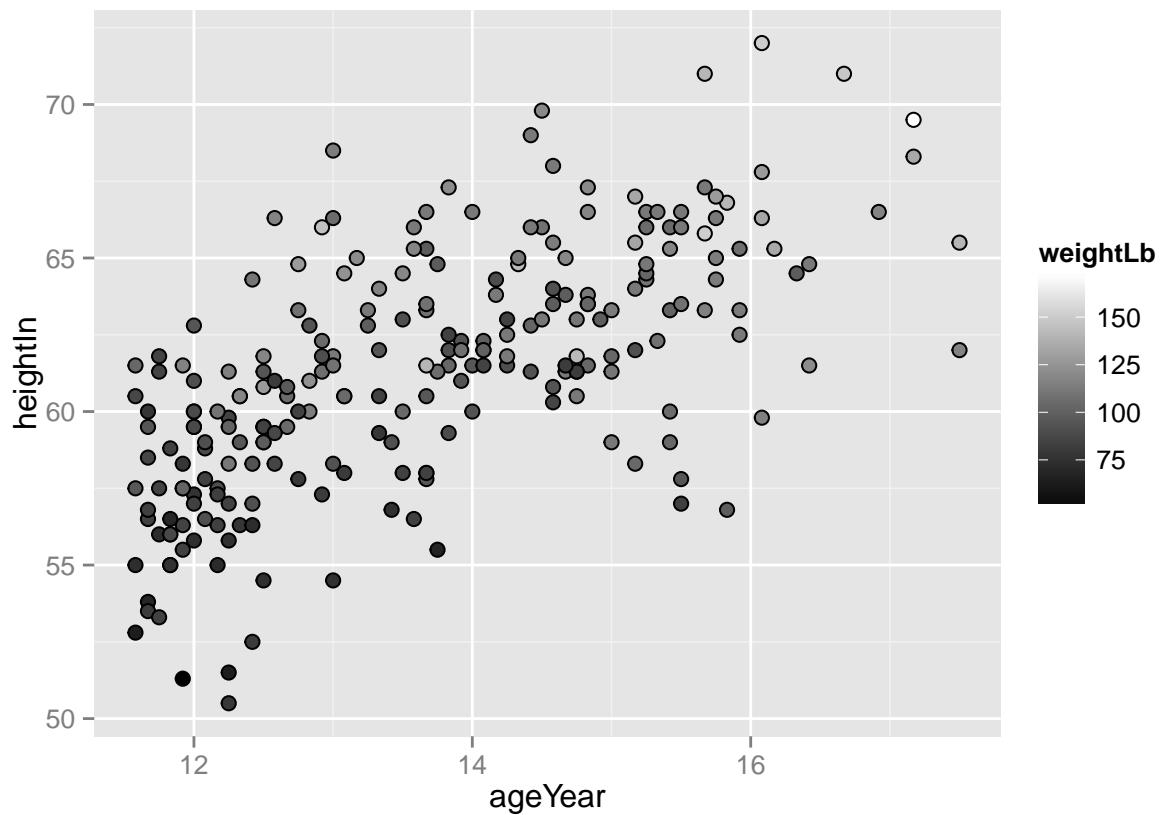
```
ggplot( heightweight, aes( x = ageYear, y = heightIn, colour = weightLb)) +
  geom_point()
```



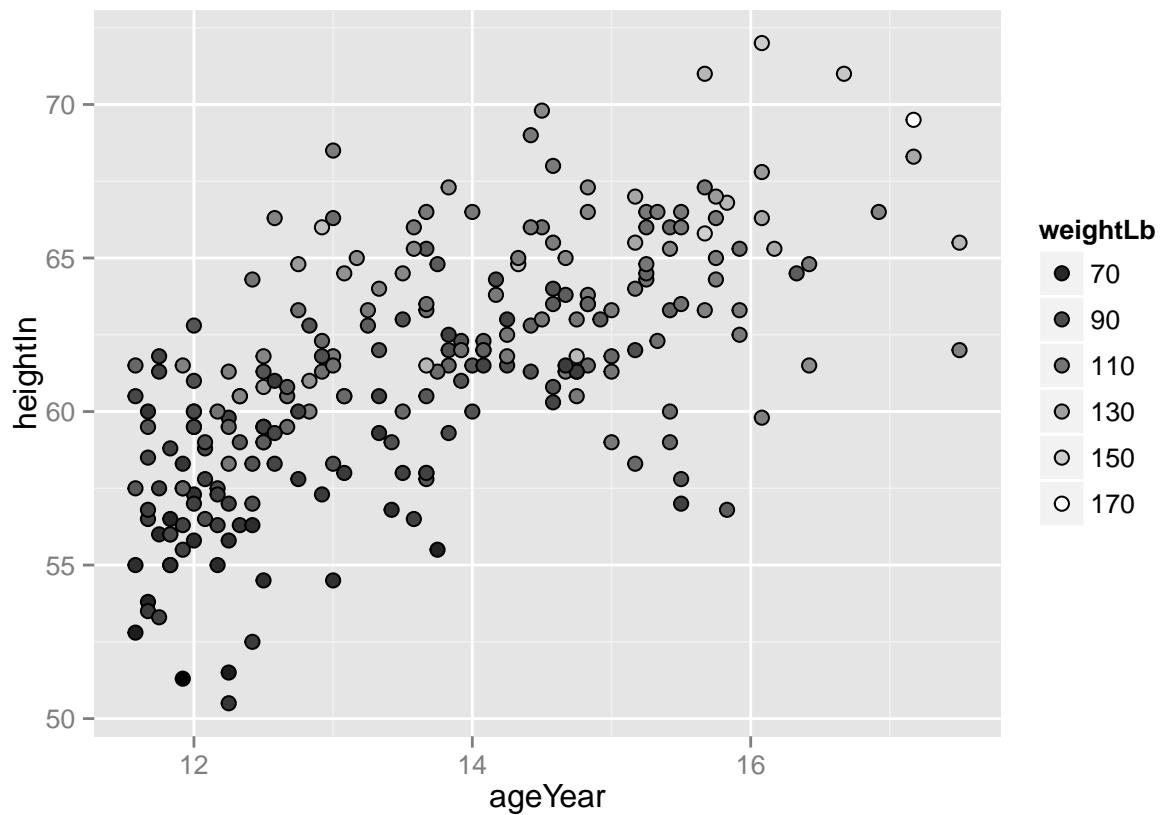
```
ggplot( heightweight, aes( x = ageYear, y = heightIn, size = weightLb ) ) +  
  geom_point()
```



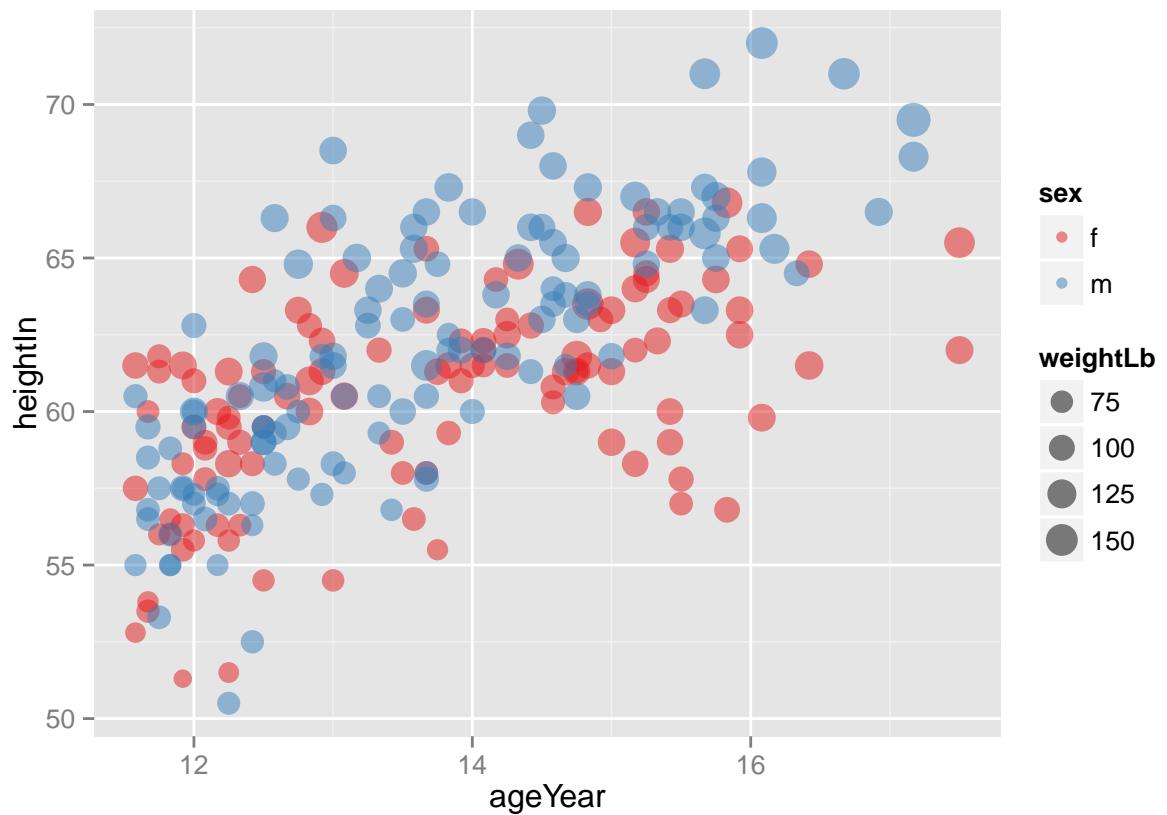
```
## set the fill gradient to go from black to white and make the points larger
ggplot( heightweight, aes( x = ageYear, y = heightIn, fill = weightLb)) +
  geom_point( shape = 21, size = 2.5) +
  scale_fill_gradient( low ="black", high ="white")
```



```
# Using guide_legend() will result in a discrete legend instead of a colorbar
ggplot( heightweight, aes( x = ageYear, y = heightIn, fill = weightLb)) +
  geom_point( shape = 21, size = 2.5) +
  scale_fill_gradient( low ="black", high ="white", breaks = seq( 70, 170, by = 20), guide = guide_leger
```



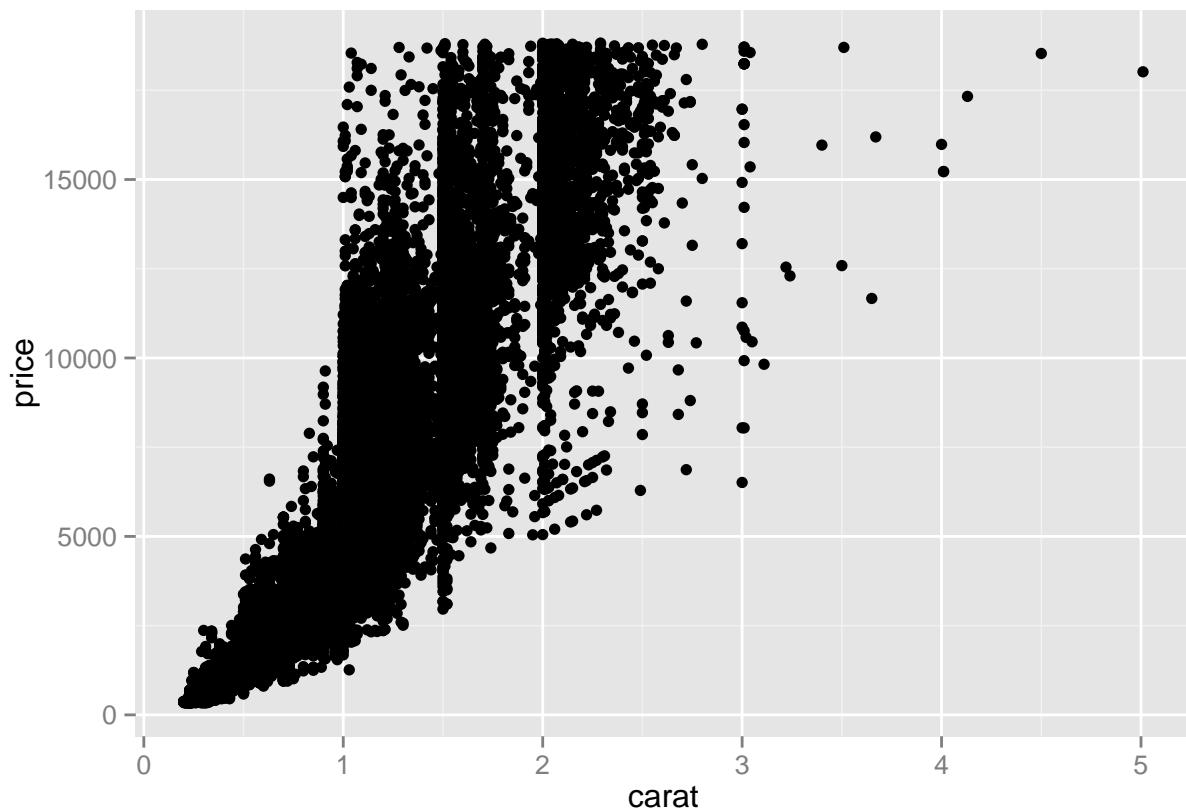
```
## best
ggplot( heightweight, aes( x = ageYear, y = heightIn, size = weightLb, colour = sex)) +
  geom_point( alpha = .5) +
  scale_size_area() + # Make area proportional to numeric value
  scale_colour_brewer( palette = "Set1")
```



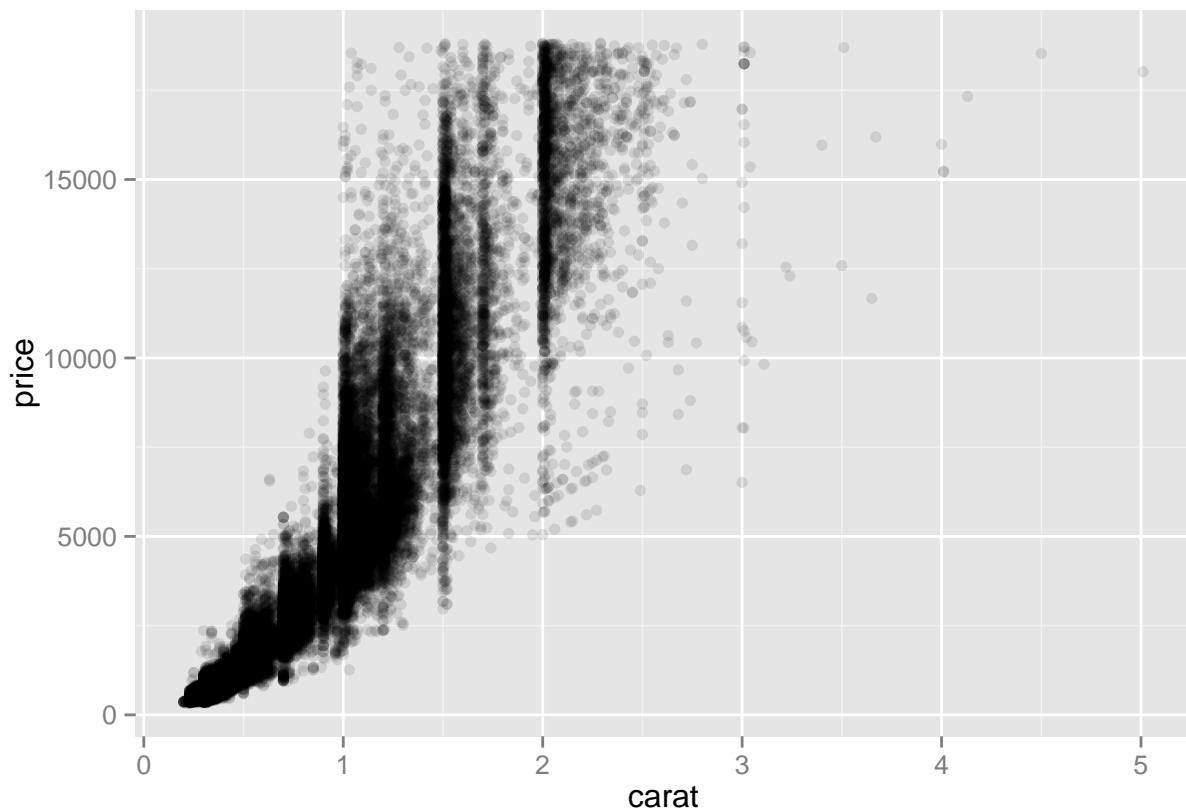
## 5. Dealing with Overplotting

You have many points and they obscure each other.

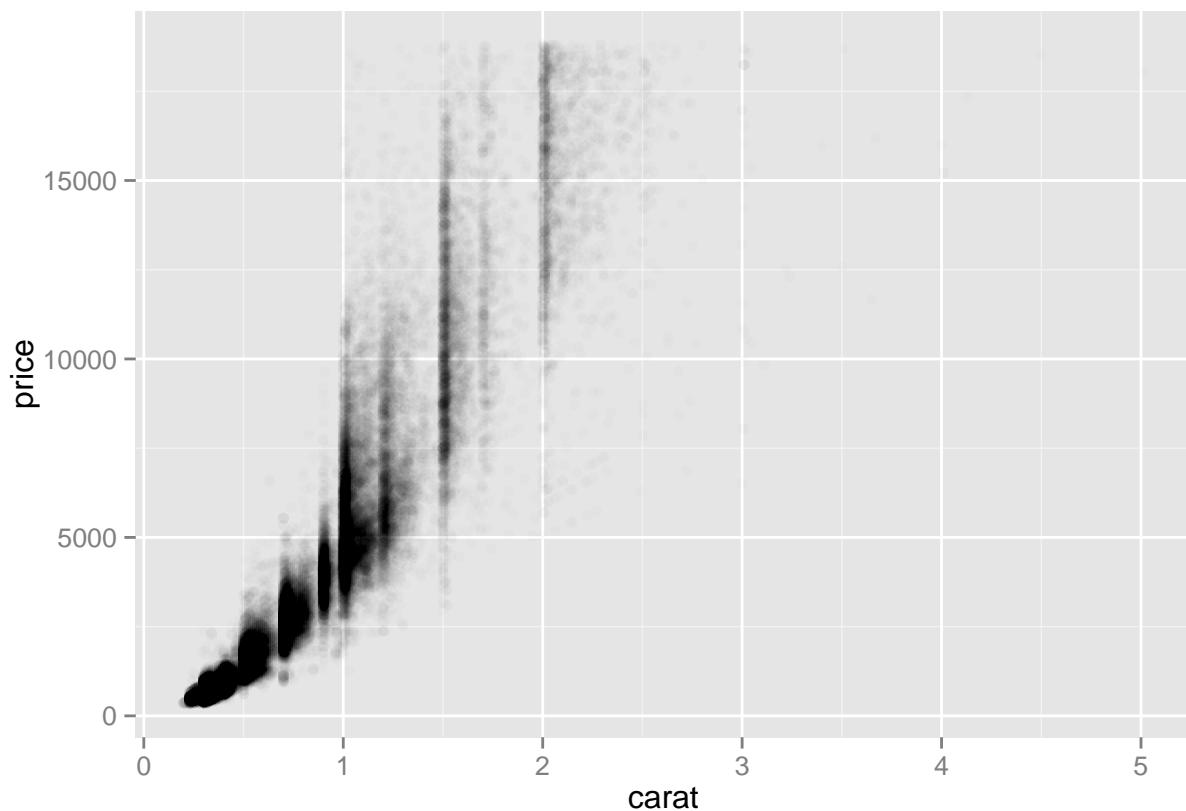
```
sp <- ggplot( diamonds, aes( x = carat, y = price))
sp +
  geom_point()
```



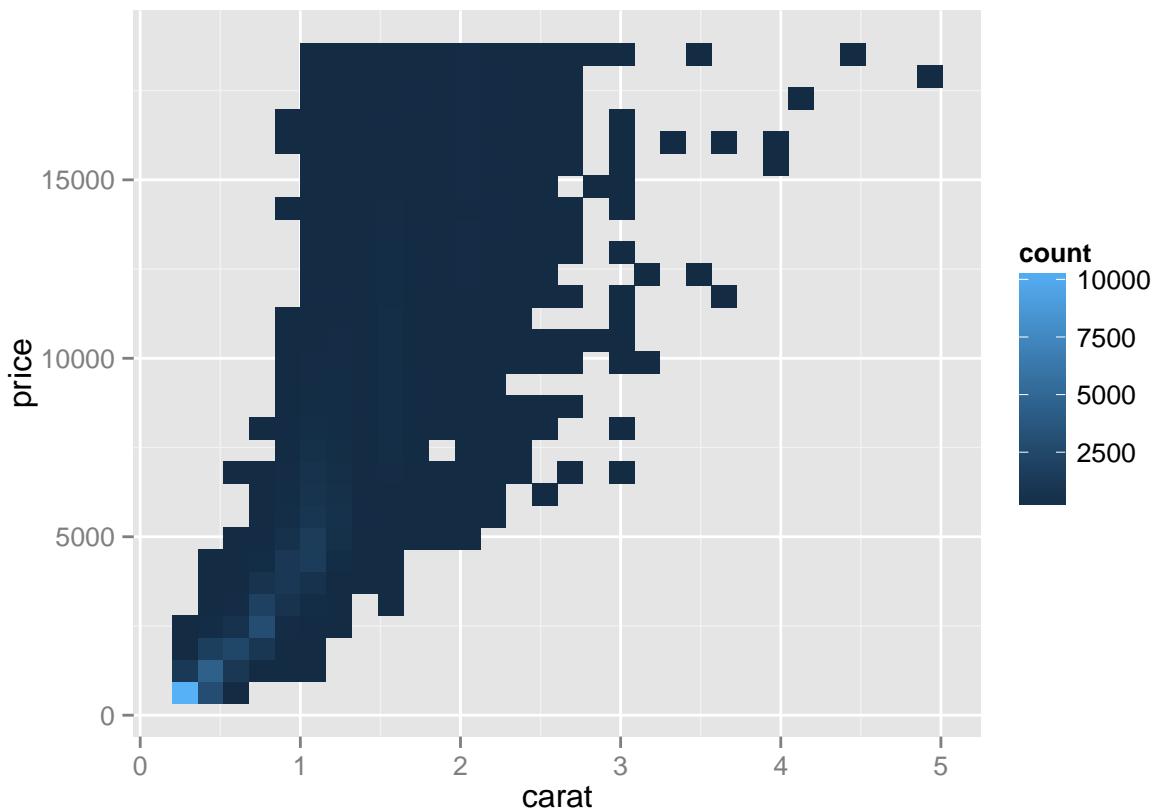
```
## We can make the points semitransparent using alpha,  
sp +  
  geom_point( alpha = .1)
```



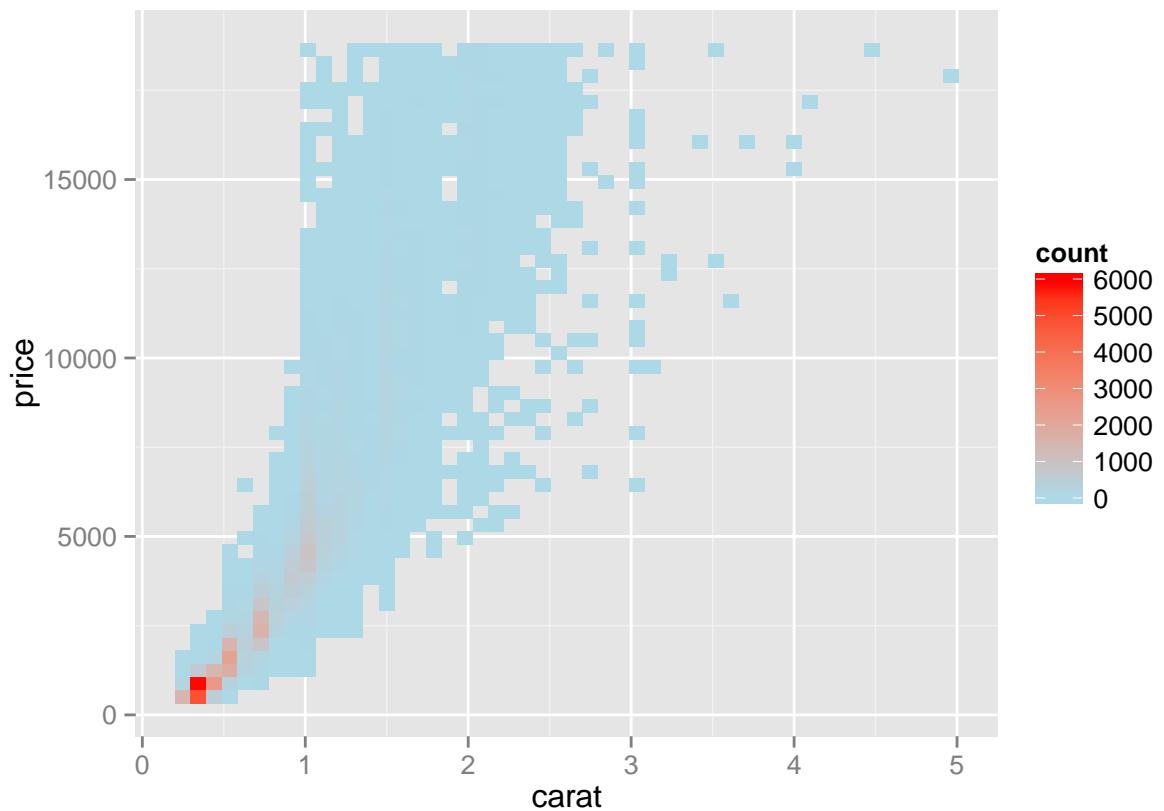
```
sp +  
  geom_point( alpha = .01)
```



```
## Another solution is to bin the points into rectangles and map the density of the points to the fill color
sp +
  stat_bin2d()
```

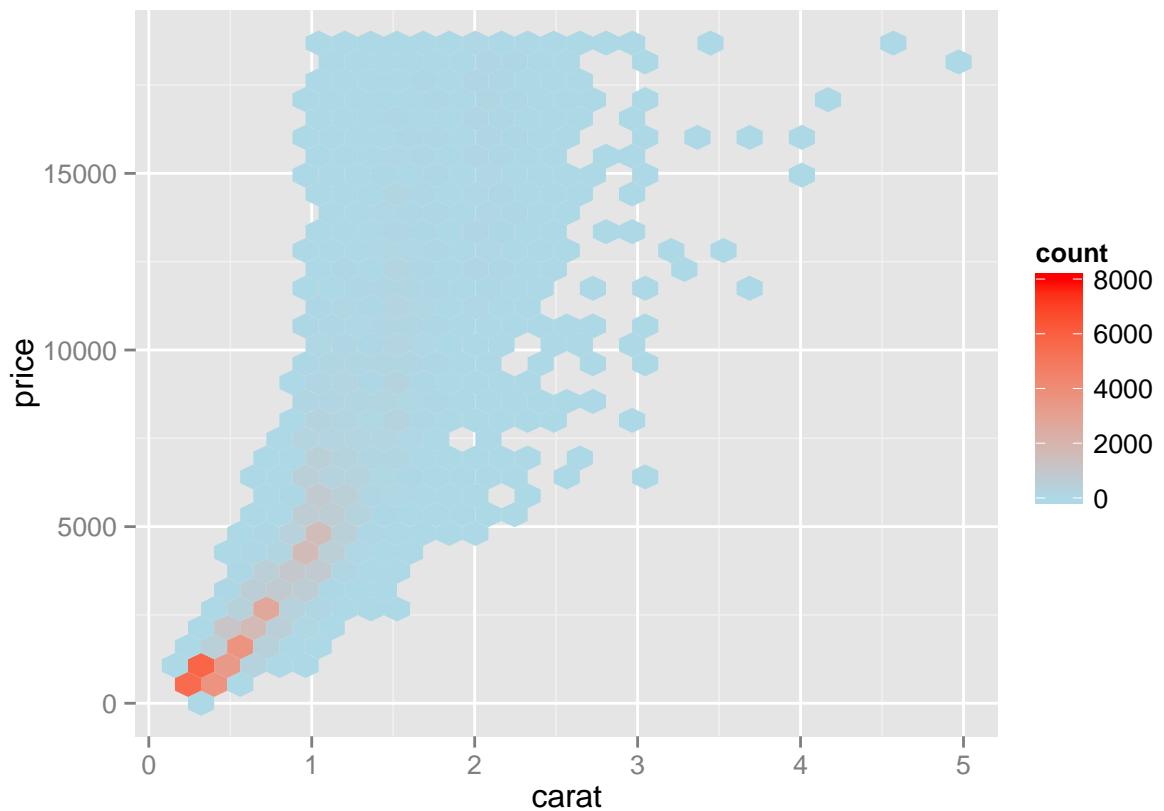


```
sp +
  stat_bin2d( bins = 50) +
  scale_fill_gradient( low ="lightblue", high ="red", limits = c( 0, 6000))
```

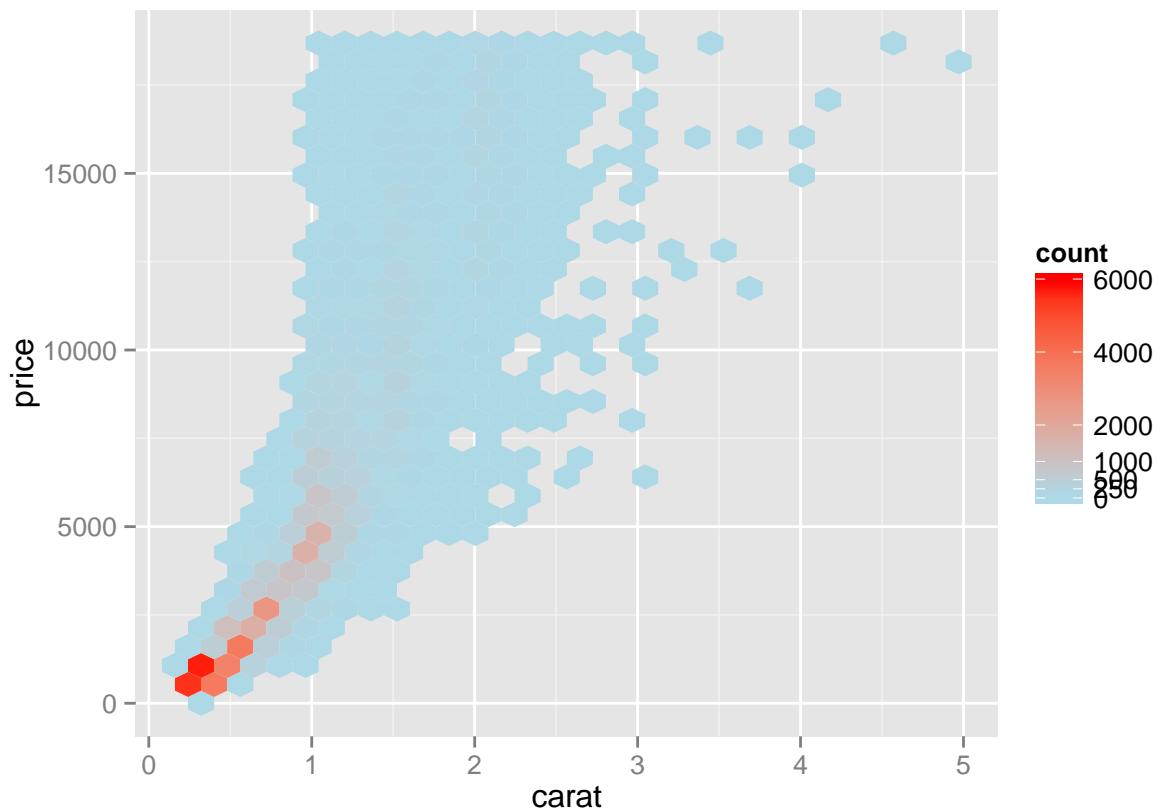


Another alternative is to bin the data into hexagons instead of rectangles, with `stat_binhex`. It works just like `stat_bin2d`().

```
library( hexbin)
sp +
  stat_binhex() +
  scale_fill_gradient( low ="lightblue", high ="red", limits = c( 0, 8000))
```



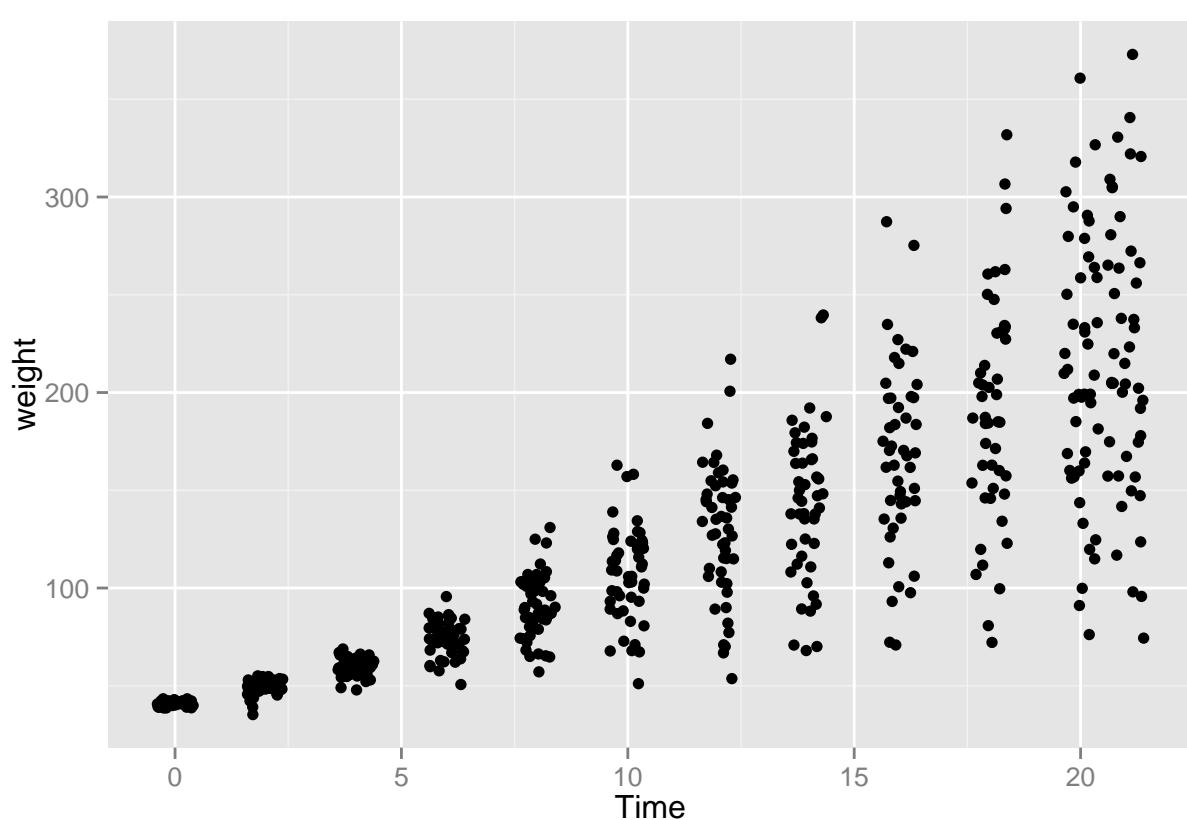
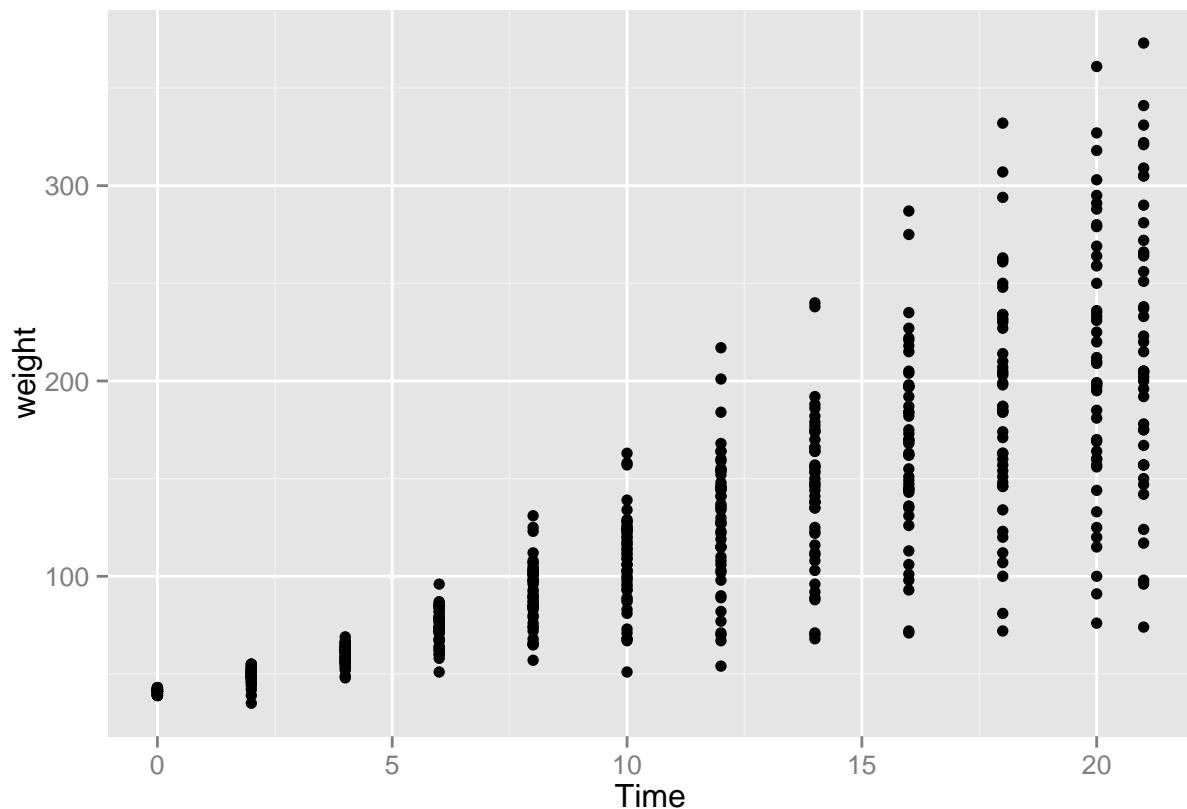
```
sp +
  stat_binhex() +
  scale_fill_gradient( low ="lightblue", high ="red", breaks = c( 0, 250, 500, 1000, 2000, 4000, 6000),
                      limits = c( 0, 6000))
```



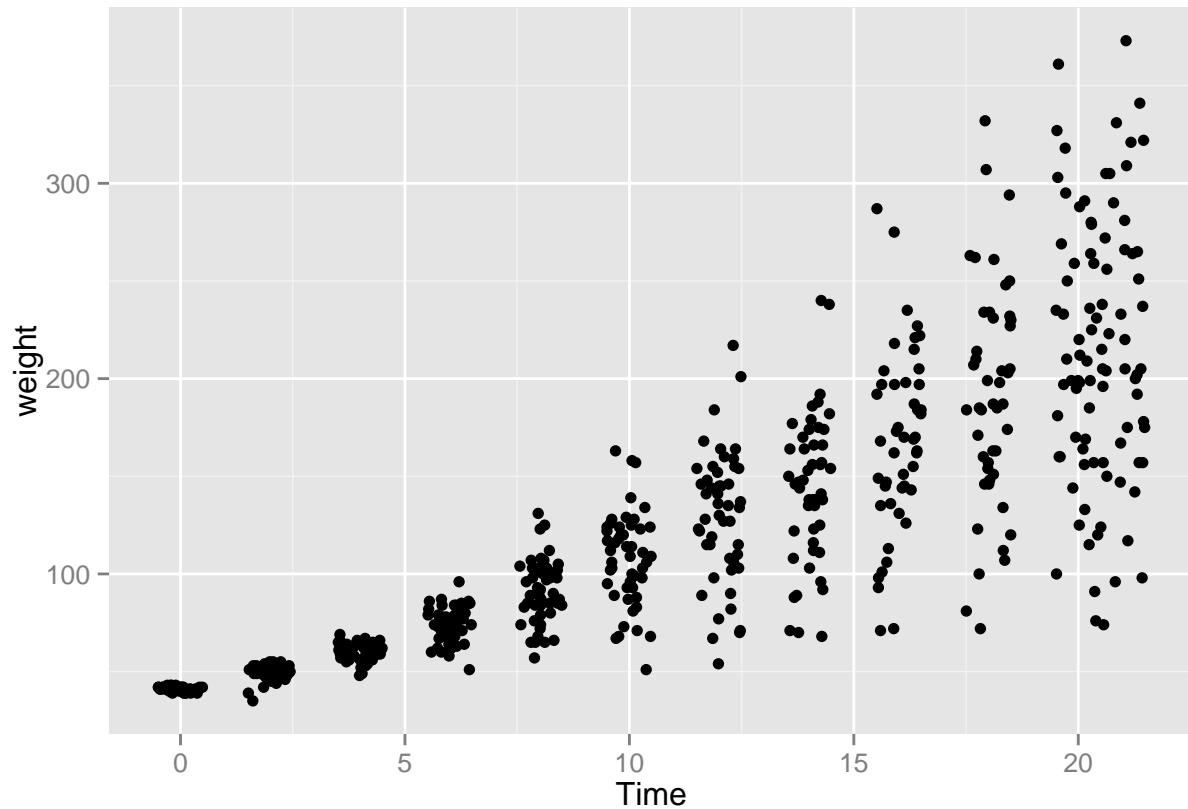
Overplotting can also occur when the data is discrete on one or both axes. In these cases, you can randomly jitter the points with `position_jitter()`.

```
sp1 <- ggplot( ChickWeight, aes( x = Time, y = weight))

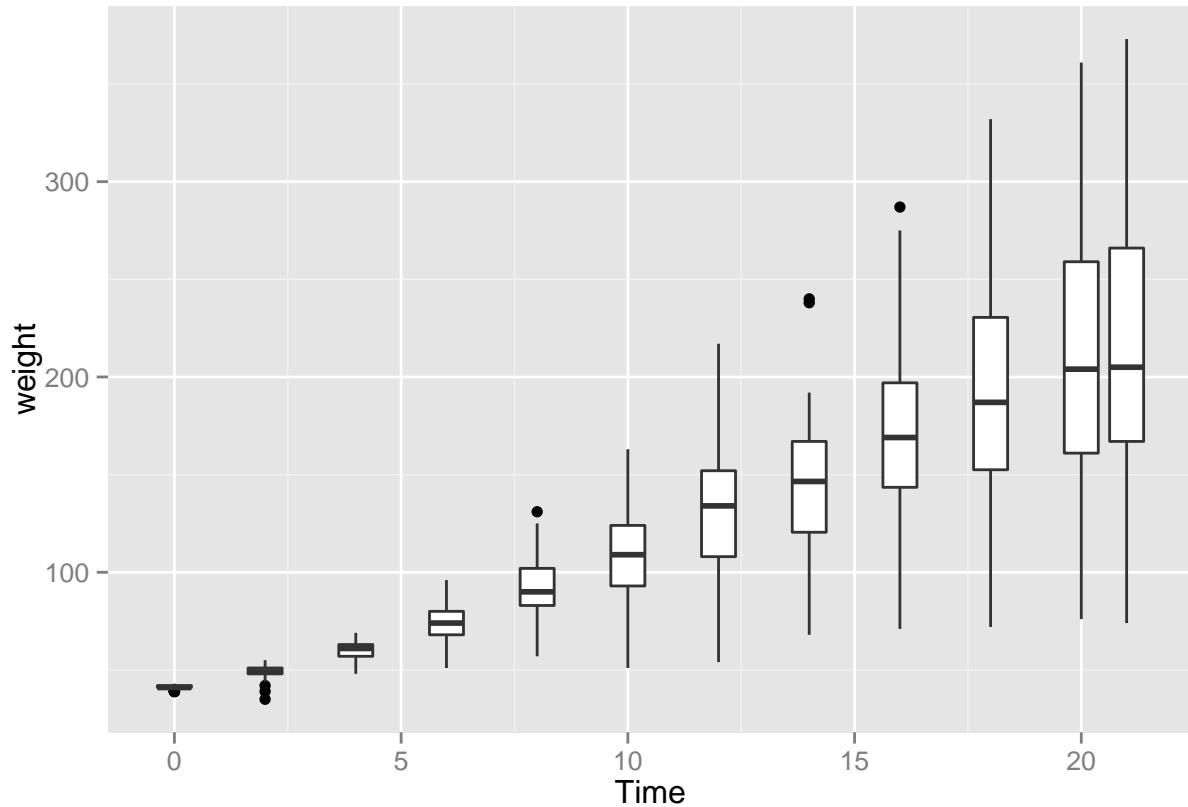
sp1 + geom_point()
```



```
# Could also use geom_jitter(), which is equivalent  
sp1 + geom_point( position = position_jitter( width = .5, height = 0))
```



```
##  
sp1 + geom_boxplot( aes( group = Time))
```



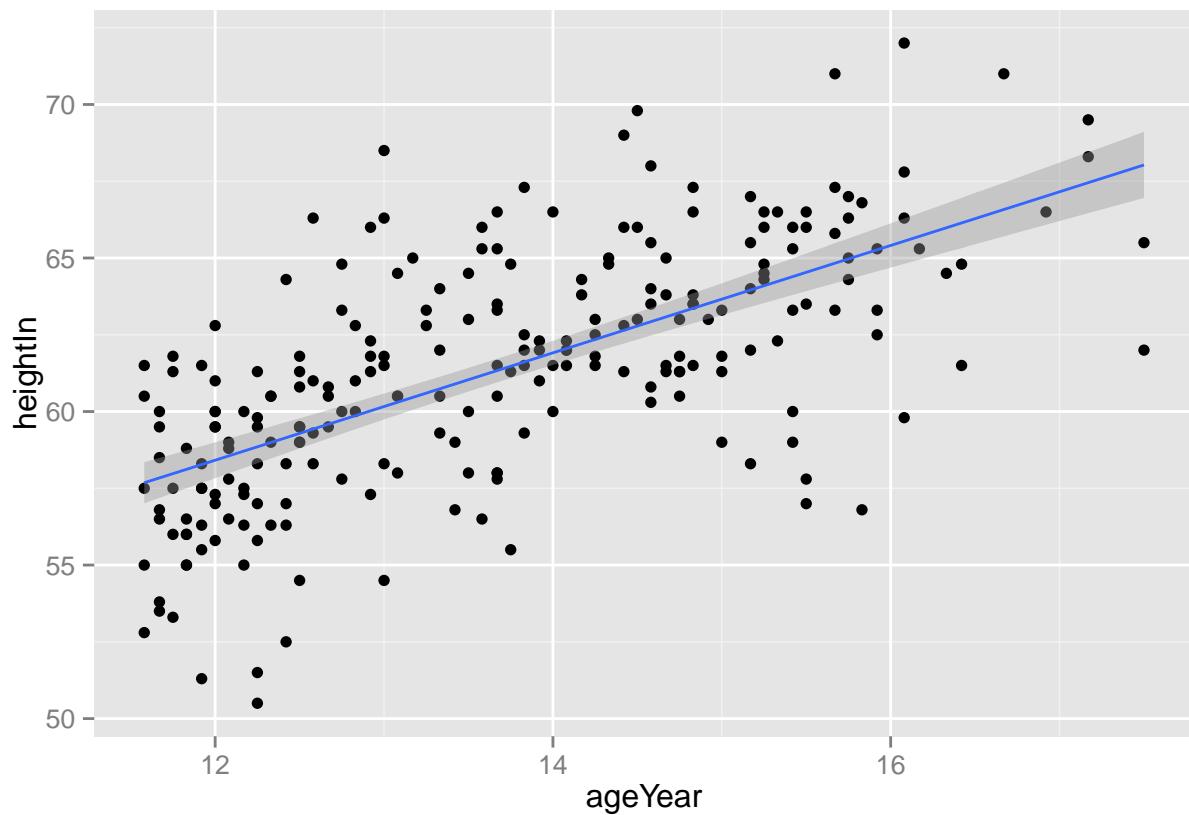
## 6. Adding Fitted Regression Model Lines

```

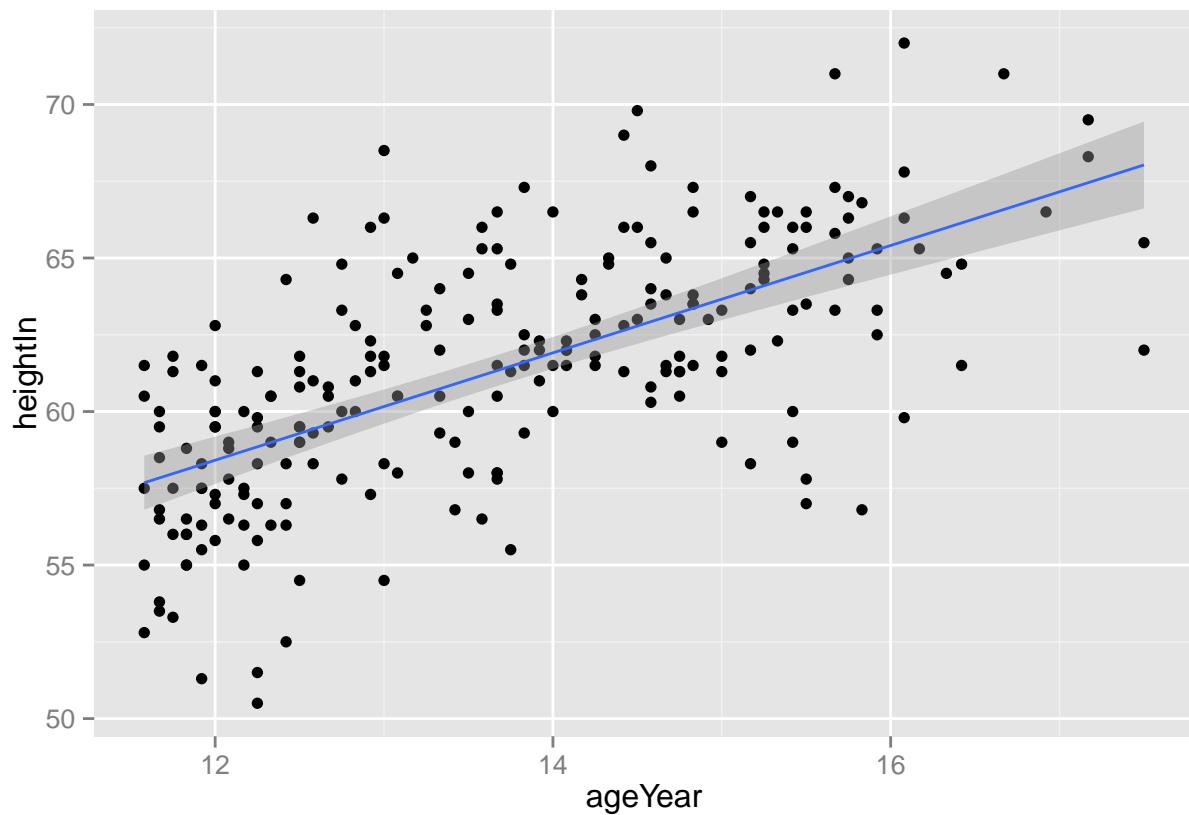
# The base plot
sp <- ggplot( heightweight, aes( x = ageYear, y = heightIn))

# By default, stat_smooth() also adds a 95% confidence region for the regression fit
sp +
  geom_point() +
  stat_smooth( method = lm)

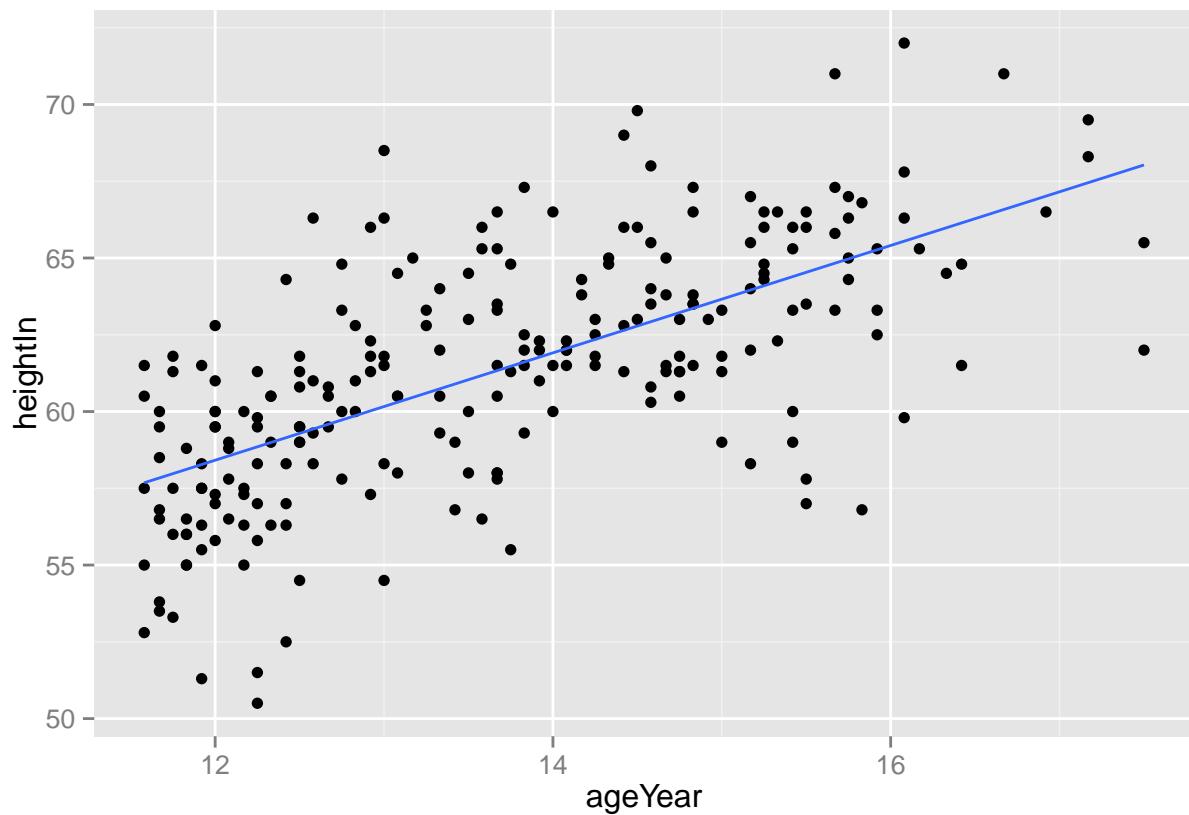
```



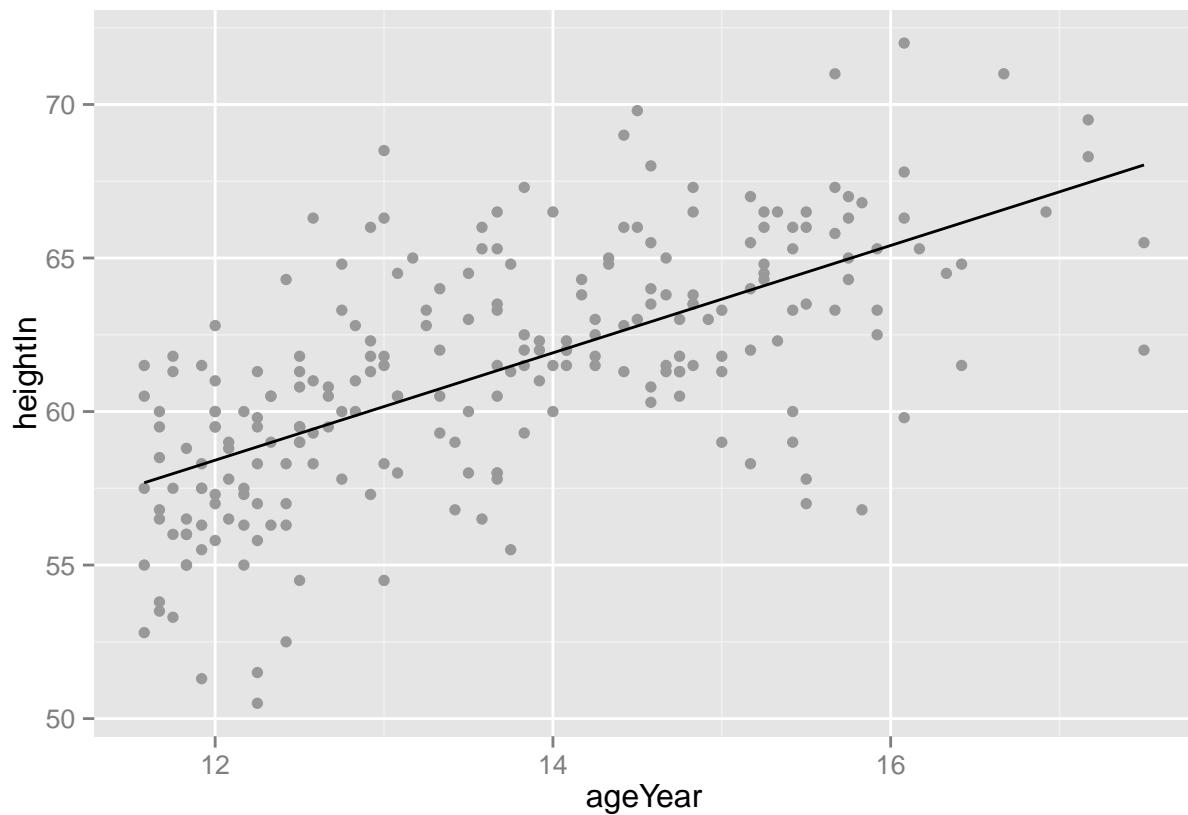
```
# 99% confidence region
sp +
  geom_point() +
  stat_smooth( method = lm, level = 0.99)
```



```
# No confidence region  
sp +  
  geom_point() +  
  stat_smooth( method = lm, se = FALSE)
```

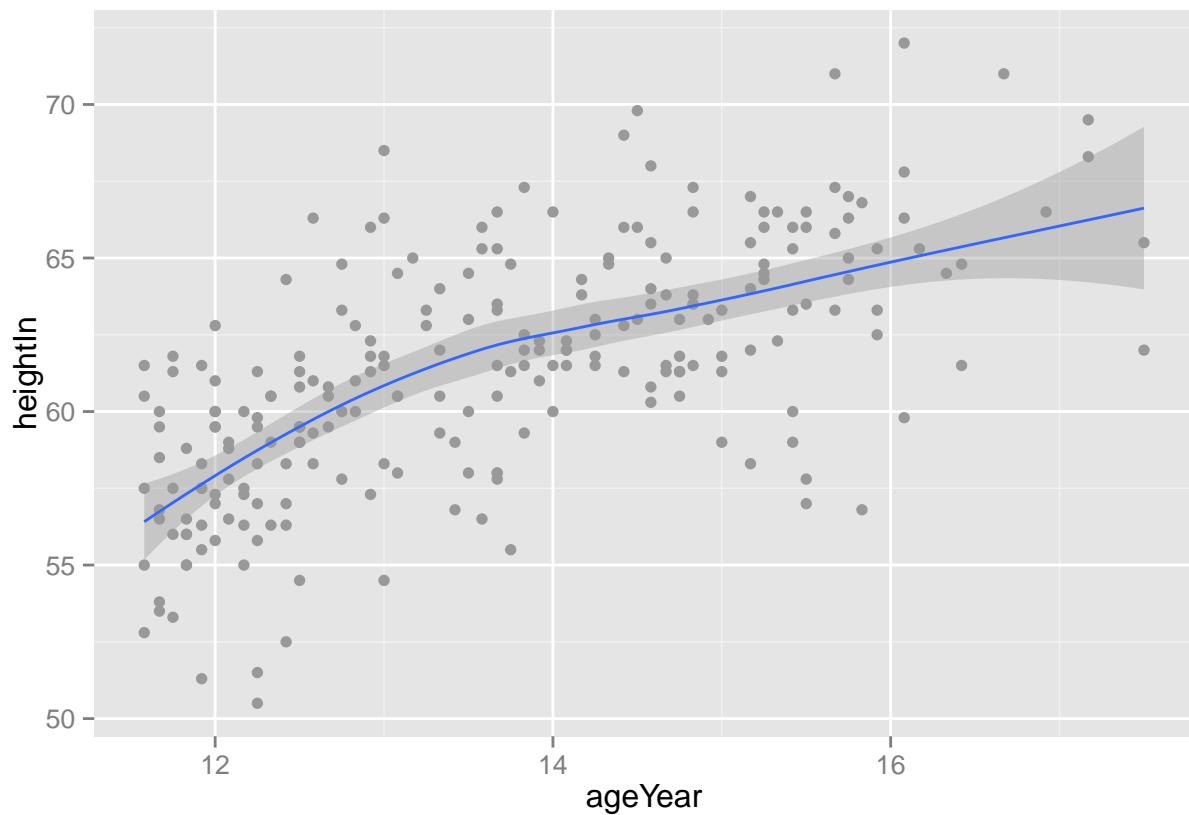


```
# The default color of the fit line is blue. This can be change by setting colour.  
sp +  
  geom_point( colour ="grey60") +  
  stat_smooth( method = lm, se = FALSE, colour ="black")
```

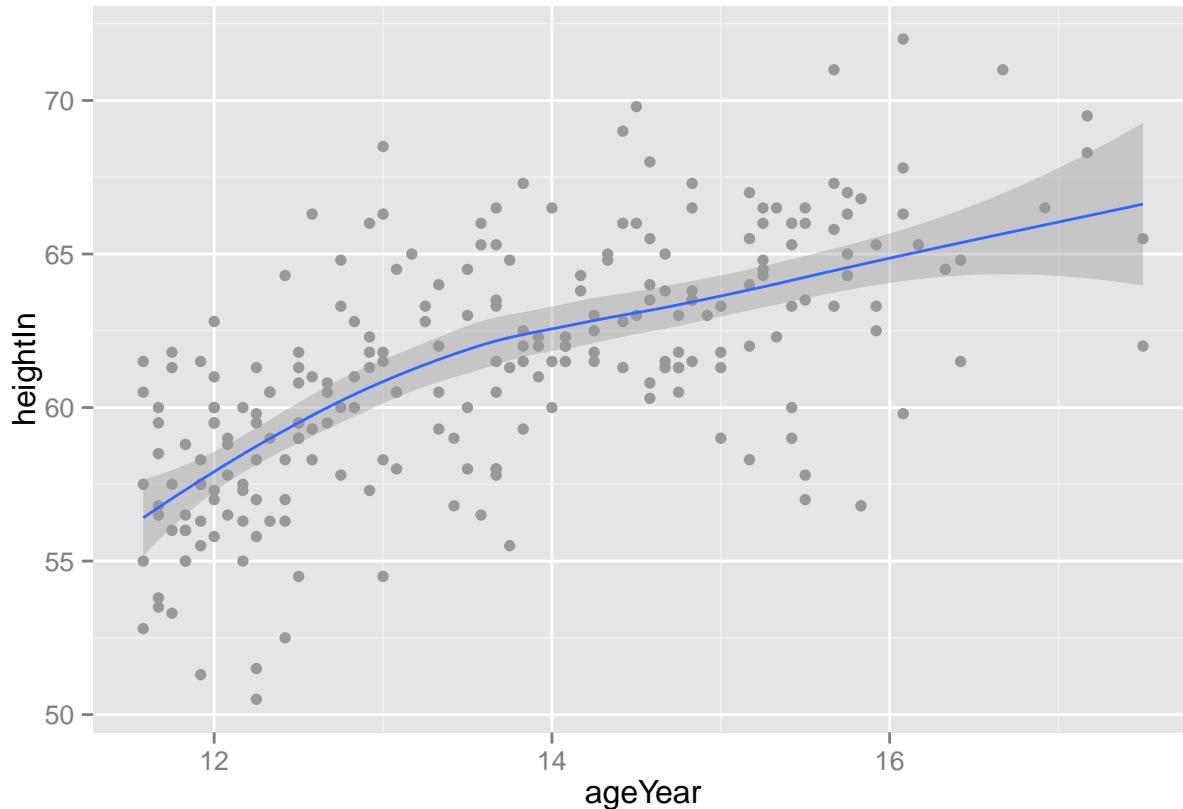


```
# If you add stat_smooth() without specifying the method, it will use a loess (locally weighted polynomial)
sp +
  geom_point( colour ="grey60") +
  stat_smooth()
```

```
## geom_smooth: method="auto" and size of largest group is <1000, so using loess. Use 'method = x' to c
```



```
sp +
  geom_point( colour ="grey60") +
  stat_smooth( method = loess)
```



Another common type of model fit is a logistic regression.

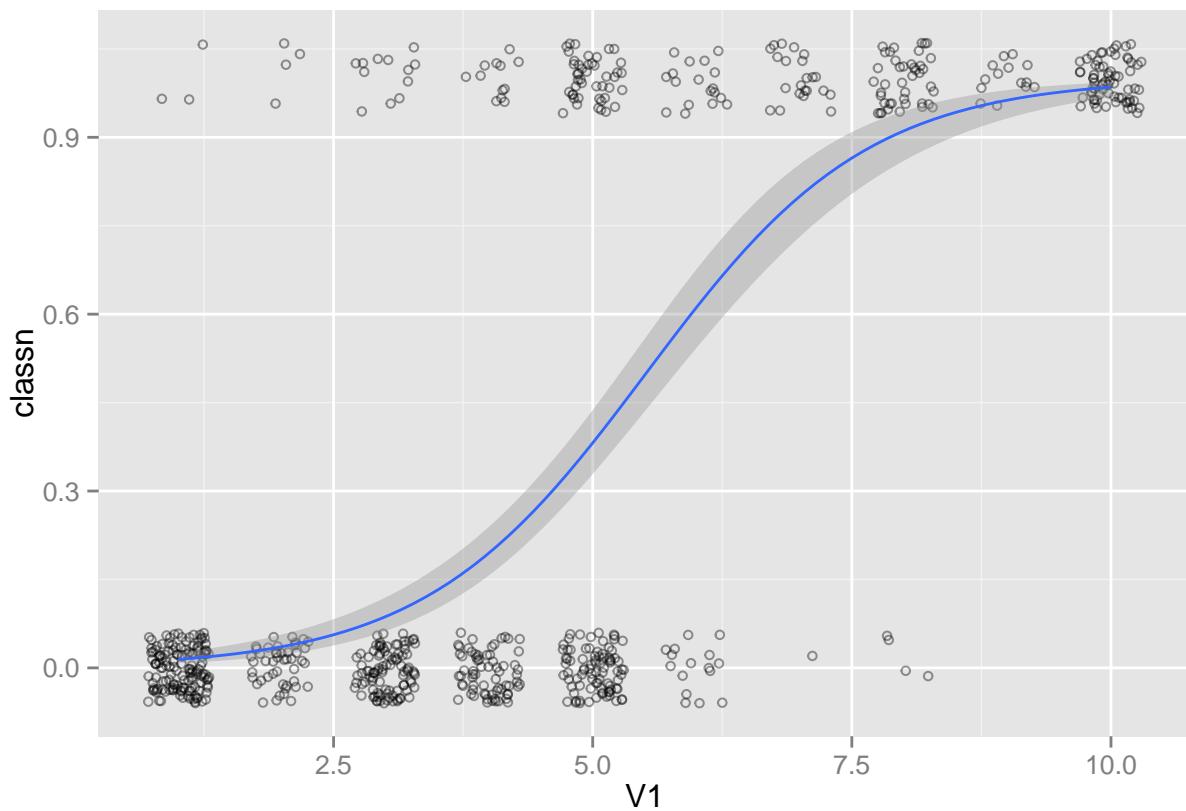
```
library( MASS ) # For the data set
b <- biopsy
b$classn[ b$class == "benign" ] <- 0
b$classn[ b$class == "malignant" ] <- 1
head(b)
```

##	ID	V1	V2	V3	V4	V5	V6	V7	V8	V9	class	classn
## 1	1000025	5	1	1	1	2	1	3	1	1	benign	0
## 2	1002945	5	4	4	5	7	10	3	2	1	benign	0
## 3	1015425	3	1	1	1	2	2	3	1	1	benign	0
## 4	1016277	6	8	8	1	3	4	3	7	1	benign	0
## 5	1017023	4	1	1	3	2	1	3	1	1	benign	0
## 6	1017122	8	10	10	8	7	10	9	7	1	malignant	1

# we'll just look at the relationship of V1 (clump thickness) and the class of the tumor.

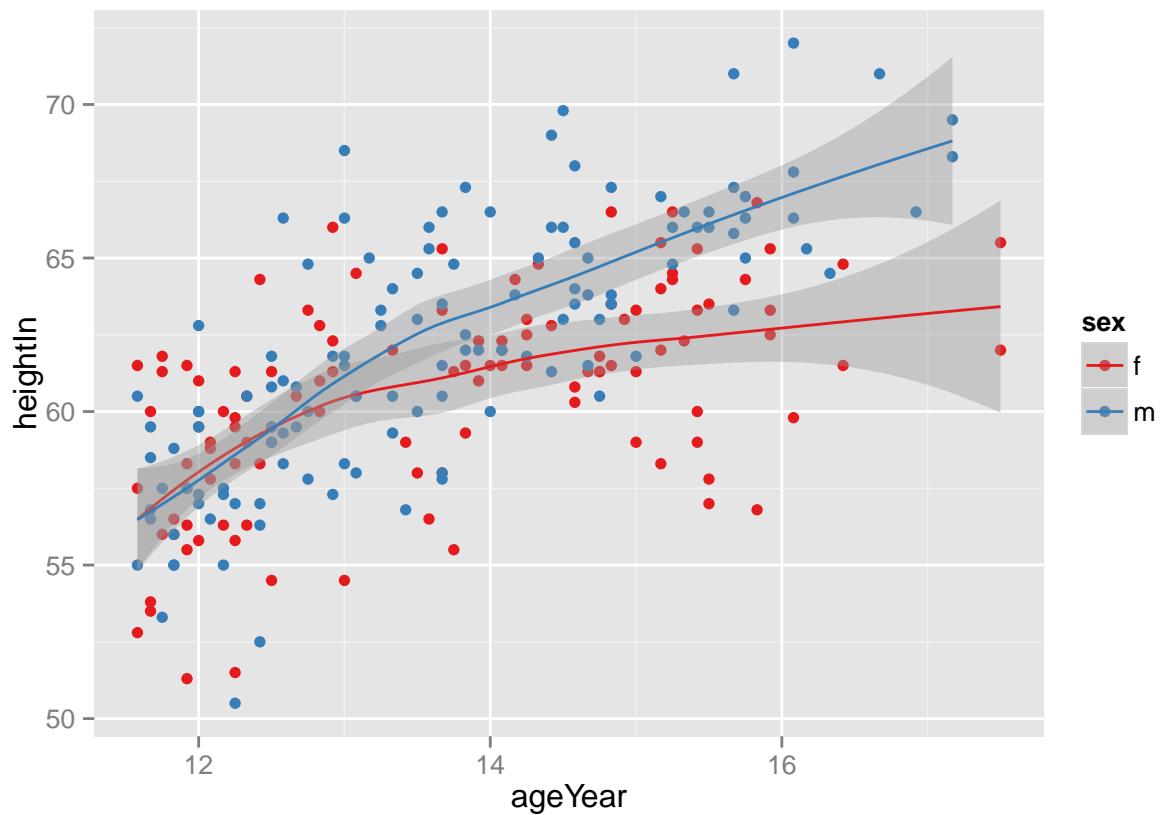
```
ggplot( b, aes( x = V1, y = classn)) +
```

```
  geom_point( position = position_jitter( width = 0.3, height = 0.06), alpha = 0.4, shape = 21, size = 1 )
```

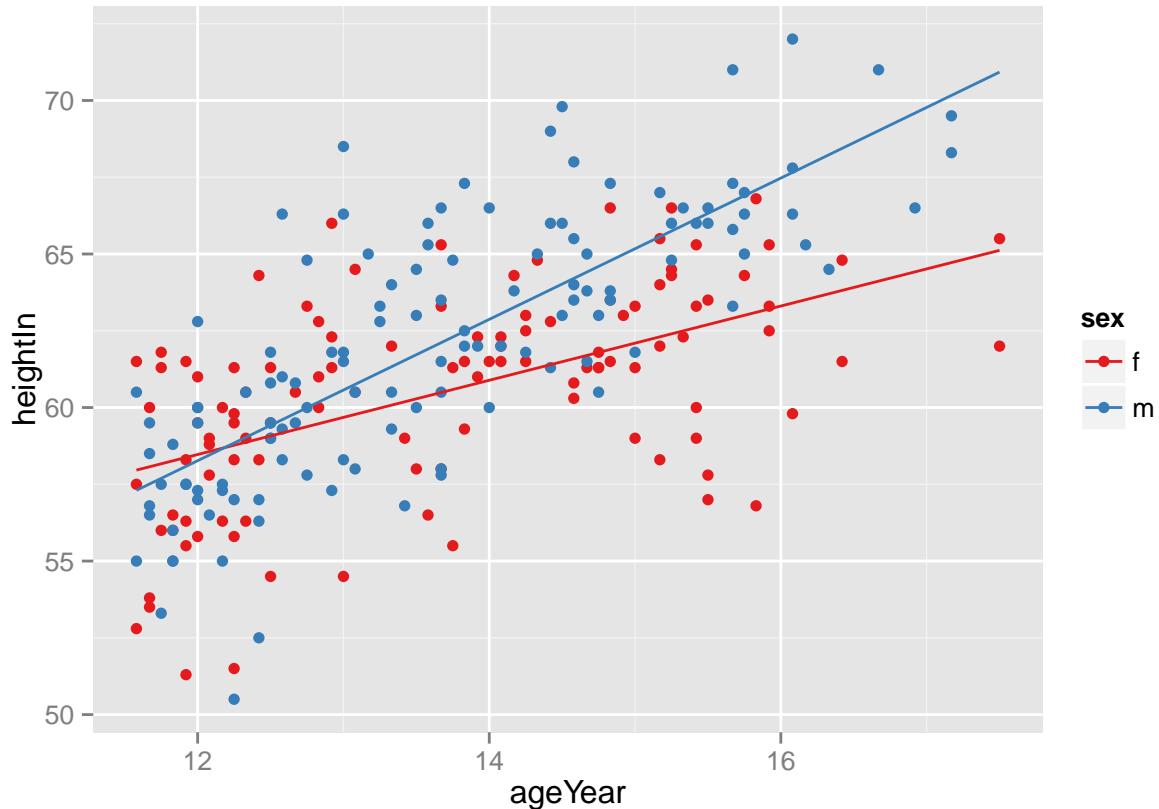


```
# If your scatter plot has points grouped by a factor, using colour or shape, one fit line will be drawn
sps <- ggplot( heightweight, aes( x = ageYear, y = heightIn, colour = sex)) +
  geom_point() +
  scale_colour_brewer( palette = "Set1")
sps + geom_smooth()
```

```
## geom_smooth: method="auto" and size of largest group is <1000, so using loess. Use 'method = x' to ch
```



```
# If you want the lines to extrapolate from the data  
sps +  
  geom_smooth( method = lm, se = FALSE, fullrange = TRUE)
```



## 7. If you want the lines to extrapolate from the data

You have already created a fitted regression model object for a data set, and you want to plot the lines for that model.

```
model <- lm( heightIn ~ ageYear + I(ageYear^2), heightweight)
model
```

```
##
## Call:
## lm(formula = heightIn ~ ageYear + I(ageYear^2), data = heightweight)
##
## Coefficients:
##   (Intercept)      ageYear    I(ageYear^2)
##     -10.3136       8.6673      -0.2478
```

```
# Create a data frame with ageYear column, interpolating across range
xmin <- min( heightweight$ageYear)
xmax <- max( heightweight$ageYear)
predicted <- data.frame( ageYear = seq( xmin, xmax, length.out = 100))
predicted$heightIn <- predict( model, predicted)
head(predicted)
```

```
##      ageYear heightIn
## 1 11.58000 56.82624
## 2 11.63980 57.00047
```

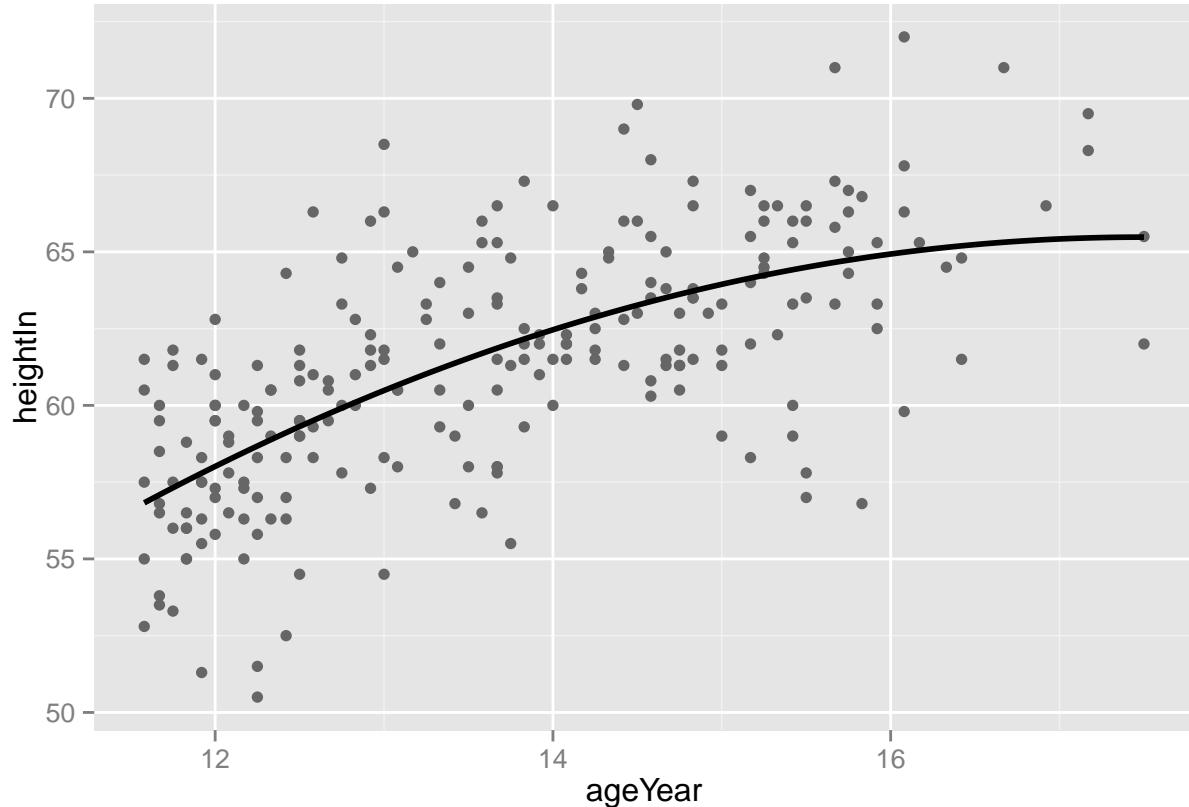
```

## 3 11.69960 57.17294
## 4 11.75939 57.34363
## 5 11.81919 57.51255
## 6 11.87899 57.67969

# We can now plot the data points along with the values predicted from the model
sp <- ggplot( heightweight, aes( x = ageYear, y = heightIn)) +
  geom_point( colour ="grey40")

sp + geom_line( data = predicted, size = 1)

```



```

# Given a model, predict values of yvar from xvar
# This supports one predictor and one predicted variable
# xrange: If NULL, determine the x range from the model object. If a vector with
# two numbers, use those as the min and max of the prediction range.
# samples: Number of samples across the x range.
# ....: Further arguments to be passed to predict()
predictvals <- function( model, xvar, yvar, xrange = NULL, samples = 100, ...) {
  # If xrange isn't passed in, determine xrange from the models.
  # Different ways of extracting the x range, depending on model type
  if (is.null(xrange)) {
    if (any( class( model ) %in% c("lm", "glm")))
      xrange <- range( model $ model[[ xvar ]])
    else if (any( class( model ) %in% "loess")) xrange <- range( model $ x)
  }
  newdata <- data.frame( x = seq( xrange[1], xrange[2], length.out = samples))
  names( newdata) <- xvar

```

```

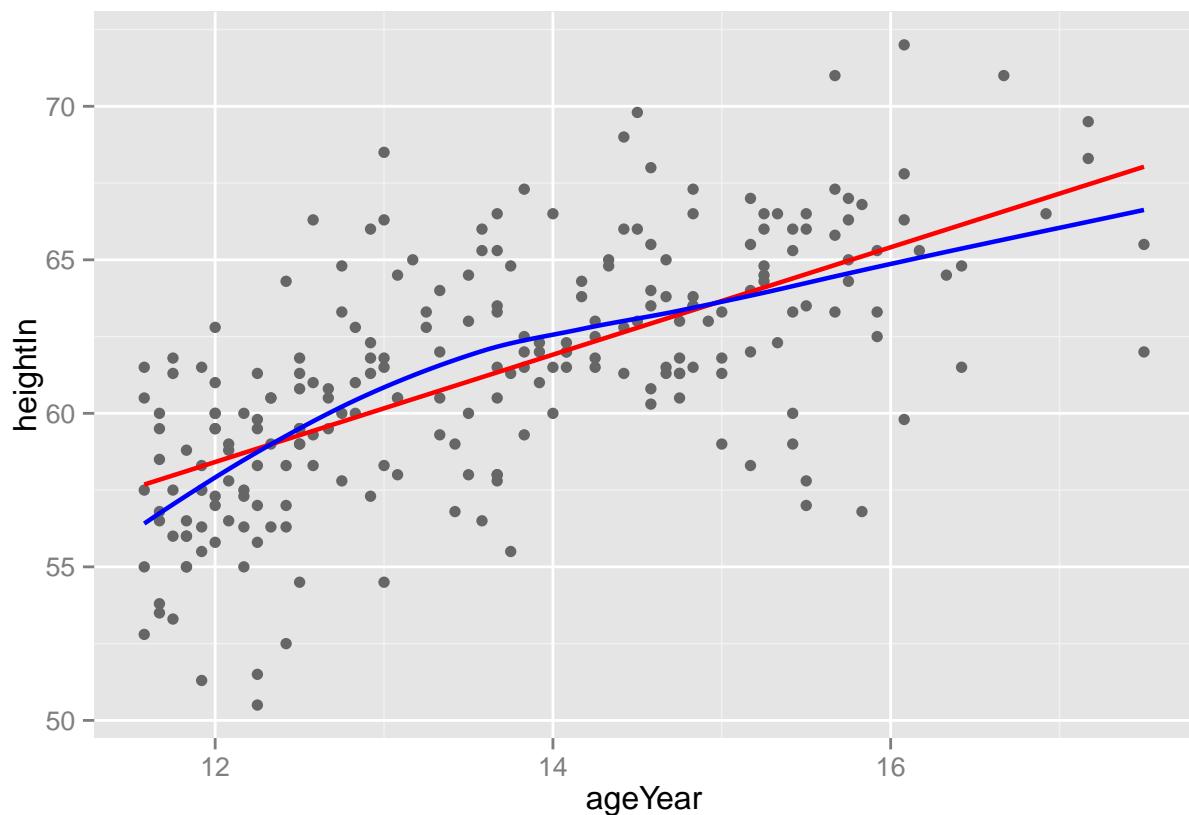
newdata[[ yvar]] <- predict( model, newdata = newdata, ...)
newdata
}

modlinear <- lm( heightIn ~ ageYear, heightweight)
modloess <- loess( heightIn ~ ageYear, heightweight)

lm_predicted <- predictvals( modlinear, "ageYear", "heightIn")
loess_predicted <- predictvals( modloess, "ageYear", "heightIn")

sp +
  geom_line( data = lm_predicted, colour ="red", size =.8) +
  geom_line( data = loess_predicted, colour ="blue", size =.8)

```



For glm models that use a nonlinear link function, you need to specify type = "response"

```

library( MASS ) # For the data set
b <- biopsy
b$classn[ b$class =="benign"] <- 0
b$classn[ b$class =="malignant"] <- 1
head(b)

```

##	ID	V1	V2	V3	V4	V5	V6	V7	V8	V9	class	classn
## 1	1000025	5	1	1	1	2	1	3	1	1	benign	0
## 2	1002945	5	4	4	5	7	10	3	2	1	benign	0
## 3	1015425	3	1	1	1	2	2	3	1	1	benign	0
## 4	1016277	6	8	8	1	3	4	3	7	1	benign	0

```

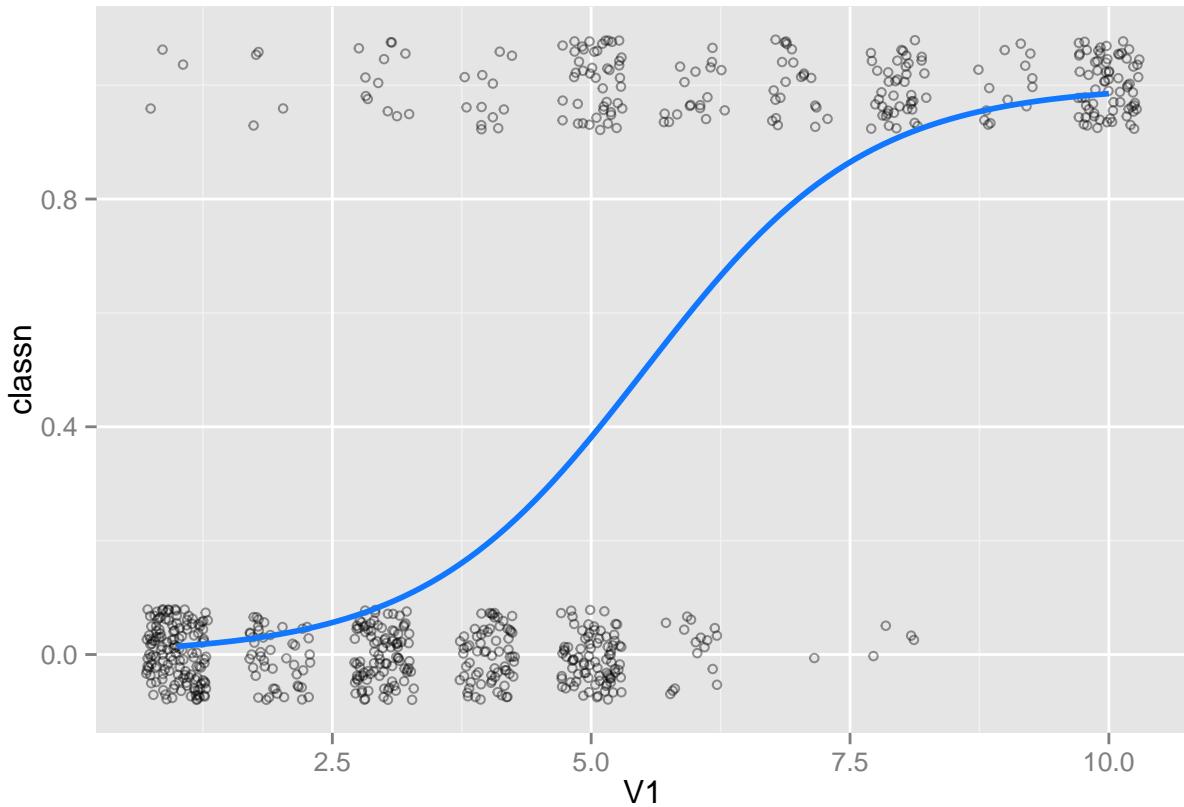
## 5 1017023 4 1 1 3 2 1 3 1 1 benign 0
## 6 1017122 8 10 10 8 7 10 9 7 1 malignant 1

# glm model
fitlogistic <- glm( classn ~ V1, b, family = binomial)

# Get predicted values
glm_predicted <- predictvals( fitlogistic, "V1", "classn", type ="response")

ggplot( b, aes( x = V1, y = classn)) +
  geom_point( position = position_jitter( width =.3, height =.08), alpha = 0.4, shape = 21, size = 1.5)
  geom_line( data = glm_predicted, colour ="#1177FF", size = 1)

```



## 8. Adding Fitted Lines from Multiple Existing Models

With the heightweight data set, we'll make a linear model with lm() for each of the levels of sex, and put those model objects in a list. The model building is done with a function, make\_model()

```

make_model <- function( data) {
  lm( heightIn ~ ageYear, data)
}
models <- dlply( heightweight, "sex", .fun = make_model) # Print out the list of two lm objects, f and m
models

## $f
##
```

```

## Call:
## lm(formula = heightIn ~ ageYear, data = data)
##
## Coefficients:
## (Intercept)      ageYear
##        43.963       1.209
##
##
## $m
##
## Call:
## lm(formula = heightIn ~ ageYear, data = data)
##
## Coefficients:
## (Intercept)      ageYear
##        30.658       2.301
##
##
## attr(,"split_type")
## [1] "data.frame"
## attr(,"split_labels")
##   sex
## 1   f
## 2   m

```

*# Now that we have the list of model objects, we can run predictvals() to get predicted values from each model.*

```

predvals <- ldply( models, .fun = predictvals, xvar ="ageYear", yvar ="heightIn")
head(predvals)

```

```

##   sex  ageYear heightIn
## 1   f 11.58000 57.96250
## 2   f 11.63980 58.03478
## 3   f 11.69960 58.10707
## 4   f 11.75939 58.17936
## 5   f 11.81919 58.25165
## 6   f 11.87899 58.32394

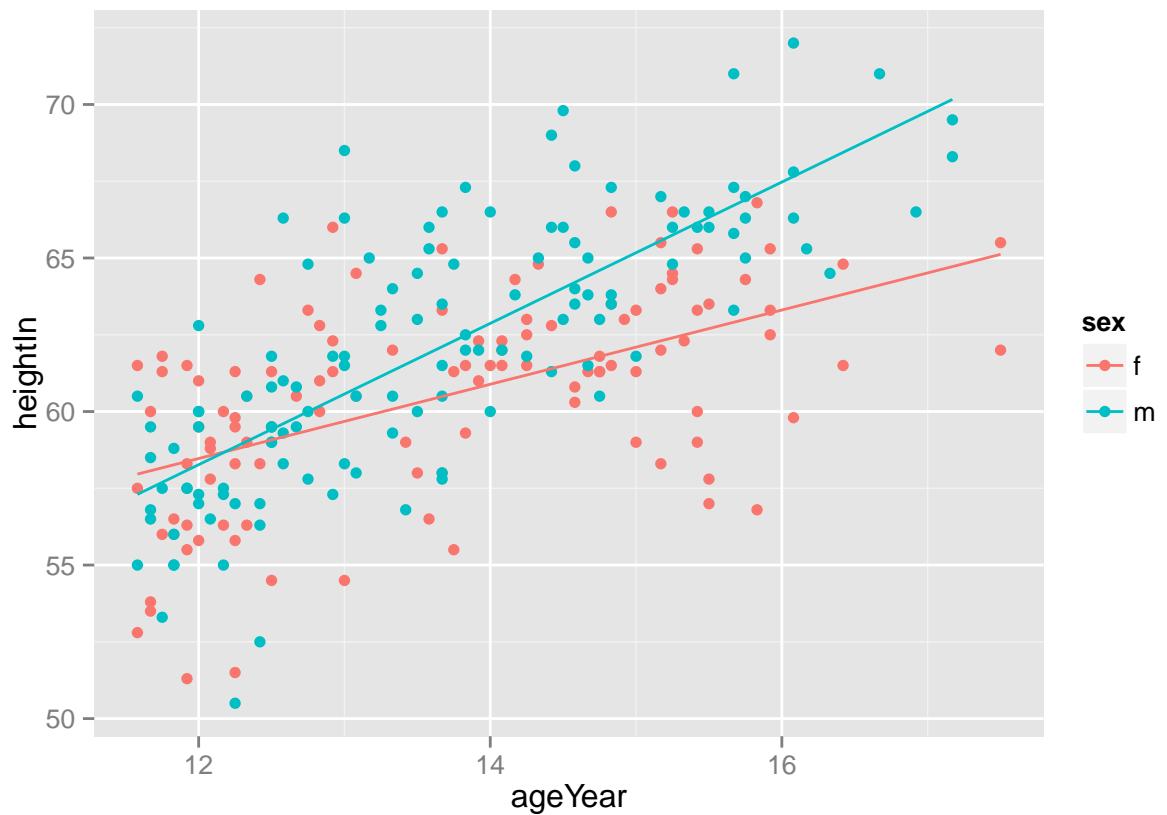
```

*# Finally, we can plot the data with the predicted values*

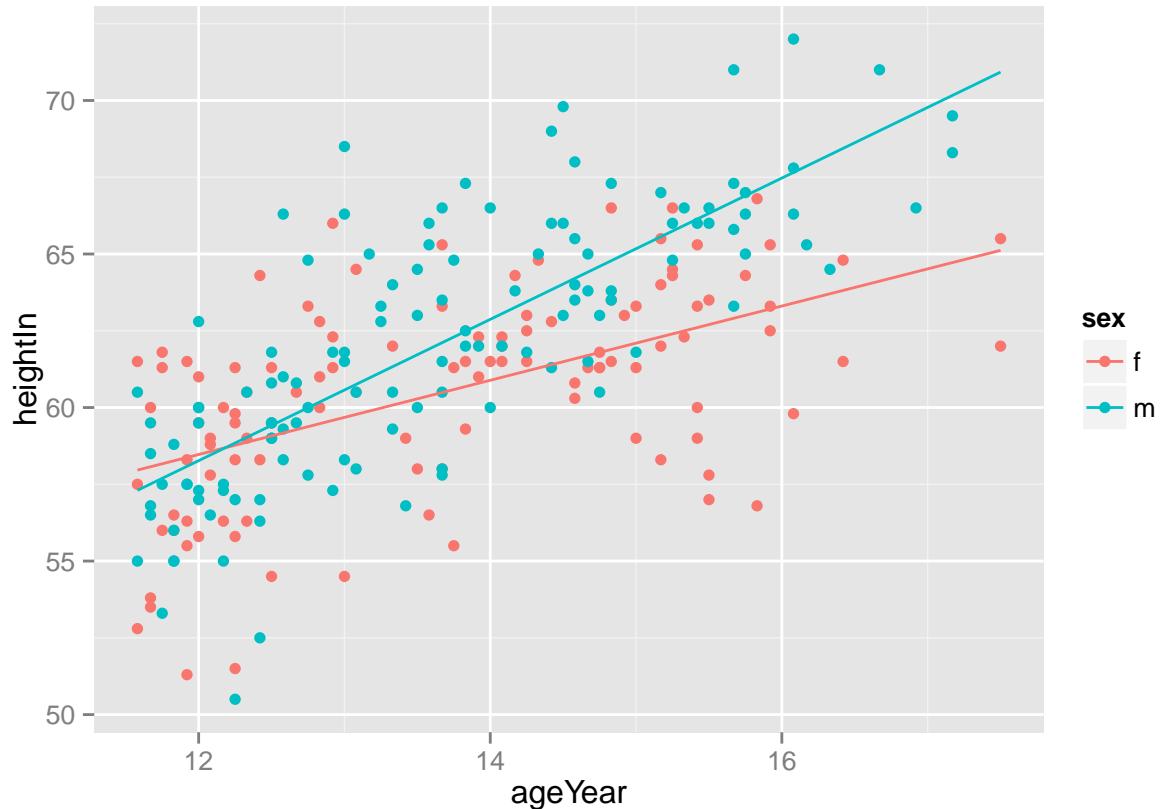
```

ggplot( heightweight, aes( x = ageYear, y = heightIn, colour = sex)) +
  geom_point() +
  geom_line( data = predvals)

```



```
predvals <- ldply( models, .fun = predictvals, xvar ="ageYear", yvar ="heightIn",
                     xrange = range( heightweight$ageYear))
ggplot( heightweight, aes( x = ageYear, y = heightIn, colour = sex)) +
  geom_point() + geom_line( data = predvals)
```



## 9. Adding Annotations with Model Coefficients

```
model <- lm( heightIn ~ ageYear, heightweight)
summary( model)

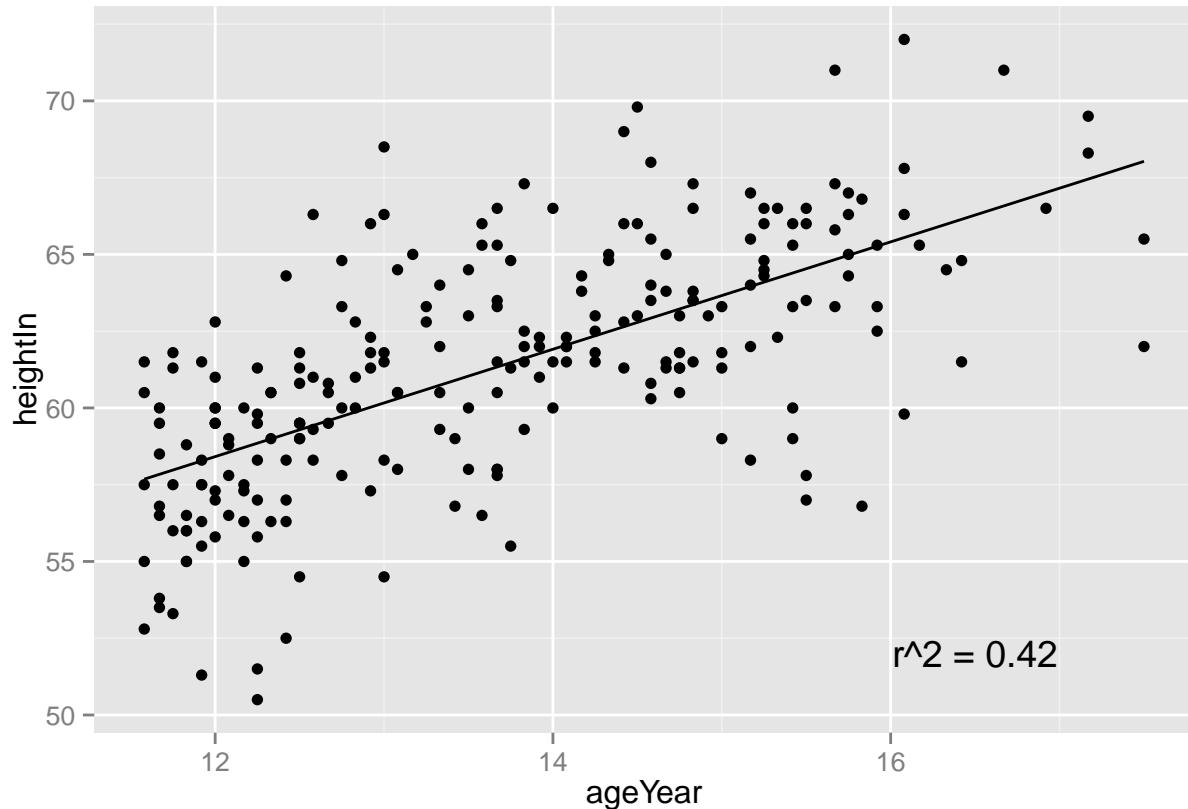
##
## Call:
## lm(formula = heightIn ~ ageYear, data = heightweight)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -8.3517 -1.9006  0.1378  1.9071  8.3371 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 37.4356    1.8281  20.48   <2e-16 ***
## ageYear      1.7483    0.1329  13.15   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.989 on 234 degrees of freedom
## Multiple R-squared:  0.4249, Adjusted R-squared:  0.4225 
## F-statistic: 172.9 on 1 and 234 DF,  p-value: < 2.2e-16
```

```

# First generate prediction data
pred <- predictvals( model, "ageYear", "heightIn")
sp <- ggplot( heightweight, aes( x = ageYear, y = heightIn)) +
  geom_point() + geom_line( data = pred)

##
sp +
  annotate("text", label ="r^2 = 0.42", x = 16.5, y = 52)

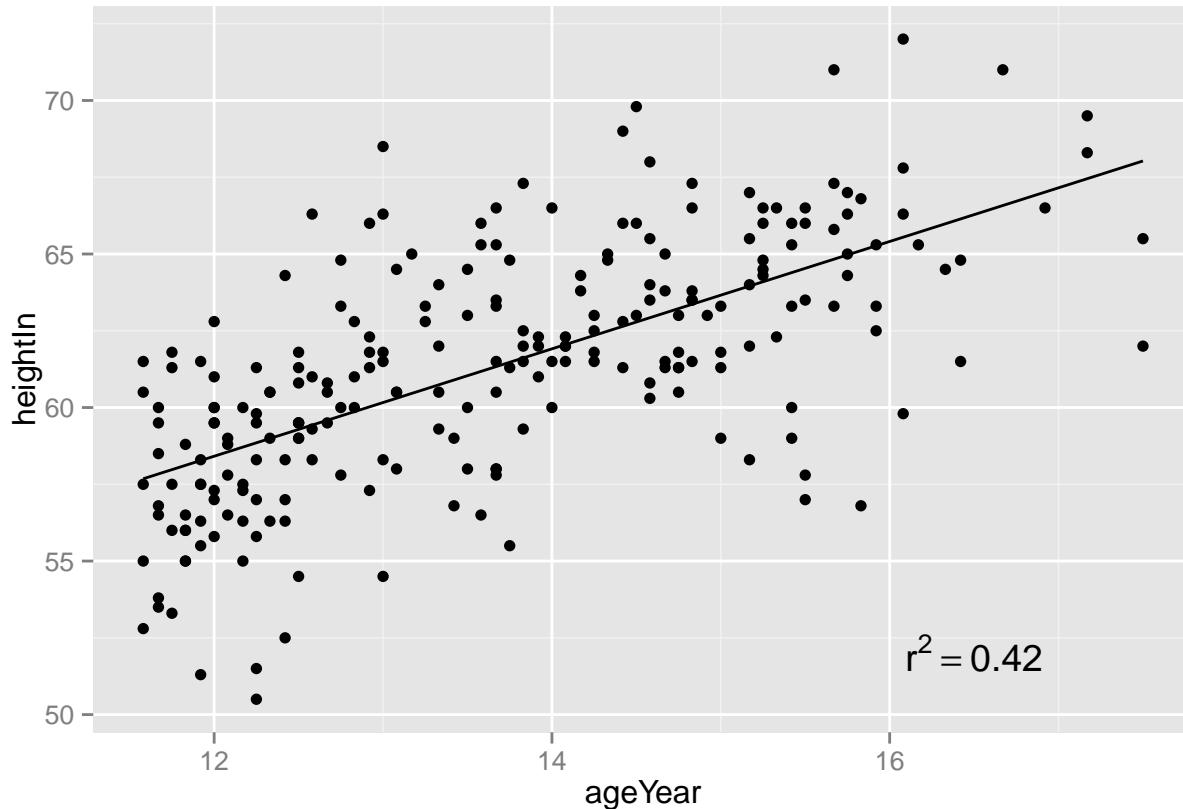
```



```

## 
sp +
  annotate("text", label ="r^2 == 0.42", parse = TRUE, x = 16.5, y = 52)

```



```

## It's possible to automatically extract values from the model object and build an expression
eqn <- as.character(
  as.expression( substitute( italic(y) == a + b * italic(x) * ",," ~ italic(r)^2 ~ "=" ~ r2,
    list( a = format( coef( model)[1], digits = 3),
         b = format( coef( model)[2], digits = 3),
         r2 = format( summary(model)$r.squared,
                      digits = 2) ))))

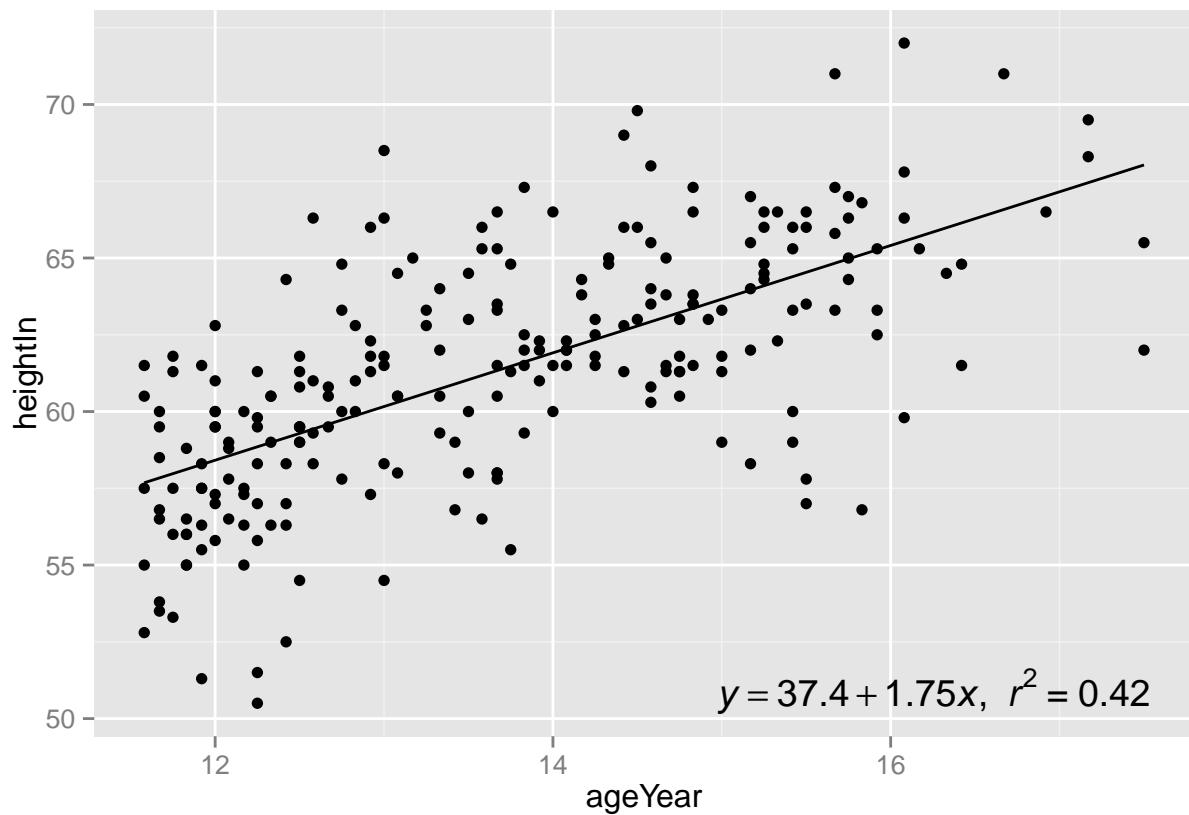
## [1] "italic(y) == \"37.4\" + \"1.75\" * italic(x) * \",,\" ~ ~italic(r)^2 ~ \"=\" ~ \"0.42\""

# Parsing turns it into an expression
parse( text = eqn)

## expression(italic(y) == "37.4" + "1.75" * italic(x) * ",," ~ ~italic(r)^2 ~
##           "=" ~ "0.42")

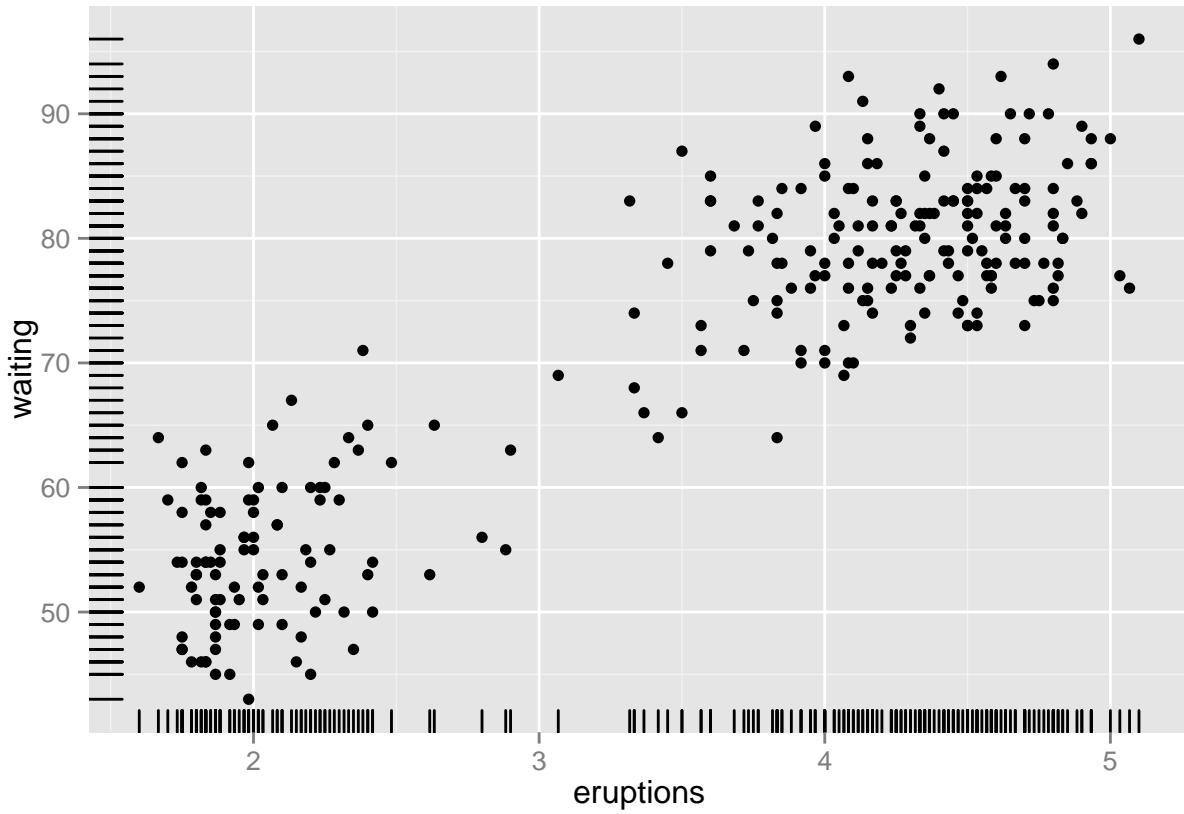
# Now that we have the expression string, we can add it to the plot.
sp + annotate("text", label = eqn, parse = TRUE, x = Inf, y = -Inf, hjust = 1.1, vjust = -.5)

```

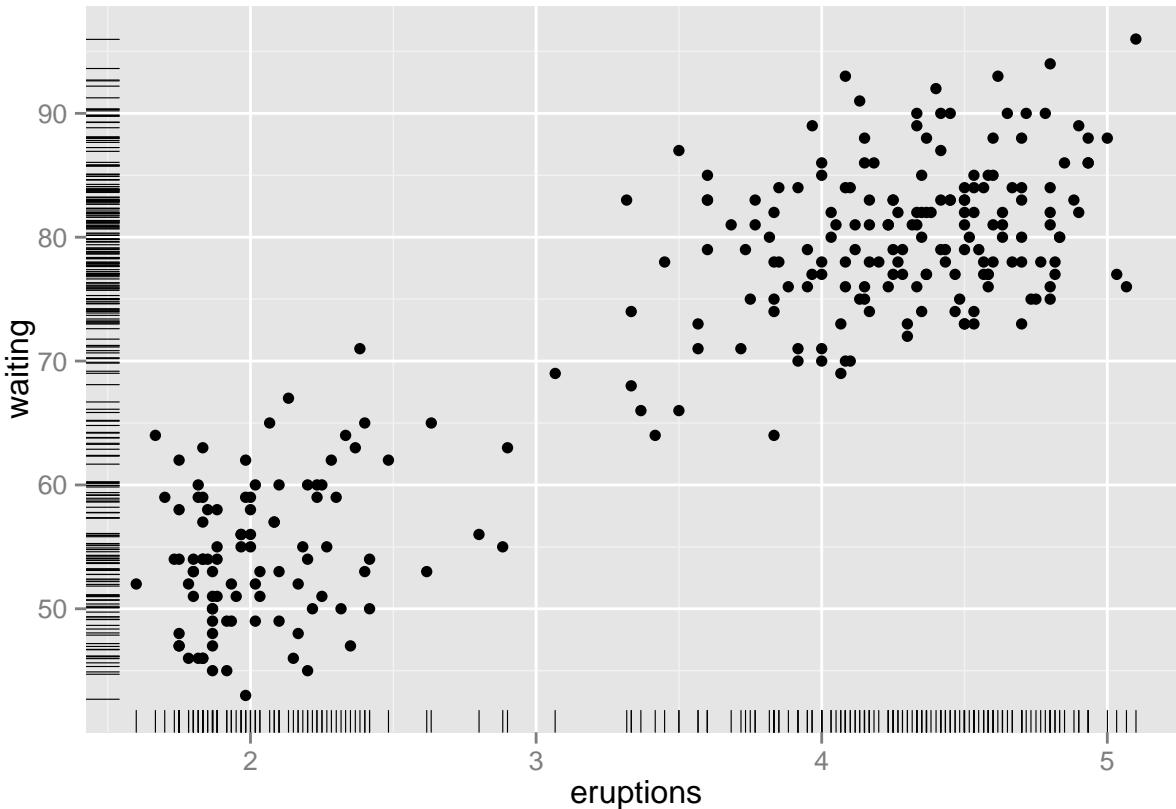


## 10. Adding Marginal Rugs to a Scatter Plot

```
ggplot( faithful, aes( x = eruptions, y = waiting)) +  
  geom_point() +  
  geom_rug()
```



```
# To reduce the overplotting, we can jitter the line positions and make them slightly thinner by specifying
ggplot( faithful, aes( x = eruptions, y = waiting)) +
  geom_point() +
  geom_rug( position ="jitter", size =.2)
```



## 11. Labeling Points in a Scatter Plot

```
# we'll just take the subset of countries that spent more than $ 2000 USD per capita:
subset(countries, Year == 2009 & healthexp > 2000)
```

	Name	Code	Year	GDP	laborrate	healthexp	infmortality
## 254	Andorra	AND	2009	NA	NA	3089.636	3.1
## 560	Australia	AUS	2009	42130.82	65.2	3867.429	4.2
## 611	Austria	AUT	2009	45555.43	60.4	5037.311	3.6
## 968	Belgium	BEL	2009	43640.20	53.5	5104.019	3.6
## 1733	Canada	CAN	2009	39599.04	67.8	4379.761	5.2
## 2702	Denmark	DNK	2009	55933.35	65.4	6272.729	3.4
## 3365	Finland	FIN	2009	44576.73	60.9	4309.604	2.5
## 3416	France	FRA	2009	40663.05	56.1	4797.966	3.5
## 3671	Germany	DEU	2009	40658.58	59.8	4628.769	3.5
## 3824	Greece	GRC	2009	28936.48	53.7	3040.734	3.5
## 4436	Iceland	ISL	2009	37972.24	77.5	3130.391	1.7
## 4691	Ireland	IRL	2009	49737.93	63.6	4951.845	3.4
## 4844	Italy	ITA	2009	35073.32	49.1	3327.630	3.2
## 4946	Japan	JPN	2009	39456.44	59.5	3321.466	2.4
## 5864	Luxembourg	LUX	2009	106252.24	55.5	8182.855	2.2
## 6680	Monaco	MCO	2009	172676.34	NA	7137.390	3.4
## 7088	Netherlands	NLD	2009	48068.35	66.1	5163.740	3.8
## 7190	New Zealand	NZL	2009	29352.45	68.6	2633.625	4.9
## 7445	Norway	NOR	2009	78408.71	66.9	7661.610	2.9

```

## 7955      Portugal PRT 2009  22029.87    62.5  2409.661   3.2
## 8312     San Marino SMR 2009       NA      NA 4089.271   1.9
## 8822      Slovenia SVN 2009  24101.26    58.9  2174.718   2.5
## 9077      Spain   ESP 2009  31891.39    58.6  3075.013   4.0
## 9536      Sweden  SWE 2009  43406.17    64.8  4251.976   2.4
## 9587 Switzerland CHE 2009  63524.65    66.9  7140.729   4.1
## 10454 United Kingdom GBR 2009  35163.41    62.2  3285.050   4.7
## 10505 United States USA 2009  45744.56    65.0  7410.163   6.6

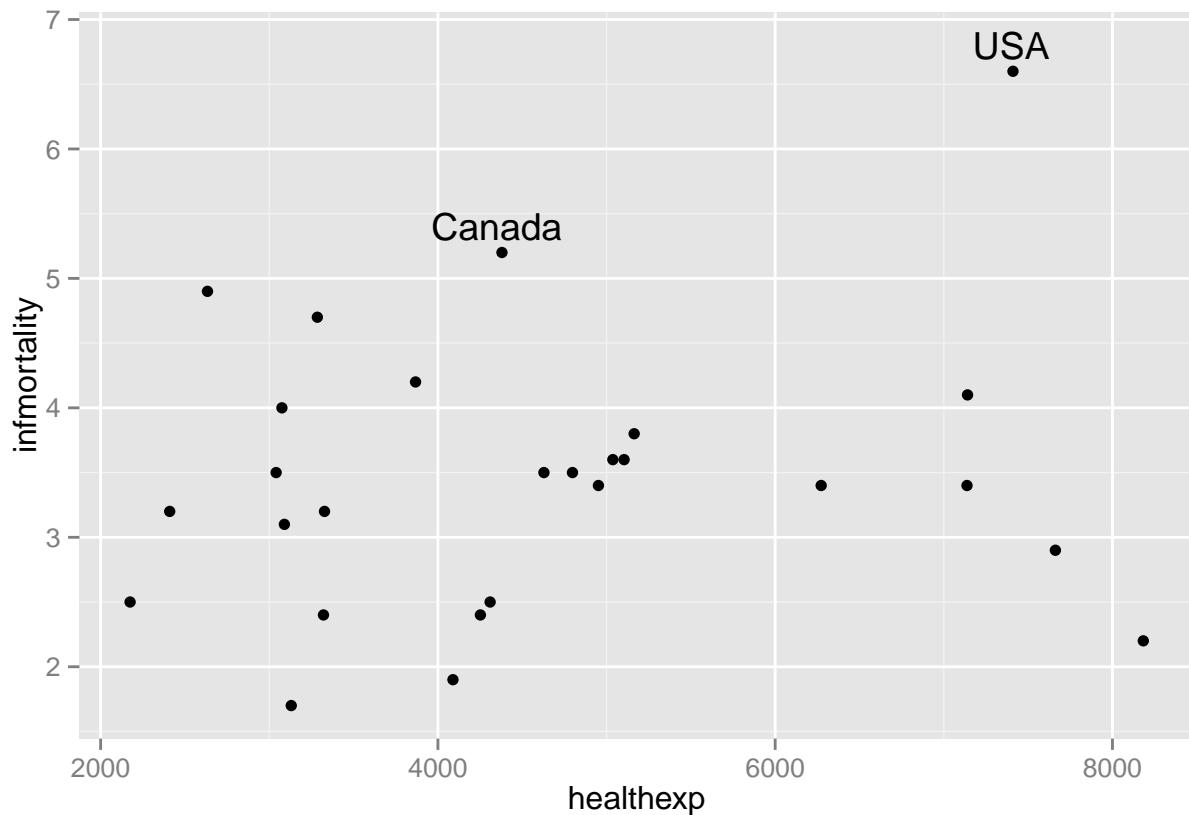
```

```

# To manually add annotations, use annotate(),
sp <- ggplot( subset( countries, Year == 2009 & healthexp > 2000), aes( x = healthexp, y = infmortality)
  geom_point()

sp +
  annotate("text", x = 4350, y = 5.4, label ="Canada") +
  annotate("text", x = 7400, y = 6.8, label ="USA")

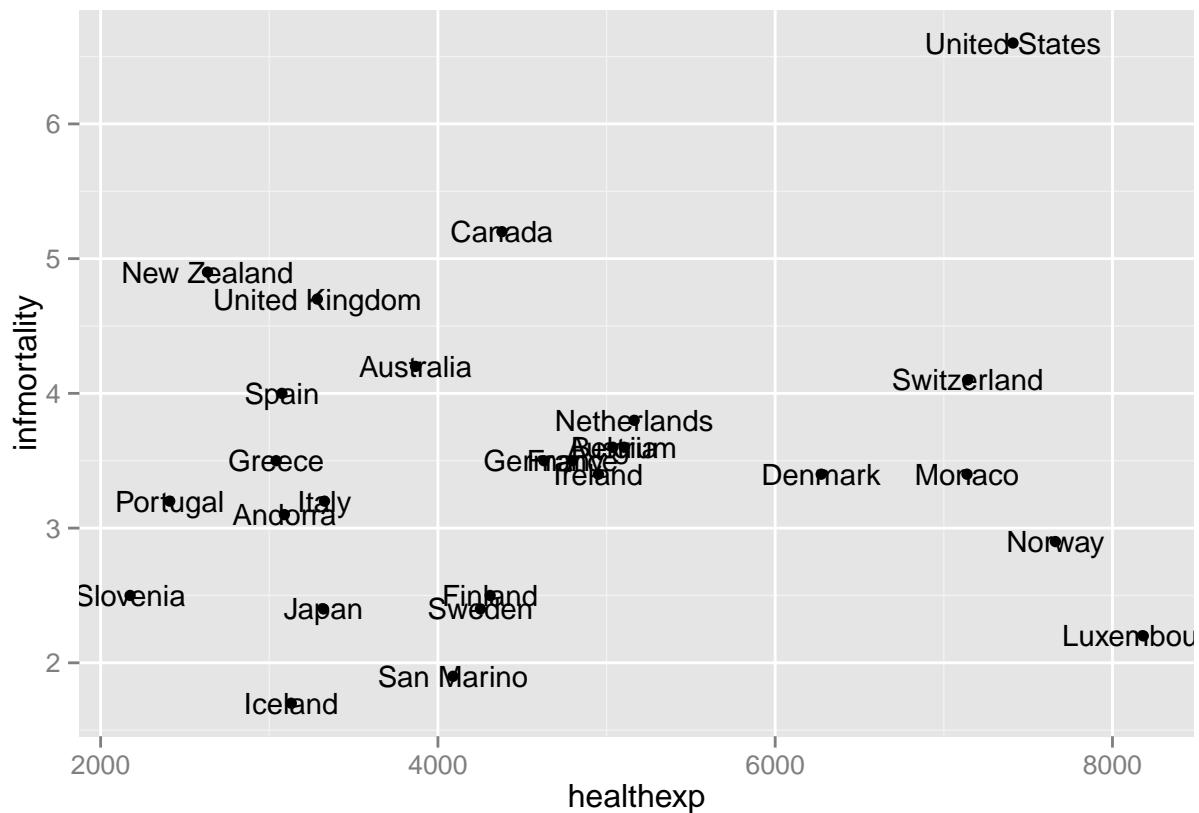
```



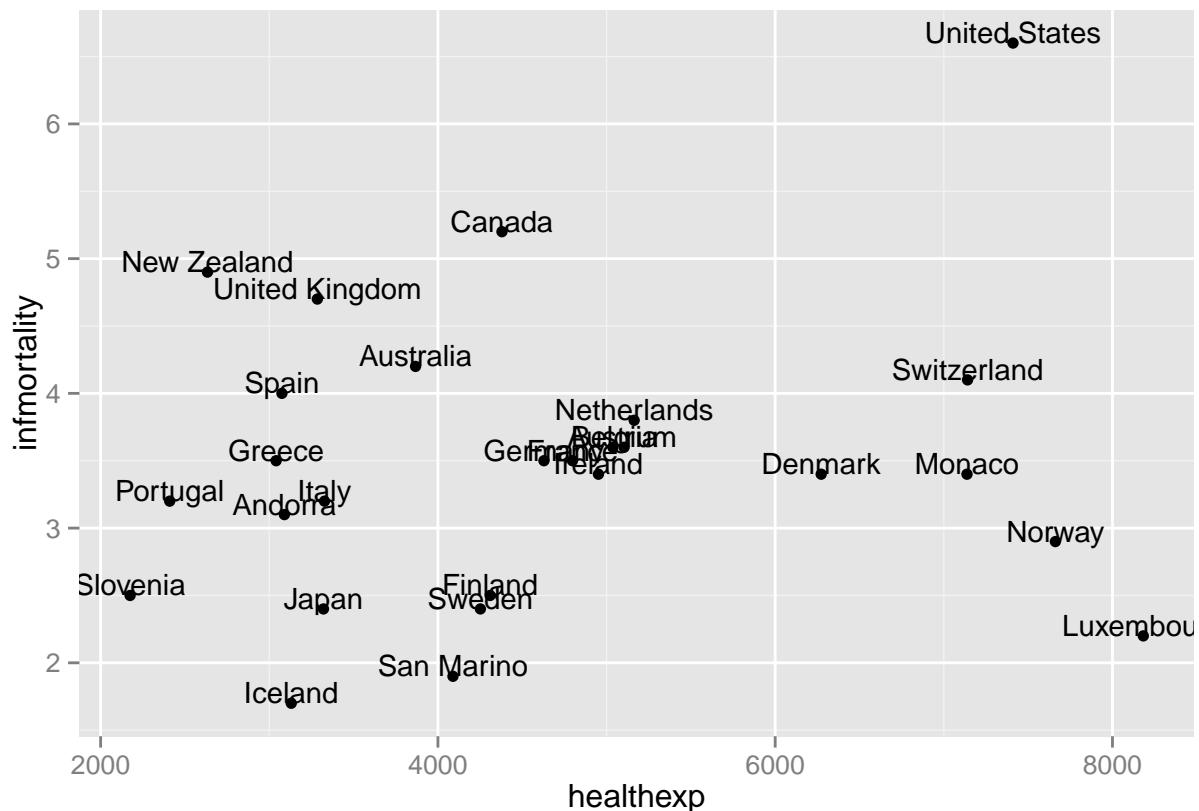
```

# To automatically add the labels from your data we use geom_text() and map a column that is a factor of
sp +
  geom_text( aes( label = Name), size = 4)

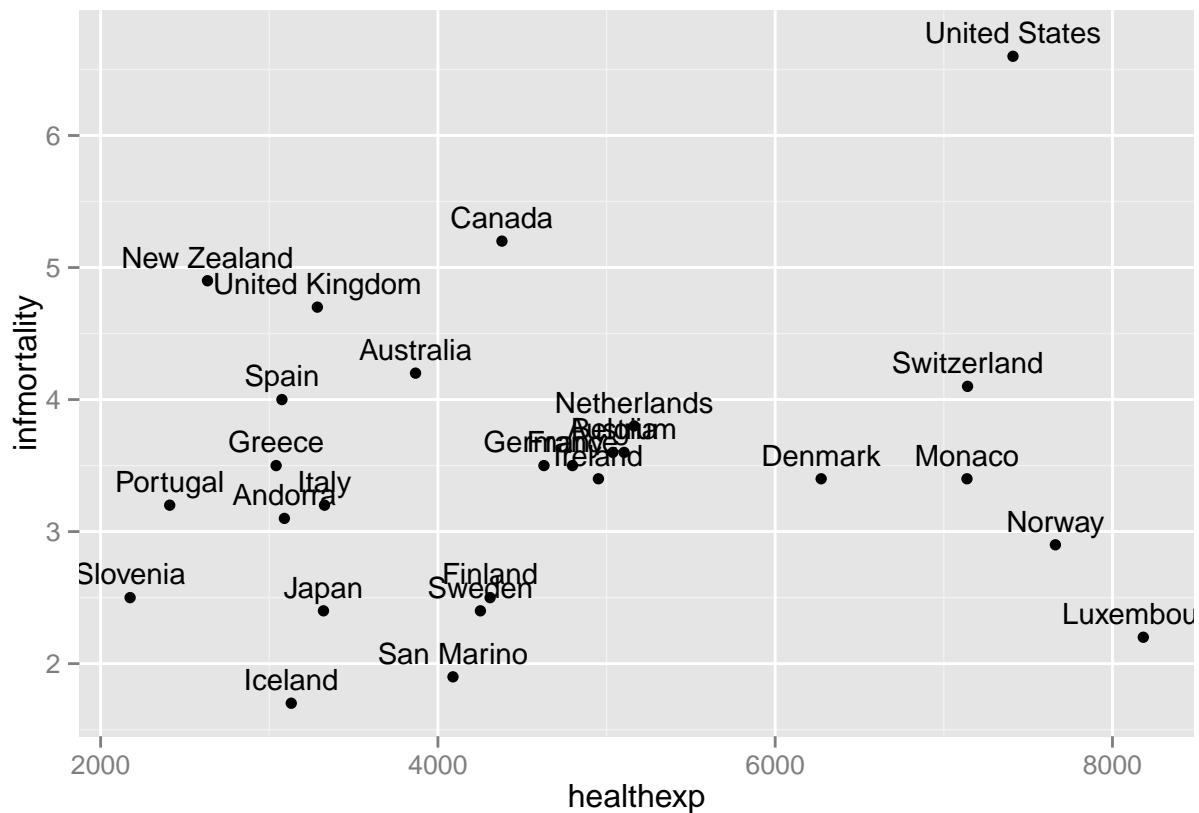
```



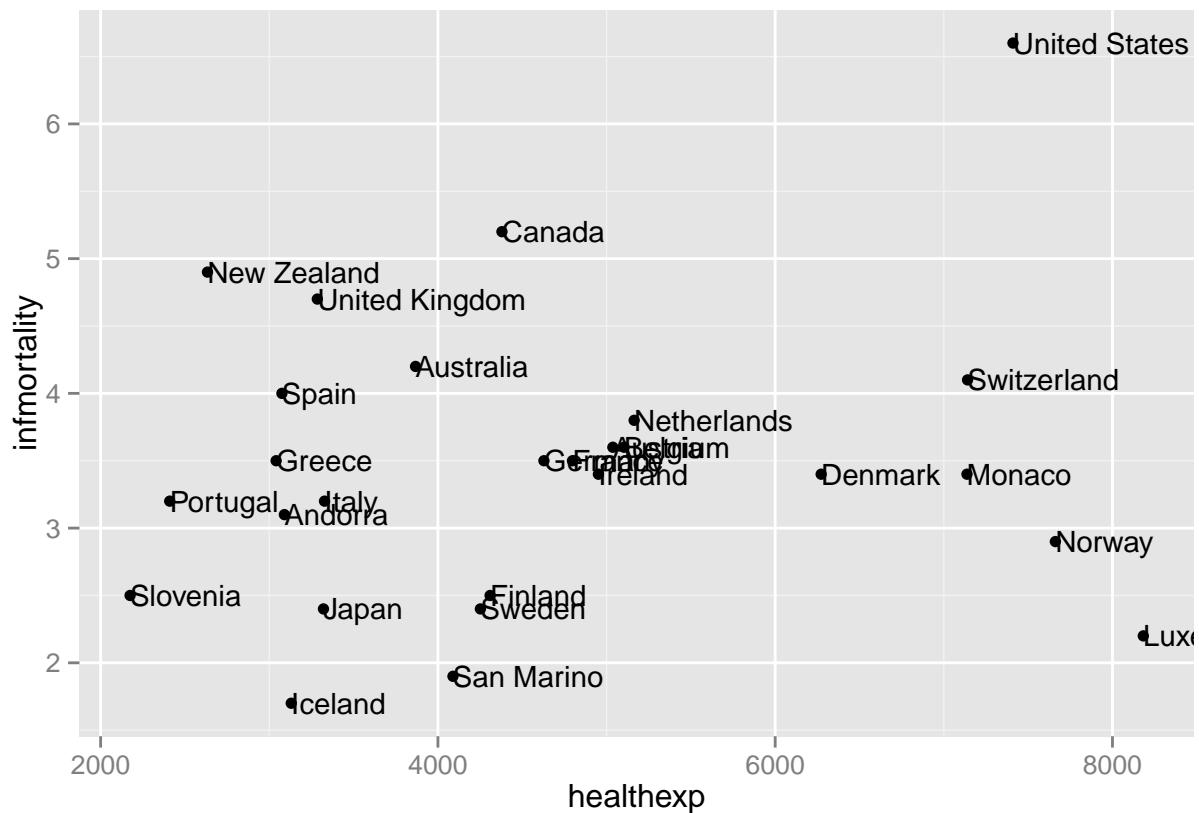
```
sp +
  geom_text( aes( label = Name), size = 4, vjust = 0)
```



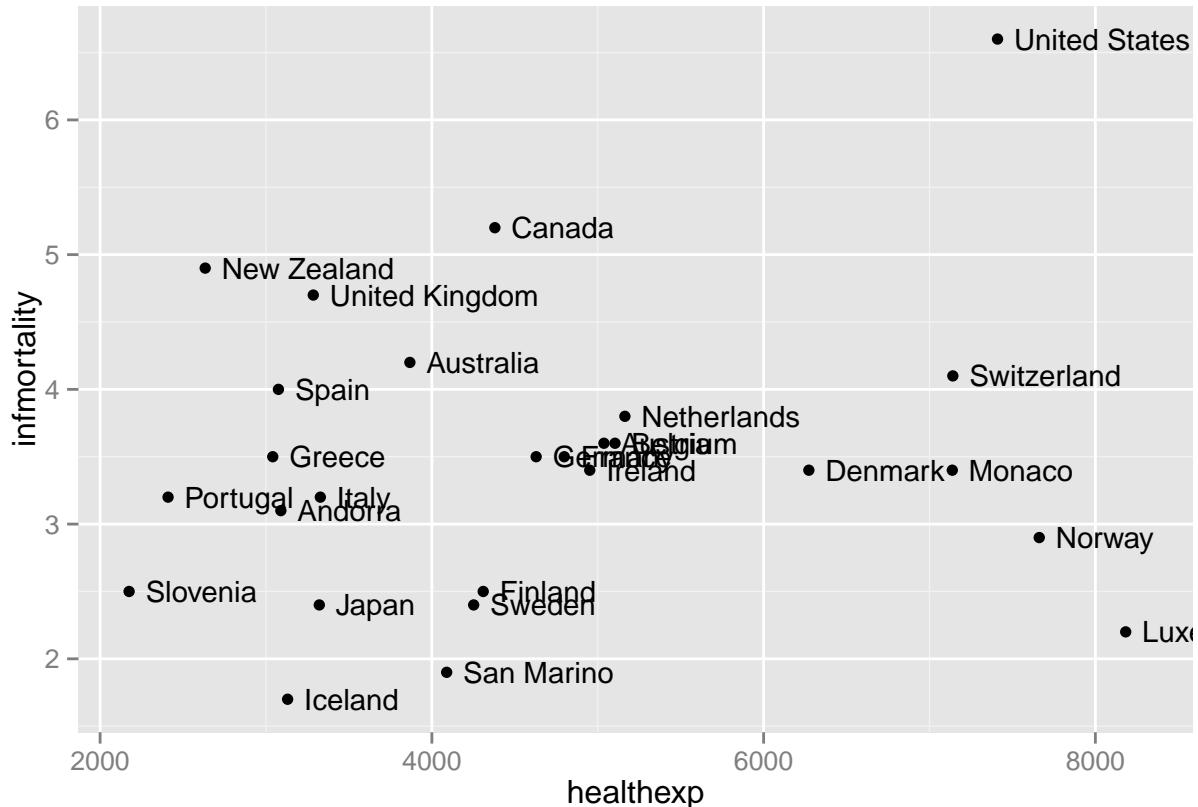
```
# Add a little extra to y
sp +
  geom_text( aes( y = infmtmortality +.1, label = Name), size = 4, vjust = 0)
```



```
sp +
  geom_text( aes( label = Name), size = 4, hjust = 0)
```



```
sp +
  geom_text( aes( x = healthexp + 100, label = Name), size = 4, hjust = 0)
```



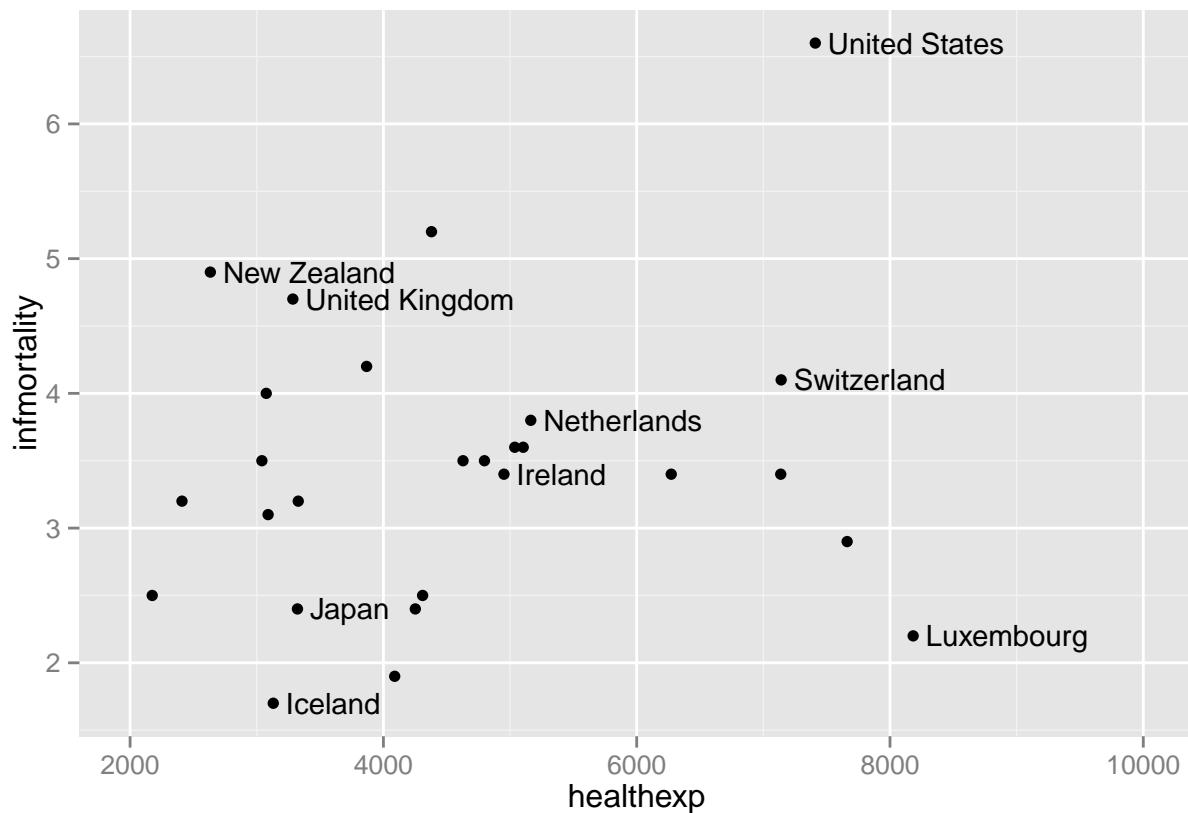
```
# you can add a new column to your data frame containing just the labels you want.
cdat <- subset(countries, Year == 2009 & healthexp > 2000)
cdat$Name1 <- cdat$name
idx <- cdat$name1 %in% c("Canada", "Ireland", "United Kingdom", "United States", "New Zealand", "Iceland")
head(idx)

## [1] FALSE FALSE FALSE FALSE FALSE FALSE

# Then we'll use that Boolean vector to overwrite all the other entries in Name1 with NA:
cdat$name1[!idx] <- NA

# Now we can make the plot
ggplot(cdat, aes(x = healthexp, y = infmortality)) +
  geom_point() +
  geom_text(aes(x = healthexp + 100, label = Name1), size = 4, hjust = 0) +
  xlim(2000, 10000)

## Warning: Removed 18 rows containing missing values (geom_text).
```

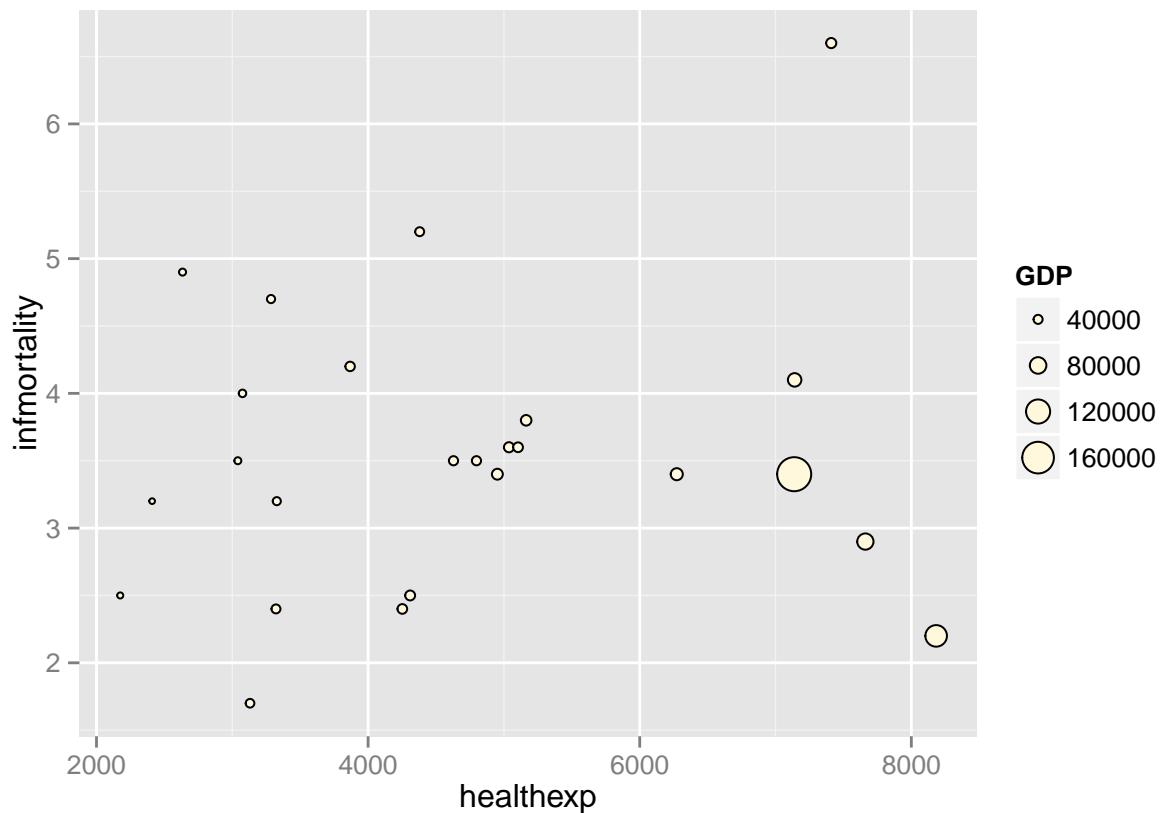


## 12. Creating a Balloon Plot

```
p <- ggplot( cdat, aes( x = healthexp, y = infmortality, size = GDP)) +
  geom_point( shape = 21, colour ="black", fill ="cornsilk")

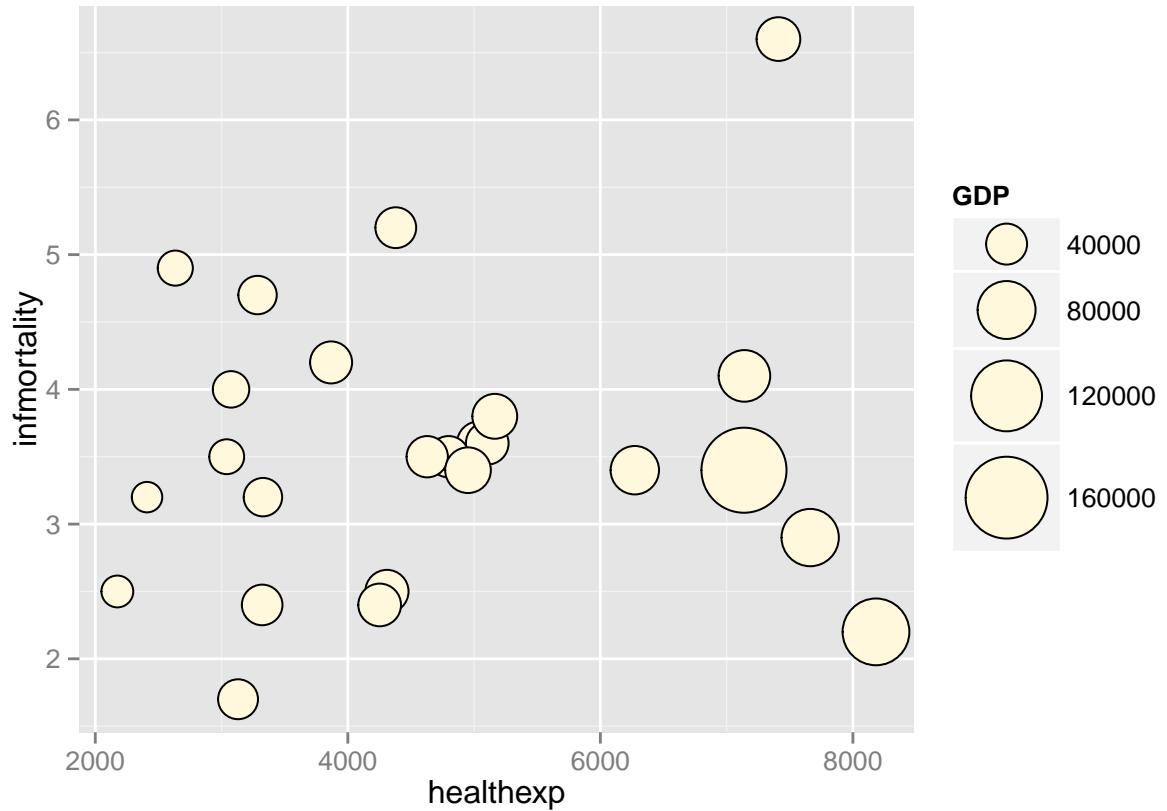
# GDP mapped to radius (default with scale_size_continuous)
p
```

## Warning: Removed 2 rows containing missing values (geom\_point).



```
# GDP mapped to area instead, and larger circles  
p + scale_size_area( max_size = 15)
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```



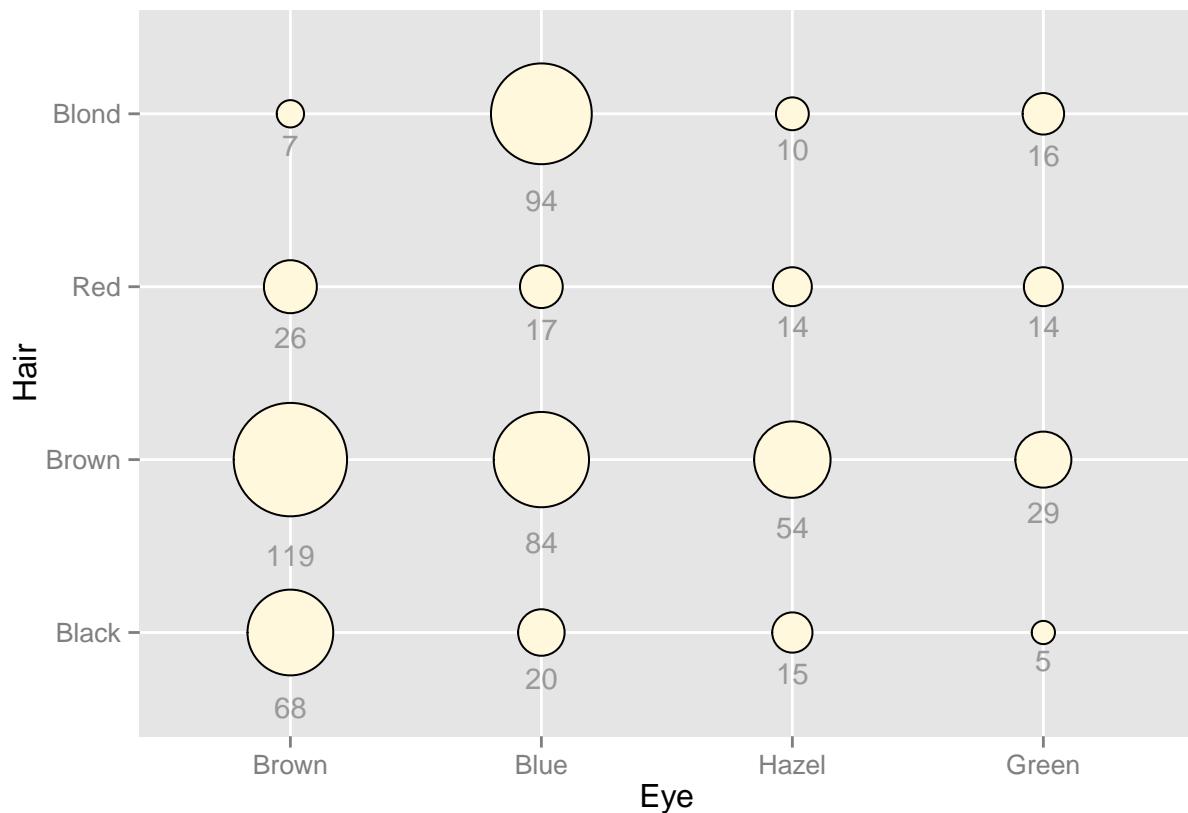
```

# Add up counts for male and female
hec <- HairEyeColor[, "Male"] + HairEyeColor[, "Female"]

# Convert to long format
library( reshape2)
hec <- melt( hec, value.name ="count")

ggplot( hec, aes( x = Eye, y = Hair)) +
  geom_point( aes( size = count), shape = 21, colour ="black", fill ="cornsilk") +
  scale_size_area( max_size = 20, guide = FALSE) +
  geom_text( aes( y = as.numeric(Hair)-sqrt(count)/ 22, label = count), vjust = 1, colour ="grey60", si

```

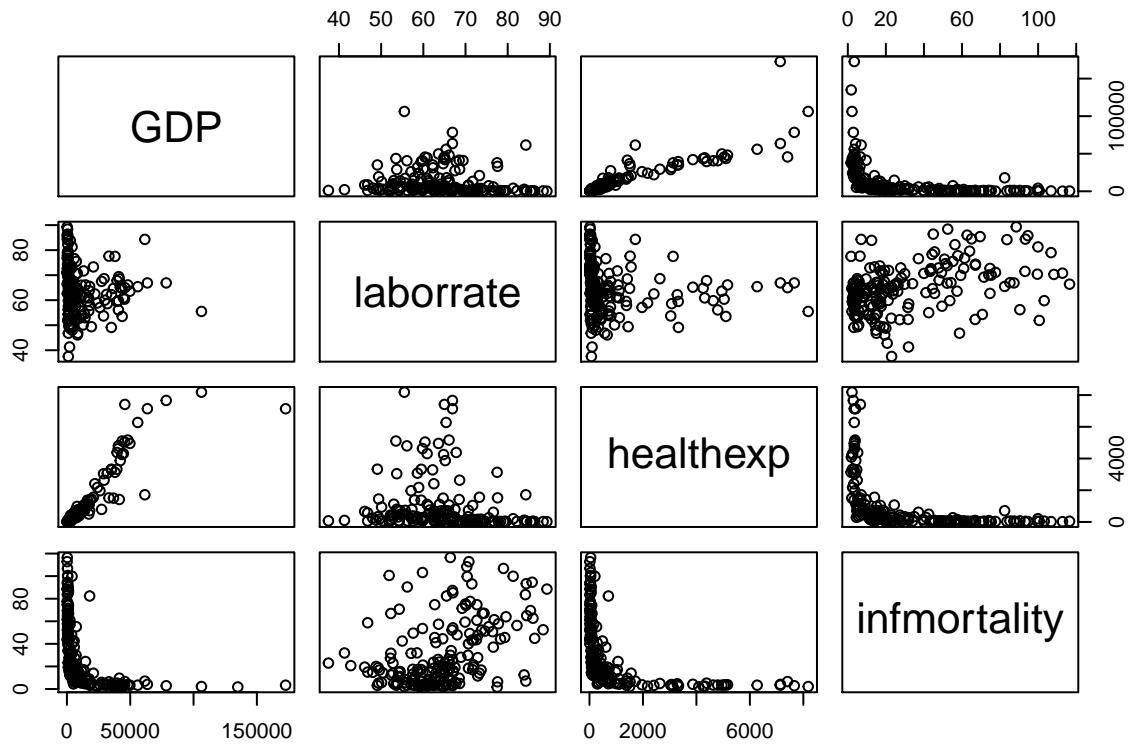


### 13. Making a Scatter Plot Matrix

```
c2009 <- subset( countries, Year == 2009, select = c( Name, GDP, laborrate, healthexp, infmortality))
head(c2009)
```

```
##          Name      GDP laborrate  healthexp infmortality
## 50    Afghanistan     NA     59.8   50.88597    103.2
## 101   Albania 3772.605     59.5   264.60406     17.2
## 152   Algeria 4022.199     58.5   267.94653     32.0
## 203 American Samoa     NA       NA        NA        NA
## 254   Andorra     NA       NA 3089.63589      3.1
## 305   Angola 4068.576     81.3   203.80787    99.9
```

```
# make the scatter plot matrix
pairs( c2009[, 2: 5])
```



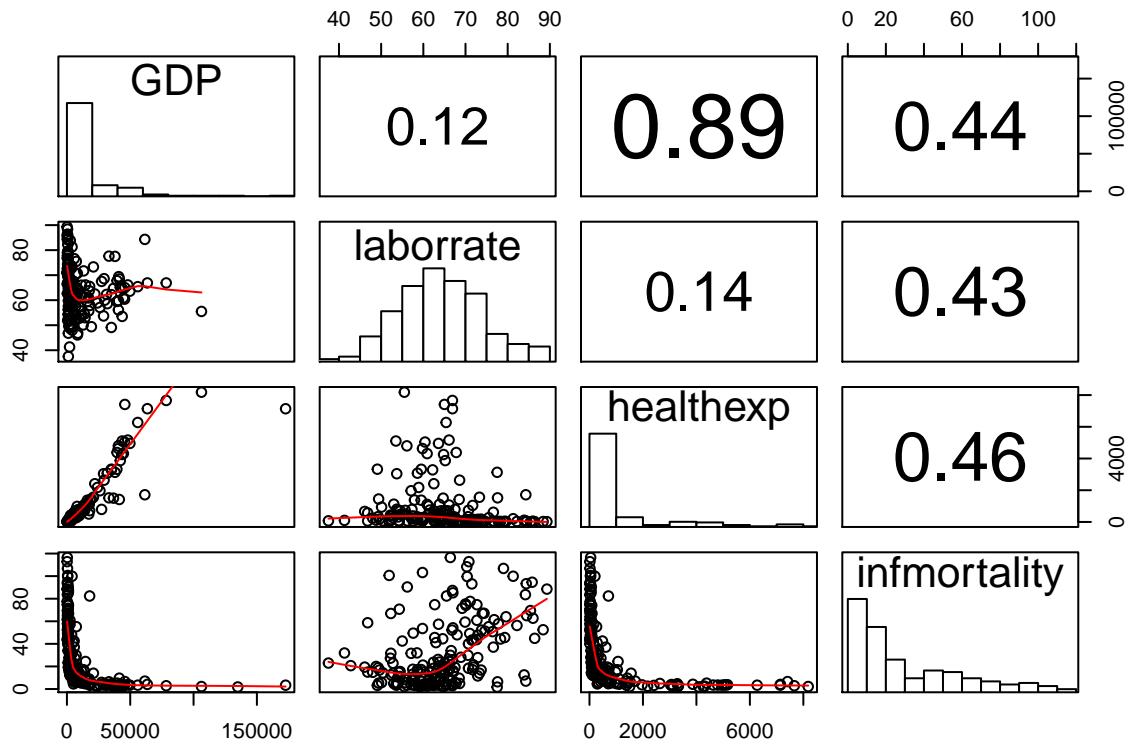
```

panel.cor <- function( x, y, digits = 2, prefix = "", cex.cor, ... ) {
  usr <- par("usr")
  on.exit( par(usr) )
  par( usr = c( 0, 1, 0, 1 ) )
  r <- abs( cor( x, y, use ="complete.obs" ) )
  txt <- format( c( r, 0.123456789 ), digits = digits )[1]
  txt <- paste( prefix, txt, sep = "" )
  if( missing( cex.cor) ) cex.cor <- 0.8 / strwidth(txt)
  text( 0.5, 0.5, txt, cex = cex.cor * (1 + r) / 2 )
}

panel.hist <- function( x, ... ) {
  usr <- par("usr")
  on.exit( par(usr) )
  par( usr = c( usr[1:2], 0, 1.5 ) )
  h <- hist( x, plot = FALSE )
  breaks <- h$breaks
  nB <- length(breaks)
  y <- h$counts
  y <- y/max( y )
  rect( breaks[-nB], 0, breaks[-1], y, col ="white", ... )
}

pairs( c2009[,2:5], upper.panel = panel.cor,
       diag.panel = panel.hist,
       lower.panel = panel.smooth)

```



```
# It may be more desirable to use linear regression lines instead of LOWESS lines.
panel.lm <- function (x, y, col = par("col"), bg = NA, pch = par("pch"), cex = 1, col.smooth = "black",
  points( x, y, pch = pch, col = col, bg = bg, cex = cex)
  abline( stats::lm( y ~ x), col = col.smooth, ...)
}

pairs( c2009[, 2:5], pch =".",
  upper.panel = panel.cor,
  diag.panel = panel.hist,
  lower.panel = panel.lm)
```

