Example R Code/Graphs Using NHANES

A.S. Wagaman

Last Updated: July 23, 2018

Contents

Introduction to This Resource

In this document, you will find a series of coded examples on the NHANES data set, to demonstrate the code used in the course. The code uses the formula syntax of *mosaic* along with *ggformula* for graphics. This may mean some of the code looks different to you, depending on your introductory statistics course experience. The document is designed to be culmulative, covering the entire course, and is organized loosely by chapters/topics. All necessary packages are loaded at the start, and some may need to be installed, especially if you are working from your own machine.

Data Set - NHANES

First, we load the data set and learn a little bit about it.

```r
data(NHANES)
```

Please reference the help file for NHANES for a lot more detail.

```r
?NHANES
```

The three points below are from the help file.

"This is survey data collected by the US National Center for Health Statistics (NCHS) which has conducted a series of health and nutrition surveys since the early 1960's. Since 1999 approximately 5,000 individuals of all ages are interviewed in their homes every year and complete the health examination component of the survey. The health examination is conducted in a mobile examination centre (MEC)."

"NHANES can be treated, for educational purposes, as if it were a simple random sample from the American population."

"Please note that the data sets provided in this package are derived from the NHANES database and have been adapted for educational purposes. As such, they are NOT suitable for use as a research database."

The educational purpose data set has a large number of variables (76) and 10000 observations. It is useful to demonstrate methods due to having many characteristics of real data, but it has been fairly cleaned for our use.

Review from Intro Stat (also Chapter 0 from Stat2)

Data Management

These are basic commands to look at the data set and see what is there.

First, you may want a quick look at a few observations. There is a quick way to look at either the first 6 or last 6 observations.

```r
head(NHANES)
```

```
## # A tibble: 6 x 76
##      ID SurveyYr Gender   Age AgeDecade AgeMonths Race1 Race3 Education
##   <int> <fct>    <fct>  <int> <fct>         <int> <fct> <fct> <fct>
## 1 51624 2009_10  male      34 " 30-39"        409 White <NA>  High Sch~
## 2 51624 2009_10  male      34 " 30-39"        409 White <NA>  High Sch~
## 3 51624 2009_10  male      34 " 30-39"        409 White <NA>  High Sch~
## 4 51625 2009_10  male       4 " 0-9"           49 Other <NA>  <NA>
## 5 51630 2009_10  female    49 " 40-49"        596 White <NA>  Some Col~
## 6 51638 2009_10  male       9 " 0-9"          115 White <NA>  <NA>
## # ... with 67 more variables: MaritalStatus <fct>, HHIncome <fct>,
## #   HHIncomeMid <int>, Poverty <dbl>, HomeRooms <int>, HomeOwn <fct>,
## #   Work <fct>, Weight <dbl>, Length <dbl>, HeadCirc <dbl>, Height <dbl>,
## #   BMI <dbl>, BMICatUnder20yrs <fct>, BMI_WHO <fct>, Pulse <int>,
## #   BPSysAve <int>, BPDiaAve <int>, BPSys1 <int>, BPDia1 <int>,
## #   BPSys2 <int>, BPDia2 <int>, BPSys3 <int>, BPDia3 <int>,
## #   Testosterone <dbl>, DirectChol <dbl>, TotChol <dbl>, UrineVol1 <int>,
## #   UrineFlow1 <dbl>, UrineVol2 <int>, UrineFlow2 <dbl>, Diabetes <fct>,
## #   DiabetesAge <int>, HealthGen <fct>, DaysPhysHlthBad <int>,
## #   DaysMentHlthBad <int>, LittleInterest <fct>, Depressed <fct>,
## #   nPregnancies <int>, nBabies <int>, Age1stBaby <int>,
## #   SleepHrsNight <int>, SleepTrouble <fct>, PhysActive <fct>,
## #   PhysActiveDays <int>, TVHrsDay <fct>, CompHrsDay <fct>,
## #   TVHrsDayChild <int>, CompHrsDayChild <int>, Alcohol12PlusYr <fct>,
## #   AlcoholDay <int>, AlcoholYear <int>, SmokeNow <fct>, Smoke100 <fct>,
## #   Smoke100n <fct>, SmokeAge <int>, Marijuana <fct>, AgeFirstMarij <int>,
## #   RegularMarij <fct>, AgeRegMarij <int>, HardDrugs <fct>, SexEver <fct>,
## #   SexAge <int>, SexNumPartnLife <int>, SexNumPartYear <int>,
## #   SameSex <fct>, SexOrientation <fct>, PregnantNow <fct>
```

```
tail(NHANES)
```

```
## # A tibble: 6 x 76
##      ID SurveyYr Gender   Age AgeDecade AgeMonths Race1 Race3 Education
##   <int> <fct>    <fct>  <int> <fct>         <int> <fct> <fct> <fct>
## 1 71909 2011_12  male      28 " 20-29"         NA Mexi~ Mexi~ 9 - 11th~
## 2 71909 2011_12  male      28 " 20-29"         NA Mexi~ Mexi~ 9 - 11th~
## 3 71910 2011_12  female     0 " 0-9"            5 White White <NA>
## 4 71911 2011_12  male      27 " 20-29"         NA Mexi~ Mexi~ College ~
## 5 71915 2011_12  male      60 " 60-69"         NA White White College ~
## 6 71915 2011_12  male      60 " 60-69"         NA White White College ~
## # ... with 67 more variables: MaritalStatus <fct>, HHIncome <fct>,
## #   HHIncomeMid <int>, Poverty <dbl>, HomeRooms <int>, HomeOwn <fct>,
## #   Work <fct>, Weight <dbl>, Length <dbl>, HeadCirc <dbl>, Height <dbl>,
## #   BMI <dbl>, BMICatUnder20yrs <fct>, BMI_WHO <fct>, Pulse <int>,
## #   BPSysAve <int>, BPDiaAve <int>, BPSys1 <int>, BPDia1 <int>,
## #   BPSys2 <int>, BPDia2 <int>, BPSys3 <int>, BPDia3 <int>,
## #   Testosterone <dbl>, DirectChol <dbl>, TotChol <dbl>, UrineVol1 <int>,
## #   UrineFlow1 <dbl>, UrineVol2 <int>, UrineFlow2 <dbl>, Diabetes <fct>,
## #   DiabetesAge <int>, HealthGen <fct>, DaysPhysHlthBad <int>,
## #   DaysMentHlthBad <int>, LittleInterest <fct>, Depressed <fct>,
## #   nPregnancies <int>, nBabies <int>, Age1stBaby <int>,
## #   SleepHrsNight <int>, SleepTrouble <fct>, PhysActive <fct>,
## #   PhysActiveDays <int>, TVHrsDay <fct>, CompHrsDay <fct>,
## #   TVHrsDayChild <int>, CompHrsDayChild <int>, Alcohol12PlusYr <fct>,
## #   AlcoholDay <int>, AlcoholYear <int>, SmokeNow <fct>, Smoke100 <fct>,
## #   Smoke100n <fct>, SmokeAge <int>, Marijuana <fct>, AgeFirstMarij <int>,
## #   RegularMarij <fct>, AgeRegMarij <int>, HardDrugs <fct>, SexEver <fct>,
## #   SexAge <int>, SexNumPartnLife <int>, SexNumPartYear <int>,
## #   SameSex <fct>, SexOrientation <fct>, PregnantNow <fct>
```

You may also want to know how many rows and columns are in the data set. You can read this in the Environment window, or try these commands.

```
ncol(NHANES)
```

```
## [1] 76
```

```
nrow(NHANES)
```

```
## [1] 10000
```

Finally, you may want a quick summary of the overarching data set. We demo three ways to do this below. You'd only really need to run one of these. Pick the one you like - though str and glimpse are probably more useful than summary.

```
#summary(NHANES) #very long output!
str(NHANES) #may be new
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    10000 obs. of  76 variables:
##  $ ID              : int  51624 51624 51624 51625 51630 51638 51646 51647 51647 51647 ...
##  $ SurveyYr        : Factor w/ 2 levels "2009_10","2011_12": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Gender          : Factor w/ 2 levels "female","male": 2 2 2 2 1 2 2 1 1 1 ...
##  $ Age             : int  34 34 34 4 49 9 8 45 45 45 ...
##  $ AgeDecade       : Factor w/ 8 levels " 0-9"," 10-19",..: 4 4 4 1 5 1 1 5 5 5 ...
##  $ AgeMonths       : int  409 409 409 49 596 115 101 541 541 541 ...
##  $ Race1           : Factor w/ 5 levels "Black","Hispanic",..: 4 4 4 5 4 4 4 4 4 4 ...
##  $ Race3           : Factor w/ 6 levels "Asian","Black",..: NA NA NA NA NA NA NA NA NA NA ...
##  $ Education       : Factor w/ 5 levels "8th Grade","9 - 11th Grade",..: 3 3 3 NA 4 NA NA 5 5 5 ...
##  $ MaritalStatus   : Factor w/ 6 levels "Divorced","LivePartner",..: 3 3 3 NA 2 NA NA 3 3 3 ...
##  $ HHIncome        : Factor w/ 12 levels " 0-4999"," 5000-9999",..: 6 6 6 5 7 11 9 11 11 11 ...
##  $ HHIncomeMid     : int  30000 30000 30000 22500 40000 87500 60000 87500 87500 87500 ...
##  $ Poverty         : num  1.36 1.36 1.36 1.07 1.91 1.84 2.33 5 5 5 ...
##  $ HomeRooms       : int  6 6 6 9 5 6 7 6 6 6 ...
##  $ HomeOwn         : Factor w/ 3 levels "Own","Rent","Other": 1 1 1 1 2 2 1 1 1 1 ...
##  $ Work            : Factor w/ 3 levels "Looking","NotWorking",..: 2 2 2 NA 2 NA NA 3 3 3 ...
##  $ Weight          : num  87.4 87.4 87.4 17 86.7 29.8 35.2 75.7 75.7 75.7 ...
##  $ Length          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ HeadCirc        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ Height          : num  165 165 165 105 168 ...
##  $ BMI             : num  32.2 32.2 32.2 15.3 30.6 ...
##  $ BMICatUnder20yrs: Factor w/ 4 levels "UnderWeight",..: NA NA NA NA NA NA NA NA NA NA ...
##  $ BMI_WHO         : Factor w/ 4 levels "12.0_18.5","18.5_to_24.9",..: 4 4 4 1 4 1 2 3 3 3 ...
##  $ Pulse           : int  70 70 70 NA 86 82 72 62 62 62 ...
##  $ BPSysAve        : int  113 113 113 NA 112 86 107 118 118 118 ...
##  $ BPDiaAve        : int  85 85 85 NA 75 47 37 64 64 64 ...
##  $ BPSys1          : int  114 114 114 NA 118 84 114 106 106 106 ...
##  $ BPDia1          : int  88 88 88 NA 82 50 46 62 62 62 ...
##  $ BPSys2          : int  114 114 114 NA 108 84 108 118 118 118 ...
##  $ BPDia2          : int  88 88 88 NA 74 50 36 68 68 68 ...
##  $ BPSys3          : int  112 112 112 NA 116 88 106 118 118 118 ...
##  $ BPDia3          : int  82 82 82 NA 76 44 38 60 60 60 ...
##  $ Testosterone    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ DirectChol      : num  1.29 1.29 1.29 NA 1.16 1.34 1.55 2.12 2.12 2.12 ...
##  $ TotChol         : num  3.49 3.49 3.49 NA 6.7 4.86 4.09 5.82 5.82 5.82 ...
##  $ UrineVol1       : int  352 352 352 NA 77 123 238 106 106 106 ...
##  $ UrineFlow1      : num  NA NA NA NA 0.094 ...
##  $ UrineVol2       : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ UrineFlow2      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ Diabetes        : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ DiabetesAge     : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ HealthGen       : Factor w/ 5 levels "Excellent","Vgood",..: 3 3 3 NA 3 NA NA 2 2 2 ...
##  $ DaysPhysHlthBad : int  0 0 0 NA 0 NA NA 0 0 0 ...
##  $ DaysMentHlthBad : int  15 15 15 NA 10 NA NA 3 3 3 ...
##  $ LittleInterest  : Factor w/ 3 levels "None","Several",..: 3 3 3 NA 2 NA NA 1 1 1 ...
##  $ Depressed       : Factor w/ 3 levels "None","Several",..: 2 2 2 NA 2 NA NA 1 1 1 ...
##  $ nPregnancies    : int  NA NA NA NA 2 NA NA 1 1 1 ...
##  $ nBabies         : int  NA NA NA NA 2 NA NA NA NA NA ...
##  $ Age1stBaby      : int  NA NA NA NA 27 NA NA NA NA NA ...
##  $ SleepHrsNight   : int  4 4 4 NA 8 NA NA 8 8 8 ...
##  $ SleepTrouble    : Factor w/ 2 levels "No","Yes": 2 2 2 NA 2 NA NA 1 1 1 ...
##  $ PhysActive      : Factor w/ 2 levels "No","Yes": 1 1 1 NA 1 NA NA 2 2 2 ...
##  $ PhysActiveDays  : int  NA NA NA NA NA NA NA 5 5 5 ...
##  $ TVHrsDay        : Factor w/ 7 levels "0_hrs","0_to_1_hr",..: NA NA NA NA NA NA NA NA NA NA ...
##  $ CompHrsDay      : Factor w/ 7 levels "0_hrs","0_to_1_hr",..: NA NA NA NA NA NA NA NA NA NA ...
##  $ TVHrsDayChild   : int  NA NA NA 4 NA 5 1 NA NA NA ...
##  $ CompHrsDayChild : int  NA NA NA 1 NA 0 6 NA NA NA ...
##  $ Alcohol12PlusYr : Factor w/ 2 levels "No","Yes": 2 2 2 NA 2 NA NA 2 2 2 ...
##  $ AlcoholDay      : int  NA NA NA NA 2 NA NA 3 3 3 ...
##  $ AlcoholYear     : int  0 0 0 NA 20 NA NA 52 52 52 ...
##  $ SmokeNow        : Factor w/ 2 levels "No","Yes": 1 1 1 NA 2 NA NA NA NA NA ...
##  $ Smoke100        : Factor w/ 2 levels "No","Yes": 2 2 2 NA 2 NA NA 1 1 1 ...
##  $ Smoke100n       : Factor w/ 2 levels "Non-Smoker","Smoker": 2 2 2 NA 2 NA NA 1 1 1 ...
##  $ SmokeAge        : int  18 18 18 NA 38 NA NA NA NA NA ...
##  $ Marijuana       : Factor w/ 2 levels "No","Yes": 2 2 2 NA 2 NA NA 2 2 2 ...
##  $ AgeFirstMarij   : int  17 17 17 NA 18 NA NA 13 13 13 ...
##  $ RegularMarij    : Factor w/ 2 levels "No","Yes": 1 1 1 NA 1 NA NA 1 1 1 ...
##  $ AgeRegMarij     : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ HardDrugs       : Factor w/ 2 levels "No","Yes": 2 2 2 NA 2 NA NA 1 1 1 ...
##  $ SexEver         : Factor w/ 2 levels "No","Yes": 2 2 2 NA 2 NA NA 2 2 2 ...
##  $ SexAge          : int  16 16 16 NA 12 NA NA 13 13 13 ...
##  $ SexNumPartnLife : int  8 8 8 NA 10 NA NA 20 20 20 ...
##  $ SexNumPartYear  : int  1 1 1 NA 1 NA NA 0 0 0 ...
##  $ SameSex         : Factor w/ 2 levels "No","Yes": 1 1 1 NA 2 NA NA 2 2 2 ...
##  $ SexOrientation  : Factor w/ 3 levels "Bisexual","Heterosexual",..: 2 2 2 NA 2 NA NA 1 1 1 ...
##  $ PregnantNow     : Factor w/ 3 levels "Yes","No","Unknown": NA NA NA NA NA NA NA NA NA NA ...
```

```r
glimpse(NHANES) #may be new
```

```
## Observations: 10,000
## Variables: 76
## $ ID               <int> 51624, 51624, 51624, 51625, 51630, 51638, 516...
## $ SurveyYr         <fct> 2009_10, 2009_10, 2009_10, 2009_10, 2009_10, ...
## $ Gender           <fct> male, male, male, male, female, male, male, f...
## $ Age              <int> 34, 34, 34, 4, 49, 9, 8, 45, 45, 45, 66, 58, ...
## $ AgeDecade        <fct>  30-39,  30-39,  30-39,  0-9,  40-49,  0-9,  ...
## $ AgeMonths        <int> 409, 409, 409, 49, 596, 115, 101, 541, 541, 5...
## $ Race1            <fct> White, White, White, Other, White, White, Whi...
## $ Race3            <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ Education        <fct> High School, High School, High School, NA, So...
## $ MaritalStatus    <fct> Married, Married, Married, NA, LivePartner, N...
## $ HHIncome         <fct> 25000-34999, 25000-34999, 25000-34999, 20000-...
## $ HHIncomeMid      <int> 30000, 30000, 30000, 22500, 40000, 87500, 600...
## $ Poverty          <dbl> 1.36, 1.36, 1.36, 1.07, 1.91, 1.84, 2.33, 5.0...
## $ HomeRooms        <int> 6, 6, 6, 9, 5, 6, 7, 6, 6, 6, 5, 10, 6, 10, 1...
## $ HomeOwn          <fct> Own, Own, Own, Own, Rent, Rent, Own, Own, Own...
## $ Work             <fct> NotWorking, NotWorking, NotWorking, NA, NotWo...
## $ Weight           <dbl> 87.4, 87.4, 87.4, 17.0, 86.7, 29.8, 35.2, 75....
## $ Length           <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ HeadCirc         <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ Height           <dbl> 164.7, 164.7, 164.7, 105.4, 168.4, 133.1, 130...
## $ BMI              <dbl> 32.22, 32.22, 32.22, 15.30, 30.57, 16.82, 20....
## $ BMICatUnder20yrs <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ BMI_WHO          <fct> 30.0_plus, 30.0_plus, 30.0_plus, 12.0_18.5, 3...
## $ Pulse            <int> 70, 70, 70, NA, 86, 82, 72, 62, 62, 62, 60, 6...
## $ BPSysAve         <int> 113, 113, 113, NA, 112, 86, 107, 118, 118, 11...
## $ BPDiaAve         <int> 85, 85, 85, NA, 75, 47, 37, 64, 64, 64, 63, 7...
## $ BPSys1           <int> 114, 114, 114, NA, 118, 84, 114, 106, 106, 10...
## $ BPDia1           <int> 88, 88, 88, NA, 82, 50, 46, 62, 62, 62, 64, 7...
## $ BPSys2           <int> 114, 114, 114, NA, 108, 84, 108, 118, 118, 11...
## $ BPDia2           <int> 88, 88, 88, NA, 74, 50, 36, 68, 68, 68, 62, 7...
## $ BPSys3           <int> 112, 112, 112, NA, 116, 88, 106, 118, 118, 11...
## $ BPDia3           <int> 82, 82, 82, NA, 76, 44, 38, 60, 60, 60, 64, 7...
## $ Testosterone     <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ DirectChol       <dbl> 1.29, 1.29, 1.29, NA, 1.16, 1.34, 1.55, 2.12,...
## $ TotChol          <dbl> 3.49, 3.49, 3.49, NA, 6.70, 4.86, 4.09, 5.82,...
## $ UrineVol1        <int> 352, 352, 352, NA, 77, 123, 238, 106, 106, 10...
## $ UrineFlow1       <dbl> NA, NA, NA, NA, 0.094, 1.538, 1.322, 1.116, 1...
## $ UrineVol2        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ UrineFlow2       <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ Diabetes         <fct> No, No, No, No, No, No, No, No, No, No, No, N...
## $ DiabetesAge      <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ HealthGen        <fct> Good, Good, Good, NA, Good, NA, NA, Vgood, Vg...
## $ DaysPhysHlthBad  <int> 0, 0, 0, NA, 0, NA, NA, 0, 0, 0, 10, 0, 4, NA...
## $ DaysMentHlthBad  <int> 15, 15, 15, NA, 10, NA, NA, 3, 3, 3, 0, 0, 0,...
## $ LittleInterest   <fct> Most, Most, Most, NA, Several, NA, NA, None, ...
## $ Depressed        <fct> Several, Several, Several, NA, Several, NA, N...
## $ nPregnancies     <int> NA, NA, NA, NA, 2, NA, NA, 1, 1, 1, NA, NA, N...
## $ nBabies          <int> NA, NA, NA, NA, 2, NA, NA, NA, NA, NA, NA, NA...
## $ Age1stBaby       <int> NA, NA, NA, NA, 27, NA, NA, NA, NA, NA, NA, N...
## $ SleepHrsNight    <int> 4, 4, 4, NA, 8, NA, NA, 8, 8, 8, 7, 5, 4, NA,...
## $ SleepTrouble     <fct> Yes, Yes, Yes, NA, Yes, NA, NA, No, No, No, N...
## $ PhysActive       <fct> No, No, No, NA, No, NA, NA, Yes, Yes, Yes, Ye...
## $ PhysActiveDays   <int> NA, NA, NA, NA, NA, NA, NA, 5, 5, 5, 7, 5, 1,...
## $ TVHrsDay         <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ CompHrsDay       <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ TVHrsDayChild    <int> NA, NA, NA, 4, NA, 5, 1, NA, NA, NA, NA, NA, ...
## $ CompHrsDayChild  <int> NA, NA, NA, 1, NA, 0, 6, NA, NA, NA, NA, NA, ...
## $ Alcohol12PlusYr  <fct> Yes, Yes, Yes, NA, Yes, NA, NA, Yes, Yes, Yes...
## $ AlcoholDay       <int> NA, NA, NA, NA, 2, NA, NA, 3, 3, 3, 1, 2, 6, ...
## $ AlcoholYear      <int> 0, 0, 0, NA, 20, NA, NA, 52, 52, 52, 100, 104...
## $ SmokeNow         <fct> No, No, No, NA, Yes, NA, NA, NA, NA, NA, No, ...
## $ Smoke100         <fct> Yes, Yes, Yes, NA, Yes, NA, NA, No, No, No, Y...
## $ Smoke100n        <fct> Smoker, Smoker, Smoker, NA, Smoker, NA, NA, N...
## $ SmokeAge         <int> 18, 18, 18, NA, 38, NA, NA, NA, NA, NA, 13, N...
## $ Marijuana        <fct> Yes, Yes, Yes, NA, Yes, NA, NA, Yes, Yes, Yes...
## $ AgeFirstMarij    <int> 17, 17, 17, NA, 18, NA, NA, 13, 13, 13, NA, 1...
## $ RegularMarij     <fct> No, No, No, NA, No, NA, NA, No, No, No, NA, Y...
## $ AgeRegMarij      <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 2...
## $ HardDrugs        <fct> Yes, Yes, Yes, NA, Yes, NA, NA, No, No, No, N...
## $ SexEver          <fct> Yes, Yes, Yes, NA, Yes, NA, NA, Yes, Yes, Yes...
## $ SexAge           <int> 16, 16, 16, NA, 12, NA, NA, 13, 13, 13, 17, 2...
## $ SexNumPartnLife  <int> 8, 8, 8, NA, 10, NA, NA, 20, 20, 20, 15, 7, 1...
## $ SexNumPartYear   <int> 1, 1, 1, NA, 1, NA, NA, 0, 0, 0, NA, 1, 1, NA...
## $ SameSex          <fct> No, No, No, NA, Yes, NA, NA, Yes, Yes, Yes, N...
## $ SexOrientation   <fct> Heterosexual, Heterosexual, Heterosexual, NA,...
## $ PregnantNow      <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
```

Univariate Plots and Summaries

To start with some data exploration, let's consider the relationship between Age and Education, so we have both a quantitative and a categorical variable of interest.

Descriptive Statistics

```
#Quantitative Variable
favstats(~ Age, data = NHANES)
```

```
##  min Q1 median Q3 max    mean      sd     n missing
##    0 17     36 54  80 36.7421 22.3976 10000       0
```

```
mean(~ Age, data = NHANES)
```

```
## [1] 36.7421
```

```
sd(~ Age, data = NHANES)
```

```
## [1] 22.3976
```

```
median(~ Age, data = NHANES)
```

```
## [1] 36
```

```
IQR(~ Age, data = NHANES)
```

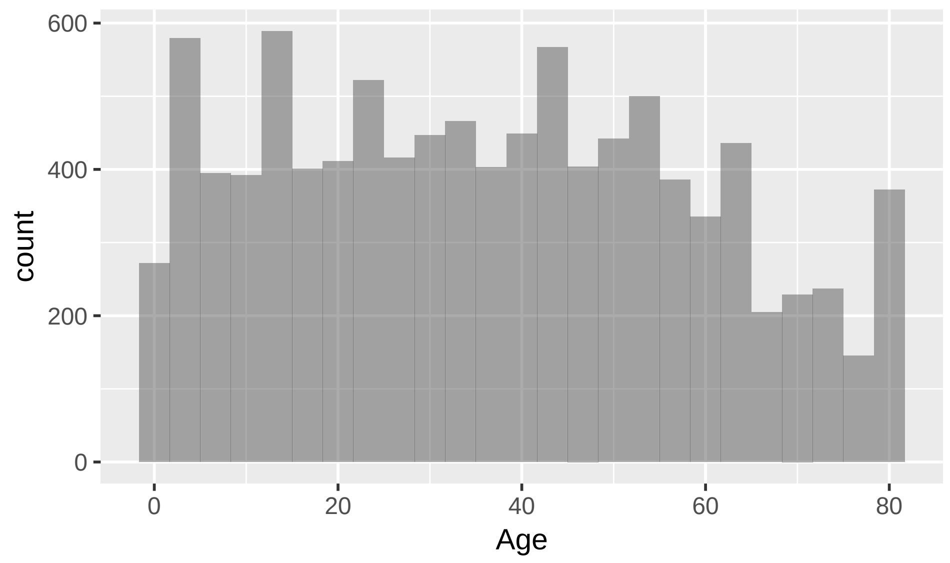
```
## [1] 37
```

```
#Categorical Variable
tally(~ Education, data = NHANES)
```

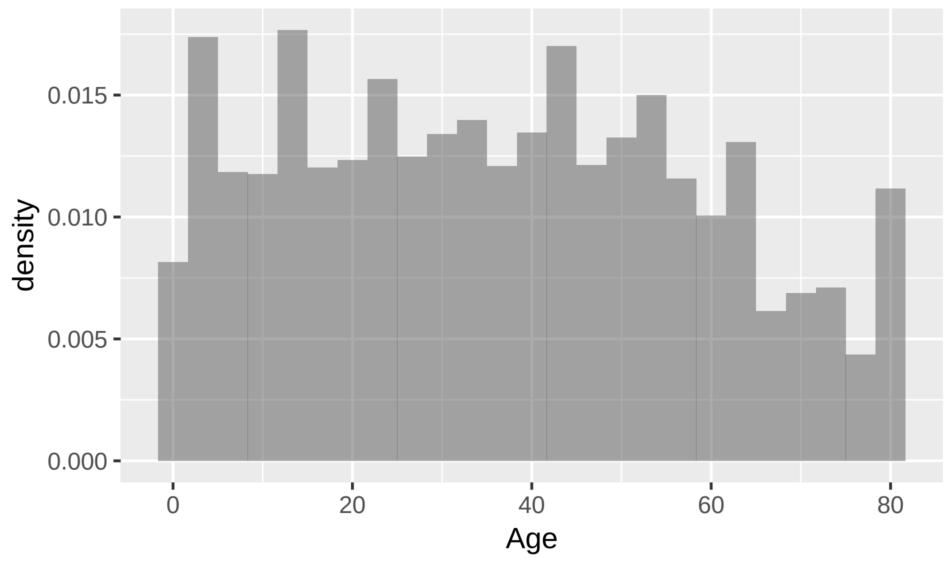
```
## Education
##      8th Grade 9 - 11th Grade    High School   Some College   College Grad
##            451            888           1517           2267           2098
##           <NA>
##           2779
```

Plots

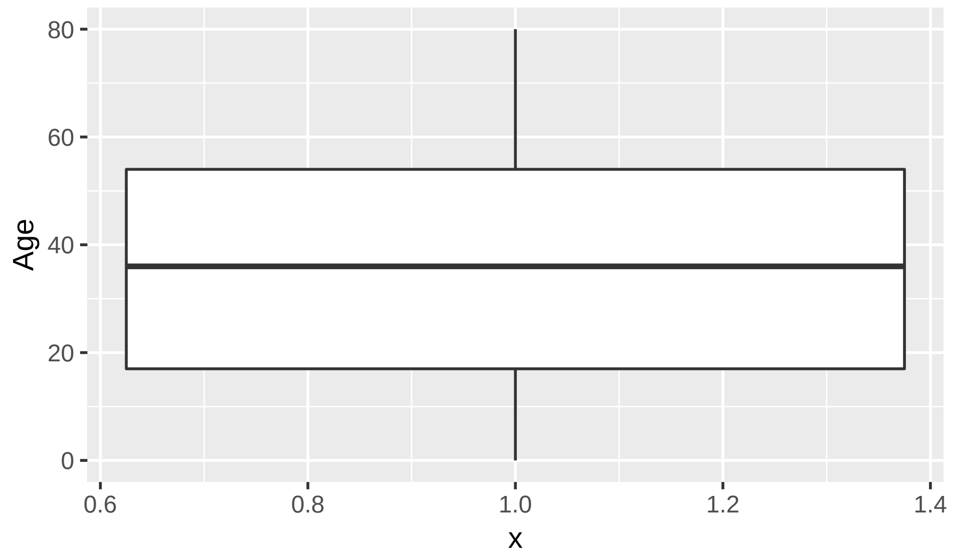
```
#Quantitative Variable
gf_histogram(~ Age, data = NHANES) #count histogram
```



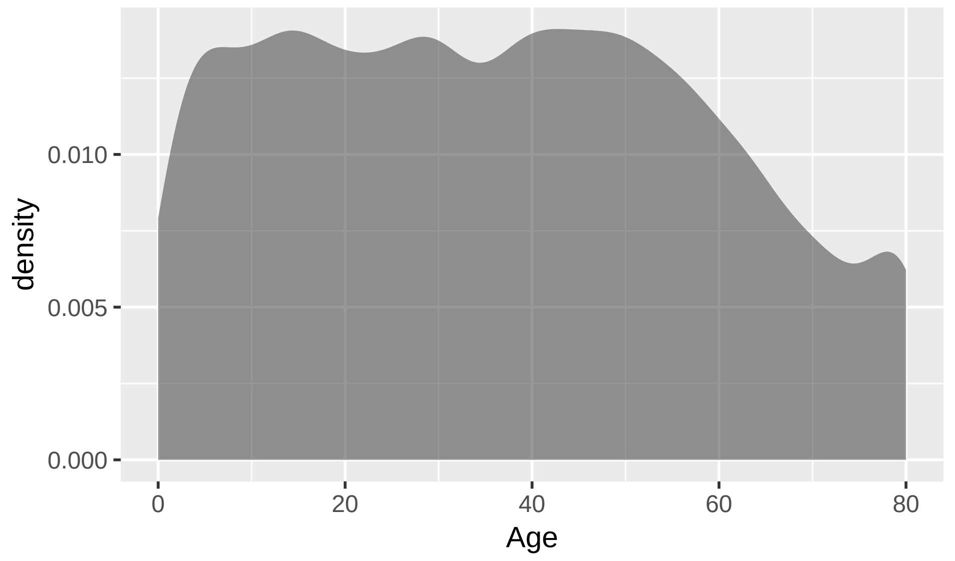
```
gf_dhistogram(~ Age, data = NHANES) #density histogram
```



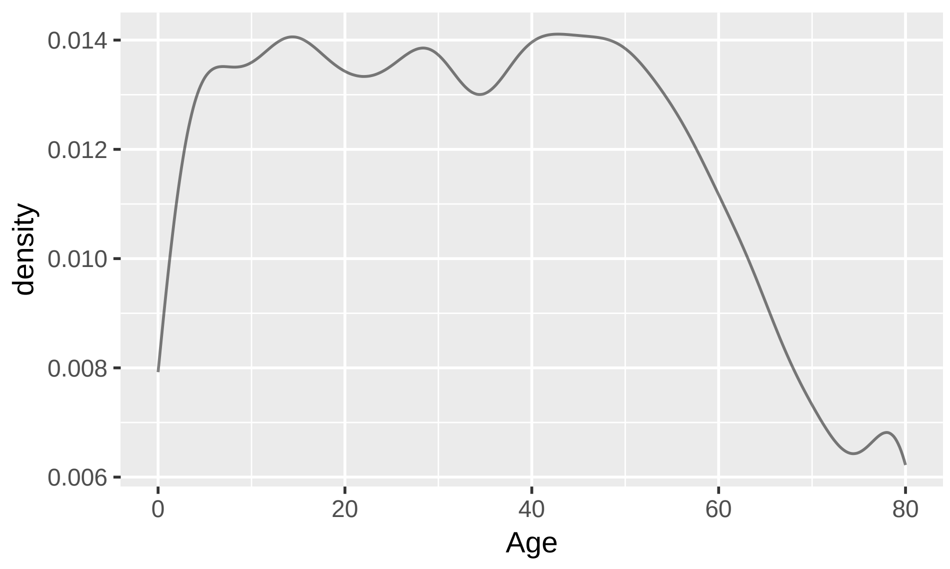
```
#the 1 is here because of how ggplot2 works with boxplots;
#more appropriate for comparing distributions; see below;
#will update when support added for ~ var structure
gf_boxplot(Age ~ 1, data = NHANES)
```



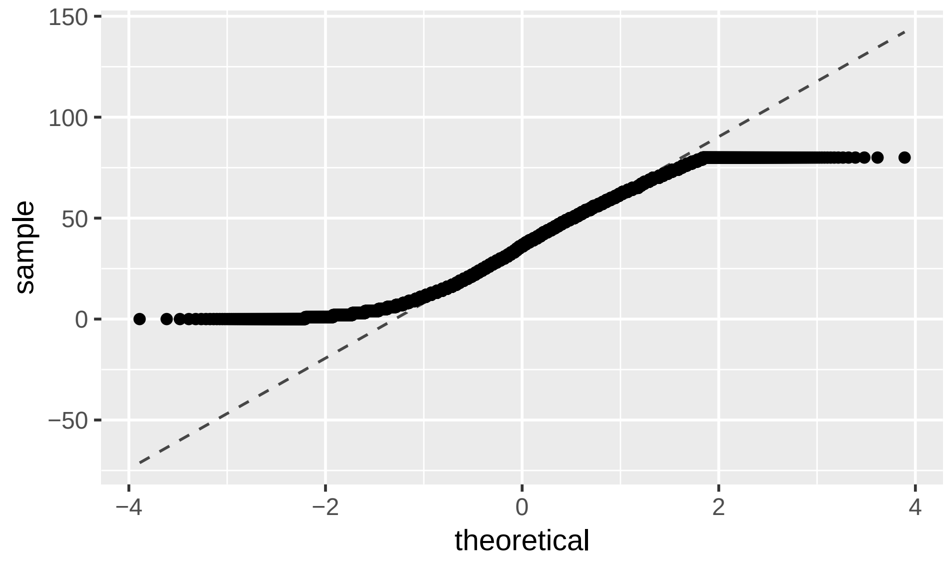
```
gf_density(~ Age, data = NHANES) #shaded densityplot
```



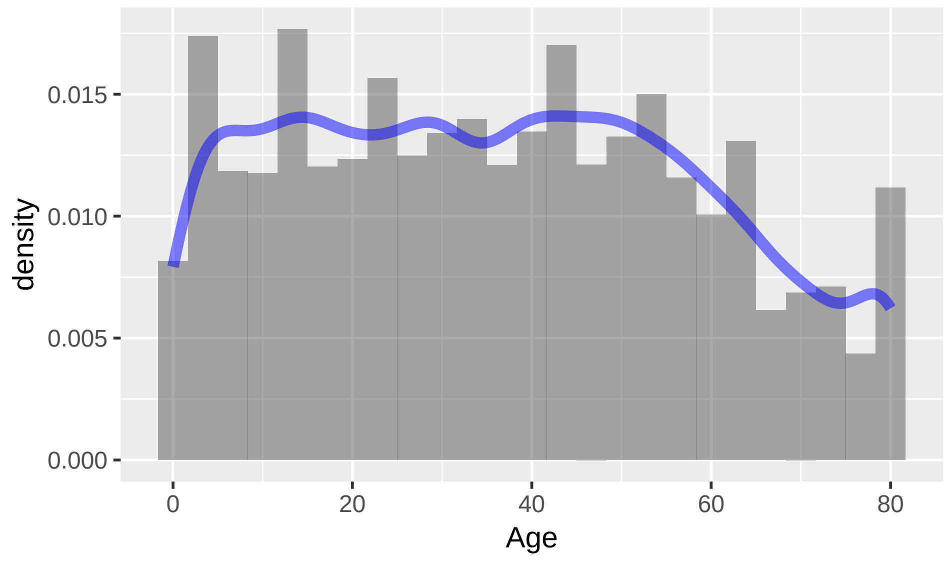
```
gf_dens(~ Age, data = NHANES) #unshaded
```



```
gf_qq(~ Age, data = NHANES) %>% gf_qqline()
```

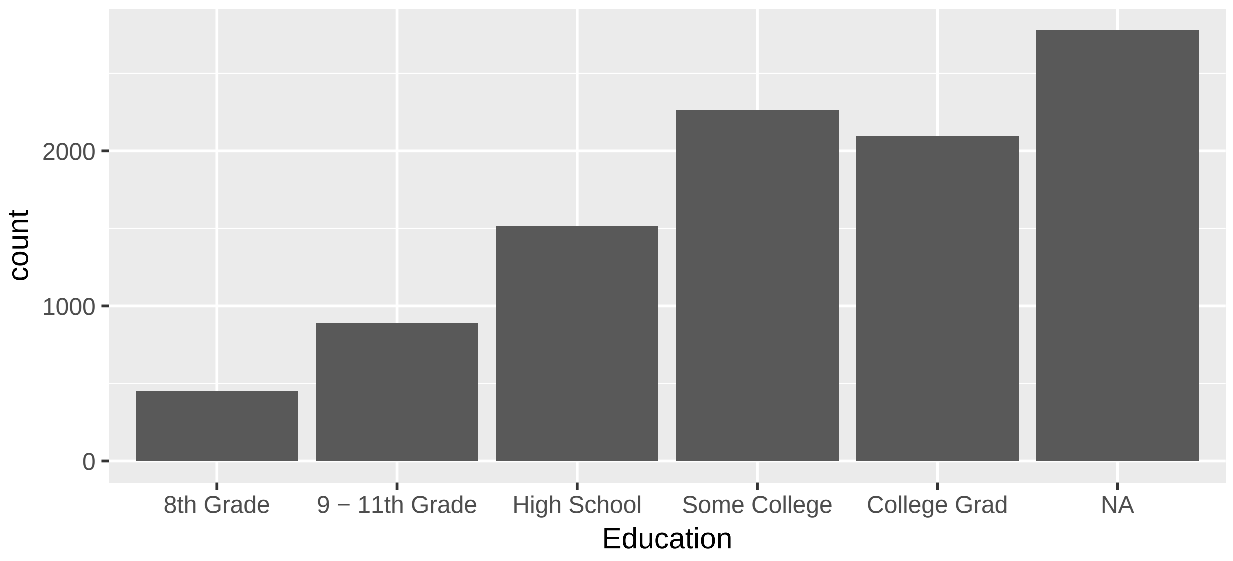


```
#Example overlay
gf_dhistogram(~ Age, data = NHANES) %>%      #the symbol %>% is called a pipe or pipe operator
  gf_dens(size = 2, col = "blue")
```

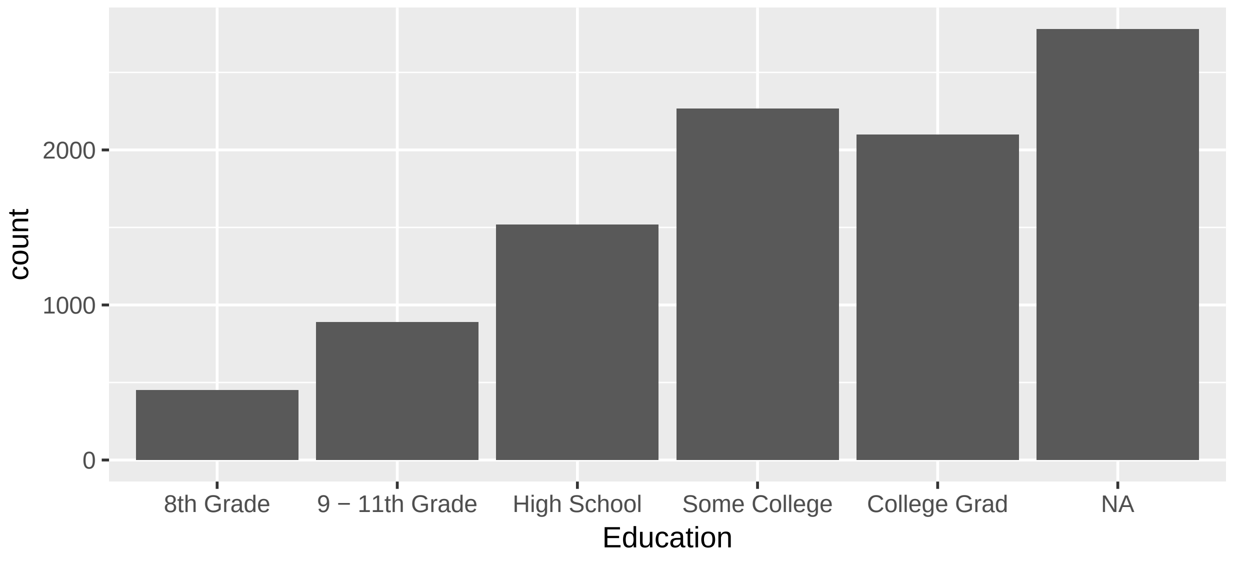


The pipe operator sends output from the first command to the second, reducing what you have to write, so it is fairly convenient.

```
#Categorical Variable
gf_bar(~ Education, data = NHANES)
```



```
gf_counts(~ Education, data = NHANES) #equivalent
```



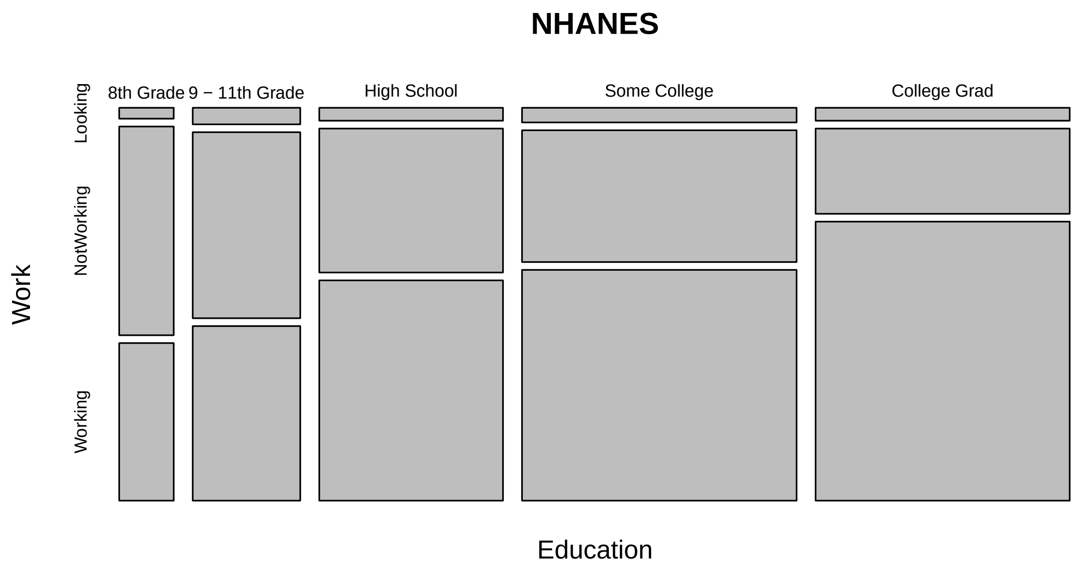
Bivariate Plots and Summaries

As we turn to bivariate plots and summaries, we need to add two more variables to our discussion. So, let's add Weight and Work to go with Age and Education.

```
#Two Categorical Variables
tally(Education ~ Work, data = NHANES)
```

```
##                   Work
## Education         Looking NotWorking Working <NA>
##   8th Grade            13        249     188    1
##   9 - 11th Grade       39        438     411    0
##   High School          52        579     886    0
##   Some College         88        792    1387    0
##   College Grad         72        474    1552    0
##   <NA>                 47        315     189 2228
```

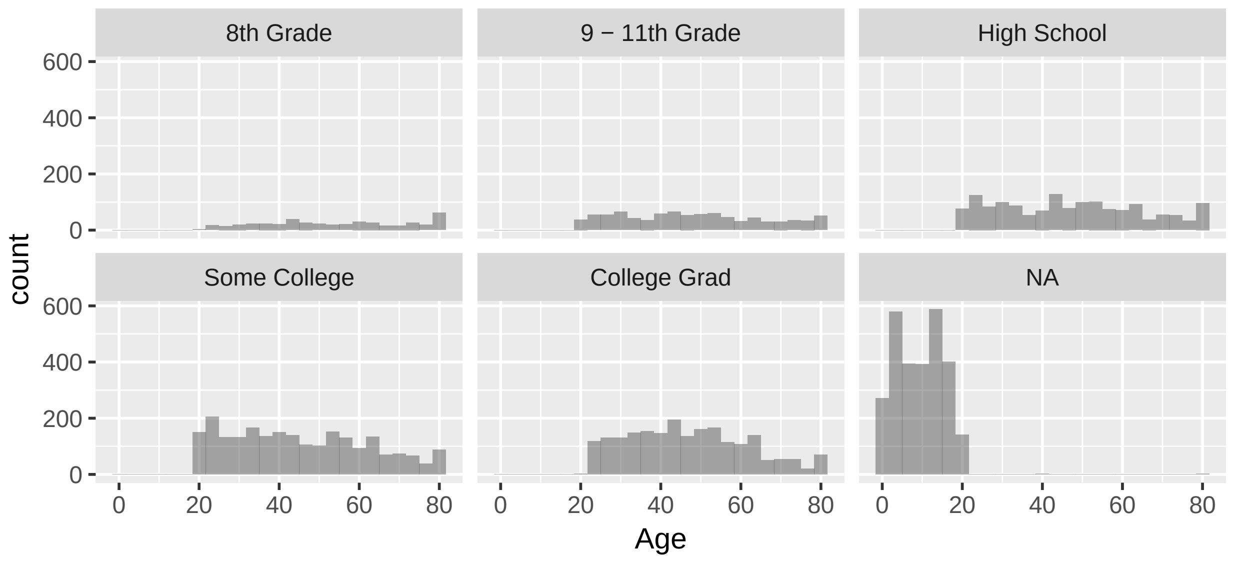
```
mosaicplot(Education ~ Work, data = NHANES)
```



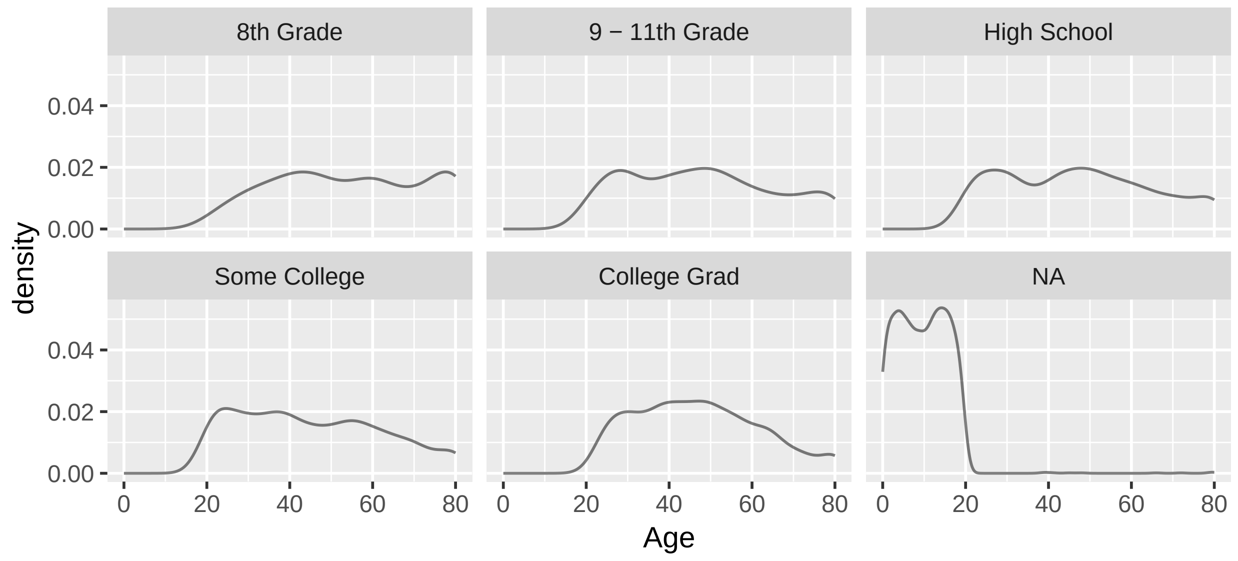
```r
#One Quantitative and One Categorical Variable
favstats(Age ~ Education, data = NHANES)
```

```
##        Education min Q1 median Q3 max    mean      sd    n missing
## 1      8th Grade  20 40     54 70  80 54.2971 17.7449  451       0
## 2 9 - 11th Grade  20 32     47 62  80 48.1363 17.6414  888       0
## 3    High School  20 31     47 61  80 47.5339 17.7744 1517       0
## 4   Some College  20 30     43 58  80 45.2730 17.2639 2267       0
## 5   College Grad  20 35     46 57  80 47.0157 14.9185 2098       0
```

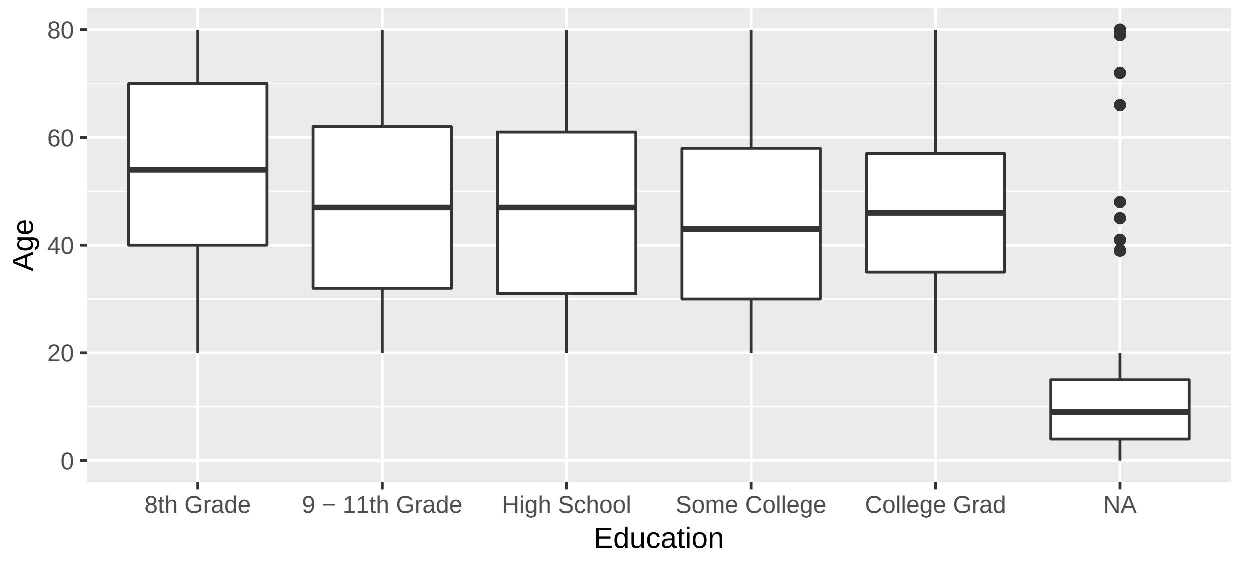
```r
gf_histogram(~ Age | Education, data = NHANES)
```



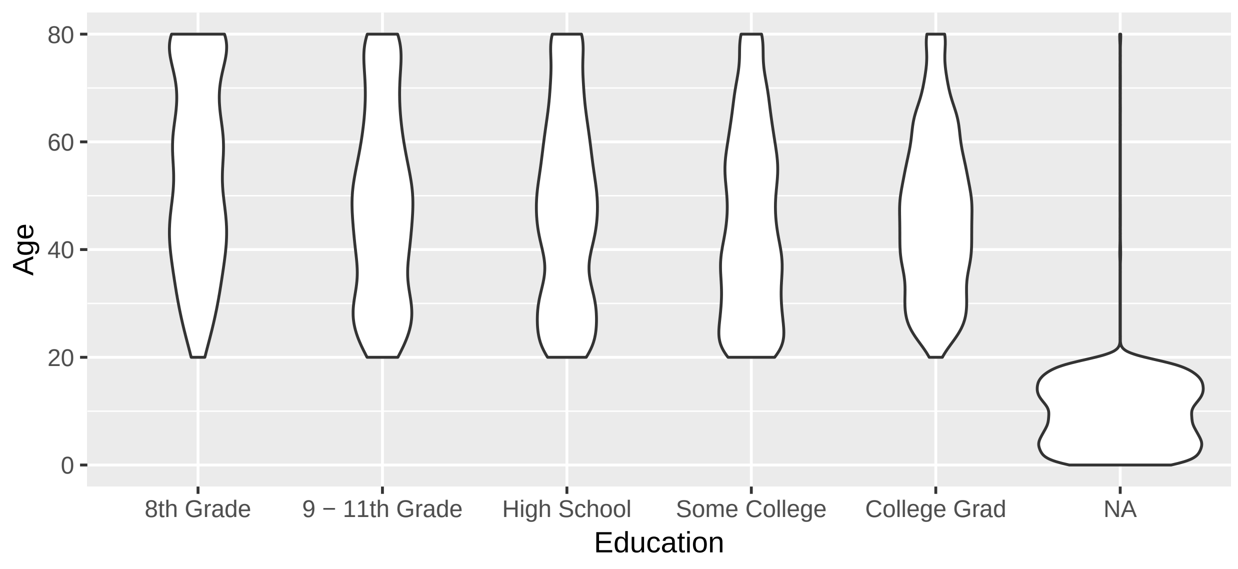
```r
gf_dens(~ Age | Education, data = NHANES)
```



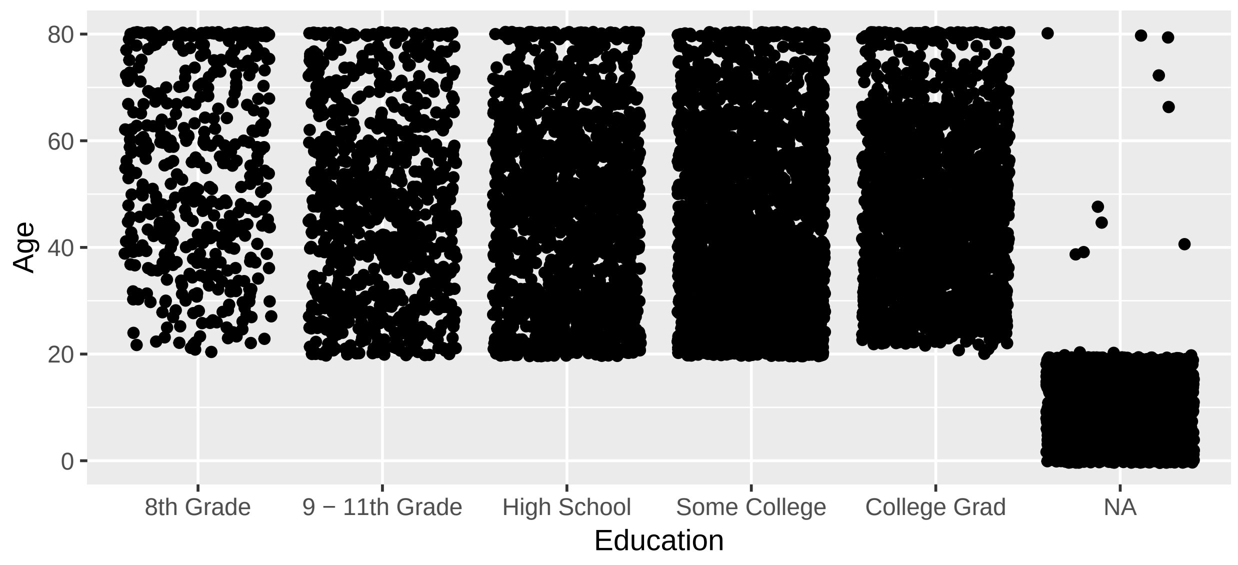
```
gf_boxplot(Age ~ Education, data = NHANES)
```



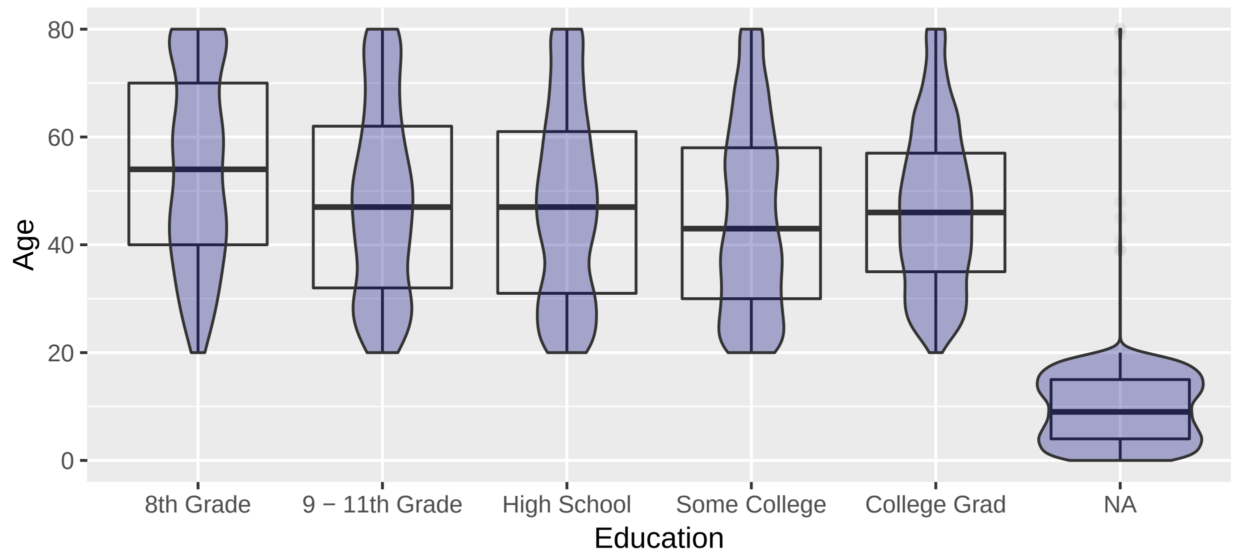
```
gf_violin(Age ~ Education, data = NHANES)
```



```r
gf_jitter(Age ~ Education, data = NHANES) #gf_point can be used if less overlap
```



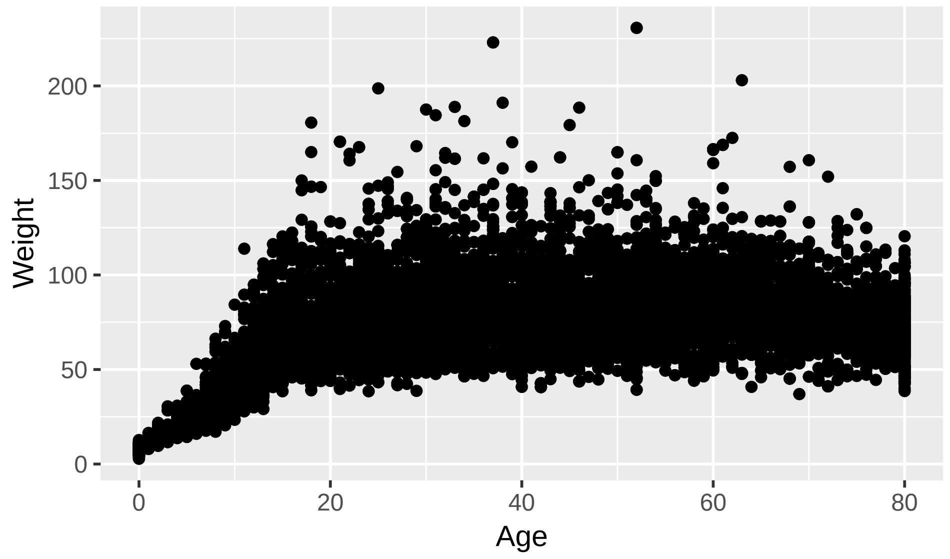
```r
#Example Overlay
gf_boxplot(Age ~ Education, data = NHANES, alpha = 0.05) %>%
  gf_violin(alpha = 0.3, fill = "navy")
```



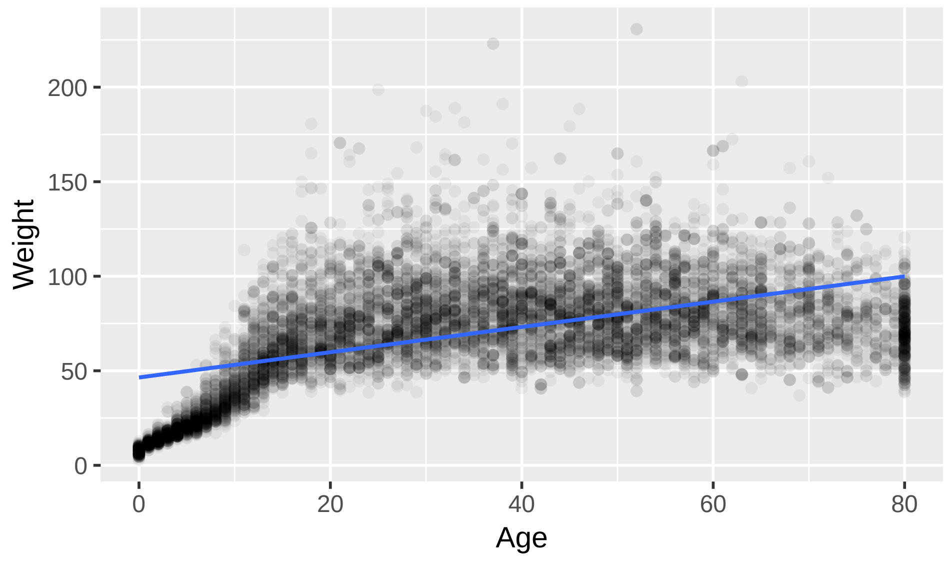
```
#Two Quantitative Variables
with(NHANES, cor(Weight, Age, use = "pairwise.complete.obs"))
```

```
## [1] 0.513243
```

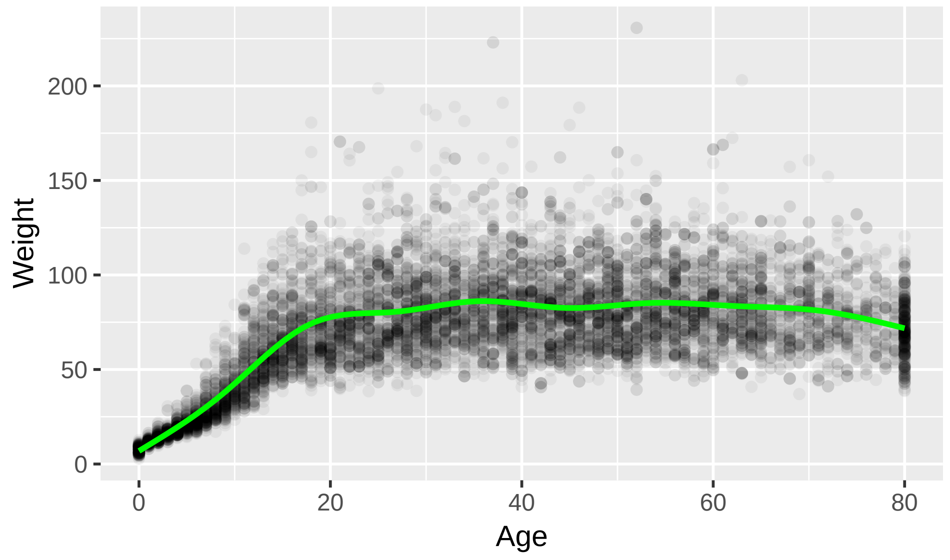
```
gf_point(Weight ~ Age, data = NHANES)
```



```
gf_point(Weight ~ Age, data = NHANES, alpha = 0.05) %>%
  gf_lm()
```



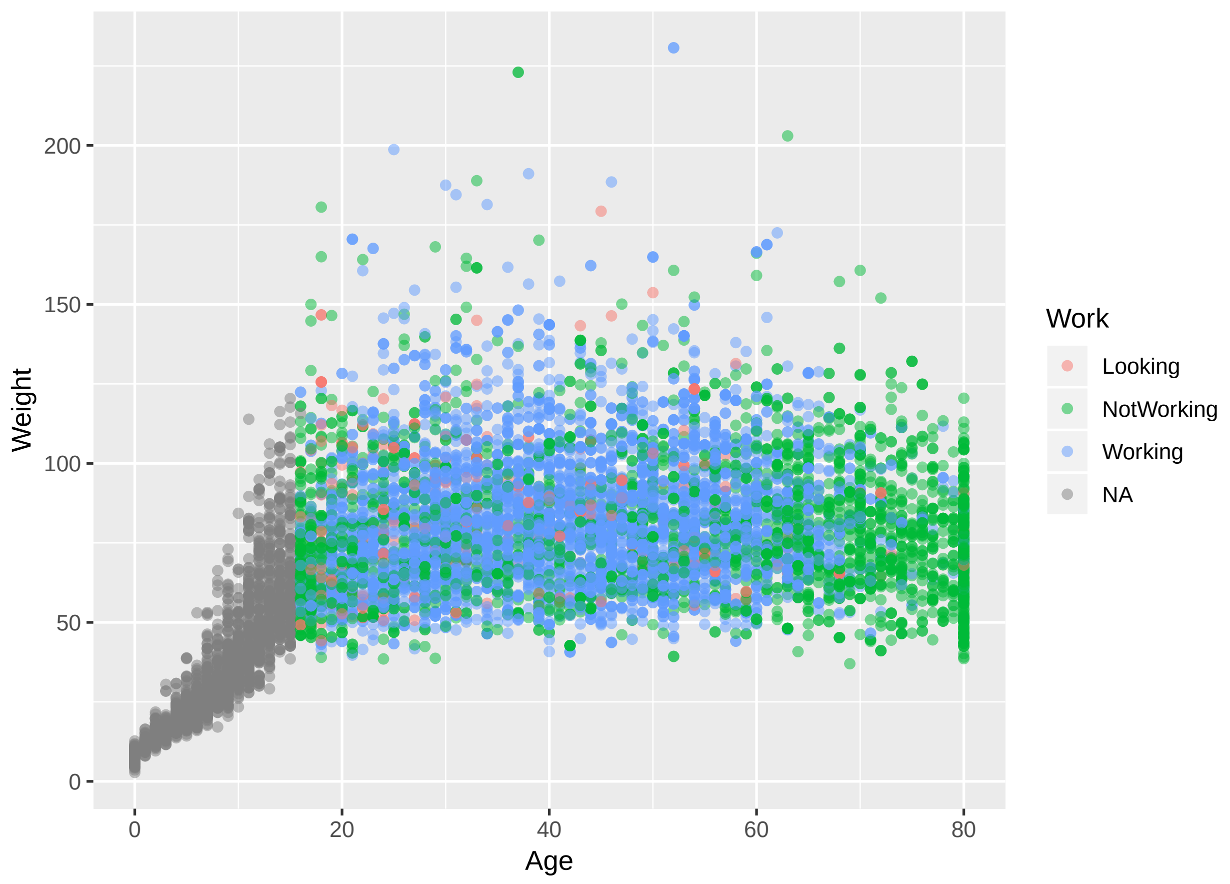
```r
gf_point(Weight ~ Age, data = NHANES, alpha = 0.05) %>%
  gf_smooth(se = FALSE, col = "green")
```



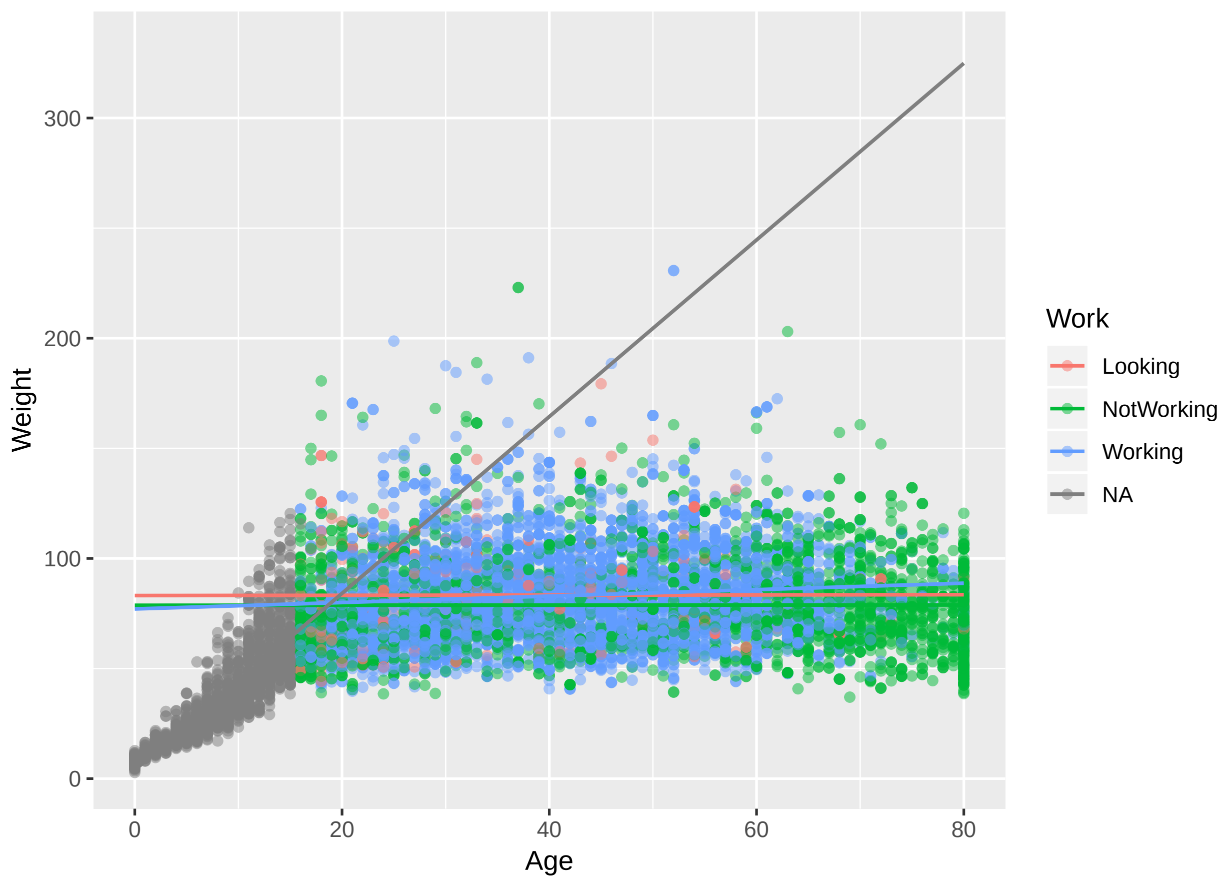
Multivariate Plots

A variety of plots are available that allow you to explore relationships between more than 2 variables. This is just a set of examples. We'll add one more quantitative variable here, HomeRooms.

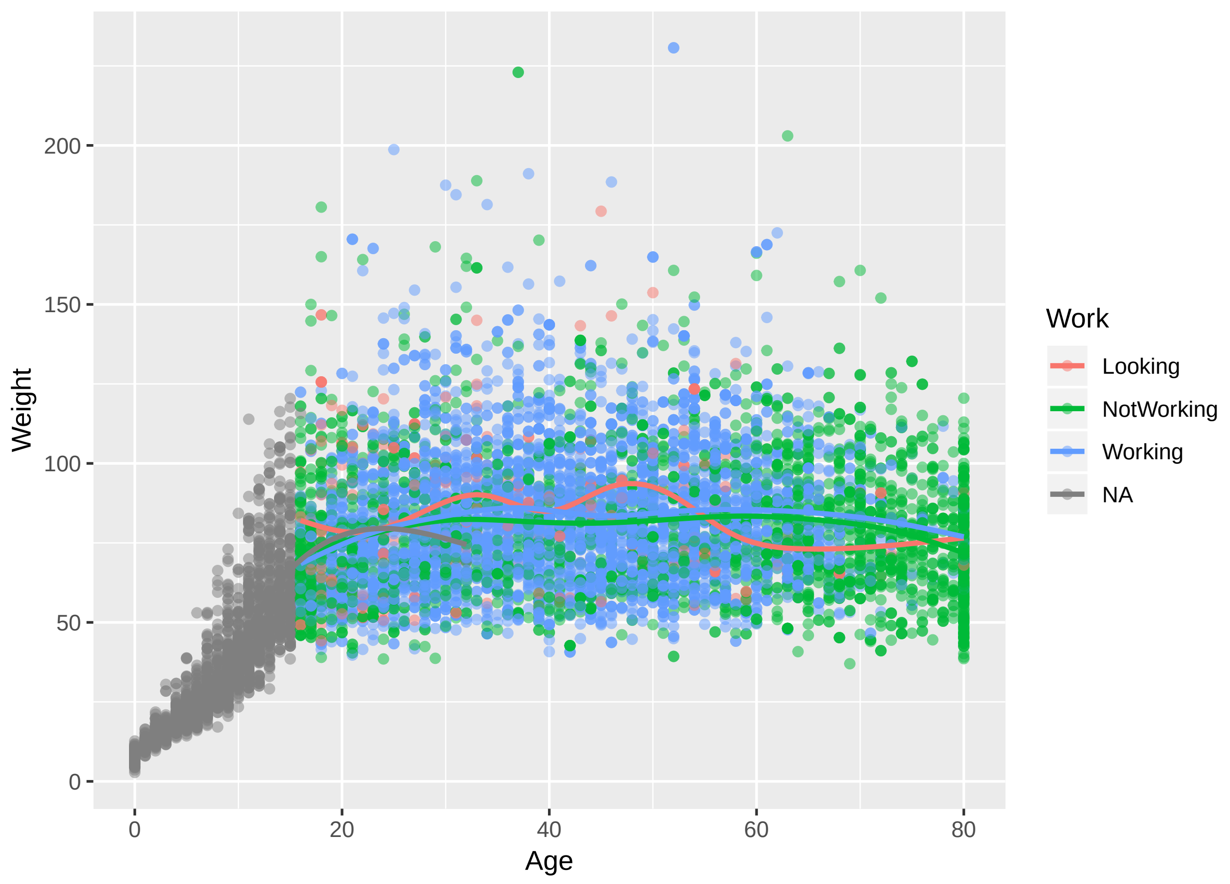
```r
gf_point(Weight ~ Age, data = NHANES, alpha = 0.5, color = ~ Work)
```



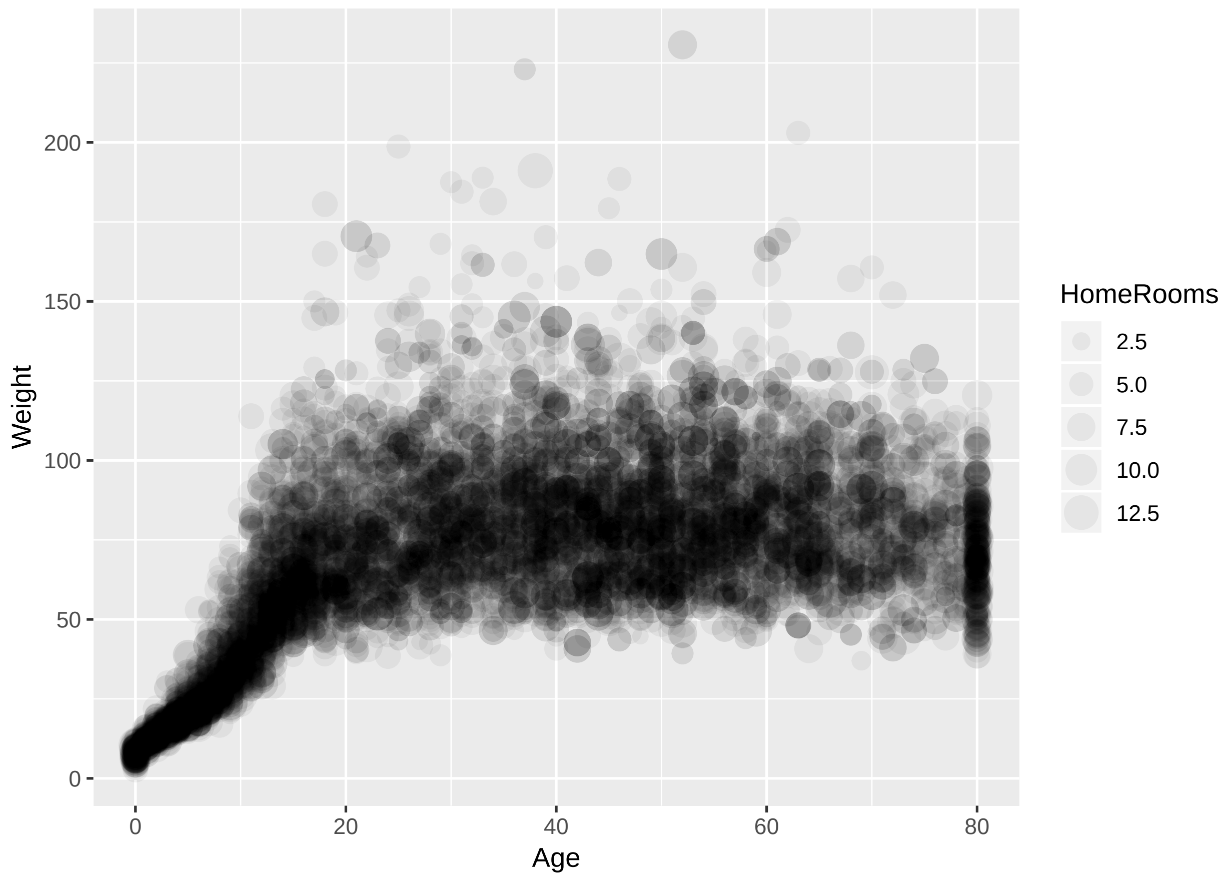
```r
gf_point(Weight ~ Age, data = NHANES, alpha = 0.5, color = ~ Work) %>%
  gf_lm()
```



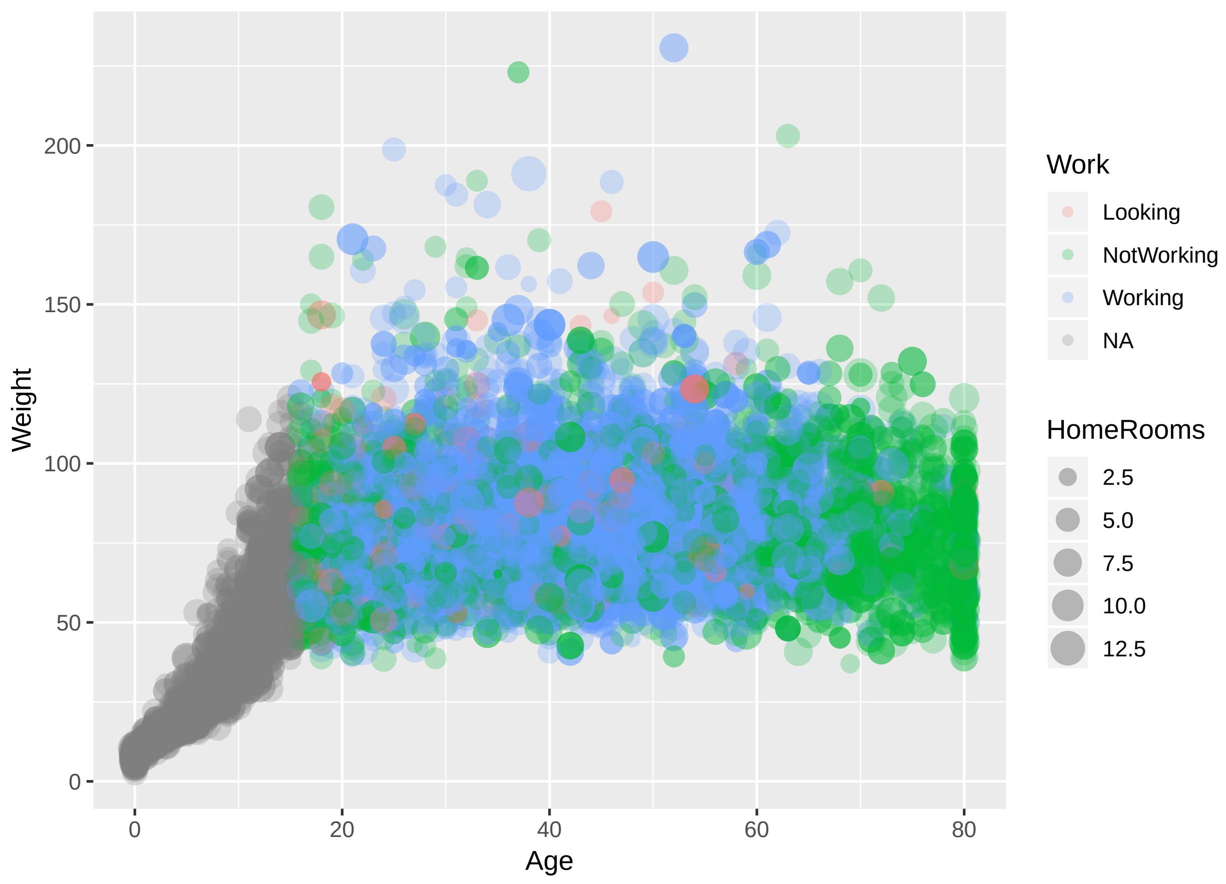
```r
gf_point(Weight ~ Age, data = NHANES, alpha = 0.5, color = ~ Work) %>%
  gf_smooth(se = FALSE)
```



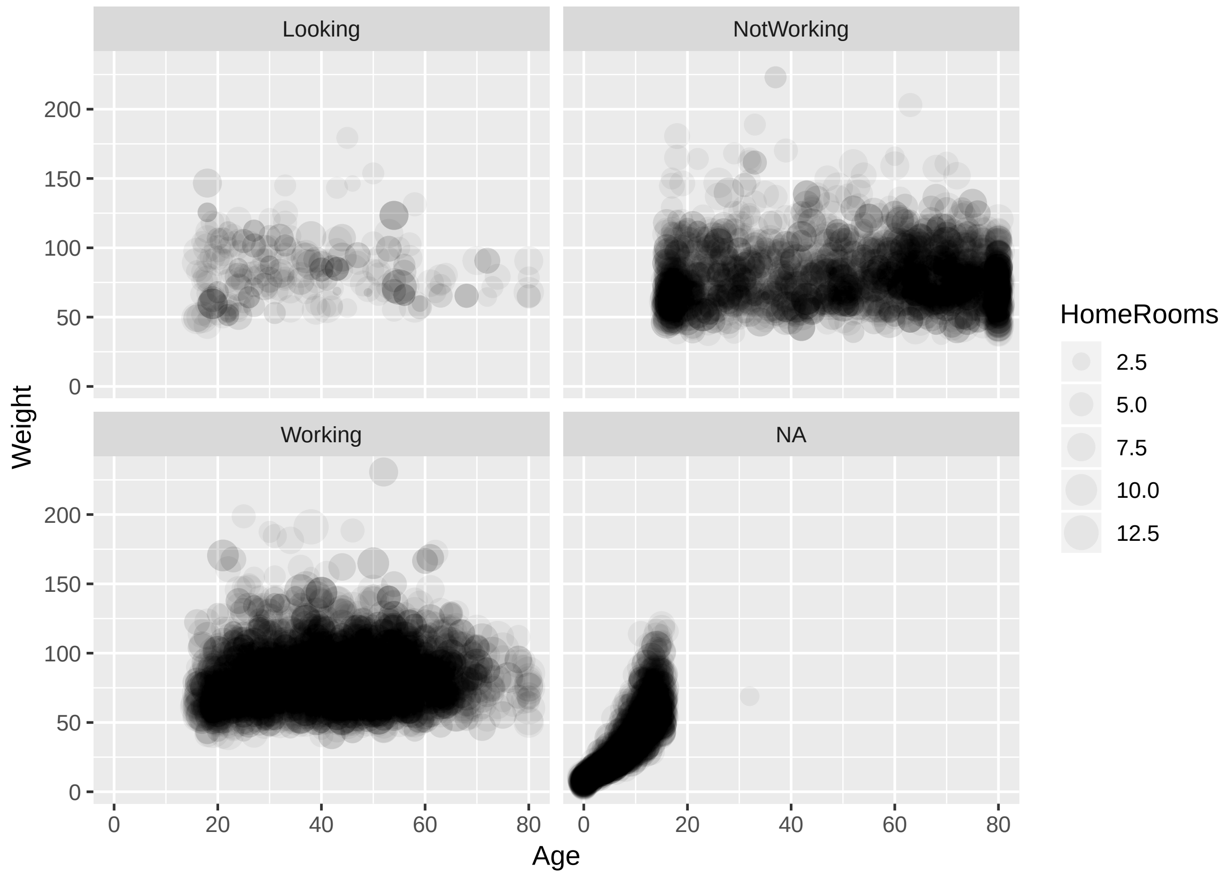
```
gf_point(Weight ~ Age, data = NHANES, alpha = 0.05, size = ~ HomeRooms)
```



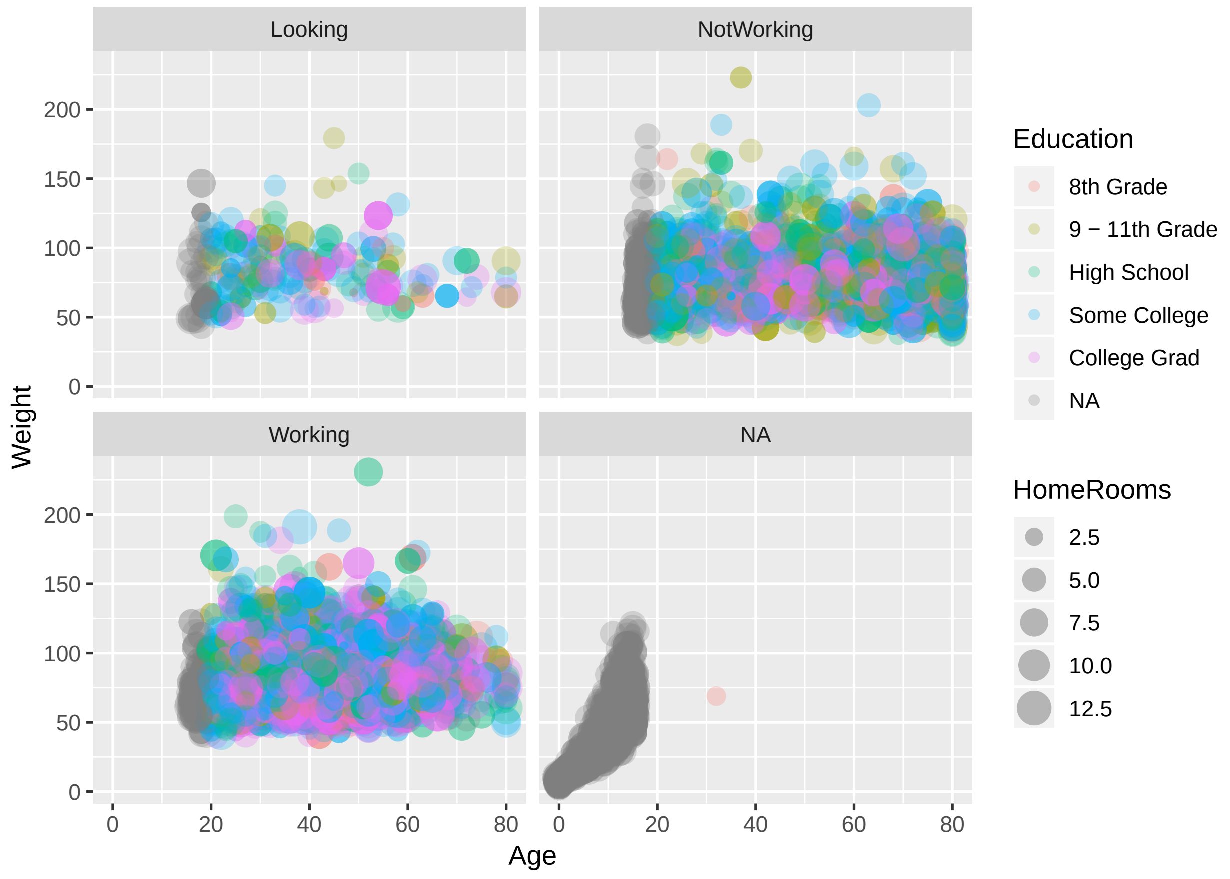
```
gf_point(Weight ~ Age, data = NHANES, alpha = 0.25, size = ~ HomeRooms, color = ~ Work)
```



```r
gf_point(Weight ~ Age | Work, data = NHANES, alpha = 0.05, size = ~ HomeRooms)
```



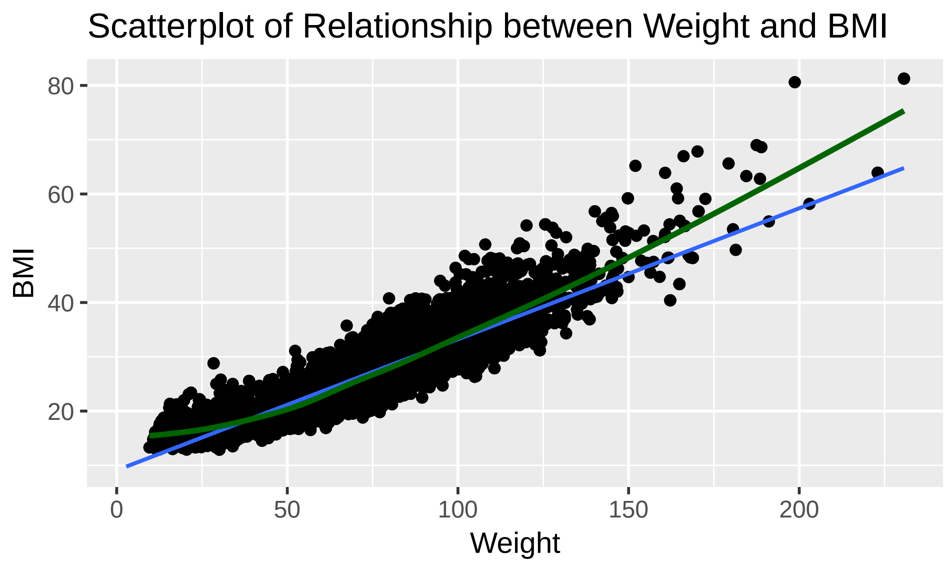
```
gf_point(Weight ~ Age | Work, data = NHANES, alpha = 0.25, size = ~ HomeRooms, color = ~ Education)
```



Fitting Linear Models and Related Plots

One last bit of code that you should have some familiarity with is the code associated with fitting and assessing a linear model (plots and output). Here, we examine the relationship between Weight and BMI, with BMI as the response.

```r
gf_point(BMI ~ Weight, data = NHANES) %>%
  gf_lm() %>%
  gf_smooth(se = FALSE, color = "dark green") %>%
  gf_labs(title = "Scatterplot of Relationship between Weight and BMI")
```



The scatterplot seems to show an expected positive relationship, though there may be some curvature in the pattern. A smoother can help illustrate this, so it was added to the plot. A title was also added to remind you about how to do that. X and Y axis labels can also be changed with that command.

Now, suppose we want to fit an SLR here. We fit and save a model, can examine the output, and related plots.

```
mod <- lm(BMI ~ Weight, data = NHANES)
msummary(mod) #summary(mod) provides additional info, but isn't always needed
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.09326    0.09151    99.4   <2e-16 ***
## Weight       0.24138    0.00118   205.3   <2e-16 ***
##
## Residual standard error: 3.18 on 9632 degrees of freedom
##   (366 observations deleted due to missingness)
## Multiple R-squared:  0.814,  Adjusted R-squared:  0.814
## F-statistic: 4.21e+04 on 1 and 9632 DF,  p-value: <2e-16
```

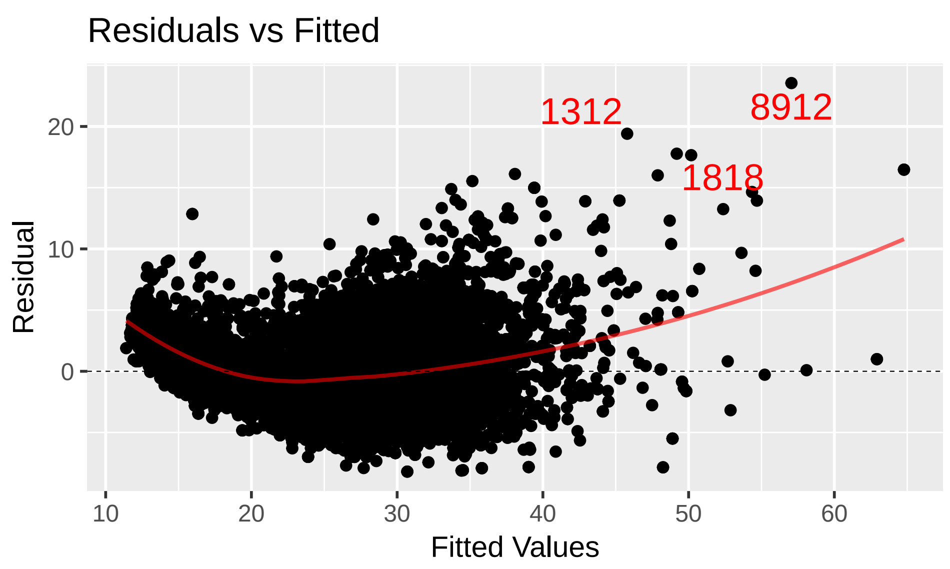
```
confint(mod, conf.level  = 0.95) #obtain CIs for model parameters
```

```
##                2.5 %   97.5 %
## (Intercept) 8.913869 9.272644
## Weight      0.239071 0.243681
```

The most commonly used assessment plots are accessed with mplot (or just plot if you want the base R versions).

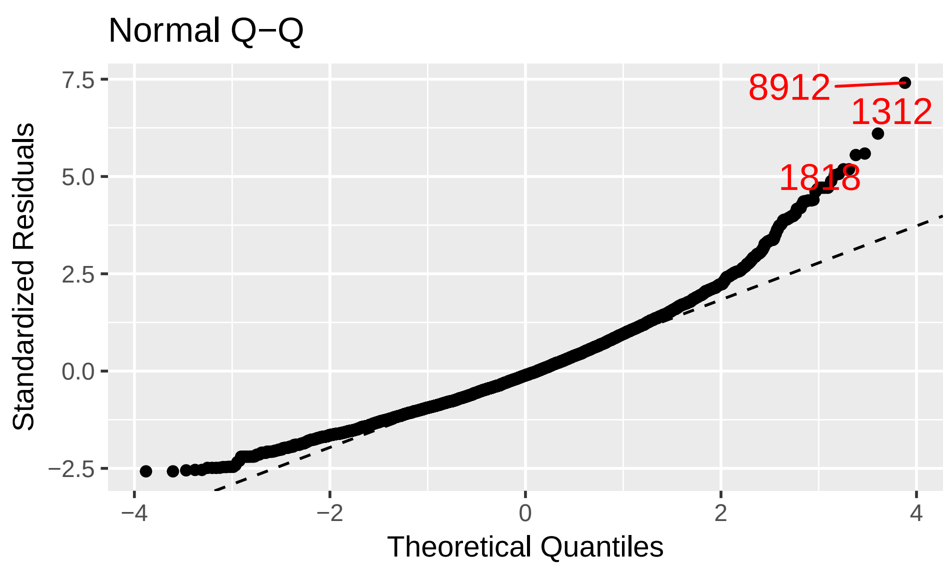
```
mplot(mod, which = 1)
```

```
## [[1]]
```



```r
mplot(mod, which = 2)
```

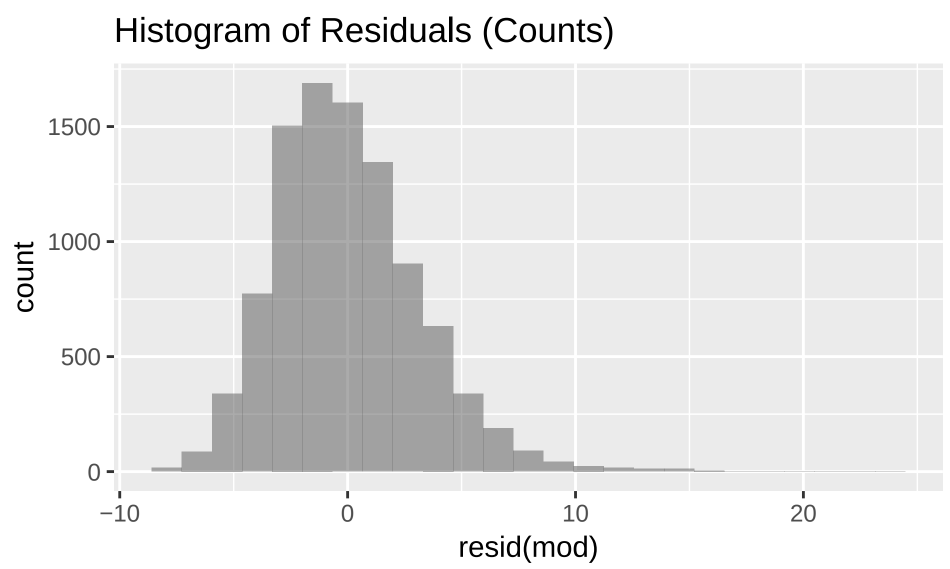
```
## [[1]]
```



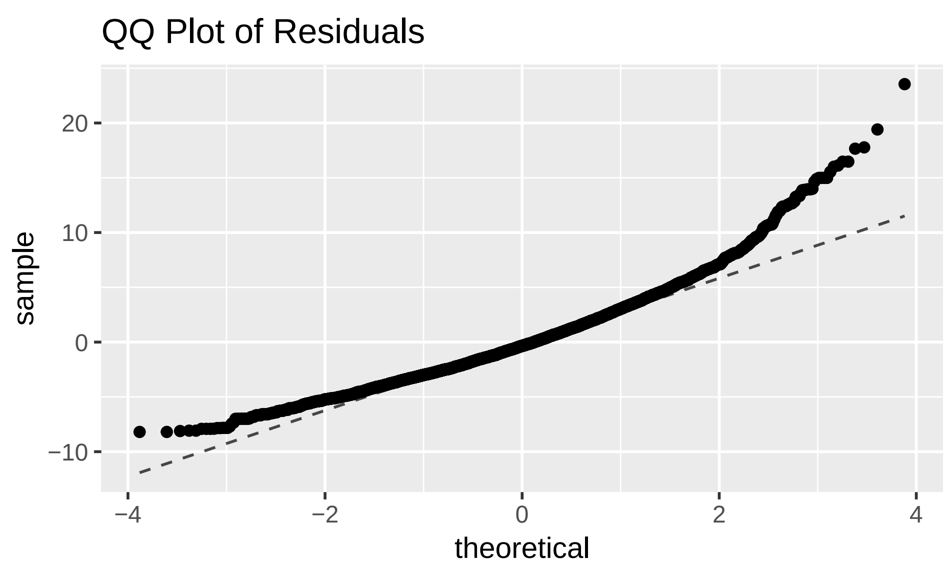
This clearly shows the curvature in the relationship, so we should make appropriate transformations and perhaps try other techniques to understand the relationship rather than relying on this SLR.

You could also make these plots directly (with some enhanced titles) as follows:

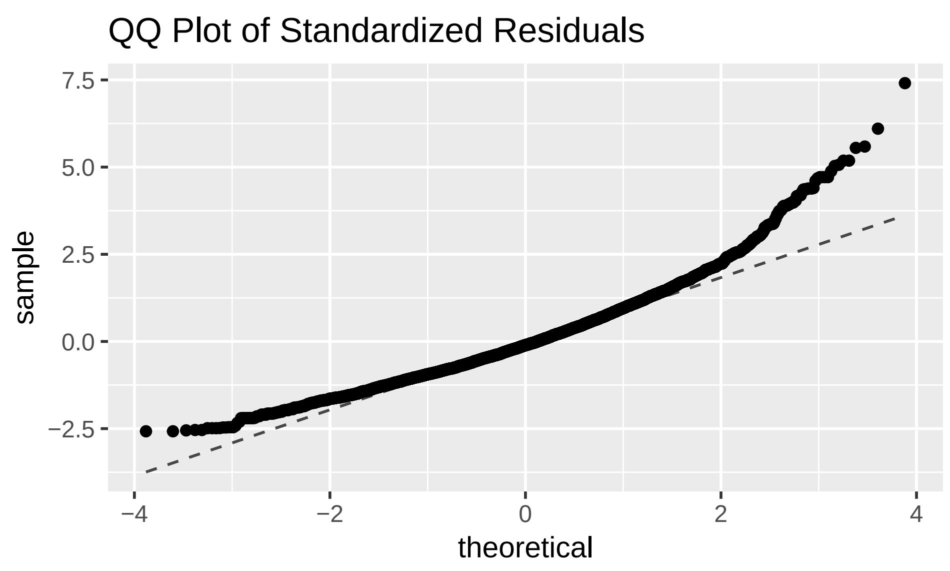
```r
gf_histogram(~ resid(mod)) %>% gf_labs(title = "Histogram of Residuals (Counts)")
```



```r
gf_qq(~ resid(mod)) %>% gf_qqline() %>%
  gf_labs(title = "QQ Plot of Residuals")
```



```r
gf_qq(~ rstandard(mod)) %>% gf_qqline() %>%
  gf_labs(title = "QQ Plot of Standardized Residuals")
```



Finally, you may have some experience with making predictions. We will get more practice with this in class.

```
predictFun <- makeFun(mod)
predictFun(Weight = 130) #predict mean BMI for a Weight of 130
```

```
##       1
## 40.4721
```

We will explore accessing residuals and fitted values with a new package *broom* in class, but those are also accessible this way:

```
head(resid(mod))
```

```
##        1        2        3        4        5        6
## 2.030486 2.030486 2.030486 2.103353 0.549449 0.533740
```

```
resid(mod)[100] #access 100th residual, due to missing data, is for observation 101
```

```
##     101
## 1.89534
```

```
head(fitted(mod))
```

```
##       1       2       3       4       5       6
## 30.1895 30.1895 30.1895 13.1966 30.0206 16.2863
```

```
fitted(mod)[100] #access 100th fitted value, due to missing data, is for observation 101
```

```
##     101
## 32.1447
```

Data Wrangling

The Datacamp course on introduction to the tidyverse covers a lot of this material. You should be comfortable with at least the four verbs - rename, mutate, select, and filter. You will also have exposure to group_by and summarize, as well as a little bit with joins, over the course of the class. We will have practice with these via Datacamp and in-class activities, so very brief examples are provided here.

```
NHANES <- mutate(NHANES, BMI.Sq = BMI^2)
NHANES2 <- NHANES %>% filter( Age < 35) %>%
  dplyr::select(Age, Weight, Education, BMI, HomeRooms)
#select can sometimes have interference from select functions in other packages
#the addition of dplyr:: forces select to be pulled from dplyr
NHANES2 <- rename(NHANES2, Education.Level = Education)
```

Next, we examine code relevant to material from our course using Stat2.

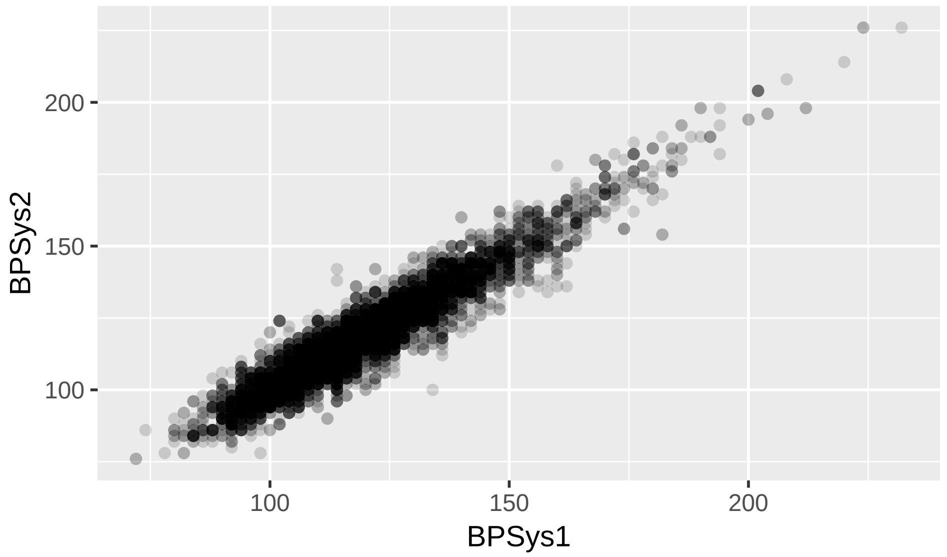
Simple Linear Regression (Chapters 1-2)

The material in 1.1-1.3 and 2.1 had relevant code covered in the review above.

Outliers (1.4)

Here, we start to look at the broom package, which has some nice features. We can access residuals and fitted values from our models already, but broom allows us to add those, as well as some other useful statistics, to our data set for easier access. Recall we have a model predicting BMI using Weight, called mod, though the pattern is curved, so it's not really appropriate for linear regression without some adjustments. Let's examine a different relationship, one between BPSys1 and BPSys2, which are systolic blood pressure readings. These are expected to be highly correlated.

```
gf_point(BPSys2 ~ BPSys1, data = NHANES, alpha = 0.15)
```



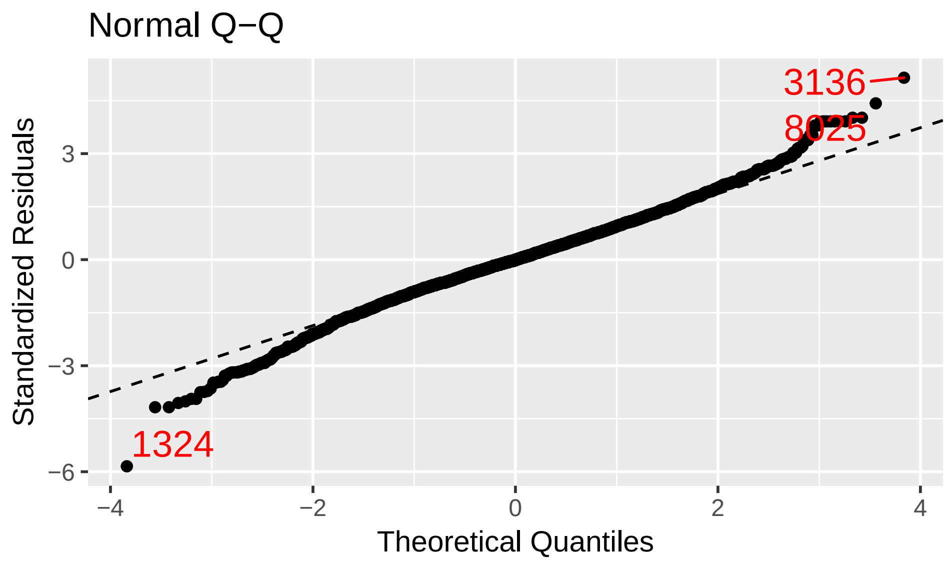
Let's look at the QQplot of residuals and examine outliers via *augment* in broom.

```
mod <- lm(BPSys2 ~ BPSys1, data = NHANES)
msummary(mod)
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.29698    0.42468    17.2   <2e-16 ***
## BPSys1       0.93252    0.00353   264.4   <2e-16 ***
##
## Residual standard error: 5.52 on 8059 degrees of freedom
##   (1939 observations deleted due to missingness)
## Multiple R-squared:  0.897,  Adjusted R-squared:  0.897
## F-statistic: 6.99e+04 on 1 and 8059 DF,  p-value: <2e-16
```

```
mplot(mod, which = 2)
```

```
## [[1]]
```



```
modData <- augment(mod)
glimpse(modData)
```

```
## Observations: 8,061
## Variables: 10
## $ .rownames  <chr> "1", "2", "3", "5", "6", "7", "8", "9", "10", "11",...
## $ BPSys2     <int> 114, 114, 114, 108, 84, 108, 118, 118, 118, 108, 10...
## $ BPSys1     <int> 114, 114, 114, 118, 84, 114, 106, 106, 106, 124, 10...
## $ .fitted    <dbl> 113.6045, 113.6045, 113.6045, 117.3346, 85.6289, 11...
## $ .se.fit    <dbl> 0.0640875, 0.0640875, 0.0640875, 0.0615923, 0.13836...
## $ .resid     <dbl> 0.395459, 0.395459, 0.395459, -9.334631, -1.628867,...
## $ .hat       <dbl> 0.000134896, 0.000134896, 0.000134896, 0.000124596,...
## $ .sigma     <dbl> 5.51825, 5.51825, 5.51825, 5.51727, 5.51822, 5.5178...
## $ .cooksd    <dbl> 3.46529e-07, 3.46529e-07, 3.46529e-07, 1.78331e-04,...
## $ .std.resid <dbl> 0.0716731, 0.0716731, 0.0716731, -1.6918037, -0.295...
```

Note this gives us the predictor(s), response, and various statistics in their own data set! It also removed NAs, but we still have the rownames (observation numbers), so we can identify observations in the original data set with large standardized residuals, for example.

```
as.numeric(with(modData, which(abs(.std.resid) > 3)))
```

```
##  [1]  182  615  616  747  748 1324 1523 1524 1892 1975 2376 2377 2429 2581
## [15] 2736 2786 2787 2788 2789 2885 2886 3136 3612 3753 3754 4015 4059 4060
## [29] 4237 4238 4239 4645 4646 5227 5288 5496 5497 6092 6142 6143 6144 6145
## [43] 6146 6147 6267 6382 6426 6531 6532 6533 6607 6608 6609 6857 6858 6859
## [57] 6860 6861 7253 7395 7396 7685 7686 7850 7851 8025
```

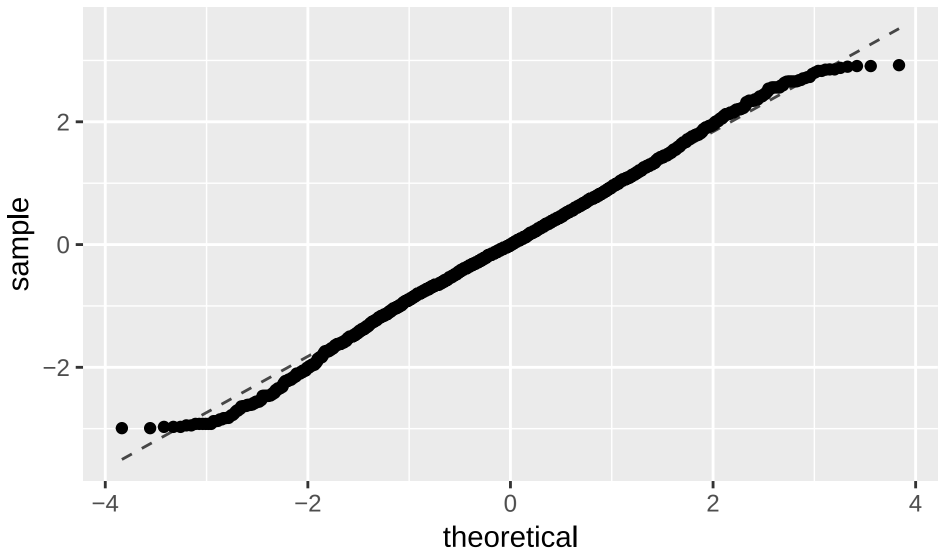
You could do this without the modData as well.

```
as.numeric(which(abs(rstandard(mod)) > 3))
```

```
##  [1]  182  615  616  747  748 1324 1523 1524 1892 1975 2376 2377 2429 2581
## [15] 2736 2786 2787 2788 2789 2885 2886 3136 3612 3753 3754 4015 4059 4060
## [29] 4237 4238 4239 4645 4646 5227 5288 5496 5497 6092 6142 6143 6144 6145
## [43] 6146 6147 6267 6382 6426 6531 6532 6533 6607 6608 6609 6857 6858 6859
## [57] 6860 6861 7253 7395 7396 7685 7686 7850 7851 8025
```

Let's demonstrate how to use modData to filter out unusual points based on standardized residuals with absolute value greater than 3 and examine the updated QQplot.

```
modData %>%
  filter(abs(.std.resid) <= 3) %>%
  gf_qq(~ .std.resid) %>% gf_qqline()
```



This helps us to assess the impact of these points on the regression and underlying conditions.

Chapter 2 - More on Simple Linear Regression

This chapter contains a little bit more information on linear regression, so we continue to examine the relationship we had between the two blood pressure measurements.

We learn about Analysis of Variance as used to break down variation in the response in the context of regression. Typically this is viewed via an ANOVA table, which is easily accessed from the model object.

```
aov(mod)
```

```
## Call:
##    aov(formula = mod)
##
## Terms:
##                  BPSys1 Residuals
## Sum of Squares  2128779    245375
## Deg. of Freedom       1      8059
##
## Residual standard error: 5.51791
## Estimated effects may be unbalanced
## 1939 observations deleted due to missingness
```

```r
anova(mod)#preferred command
```

```
## Analysis of Variance Table
##
## Response: BPSys2
##             Df  Sum Sq Mean Sq F value Pr(>F)
## BPSys1       1 2128779 2128779   69917 <2e-16 ***
## Residuals 8059  245375      30
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Sometimes you don't need the whole table, and might want specific parts of the output, such as the F statistic, or the model coefficients. There are easy ways to access those as well from the model object. Here, we can see what is accessible in the *mod* object.

```r
names(mod)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "na.action"     "xlevels"       "call"          "terms"
## [13] "model"
```

We can also access values from the anova applied to the model object.

```r
names(anova(mod))
```

```
## [1] "Df"      "Sum Sq"  "Mean Sq" "F value" "Pr(>F)"
```

So, to pull out the model coefficients, F statistic, MSE, and R-squared respectively, we use:

```r
mod$coefficients
```

```
## (Intercept)      BPSys1
##    7.296979    0.932522
```

```r
anova(mod)$F
```

```
## [1] 69916.9      NA
```

```r
anova(mod)$Mean[2]
```

```
## [1] 30.4473
```

```r
rsquared(mod)
```

```
## [1] 0.896648
```

Note that the F statistic is returned as a vector - the first entry matches the entry from the table. The Mean Square error is also returned as part of a vector, but it was in the second row of the table, so this is how to access it directly. You may also want to be familiar with accessing the Dfs for the degrees of freedom.

This chapter shows the test for correlation between two quantitative variables, though this is not really used at any other point in the course. Here, we demonstrate this on the blood pressure measurements. Strictly speaking, the method option is not needed because pearson is the default.

```r
with(NHANES, cor.test(BPSys1, BPSys2), method = c("pearson"))
```

```
##
##  Pearson's product-moment correlation
##
## data:  x and y
## t = 264.4, df = 8059, p-value <2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.944611 0.949125
## sample estimates:
##      cor
## 0.946915
```

Finally, the chapter concludes with confidence intervals for the mean response and prediction intervals for an individual response. In order to access these, you need to create a new data frame that you want the predictions for. Basically, you need some values for the predictor variable. The predict function is used after that for both types of intervals, you just need to set the int option appropriately.

```
new.data <- data.frame(BPSys1 = c(109, 115, 122))

predict(mod, new.data, int = "confidence", level = 0.90) #for CIs for mean response
```

```
##       fit     lwr     upr
## 1 108.942 108.825 109.059
## 2 114.537 114.433 114.641
## 3 121.065 120.962 121.167
```

```
predict(mod, new.data, int = "prediction", level = 0.90) #for prediction intervals
```

```
##       fit     lwr     upr
## 1 108.942  99.864 118.020
## 2 114.537 105.459 123.615
## 3 121.065 111.987 130.142
```

Multiple Linear Regression (Chapter 3)

Now we enter into multiple linear regression. The good news is that you don't really need many new commands to work with this setting. We can literally *add* variables to our model (with plus signs!).

Intro to MLR (3.1 - 3.2)

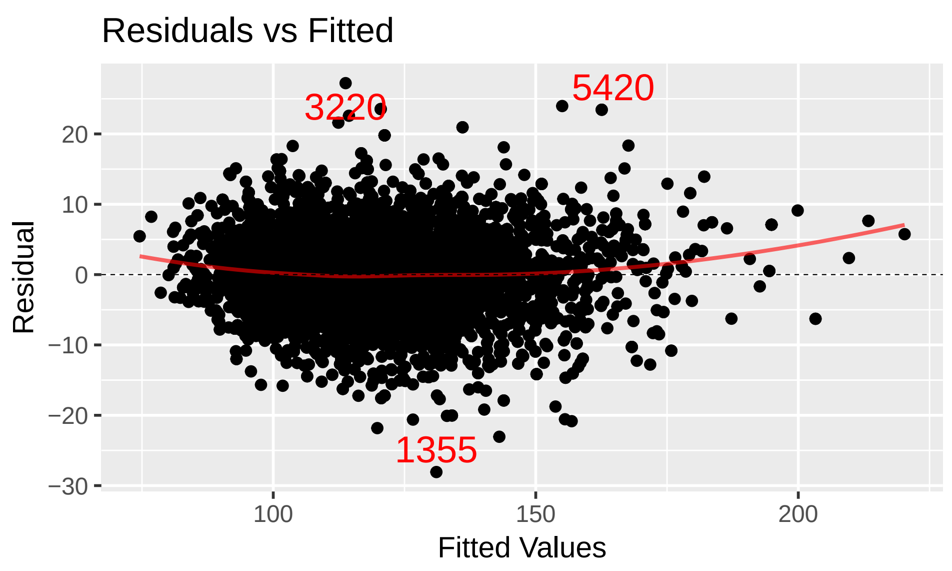
As a light intro, let's try to predict the Systolic blood pressure average using just the first systolic and diastolic measurements. This means that we have two quantitative predictors. We fit the model, examine it's summary, basic plots, and ANOVA output with the same commands we already know!

```
mlrmod <- lm(BPSysAve ~ BPSys1 + BPDia1, data = NHANES)
msummary(mlrmod)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.11133    0.40743    17.4   <2e-16 ***
## BPSys1       0.90141    0.00355   254.1   <2e-16 ***
## BPDia1       0.05283    0.00450    11.7   <2e-16 ***
##
## Residual standard error: 5.1 on 8234 degrees of freedom
##   (1763 observations deleted due to missingness)
## Multiple R-squared:  0.909,  Adjusted R-squared:  0.909
## F-statistic: 4.1e+04 on 2 and 8234 DF,  p-value: <2e-16
```

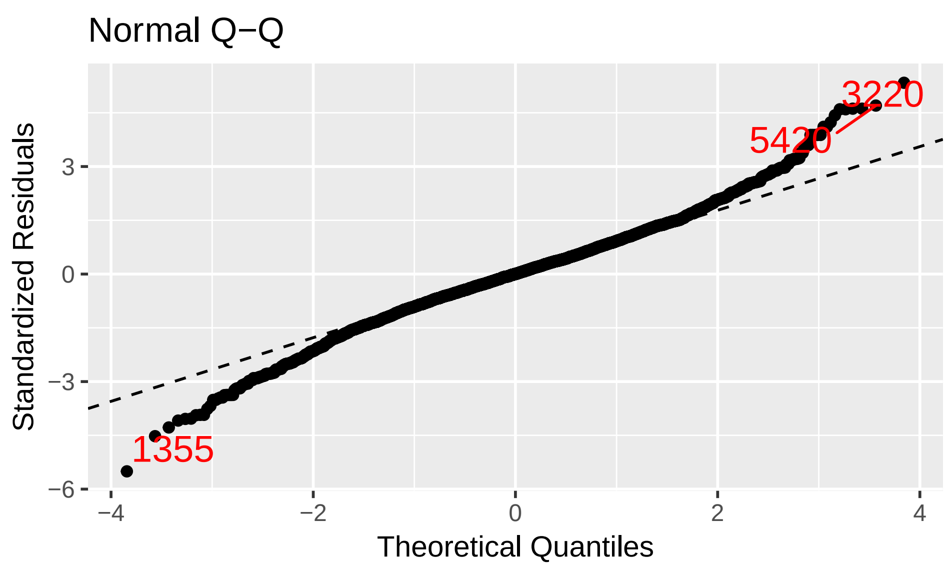
```
mplot(mlrmod, which = 1)
```

```
## [[1]]
```



```
mplot(mlrmod, which = 2)
```

```
## [[1]]
```



```
anova(mlrmod)
```

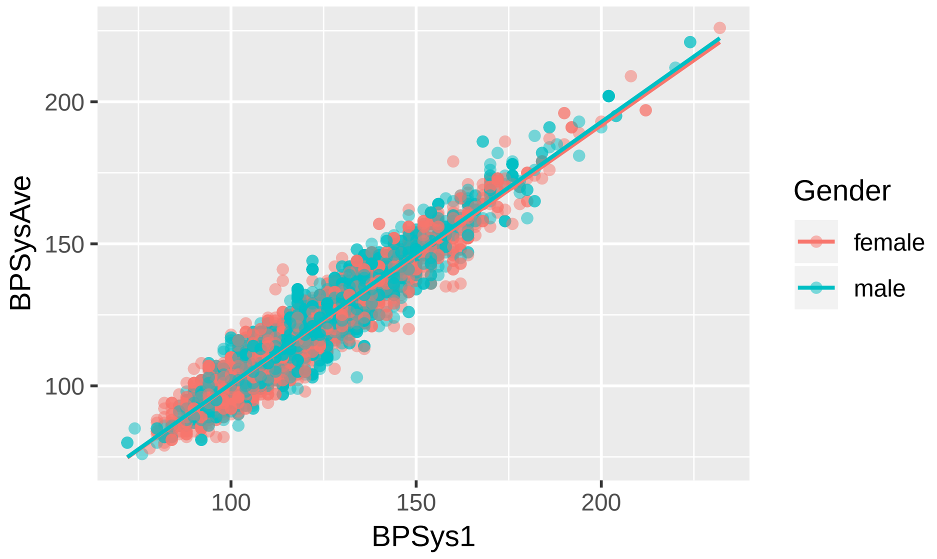
```
## Analysis of Variance Table
##
## Response: BPSysAve
##             Df  Sum Sq Mean Sq F value Pr(>F)
## BPSys1       1 2129580 2129580 81848.1 <2e-16 ***
## BPDia1       1    3580    3580   137.6 <2e-16 ***
## Residuals 8234  214238      26
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that you just have more rows in the output and the table now, corresponding to the increase in predictors.

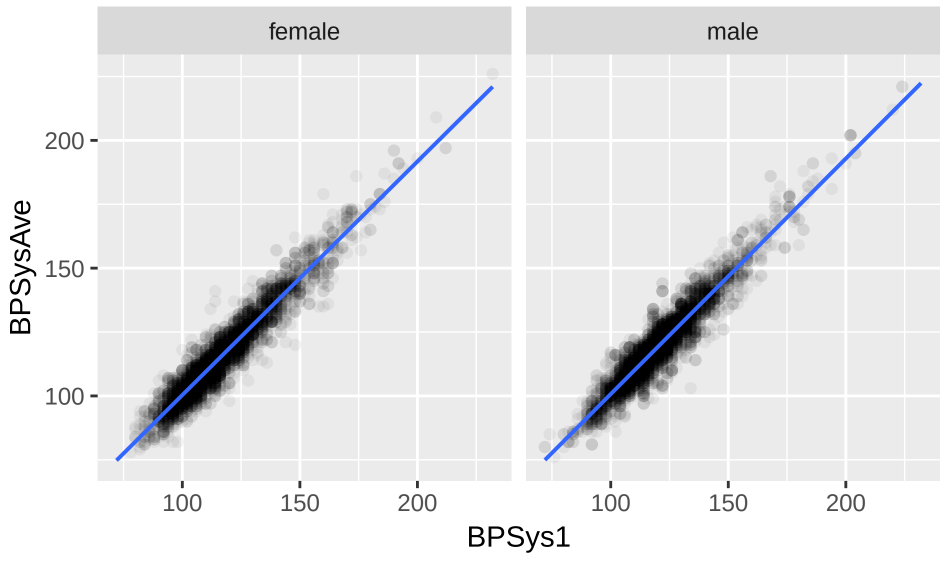
Parallel Lines (3.3)

Our first MLR example had 2 quantitative predictors. But what happens when one is quantitative and one is categorical? In terms of the commands used, absolutely nothing. But the output will look a little different. Let's use Gender and BPSys1 to try to predict BPSysAve. We can visualize this a bit first (these are similar to plots shown above).

```r
gf_point(BPSysAve ~ BPSys1, data = NHANES, alpha = 0.5, color = ~ Gender) %>%
  gf_lm()
```



```r
gf_point(BPSysAve ~ BPSys1 | Gender, data = NHANES, alpha = 0.05) %>%
  gf_lm()
```



This example was chosen because the relationship looks very similar between males and females. When considering this relationship, we have several models available to us:

BPSysAve ~ BPSys1 is the usual SLR.
BPSysAve ~ Gender would result in just examining the mean for each Gender, and likely not be very

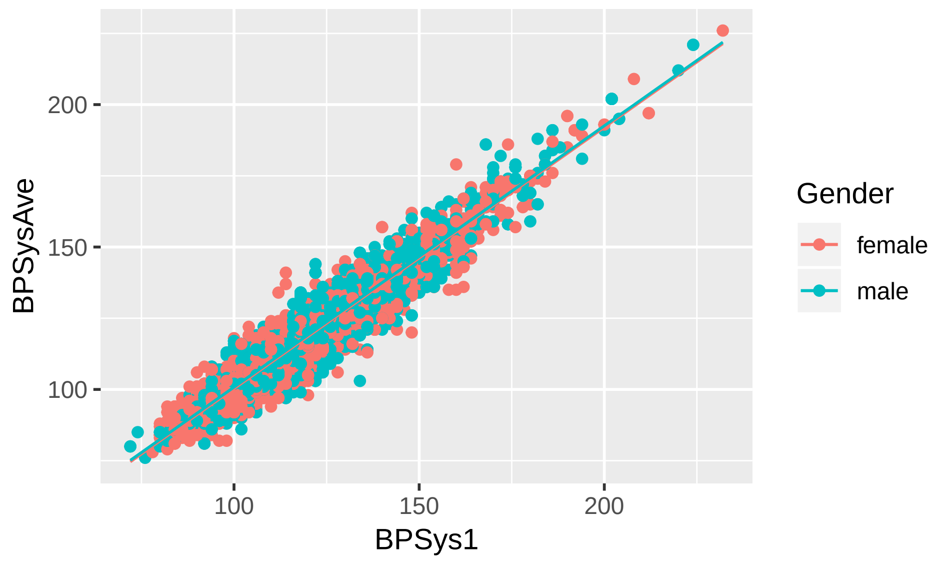
informative.
BPSysAve ~ BPSys1 + Gender is a model that would result in parallel lines, one for each Gender. In other words, the two groups would have the same slope for BPSys1, but could have different intercepts.
BPSysAve ~ BPSys1 * Gender is a model that would result in potentially different lines with differents slopes and intercepts. The interaction term, which can be examined individually as BPSys1:Gender is what allows the slopes to potentially differ. We examine the two MLRs below. Note that because "f" comes before "m" in the alphabet, female is taken to be the reference level.

```
paramod <- lm(BPSysAve ~ BPSys1 + Gender, data = NHANES)
msummary(paramod)
```

```
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    8.53752    0.38978    21.9  < 2e-16 ***
## BPSys1         0.91752    0.00325   282.1  < 2e-16 ***
## Gendermale     0.52309    0.11379     4.6  4.4e-06 ***
##
## Residual standard error: 5.14 on 8234 degrees of freedom
##   (1763 observations deleted due to missingness)
## Multiple R-squared:  0.907,  Adjusted R-squared:  0.907
## F-statistic: 4.04e+04 on 2 and 8234 DF,  p-value: <2e-16
```

Now suppose we want to plot this solution - it should have parallel lines. Here is how to do that, including use of *makeFun* to make predictions for the model to make the plot.

```
myFun <- makeFun(paramod)

gf_point(BPSysAve ~ BPSys1, data = NHANES, color = ~ Gender) %>%
  gf_fun(myFun(BPSys1, Gender = "female") ~ BPSys1, color = ~ "female") %>%
  gf_fun(myFun(BPSys1, Gender = "male") ~ BPSys1, color = ~ "male")
```



The model with interaction is easily fit as well.

```
intmod <- lm(BPSysAve ~ BPSys1 * Gender, data = NHANES)
# could also do BPSys1 + Gender + BPSys1:Gender
msummary(intmod)
```

```
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          8.94968    0.52386   17.08   <2e-16 ***
## BPSys1             0.91401    0.00441  207.09   <2e-16 ***
## Gendermale        -0.39358    0.78671   -0.50     0.62
## BPSys1:Gendermale  0.00769    0.00653    1.18     0.24
##
## Residual standard error: 5.14 on 8233 degrees of freedom
##   (1763 observations deleted due to missingness)
## Multiple R-squared:  0.907,  Adjusted R-squared:  0.907
## F-statistic: 2.69e+04 on 3 and 8233 DF,  p-value: <2e-16
```

Plotting these lines was easy, since it was the default used above when we specified color as Gender in our first plot in this section.

Accessing the model coefficients, ANOVA table, residuals, etc. is all done with the same commands as previously shown. But, we are learning to build models, expanding our options of what we can incorporate, and thinking about relationships between predictors and how that could be related to the response.

Polynomial Regression (3.4)

In this section, we are just introduced to some models that use powers of predictors or some other common combinations of predictors. There are two notes I'd like to make for this section.

First, to add a power of a variable to the model, you can't just raise it to the power in the lm command. See for example below.

```
mod2 <- lm(BPSysAve ~ BPSys1 + BPSys1^2, data = NHANES)
msummary(mod2)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.61705    0.38987    22.1   <2e-16 ***
## BPSys1       0.91905    0.00324   283.8   <2e-16 ***
##
## Residual standard error: 5.14 on 8235 degrees of freedom
##   (1763 observations deleted due to missingness)
## Multiple R-squared:  0.907,  Adjusted R-squared:  0.907
## F-statistic: 8.05e+04 on 1 and 8235 DF,  p-value: <2e-16
```

The predictor doesn't show up squared, does it? (Though, you CAN do log, exp, and sqrt transformations in line). There are two ways around this. First, you can code it like this:

```
mod3 <- lm(BPSysAve ~ BPSys1 + I(BPSys1^2), data = NHANES)
msummary(mod3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.60e+01   1.71e+00    9.39  < 2e-16 ***
## BPSys1      8.00e-01   2.69e-02   29.79  < 2e-16 ***
## I(BPSys1^2) 4.65e-04   1.04e-04    4.46  8.1e-06 ***
##
## Residual standard error: 5.14 on 8234 degrees of freedom
##   (1763 observations deleted due to missingness)
## Multiple R-squared:  0.907,  Adjusted R-squared:  0.907
## F-statistic: 4.04e+04 on 2 and 8234 DF,  p-value: <2e-16
```

The I() makes the system realize it should be squaring the predictor to include, and can be done with any power. You could alternatively add the squared variable to the data set, and then fit the model.

```
NHANES <- mutate(NHANES, BPSys1Sq = BPSys1^2)
mod4 <- lm(BPSysAve ~ BPSys1 + BPSys1Sq, data = NHANES)
msummary(mod4)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.60e+01   1.71e+00    9.39  < 2e-16 ***
## BPSys1      8.00e-01   2.69e-02   29.79  < 2e-16 ***
## BPSys1Sq    4.65e-04   1.04e-04    4.46  8.1e-06 ***
##
## Residual standard error: 5.14 on 8234 degrees of freedom
##   (1763 observations deleted due to missingness)
## Multiple R-squared:  0.907,  Adjusted R-squared:  0.907
## F-statistic: 4.04e+04 on 2 and 8234 DF,  p-value: <2e-16
```

Next, remember that Var1 * Var2 includes both main effects and the interaction. So for example, the code below has superfluous pieces.

```r
intmod2 <- lm(BPSysAve ~  BPSys1 + Gender + BPSys1 * Gender, data = NHANES)
msummary(intmod2)
```

```
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        8.94968    0.52386   17.08   <2e-16 ***
## BPSys1             0.91401    0.00441  207.09   <2e-16 ***
## Gendermale        -0.39358    0.78671   -0.50     0.62
## BPSys1:Gendermale  0.00769    0.00653    1.18     0.24
##
## Residual standard error: 5.14 on 8233 degrees of freedom
##   (1763 observations deleted due to missingness)
## Multiple R-squared:  0.907,  Adjusted R-squared:  0.907
## F-statistic: 2.69e+04 on 3 and 8233 DF,  p-value: <2e-16
```

This is the same output we had above from intmod, which was just BPSys1 * Gender.

The text describes a complete second order model which involves two quantitative predictors, both with linear and quadratic terms, and their interaction. We fit an example model using the blood pressure measurements.

```r
mod5 <- lm(BPSysAve ~ BPSys1 * BPSys2 + I(BPSys1^2) + I(BPSys2^2), data = NHANES)
msummary(mod5)
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2.983224   0.838169    3.56  0.00037 ***
## BPSys1         0.251909   0.033026    7.63  2.7e-14 ***
## BPSys2         0.717247   0.033532   21.39  < 2e-16 ***
## I(BPSys1^2)   -0.000531   0.000483   -1.10  0.27220
## I(BPSys2^2)    0.000260   0.000514    0.51  0.61356
## BPSys1:BPSys2  0.000290   0.000961    0.30  0.76282
##
## Residual standard error: 2.41 on 8055 degrees of freedom
##   (1939 observations deleted due to missingness)
## Multiple R-squared:  0.979,  Adjusted R-squared:  0.979
## F-statistic: 7.66e+04 on 5 and 8055 DF,  p-value: <2e-16
```

Multicollinearity (3.5)

Sometimes the predictors are highly related to one another (either pairwise or in a linear combination). If the relationship is strong, problems can occur. To examine this issue, variance inflation factors can be examined. Here, we fit a modest sized model to predict BPSysAve and look at the VIFs.

```
mymod <- lm(BPSysAve ~ BPSys1 + Age + AgeMonths + Weight + BMI, data = NHANES)
msummary(mymod)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.85447    0.63647   15.48   <2e-16 ***
## BPSys1       0.89126    0.00563  158.30   <2e-16 ***
## Age          0.08091    0.28041    0.29   0.7730
## AgeMonths   -0.00628    0.02337   -0.27   0.7883
## Weight       0.02298    0.00789    2.91   0.0036 **
## BMI         -0.01581    0.02620   -0.60   0.5462
##
## Residual standard error: 5.11 on 3954 degrees of freedom
##   (6040 observations deleted due to missingness)
## Multiple R-squared:   0.9,   Adjusted R-squared:   0.9
## F-statistic: 7.11e+03 on 5 and 3954 DF,  p-value: <2e-16
```

```
car::vif(mymod)
```

```
##     BPSys1        Age  AgeMonths     Weight        BMI
##    1.38408 4244.43527 4248.08848    5.16424    5.08034
```

The VIFs above 4-5 indicate potential issues. Here, there is an obvious (expected!) issue between Age and AgeMonths - AgeMonths is just Age times 12, so we don't need both in the model. Also, here we see issues with BMI and Weight which is because BMI and Weight are highly related (though again, the relationship doesn't need to be pairwise). This example with Age demonstrates one issue these relationships can cause. Neither Age nor AgeMonths is significant at a 0.05 level, but if you remove AgeMonths, Age becomes significant.

```
mymod <- lm(BPSysAve ~ BPSys1 + Age + Weight + BMI, data = NHANES)
msummary(mymod)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.59280    0.43405   19.80  < 2e-16 ***
## BPSys1       0.90857    0.00388  234.26  < 2e-16 ***
## Age          0.00711    0.00336    2.11  0.03466 *
## Weight       0.02111    0.00550    3.84  0.00012 ***
## BMI         -0.02400    0.01834   -1.31  0.19076
##
## Residual standard error: 5.11 on 8171 degrees of freedom
##   (1824 observations deleted due to missingness)
## Multiple R-squared:  0.908,  Adjusted R-squared:  0.908
## F-statistic: 2.01e+04 on 4 and 8171 DF,  p-value: <2e-16
```

```
car::vif(mymod)
```

```
##  BPSys1     Age  Weight     BMI
## 1.43563 1.43037 5.07232 5.04553
```

You can also obtain generalized VIFs when categorical predictors are involved, though these ideas are not presented in the text. For example,

```
mymod <- lm(BPSysAve ~ BPSys1 + Age + Weight + Work, data = NHANES)
msummary(mymod)
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    8.01e+00   5.64e-01   14.20  < 2e-16 ***
## BPSys1         9.13e-01   4.05e-03  225.68  < 2e-16 ***
## Age            7.25e-05   3.96e-03    0.02    0.985
## Weight         1.22e-02   2.96e-03    4.13  3.7e-05 ***
## WorkNotWorking 5.55e-01   3.33e-01    1.67    0.096 .
## WorkWorking    3.03e-01   3.22e-01    0.94    0.348
##
## Residual standard error: 5.21 on 7144 degrees of freedom
##   (2850 observations deleted due to missingness)
## Multiple R-squared:  0.902,  Adjusted R-squared:  0.901
## F-statistic: 1.31e+04 on 5 and 7144 DF,  p-value: <2e-16
```

```
car::vif(mymod)
```

```
##           GVIF Df GVIF^(1/(2*Df))
## BPSys1 1.27485  1         1.12909
## Age    1.32651  1         1.15174
## Weight 1.03210  1         1.01592
## Work   1.09875  2         1.02382
```

These values do not seem to indicate any issues. Remember that descriptive statistics are meant to be a tool to assist you, and you need to be realistic with your interpretations of what you see. For example,

```
mymod <- lm(BPSysAve ~ BPSys1 + Weight + I(Weight^2), data = NHANES)
msummary(mymod)
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.93e+00   5.33e-01   13.00  < 2e-16 ***
## BPSys1       9.11e-01   3.41e-03  267.43  < 2e-16 ***
## Weight       5.23e-02   1.09e-02    4.79  1.7e-06 ***
## I(Weight^2) -2.16e-04   6.21e-05   -3.48    5e-04 ***
##
## Residual standard error: 5.12 on 8178 degrees of freedom
##   (1818 observations deleted due to missingness)
## Multiple R-squared:  0.908,  Adjusted R-squared:  0.908
## F-statistic: 2.69e+04 on 3 and 8178 DF,  p-value: <2e-16
```

```
car::vif(mymod)
```

```
##      BPSys1      Weight I(Weight^2)
##      1.1100     19.9462     19.4986
```

Should we be upset about the VIFs of 19 here for Weight and Weight^2, and use that as motivation for adjusting the model? (There may be other reasons to adjust the model, but you should realize that we would expect these values to be high.)

Nested F-tests (3.6)

To test for removing subsets of variables at a time, you need nested models, and then we run nested F procedures.

```
modmed <- lm(BPSysAve ~ BPSys1 + Weight + Age + Gender, data = NHANES)
modsmall <- lm(BPSysAve ~ BPSys1 + Weight, data = NHANES)

anova(modsmall, modmed) #ascending order of complexity
```

```
## Analysis of Variance Table
##
## Model 1: BPSysAve ~ BPSys1 + Weight
## Model 2: BPSysAve ~ BPSys1 + Weight + Age + Gender
##   Res.Df    RSS Df Sum of Sq     F   Pr(>F)    
## 1   8179 214312                                
## 2   8177 213881  2     431.1 8.241 0.000266 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that the models need to be fit on the same dataset (meaning, if observations are removed from one model due to NAs but not the other, this command will generate an error message.) You can do more than 2 comparisons at a time, but be careful with your interpretations in that case. Also, notice that if you then do pairwise comparisons, your F statistics and p-values will be slightly off compared to the multiple models at once results, because the df are being adjusted for testing multiple models at once. This is usually not a large enough concern to necessitate doing all comparisons pairwise, however, the CONCEPT of multiple testing and adjusting significance levels is one you should work at understanding.

Topics in MLR (Chapter 4)

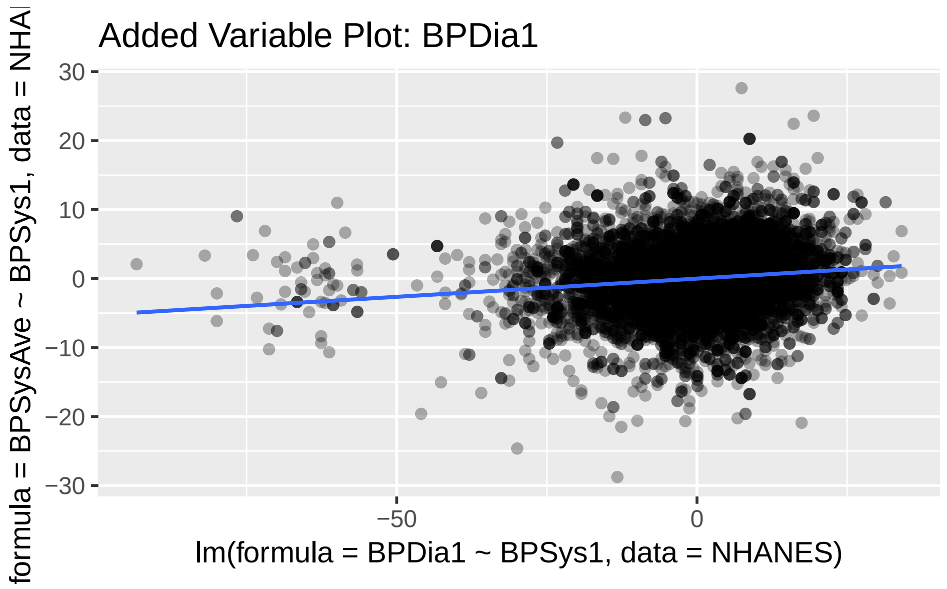
Added Variable Plots (4.1)

You can construct added variable plots by obtaining the residuals directly from the two regressions yourself or by using this function, modified from one written by Ben Baumer. Run this entire chunk to obtain the function (and be sure to put it in your markdown file as well.)

```
plotAddedVar <-
  function(
    fm, var.test,
    title = paste("Added Variable Plot:", var.test),
    ylab = deparse(fm1$call),
    xlab = deparse(fm2$call),
    ...) {
  formula <- formula(fm)
  fm1 <- update(fm, formula. = substitute(~ . - x, list(x = as.name(var.test))))
  #fm2 <- update(fm, formula. = substitute(x ~ . , list(x = as.name(var.test))))
  fm2 <- update(fm, formula. = substitute(x ~ . - x, list(x = as.name(var.test))))

  PlotData <- data.frame(y = resid(fm1), x = resid(fm2))
  gf_point(y ~ x, data = PlotData, ...) %>%
    gf_lm() %>%
    gf_labs(
      title = title,
      y = ylab, x = xlab
    )
}
```

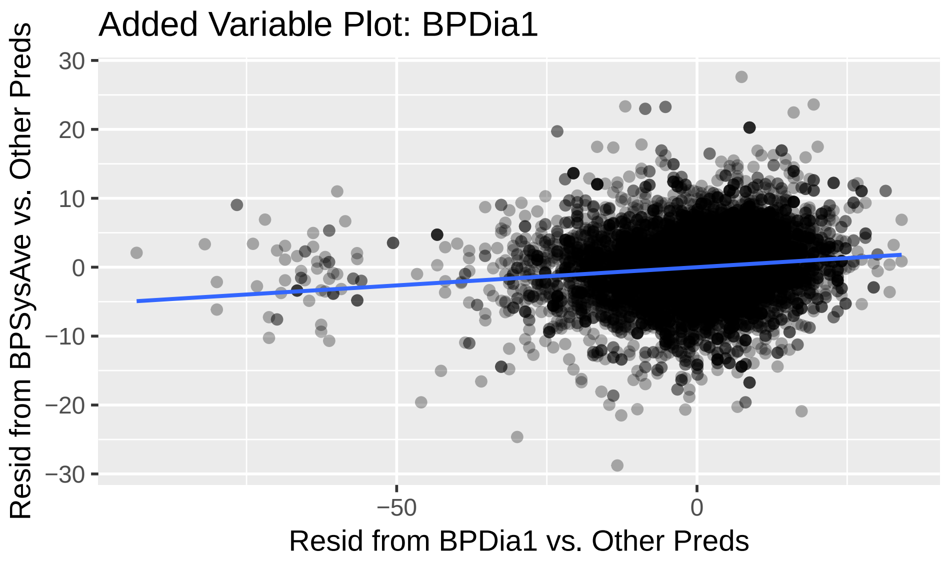
We demonstrate on the first example MLR model, using the first systolic and diastolic blood pressures to predict the systolic average.

```
plotAddedVar(mlrmod, "BPDia1", alpha = 0.3)
```



You can change the labels on the axes if you want, because they usually run off with MLRs of any decent size.

```
plotAddedVar(mlrmod, "BPDia1", alpha = 0.3, ylab = "Resid from BPSysAve vs. Other Preds",
             xlab = "Resid from BPDia1 vs. Other Preds")
```



Automated Variable Selection (4.2)

This section includes four different automated variable selection methods. Code to execute these methods in R is contained in the leaps package. Other packages have functions that can do this as well, but we'll just use *regsubsets* from leaps.

To demonstrate this code, we need a model to start from.

```
modall <- lm(BPSysAve ~ BPSys1 + BPDia1 + Weight + Age + Gender + HomeRooms + Height + BMI,
             data = NHANES)
msummary(modall)
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.49904    2.67736    1.68    0.093 .
## BPSys1       0.89200    0.00413  215.73  < 2e-16 ***
## BPDia1       0.05080    0.00477   10.64  < 2e-16 ***
## Weight      -0.02722    0.01902   -1.43    0.153
## Age          0.00804    0.00343    2.35    0.019 *
## Gendermale   0.63493    0.14075    4.51  6.5e-06 ***
## HomeRooms    0.00270    0.02467    0.11    0.913
## Height       0.01505    0.01631    0.92    0.356
## BMI          0.10177    0.05315    1.91    0.056 .
##
## Residual standard error: 5.08 on 8113 degrees of freedom
##   (1878 observations deleted due to missingness)
## Multiple R-squared:  0.909,  Adjusted R-squared:  0.909
## F-statistic: 1.02e+04 on 8 and 8113 DF,  p-value: <2e-16
```

Here, we chose a model that has some predictors we probably don't need given the presence of other predictors (based on previous exploration). We start with best subsets code. We can specify the number of best models of each size to examine, though the default is one. With the output, we can examine all the "best" models and compare their statistics, to then use to choose one.

```
best <- regsubsets(BPSysAve ~ BPSys1 + BPDia1 + Weight + Age + Gender + HomeRooms
                   + Height + BMI, data = NHANES, nbest = 1)
with(summary(best), data.frame(rsq, adjr2, cp, rss, outmat))
```

```
##                rsq    adjr2        cp    rss BPSys1 BPDia1 Weight Age
## 1  ( 1 ) 0.907484 0.907473 167.80479 213446      *
## 2  ( 1 ) 0.909011 0.908989  33.05148 209923      *      *
## 3  ( 1 ) 0.909187 0.909153  19.32589 209518      *      *
## 4  ( 1 ) 0.909311 0.909267  10.14697 209230      *      *
## 5  ( 1 ) 0.909376 0.909321   6.32336 209080      *      *          *
## 6  ( 1 ) 0.909404 0.909337   5.85397 209016      *      *      *   *
## 7  ( 1 ) 0.909413 0.909335   7.01196 208994      *      *      *   *
## 8  ( 1 ) 0.909413 0.909324   9.00000 208994      *      *      *   *
##          Gendermale HomeRooms Height BMI
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )          *
## 4  ( 1 )          *                    *
## 5  ( 1 )          *                    *
## 6  ( 1 )          *                    *
## 7  ( 1 )          *               *    *
## 8  ( 1 )          *         *     *    *
```

Mallows Cp is the statistic described in your text. Other criteria exist, including AIC and BIC which are not described in your text. AIC is what R uses by default. The AIC value for a model can be obtained using the AIC command, once you have selected a model. Here, suppose I choose the minimum Cp model as the one I want to work with, and I want it's AIC value. Remember, you want to minimize AIC.

```
bestmodel <- lm(BPSysAve ~ BPSys1 + BPDia1 + Weight + Age + Gender + Work + BMI,
                data = NHANES)
AIC(bestmodel)
```

```
## [1] 43716.6
```

Now we demonstrate the other 3 methods with *regsubsets*, and we'll put fewer statistics in the output since the book focuses on Cp. Backward elimination starts with a full model and removes predictors one at a time.

Here, you can set the maximum size of subsets to examine with the nvmax option. The default is 8, which is fine for this example. If you have a very large set of predictors, you may need to adjust that value.

```r
backward <- regsubsets(BPSysAve ~ BPSys1 + BPDia1 + Weight + Age + Gender + HomeRooms
                       + Height + BMI, data = NHANES, method = "backward", nbest = 1)
with(summary(backward), data.frame(cp, outmat))
```

```
##                  cp BPSys1 BPDia1 Weight Age Gendermale HomeRooms Height
## 1  ( 1 ) 167.80479      *
## 2  ( 1 )  33.05148      *      *
## 3  ( 1 )  19.32589      *      *                     *
## 4  ( 1 )  10.14697      *      *                     *
## 5  ( 1 )   6.32336      *      *          *          *
## 6  ( 1 )   5.85397      *      *      *   *          *
## 7  ( 1 )   7.01196      *      *      *   *          *                *
## 8  ( 1 )   9.00000      *      *      *   *          *         *      *
##          BMI
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )   *
## 5  ( 1 )   *
## 6  ( 1 )   *
## 7  ( 1 )   *
## 8  ( 1 )   *
```

Next, we examine forward selection. Again, you could adjust *nvmax* if you wanted to control how large the models get. The default is 8.

```r
forward <- regsubsets(BPSysAve ~ BPSys1 + BPDia1 + Weight + Age + Gender + HomeRooms
                      + Height + BMI, data = NHANES, method = "forward", nbest = 1)
with(summary(forward), data.frame(cp, outmat))
```

```
##                  cp BPSys1 BPDia1 Weight Age Gendermale HomeRooms Height
## 1  ( 1 ) 167.80479      *
## 2  ( 1 )  33.05148      *      *
## 3  ( 1 )  19.32589      *      *                     *
## 4  ( 1 )  10.14697      *      *                     *
## 5  ( 1 )   6.32336      *      *          *          *
## 6  ( 1 )   5.85397      *      *      *   *          *
## 7  ( 1 )   7.01196      *      *      *   *          *                *
## 8  ( 1 )   9.00000      *      *      *   *          *         *      *
##          BMI
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )   *
## 5  ( 1 )   *
## 6  ( 1 )   *
## 7  ( 1 )   *
## 8  ( 1 )   *
```

Finally, stepwise regression is performed using the *seqrep* option for the method.

```r
stepwise <- regsubsets(BPSysAve ~ BPSys1 + BPDia1 + Weight + Age + Gender + HomeRooms
                       + Height + BMI, data = NHANES, method = "seqrep", nbest = 1)
with(summary(stepwise), data.frame(cp, outmat))
```

```
##                   cp BPSys1 BPDia1 Weight Age Gendermale HomeRooms Height
## 1  ( 1 ) 167.80479      *
## 2  ( 1 )  33.05148      *      *
## 3  ( 1 )  19.32589      *      *                     *
## 4  ( 1 )  10.14697      *      *                     *
## 5  ( 1 )   6.32336      *      *          *          *
## 6  ( 1 )  12.59326      *      *      *   *          *          *
## 7  ( 1 )   7.01196      *      *      *   *          *                 *
## 8  ( 1 )   9.00000      *      *      *   *          *          *      *
##          BMI
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )   *
## 5  ( 1 )   *
## 6  ( 1 )
## 7  ( 1 )   *
## 8  ( 1 )   *
```

For each of these, you get to select the model you want to use, and then fit it in order to work with it further. Automated methods are not a substitute for a statistician!

Other functions such as stepAIC in the MASS library will perform similar operations.

Unusual Points (4.3)

To identify unusual points, we can use graphs and descriptive statistics. Your book's R companion uses *ls.diag* as the function to find the necessary descriptive statistics. But the statistics it produces, with the exception of the studentized residuals, are also available in the augmented model from *augment*. So, we demo both approaches below, and show you how to get the studentized residuals separately.

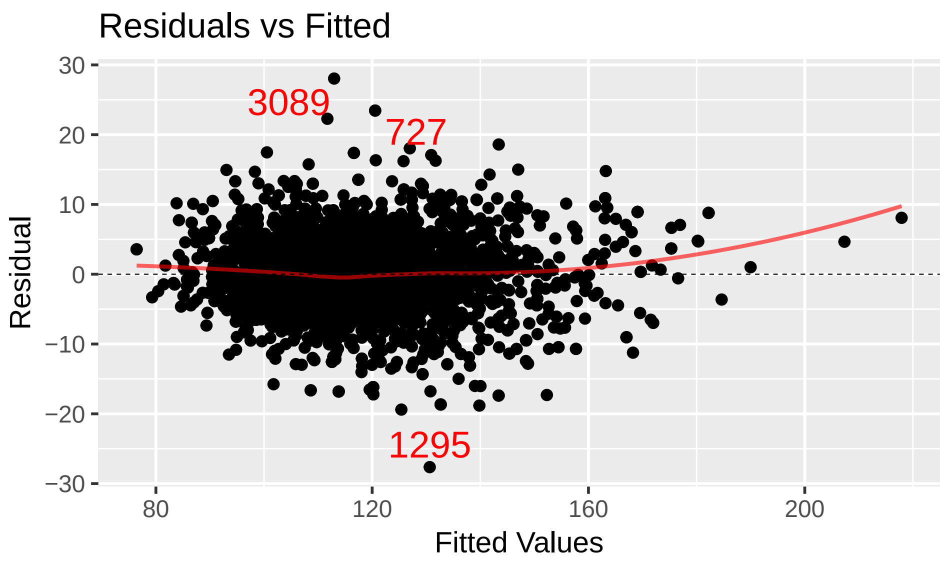
With this model, we can see that there are some points we want to investigate.

```
mymod <- lm(BPSysAve ~ BPSys1 + Age + AgeMonths + Weight + BMI, data = NHANES)
msummary(mymod)
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.85447    0.63647   15.48   <2e-16 ***
## BPSys1        0.89126    0.00563  158.30   <2e-16 ***
## Age           0.08091    0.28041    0.29   0.7730
## AgeMonths    -0.00628    0.02337   -0.27   0.7883
## Weight        0.02298    0.00789    2.91   0.0036 **
## BMI          -0.01581    0.02620   -0.60   0.5462
##
## Residual standard error: 5.11 on 3954 degrees of freedom
##   (6040 observations deleted due to missingness)
## Multiple R-squared:   0.9,   Adjusted R-squared:   0.9
## F-statistic: 7.11e+03 on 5 and 3954 DF,  p-value: <2e-16
```

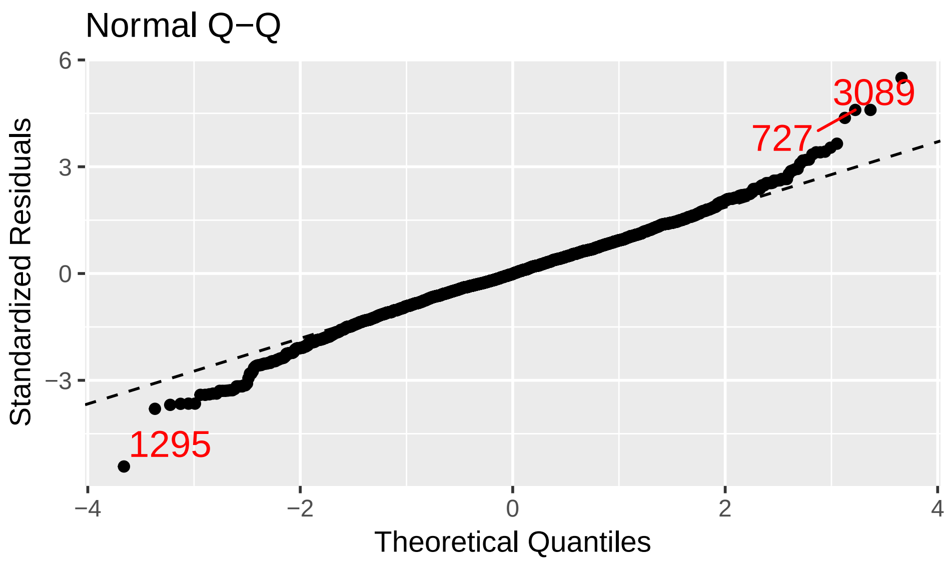
```
mplot(mymod, which = 1)
```

```
## [[1]]
```



```r
mplot(mymod, which = 2)
```

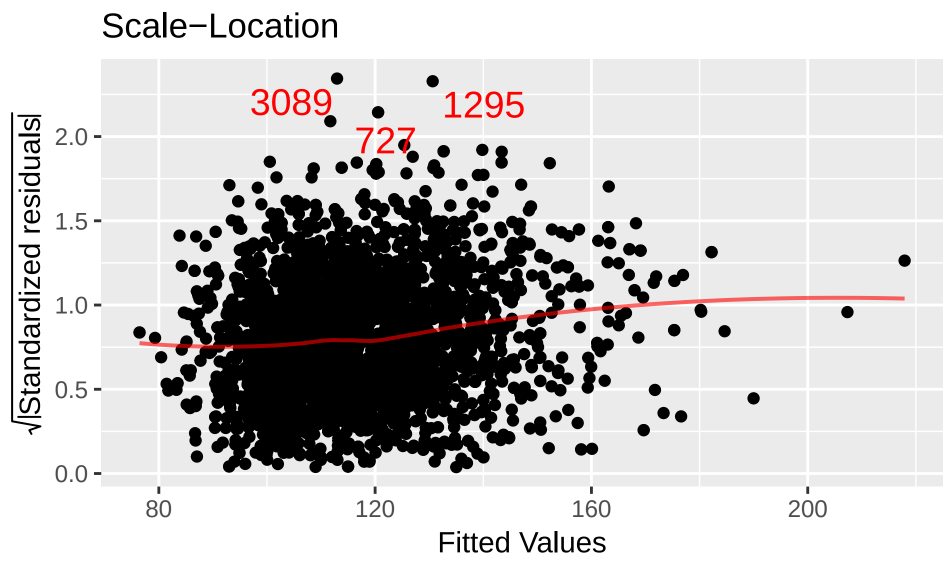
```
## [[1]]
```



First, we examine the other diagnostic plots that come with any model fit.

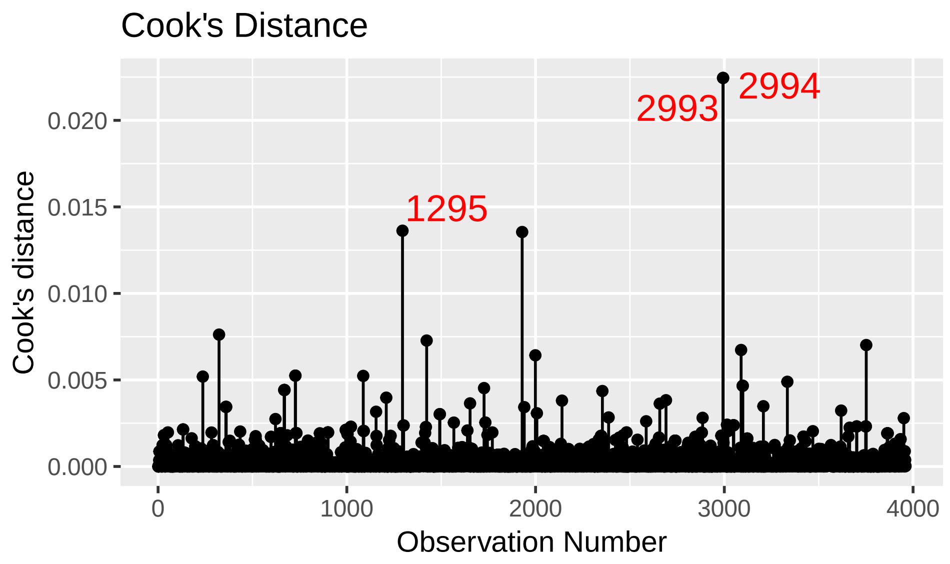
```r
mplot(mymod, which = 3)
```

```
## [[1]]
```



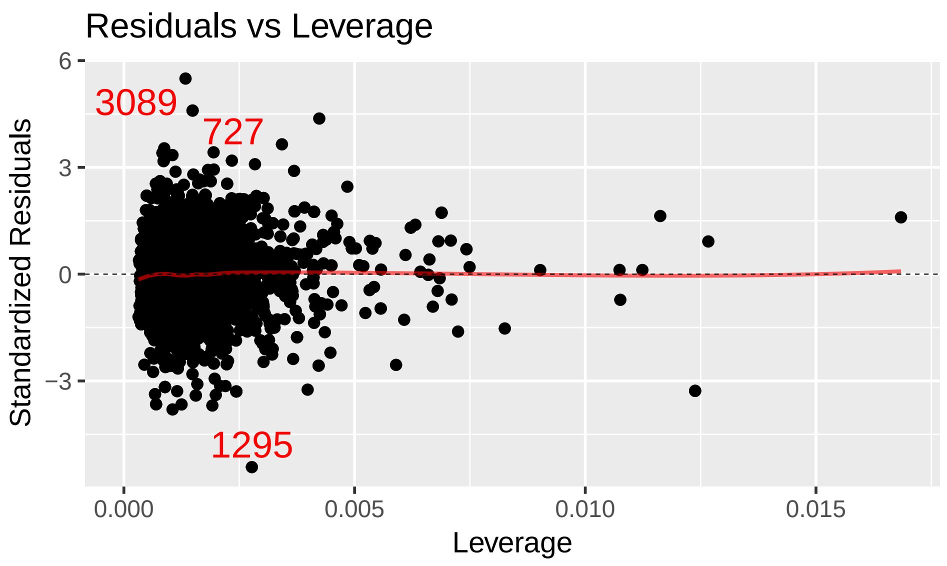
```r
mplot(mymod, which = 4)
```

```
## [[1]]
```



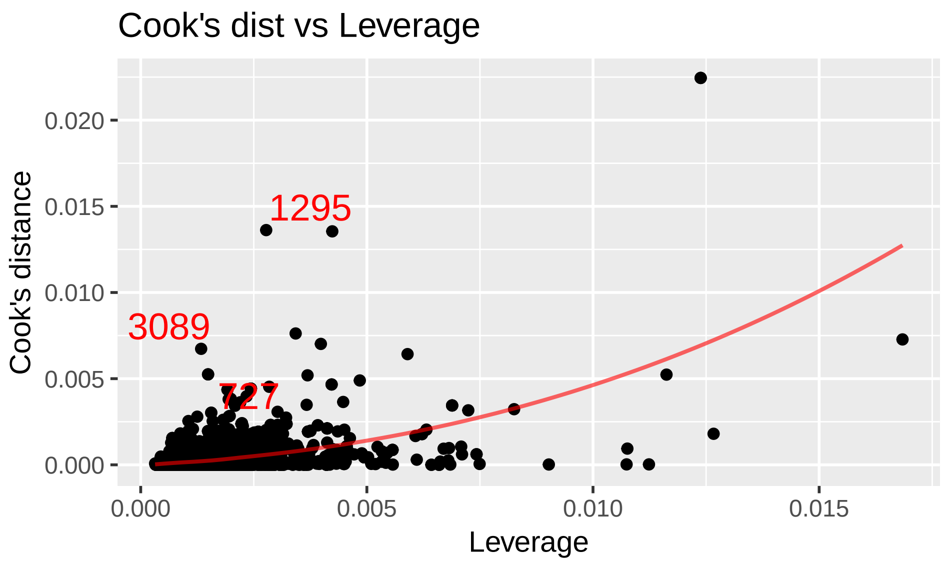
```r
mplot(mymod, which = 5)
```

```
## [[1]]
```



```r
mplot(mymod, which = 6)
```

```
## [[1]]
```



Note the combination of residuals, leverage, and cook's distance used throughout these plots.

Next, we want to examine how to get the statistics and save them.

```r
influencestat <- ls.diag(mymod)
names(influencestat)
```

```
##  [1] "std.dev"      "hat"          "std.res"      "stud.res"
##  [5] "cooks"        "dfits"        "correlation"  "std.err"
##  [9] "cov.scaled"   "cov.unscaled"
```

Note here that you have the standardized and studentized residuals, cooks distances, and leverages (hat). You can add these variables to your data set with mutate.

There are a few other ways to access the statistics. Here, we use a function to obtain each.

```r
std.res <- rstandard(mymod)
stu.res <- rstudent(mymod)
cooks <- cooks.distance(mymod)
hats <- hatvalues(mymod)
```

The augmented data set also contains some of these descriptive statistics. The only one it leaves out is the studentized residuals, so you could easily add that, as shown below.

```r
NHANESaug <- augment(mymod) %>% mutate(.stu.resid = rstudent(mymod))
names(NHANESaug)
```

```
##  [1] ".rownames"  "BPSysAve"   "BPSys1"     "Age"        "AgeMonths"
##  [6] "Weight"     "BMI"        ".fitted"    ".se.fit"    ".resid"
## [11] ".hat"       ".sigma"     ".cooksd"    ".std.resid" ".stu.resid"
```

Then, we can use commands like which and filter to help isolate observations that are unusual based on our chosen criteria and create datasets without those unusual points to study their effect on the model.

For example, suppose we want to examine the relationship only with observations with standardized residuals less than 3 in absolute value, and we want to know which observations were left out. First, we can make a new data set with those observations left out.

```r
NHANESaugnoup <- filter(NHANESaug, abs(.std.resid) < 3)
mymodnoup <- lm(BPSysAve ~ BPSys1 + Age + AgeMonths + Weight + BMI, data = NHANESaugnoup)
msummary(mymodnoup)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.59721    0.59898   16.02  < 2e-16 ***
## BPSys1       0.89325    0.00531  168.07  < 2e-16 ***
## Age         -0.08415    0.26440   -0.32    0.750
## AgeMonths    0.00748    0.02204    0.34    0.734
## Weight       0.03271    0.00747    4.38  1.2e-05 ***
## BMI         -0.04372    0.02475   -1.77    0.077 .
##
## Residual standard error: 4.79 on 3915 degrees of freedom
## Multiple R-squared:  0.911,  Adjusted R-squared:  0.911
## F-statistic: 8.06e+03 on 5 and 3915 DF,  p-value: <2e-16
```

Does it look like there have been any major changes to the model?

Now, what points were left out? We can take those observations and make them into a dataframe for further investigation.

```r
NHANESup <-filter(NHANESaug, abs(.std.resid) >= 3)
nrow(NHANESup)
```

```
## [1] 39
```

```r
NHANESup$.rownames
```

```
##  [1] "375"  "455"  "835"  "836"  "837"  "908"  "909"  "1282" "1511" "1624"
## [11] "1863" "1864" "1956" "2058" "2172" "2180" "2194" "2230" "2452" "2466"
## [21] "2705" "2939" "2940" "2962" "3352" "3391" "3454" "3455" "3456" "3457"
## [31] "3585" "3586" "3786" "3787" "3892" "3930" "3931" "4731" "4991"
```

```r
unusualpoints <- as.numeric(NHANESup$.rownames)
```

39 points were left out, and because augment passed along their observation number, we can see which observations they were fairly easily. If you wanted, you could store those entries as a vector to use in other

operations. Note that you have a data set of the unusual points in NHANESup so you can investigate them all you like.

Randomization Permutation-Based Procedures (4.5)

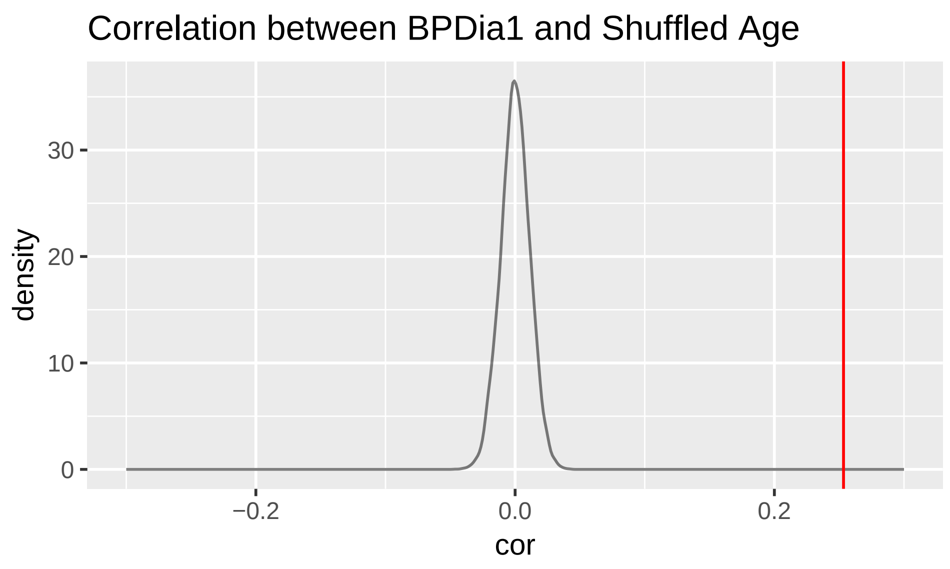
Here, we demonstrate running randomization permutation-based procedures. The examples cover examining a correlation and a slope for a main effect. Please ASK for assistance if you are trying to study an interaction effect, as that is a bit more complicated than your book examples.

First, we examine a randomization procedure for a correlation.

```
cor.actual <- with(NHANES, cor(BPDia1,Age, use="pairwise.complete.obs"))
cor.actual
```

```
## [1] 0.253356
```

```
set.seed(45)
rtest <- with(NHANES, do(10000) * cor(BPDia1, shuffle(Age), use = "pairwise.complete.obs"))
gf_dens(~ cor, data = rtest) %>%
  gf_lims(x = c(-0.3, 0.3)) %>%
  gf_labs(title = "Correlation between BPDia1 and Shuffled Age") %>%
  gf_vline(xintercept = ~ cor.actual, color = "red")
```



We can see that the observed correlation is well removed from the randomization distribution. We could compute empirical p-values and confidence intervals for the shuffled correlations as follows.

```
#provides area below cor.actual based on rtest$result distribution
pdata(~ cor, cor.actual, data = rtest)
```

```
## [1] 1
```

```
#area above (if you want upper tail)
pdata(~ cor, cor.actual, data = rtest, lower.tail = FALSE)
```

```
## [1] 0
```

```
qdata(~ cor, c(0.025, 0.975), data = rtest)
```

```
##         quantile     p
## 2.5%  -0.0218806 0.025
## 97.5%  0.0220811 0.975
```

As expected, this confidence interval is fairly symmetric around 0. We see the observed value of 0.25 is clearly outside the CI.

Next, we examine a procedure for a slope in an MLR.

```
mlrmod <- lm(BPSysAve ~ BPSys1 + Age + Weight + BMI, data = NHANES)
coef(mlrmod)
```

```
## (Intercept)      BPSys1         Age      Weight         BMI
##  8.59279972  0.90857292  0.00710519  0.02110639 -0.02400251
```

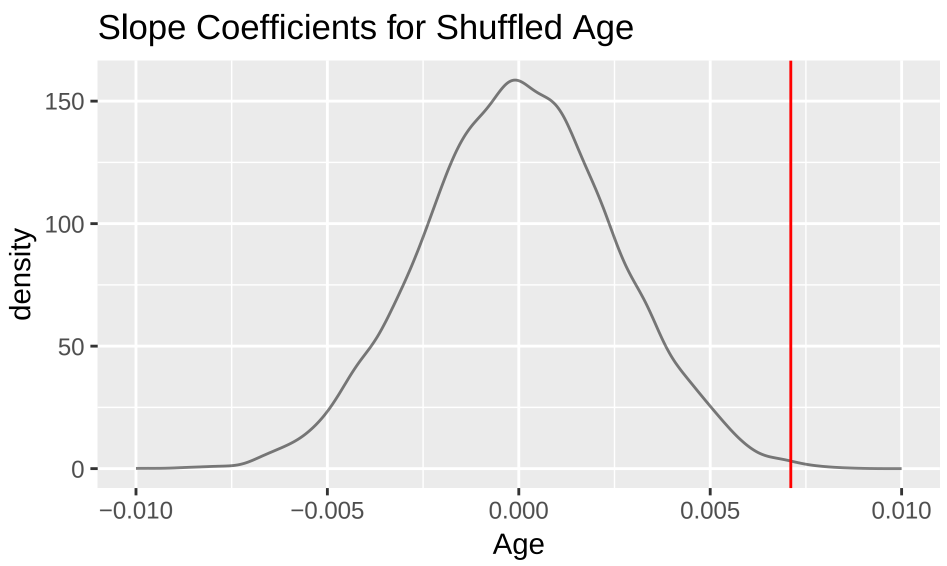
```
coef(mlrmod)[3]
```

```
##        Age
## 0.00710519
```

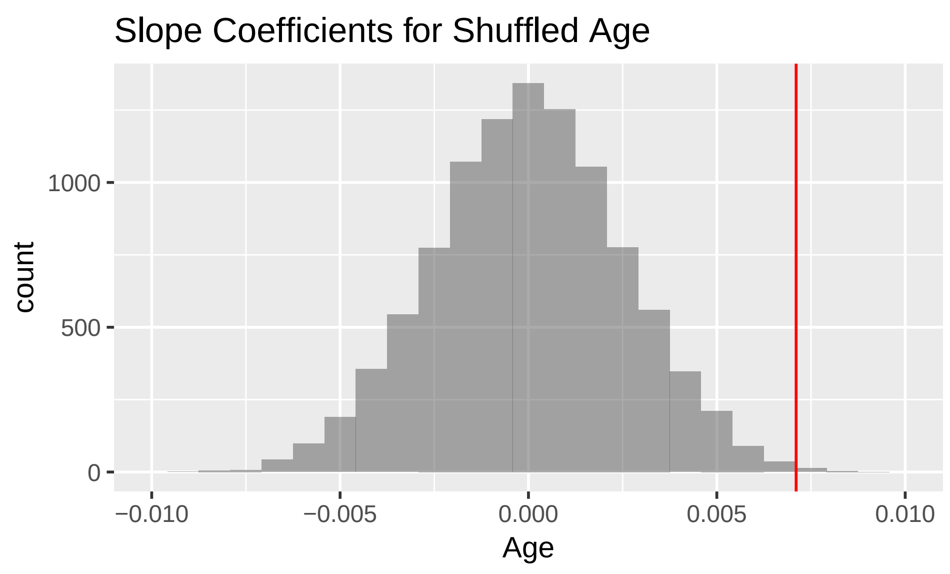
```
slopetest <- do(10000)*(lm(BPSysAve ~ BPSys1 + shuffle(Age) + Weight + BMI, data = NHANES))
names(slopetest)
```

```
##  [1] "Intercept" "BPSys1"    "Age"       "Weight"    "BMI"
##  [6] "sigma"     "r.squared" "F"         "numdf"     "dendf"
## [11] ".row"      ".index"
```

```
gf_dens(~ Age, data = slopetest) %>%
  gf_lims(x = c(-0.01, 0.01)) %>%
  gf_labs(title = "Slope Coefficients for Shuffled Age") %>%
  gf_vline(xintercept = ~ coef(mlrmod)["Age"], color = "red")
```



```
gf_histogram(~ Age, data = slopetest) %>%
  gf_lims(x = c(-0.01, 0.01)) %>%
  gf_labs(title = "Slope Coefficients for Shuffled Age") %>%
  gf_vline(xintercept = coef(mlrmod)["Age"], color = "red")
```



If you need the empirical p-value or CI, again that can be done with:

```
pdata(~ Age, coef(mlrmod)["Age"], data = slopetest, lower.tail = FALSE)
```

```
##    Age
## 0.0016
```

```
qdata(~ Age, c(0.025, 0.975), data = slopetest)
```

```
##          quantile     p
## 2.5%  -0.00490491 0.025
## 97.5%  0.00493263 0.975
```

Bootstrap (4.6)

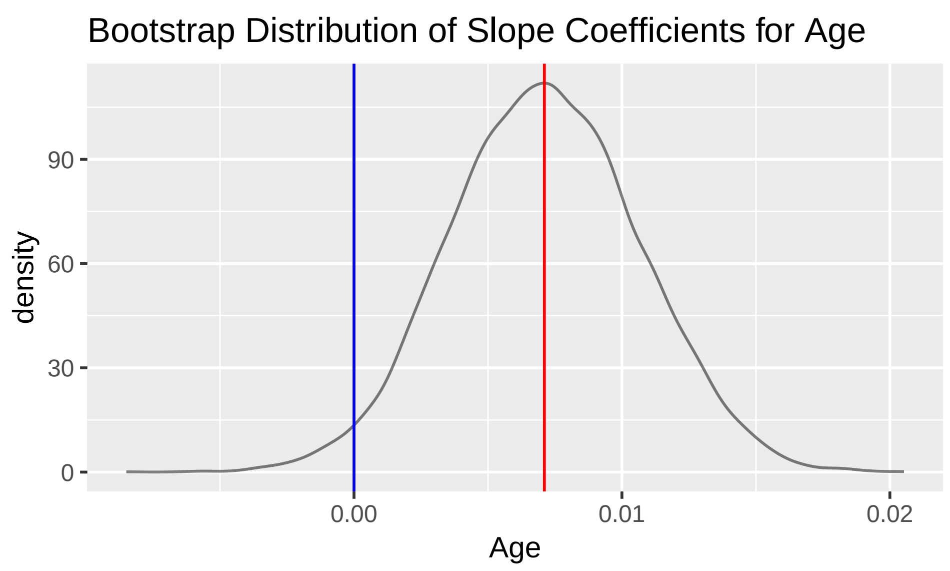
The final section of chapter 4 illustrates bootstrap procedures. These procedures are randomization-based, but differ from permutation procedures. Be sure you understand the difference!

We examine the same relationship with the slope of Age as above in the permutation section.

```
bootstrap <- do(10000) * lm(BPSysAve ~ BPSys1 + Age + Weight + BMI,
              data = resample(NHANES))$coefficients
# might take a second to run!
names(bootstrap) #saved slope coefficients
```

```
## [1] "Intercept" "BPSys1"    "Age"       "Weight"    "BMI"
```

```
gf_dens(~ Age, data = bootstrap) %>%
  gf_labs(title = "Bootstrap Distribution of Slope Coefficients for Age") %>%
  gf_vline(xintercept = ~ coef(mlrmod)["Age"], color = "red") %>%
  gf_vline(xintercept = ~ 0, color = "blue")
```



We can see that the observed slope is near the center of the distribution, and the value of 0 is off to the far side of the distribution.

Your textbook illustrates three methods to create CIs using the bootstrapped values. The shape of the bootstrapped value distribution is what determines what method is appropriate. This also shows how to attain a critical value (such as 1.96), if you want to change the confidence level, in reference to Method 1.

```
#Method 1 - Bootstrap distribution is roughly normally distributed

mycoef <- coef(mlrmod)["Age"]
mycoef + c(-1.96, 1.96) * sd(bootstrap$Age) #95% CI
```

```
## [1] 0.000224501 0.013985874
```

```
qnorm(0.95) #90% critical value. 5% in upper tail + 5% in lower tail would yield 90% CI
```

```
## [1] 1.64485
```

```
#Method 2 - Bootstrap distribution is roughly symmetric
qdata(~ Age, c(0.025, 0.975), data = bootstrap)
```

```
##          quantile     p
## 2.5%  0.000376699 0.025
## 97.5% 0.014018553 0.975
```

```
#Method 3 - Bootstrap distribution is skewed
q.ul <- qdata(~ Age, c(0.025, 0.975), data = bootstrap)$quantile
c(2*coef(mlrmod)["Age"] - q.ul[2], 2*coef(mlrmod)["Age"] - q.ul[1])
```

```
##         Age         Age
## 0.000191823 0.013833676
```

For this example, we see that all three CIs are similar, and do not contain 0.

One-Way ANOVA and Multiple Comparisons (Chapter 5, selected parts of Chapter 7)

ANOVA Output

For ANOVA commands, there is nothing really new. We use *lm* and *anova*, though you can use *aov* as well.

```
anovamod <- lm(BPSysAve ~ Race1, data = NHANES)
anova(anovamod)
```

```
## Analysis of Variance Table
##
## Response: BPSysAve
##             Df  Sum Sq Mean Sq F value Pr(>F)
## Race1        4   26934    6733   22.86 <2e-16 ***
## Residuals 8546 2516686     294
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

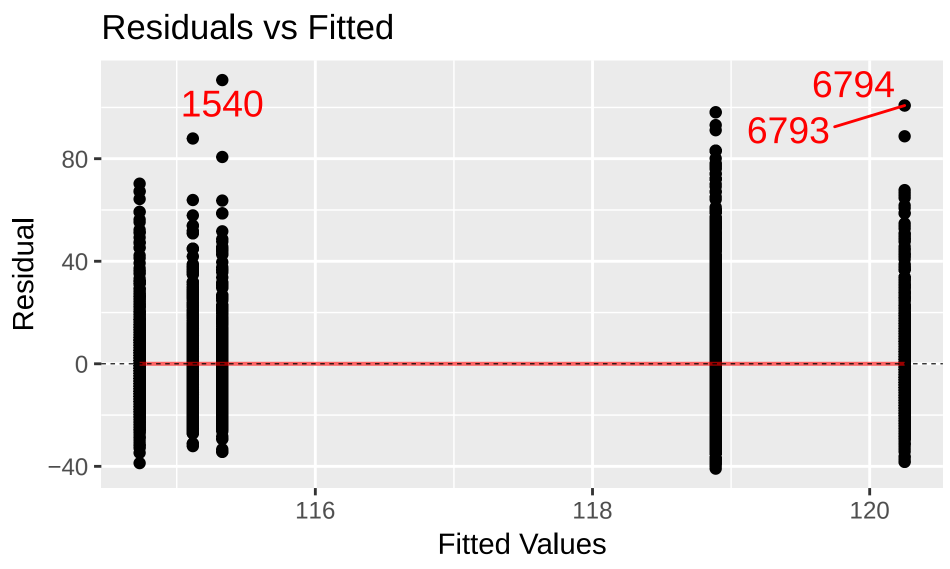
```
aovdirect <- aov(BPSysAve ~ Race1, data = NHANES)
summary(aovdirect) #has a little more information
```

```
##               Df  Sum Sq Mean Sq F value Pr(>F)
## Race1          4   26934    6733    22.9 <2e-16 ***
## Residuals   8546 2516686     294
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 1449 observations deleted due to missingness
```

Of course, we want to check conditions as well, so we return to using *mplot*.

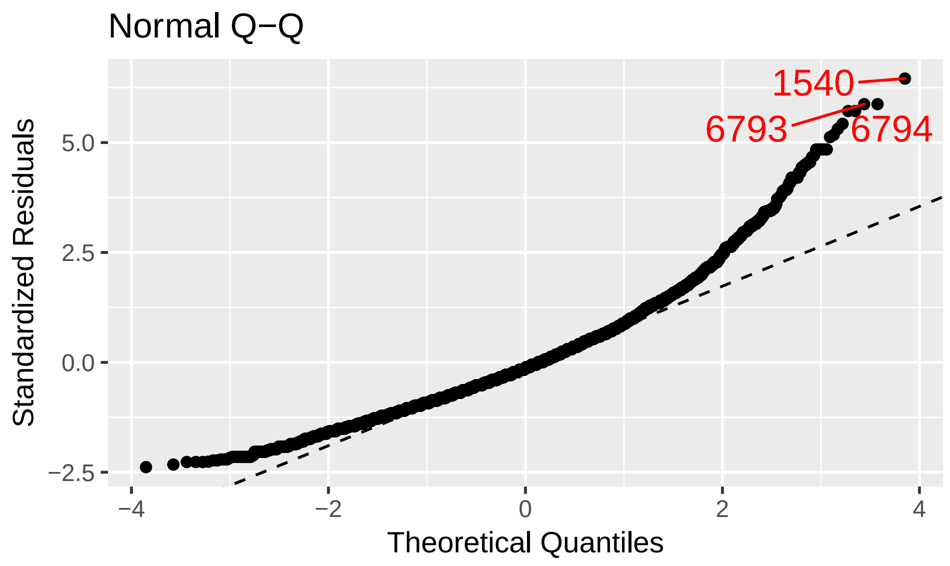
```
mplot(anovamod, which = 1)
```

```
## [[1]]
```



```
mplot(anovamod, which = 2)
```

```
## [[1]]
```



For descriptive statistics, recall we can use favstats to get summaries by group.

```
favstats(BPSysAve ~ Race1, data = NHANES)
```

```
##      Race1 min  Q1 median  Q3 max    mean      sd    n missing
## 1    Black  82 108    118 130 221 120.252 18.3226  991     206
## 2 Hispanic  81 104    112 125 226 115.329 17.7590  499     111
## 3  Mexican  76 104    113 123 185 114.732 15.9593  814     201
## 4    White  78 107    117 128 217 118.889 17.1629 5593     779
## 5    Other  83 104    112 125 203 115.116 16.2761  654     152
```

Side-by-side boxplots are also useful for checking conditions and seeing if the procedure is even necessary.

Multiple Comparisons (5.4 and 7.2)

If we believe the conditions are met, and we've determined that we believe at least one significant difference in means is present, we can use multiple comparisons procedures to find the difference. Here, we illustrate code used by your textbook as well as a function from a different package *DescTools* that does the same computations. Three multiple comparisons methods are covered in your text

```
# Book Code
with(NHANES, pairwise.t.test(BPSysAve, Race1, p.adj = "none")) #Fisher's
```

```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  BPSysAve and Race1
##
##          Black Hispanic Mexican White
## Hispanic 2e-07 -        -       -
## Mexican  1e-11 0.54     -       -
## White    0.02  9e-06    1e-10   -
## Other    3e-09 0.84     0.67    1e-07
##
## P value adjustment method: none
```

```r
with(NHANES, pairwise.t.test(BPSysAve, Race1, p.adj = "bonf")) #Bonferroni
```

```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  BPSysAve and Race1
##
##          Black Hispanic Mexican White
## Hispanic 2e-06 -        -       -
## Mexican  1e-10 1.0      -       -
## White    0.2   9e-05    1e-09   -
## Other    3e-08 1.0      1.0     1e-06
##
## P value adjustment method: bonferroni
```

```r
TukeyHSD(anovamod) #just feed it the model, Tukey's
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = x)
##
## $Race1
##                        diff      lwr       upr    p adj
## Hispanic-Black   -4.923613 -7.49366 -2.353571 0.000002
## Mexican-Black    -5.520084 -7.73483 -3.305338 0.000000
## White-Black      -1.363302 -2.97699  0.250387 0.143274
## Other-Black      -5.136062 -7.49486 -2.777260 0.000000
## Mexican-Hispanic -0.596471 -3.25845  2.065506 0.973350
## White-Hispanic    3.560311  1.37284  5.747778 0.000089
## Other-Hispanic   -0.212449 -2.99543  2.570529 0.999582
## White-Mexican     4.156782  2.40037  5.913196 0.000000
## Other-Mexican     0.384021 -2.07463  2.842669 0.993130
## Other-White      -3.772760 -5.70766 -1.837859 0.000001
```

The new function that you may like due to the structure is PostHocTest. You can call this on an aov object, or do the anova in line.

```r
myanova <- aov(BPSysAve ~ Race1, data = NHANES)
PostHocTest(myanova, method = "lsd") #use on an object
```

```
##
##   Posthoc multiple comparisons of means : Fisher LSD
##     95% family-wise confidence level
##
## $Race1
##                        diff   lwr.ci    upr.ci    pval
## Hispanic-Black   -4.923613 -6.77011 -3.077116 1.8e-07 ***
## Mexican-Black    -5.520084 -7.11131 -3.928856 1.1e-11 ***
## White-Black      -1.363302 -2.52269 -0.203915  0.0212 *
## Other-Black      -5.136062 -6.83079 -3.441335 2.9e-09 ***
## Mexican-Hispanic -0.596471 -2.50902  1.316078  0.5410
## White-Hispanic    3.560311  1.98868  5.131940 9.1e-06 ***
## Other-Hispanic   -0.212449 -2.21193  1.787036  0.8350
## White-Mexican     4.156782  2.89485  5.418712 1.1e-10 ***
## Other-Mexican     0.384021 -1.38244  2.150485  0.6700
## Other-White      -3.772760 -5.16293 -2.382593 1.1e-07 ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
PostHocTest(aov(BPSysAve ~ Race1, data = NHANES), method = "lsd") #done inline; Fisher's LSD
```

```
##
##   Posthoc multiple comparisons of means : Fisher LSD
##     95% family-wise confidence level
##
## $Race1
##                       diff    lwr.ci    upr.ci    pval
## Hispanic-Black   -4.923613 -6.77011 -3.077116 1.8e-07 ***
## Mexican-Black    -5.520084 -7.11131 -3.928856 1.1e-11 ***
## White-Black      -1.363302 -2.52269 -0.203915  0.0212 *
## Other-Black      -5.136062 -6.83079 -3.441335 2.9e-09 ***
## Mexican-Hispanic -0.596471 -2.50902  1.316078  0.5410
## White-Hispanic    3.560311  1.98868  5.131940 9.1e-06 ***
## Other-Hispanic   -0.212449 -2.21193  1.787036  0.8350
## White-Mexican     4.156782  2.89485  5.418712 1.1e-10 ***
## Other-Mexican     0.384021 -1.38244  2.150485  0.6700
## Other-White      -3.772760 -5.16293 -2.382593 1.1e-07 ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Changing methods is just changing the option.

```r
PostHocTest(aov(BPSysAve ~ Race1, data = NHANES), method = "hsd") #Tukey's HSD
```

```
##
##   Posthoc multiple comparisons of means : Tukey HSD
##     95% family-wise confidence level
##
## $Race1
##                       diff    lwr.ci    upr.ci    pval
## Hispanic-Black   -4.923613 -7.49366 -2.353571 1.8e-06 ***
## Mexican-Black    -5.520084 -7.73483 -3.305338 1.3e-10 ***
## White-Black      -1.363302 -2.97699  0.250387  0.1433
## Other-Black      -5.136062 -7.49486 -2.777260 2.9e-08 ***
## Mexican-Hispanic -0.596471 -3.25845  2.065506  0.9734
## White-Hispanic    3.560311  1.37284  5.747778 8.9e-05 ***
## Other-Hispanic   -0.212449 -2.99543  2.570529  0.9996
## White-Mexican     4.156782  2.40037  5.913196 1.1e-09 ***
## Other-Mexican     0.384021 -2.07463  2.842669  0.9931
## Other-White      -3.772760 -5.70766 -1.837859 1.1e-06 ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

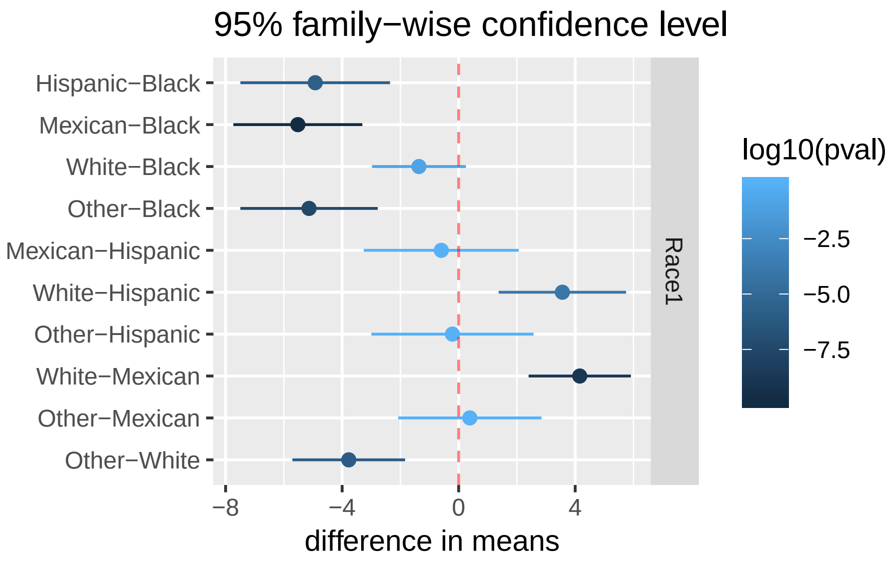
```r
PostHocTest(aov(BPSysAve ~ Race1, data = NHANES), method = "bonf") #Bonferroni
```

```
##
##   Posthoc multiple comparisons of means : Bonferroni
##     95% family-wise confidence level
##
```

```
## $Race1
##                        diff    lwr.ci    upr.ci    pval
## Hispanic-Black   -4.923613 -7.56845 -2.278772 1.8e-06 ***
## Mexican-Black    -5.520084 -7.79929 -3.240880 1.1e-10 ***
## White-Black      -1.363302 -3.02396  0.297352  0.2119
## Other-Black      -5.136062 -7.56352 -2.708610 2.9e-08 ***
## Mexican-Hispanic -0.596471 -3.33592  2.142980  1.0000
## White-Hispanic    3.560311  1.30918  5.811442 9.1e-05 ***
## Other-Hispanic   -0.212449 -3.07642  2.651524  1.0000
## White-Mexican     4.156782  2.34925  5.964315 1.1e-09 ***
## Other-Mexican     0.384021 -2.14618  2.914225  1.0000
## Other-White      -3.772760 -5.76397 -1.781546 1.1e-06 ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

There are other ways you can obtain and visualize these results too, at least with Tukey's HSD. Here, we see 95% CIs plotted. Note how there is a dashed red line at 0 to help you see which CIs miss 0.

```
mplot(TukeyHSD(lm(BPSysAve ~ Race1, data = NHANES)))
```



Two-Way ANOVA (Chapter 6)

With Two-Way ANOVA, we have two categorical predictors. There are two models to consider, the additive model and the model with interaction.

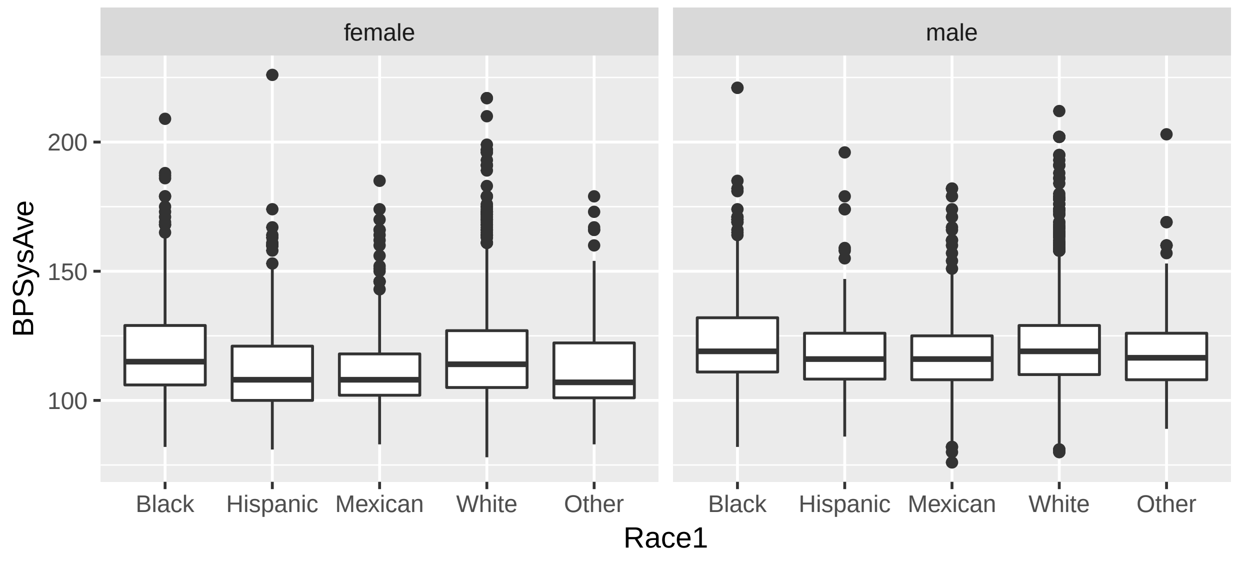
Additive Model (6.1)

Remember that examining the relationship between the response and the predictors is very important.

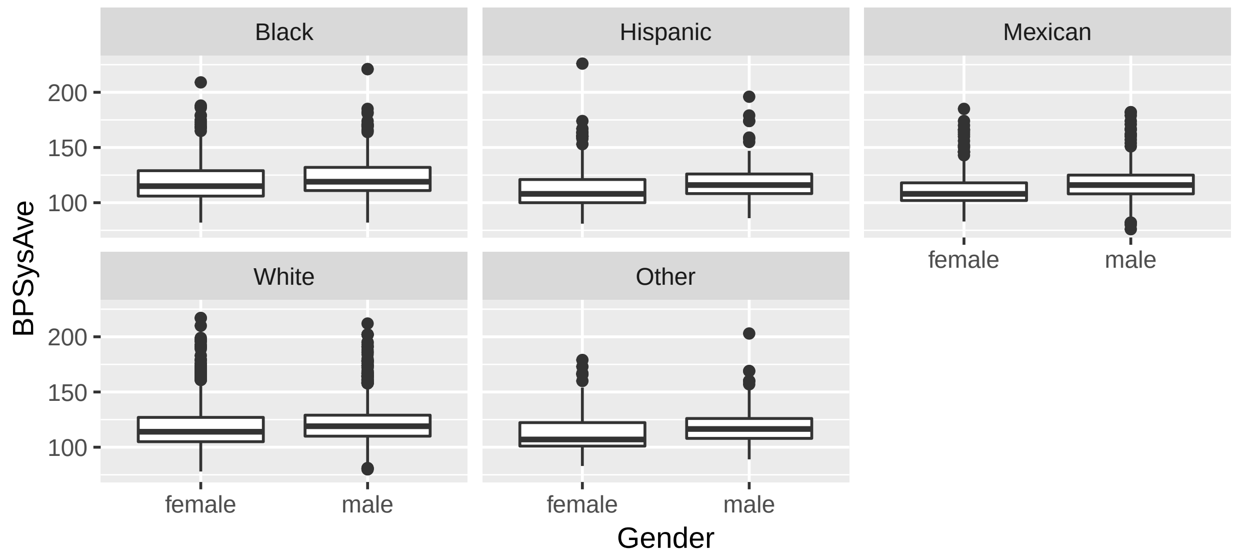
```
favstats(BPSysAve ~ Race1 + Gender, data = NHANES)
```

```
##       Race1.Gender min     Q1 median     Q3 max    mean      sd   n
## 1     Black.female  82 106.00  115.0 129.00 209 118.751 18.3268 514
## 2  Hispanic.female  81 100.00  108.0 121.00 226 112.475 18.6084 265
## 3   Mexican.female  83 102.00  108.0 118.00 185 111.347 15.6289  349
## 4     White.female  78 105.00  114.0 127.00 217 117.338 17.7838 2847
## 5     Other.female  83 101.00  107.0 122.25 179 111.843 16.7198  324
## 6       Black.male  82 111.00  119.0 132.00 221 121.870 18.1989  477
## 7    Hispanic.male  86 108.25  116.0 126.00 196 118.560 16.1861  234
## 8     Mexican.male  76 108.00  116.0 125.00 182 117.273 15.7475  465
## 9       White.male  80 110.00  119.0 129.00 212 120.497 16.3429 2746
## 10      Other.male  89 108.00  116.5 126.00 203 118.330 15.1799  330
##    missing
## 1      100
## 2       55
## 3      103
## 4      374
## 5       89
## 6      106
## 7       56
## 8       98
## 9      405
## 10      63
```

```r
gf_boxplot(BPSysAve ~ Race1 | Gender, data = NHANES)
```



```r
gf_boxplot(BPSysAve ~ Gender | Race1, data = NHANES)
```



We fit the model and check conditions using the usual commands.

```
twowaymod <- lm(BPSysAve ~ Race1 + Gender, data = NHANES)
msummary(twowaymod) #note this isn't as useful now because we need separate F statistics
```

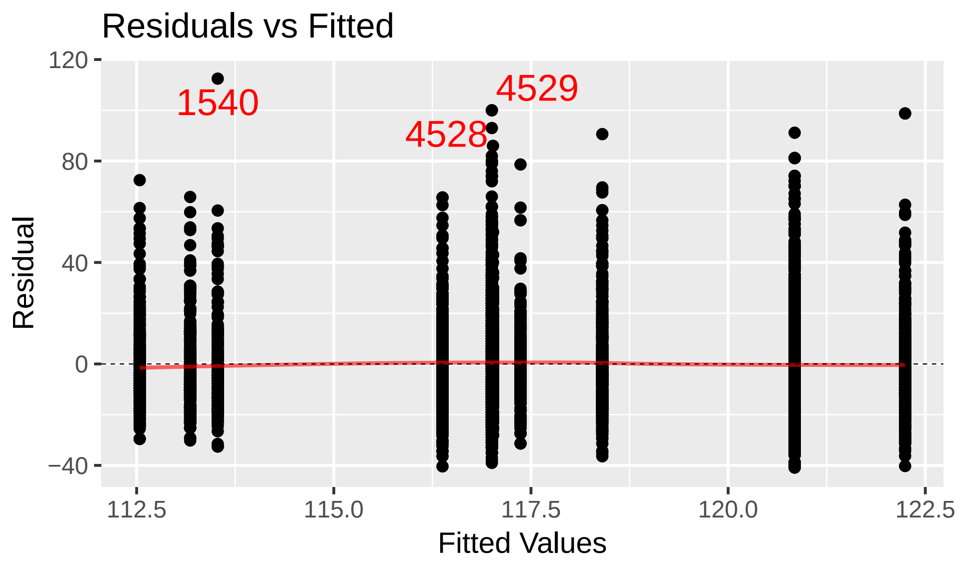
```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     118.404      0.570  207.67  < 2e-16 ***
## Race1Hispanic    -4.876      0.936   -5.21  1.9e-07 ***
## Race1Mexican     -5.865      0.807   -7.26  4.1e-13 ***
## Race1White       -1.400      0.588   -2.38    0.017 *
## Race1Other       -5.225      0.859   -6.08  1.2e-09 ***
## Gendermale        3.839      0.369   10.40  < 2e-16 ***
##
## Residual standard error: 17.1 on 8545 degrees of freedom
##   (1449 observations deleted due to missingness)
## Multiple R-squared:  0.0229, Adjusted R-squared:  0.0224
## F-statistic: 40.1 on 5 and 8545 DF,  p-value: <2e-16
```

```
anova(twowaymod) #works on the lm object to get the desired table
```

```
## Analysis of Variance Table
##
## Response: BPSysAve
##             Df  Sum Sq Mean Sq F value Pr(>F)
## Race1        4   26934    6733   23.15 <2e-16 ***
## Gender       1   31433   31433  108.07 <2e-16 ***
## Residuals 8545 2485253     291
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

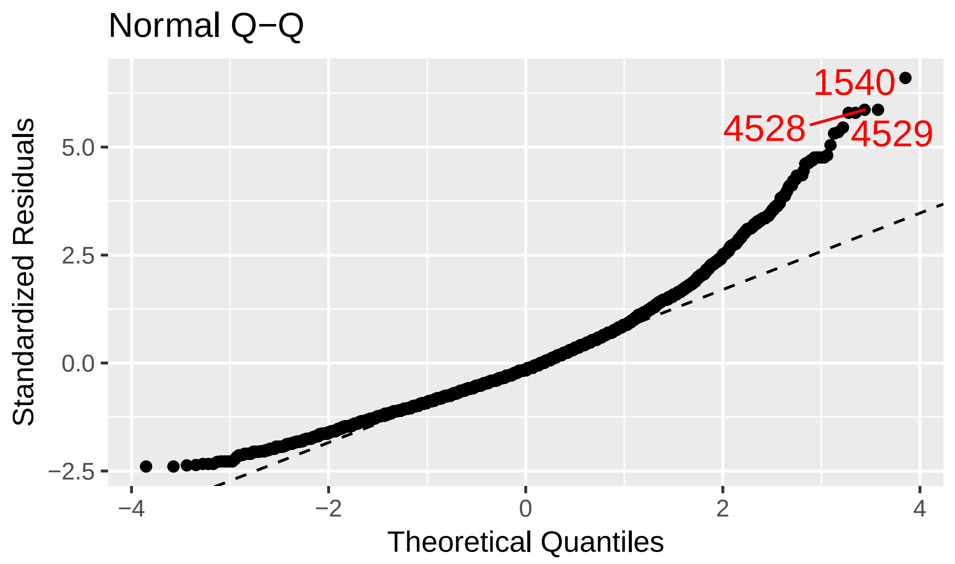
```
mplot(twowaymod, which = 1)
```

```
## [[1]]
```



```r
mplot(twowaymod, which = 2)
```

```
## [[1]]
```



I will also demo a few ways to get other tables that might be useful.

```r
#if you want to save the model as an ANOVA object
aovmod <- aov(BPSysAve ~ Race1 + Gender, data = NHANES)
summary(aovmod) #get the ANOVA summary table
```

```
##                Df  Sum Sq Mean Sq F value Pr(>F)    
## Race1           4   26934    6733    23.1 <2e-16 ***
## Gender          1   31433   31433   108.1 <2e-16 ***
## Residuals    8545 2485253     291                   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 1449 observations deleted due to missingness
```

```
model.tables(aovmod) #more summary tables; to get effect summaries
```

```
## Tables of effects
##
##  Race1
##       Black Hispanic Mexican    White   Other
##       2.097   -2.826  -3.423    0.734  -3.039
## rep 991.000  499.000 814.000 5593.000 654.000
##
##  Gender
##       female     male
##       -1.904    1.925
## rep 4299.000 4252.000
```

```
model.tables(aovmod, type = "means")
```

```
## Tables of means
## Grand mean
##
## 118.155
##
##  Race1
##     Black Hispanic Mexican  White Other
##     120.3    115.3   114.7  118.9 115.1
## rep 991.0    499.0   814.0 5593.0 654.0
##
##  Gender
##     female   male
##      116.3  120.1
## rep 4299.0 4252.0
```

Finally, for multiple comparisons, Tukey's method is still easily applied, and PostHocTest can be used as well. pairwise.t.test should NOT be used, because it only works with one categorical predictor.

```
TukeyHSD(twowaymod) #just feed it the model
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = x)
##
## $Race1
##                        diff      lwr       upr    p adj
## Hispanic-Black   -4.923613 -7.47771 -2.369521 0.000001
## Mexican-Black    -5.520084 -7.72108 -3.319083 0.000000
## White-Black      -1.363302 -2.96698  0.240372 0.138742
## Other-Black      -5.136062 -7.48023 -2.791900 0.000000
## Mexican-Hispanic -0.596471 -3.24193  2.048985 0.972736
## White-Hispanic    3.560311  1.38642  5.734202 0.000078
## Other-Hispanic   -0.212449 -2.97816  2.553257 0.999572
## White-Mexican     4.156782  2.41127  5.902295 0.000000
## Other-Mexican     0.384021 -2.05937  2.827410 0.992964
## Other-White      -3.772760 -5.69565 -1.849868 0.000001
##
## $Gender
##                 diff     lwr     upr p adj
## male-female 3.82977 3.10672 4.55282     0
```

```r
PostHocTest(aov(BPSysAve ~ Race1 + Gender, data = NHANES), method = "lsd") #Fisher's LSD
```

```
##
##   Posthoc multiple comparisons of means : Fisher LSD
##     95% family-wise confidence level
##
## $Race1
##                        diff   lwr.ci    upr.ci    pval
## Hispanic-Black   -4.923613 -6.75865 -3.088576 1.5e-07 ***
## Mexican-Black    -5.520084 -7.10144 -3.938732 8.3e-12 ***
## White-Black      -1.363302 -2.51549 -0.211111  0.0204 *
## Other-Black      -5.136062 -6.82027 -3.451853 2.4e-09 ***
## Mexican-Hispanic -0.596471 -2.49715  1.304208  0.5385
## White-Hispanic    3.560311  1.99844  5.122186 8.0e-06 ***
## Other-Hispanic   -0.212449 -2.19953  1.774626  0.8340
## White-Mexican     4.156782  2.90268  5.410880 8.6e-11 ***
## Other-Mexican     0.384021 -1.37148  2.139521  0.6681
## Other-White      -3.772760 -5.15430 -2.391221 8.9e-08 ***
##
## $Gender
##                 diff  lwr.ci  upr.ci   pval
## male-female 3.82977 3.10672 4.55282 <2e-16 ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
PostHocTest(aov(BPSysAve ~ Race1 + Gender, data = NHANES), method = "bonf")
```

```
##
##   Posthoc multiple comparisons of means : Bonferroni
##     95% family-wise confidence level
##
## $Race1
##                        diff   lwr.ci    upr.ci    pval
## Hispanic-Black   -4.923613 -7.55204 -2.295187 1.5e-06 ***
## Mexican-Black    -5.520084 -7.78514 -3.255026 8.3e-11 ***
## White-Black      -1.363302 -3.01365  0.287046  0.2040
## Other-Black      -5.136062 -7.54845 -2.723675 2.4e-08 ***
## Mexican-Hispanic -0.596471 -3.31892  2.125978  1.0000
## White-Hispanic    3.560311  1.32315  5.797471 8.0e-05 ***
## Other-Hispanic   -0.212449 -3.05865  2.633750  1.0000
## White-Mexican     4.156782  2.36047  5.953097 8.6e-10 ***
## Other-Mexican     0.384021 -2.13048  2.898522  1.0000
## Other-White      -3.772760 -5.75162 -1.793904 8.9e-07 ***
##
## $Gender
##                 diff  lwr.ci  upr.ci   pval
## male-female 3.82977 3.10672 4.55282 <2e-16 ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

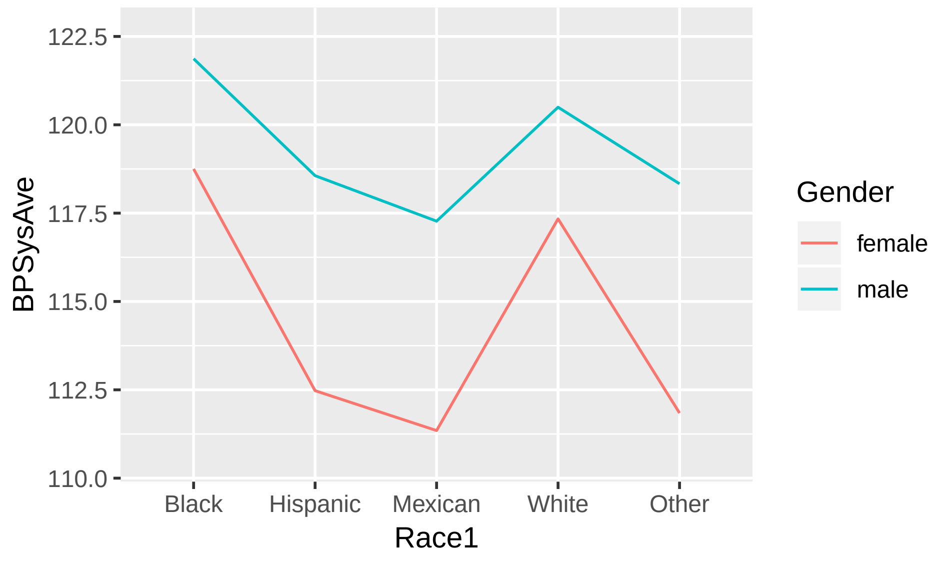
```r
PostHocTest(aov(BPSysAve ~ Race1 + Gender, data = NHANES), method = "hsd")
```

```
##
##   Posthoc multiple comparisons of means : Tukey HSD
##     95% family-wise confidence level
##
## $Race1
##                       diff    lwr.ci     upr.ci    pval
## Hispanic-Black   -4.923613 -7.47771 -2.369521 1.5e-06 ***
## Mexican-Black    -5.520084 -7.72108 -3.319083 1.0e-10 ***
## White-Black      -1.363302 -2.96698  0.240372  0.1387
## Other-Black      -5.136062 -7.48023 -2.791900 2.4e-08 ***
## Mexican-Hispanic -0.596471 -3.24193  2.048985  0.9727
## White-Hispanic    3.560311  1.38642  5.734202 7.8e-05 ***
## Other-Hispanic   -0.212449 -2.97816  2.553257  0.9996
## White-Mexican     4.156782  2.41127  5.902295 8.8e-10 ***
## Other-Mexican     0.384021 -2.05937  2.827410  0.9930
## Other-White      -3.772760 -5.69565 -1.849868 8.8e-07 ***
##
## $Gender
##                diff   lwr.ci  upr.ci  pval
## male-female 3.82977 3.10672 4.55282 2e-11 ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

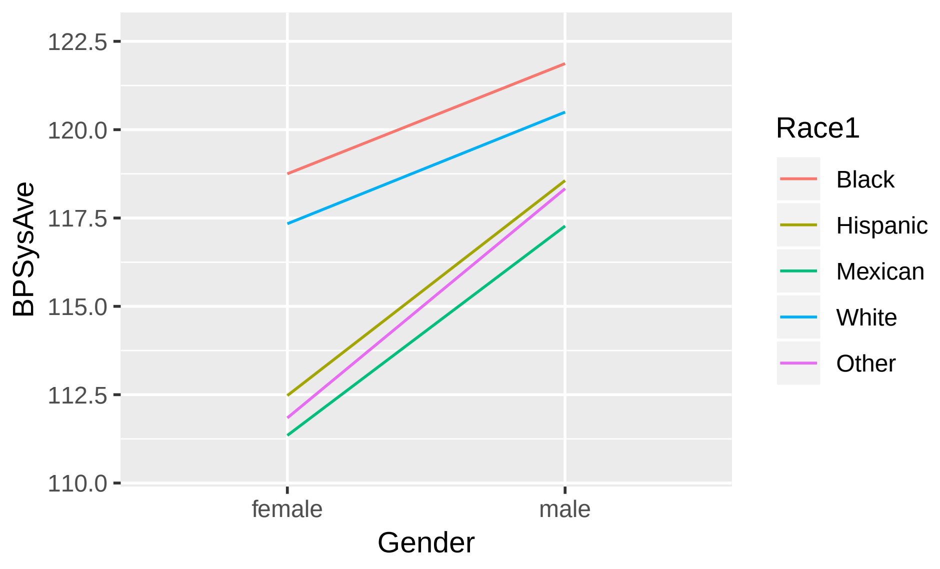
Model with Interaction (6.2-6.3)

If we believe there is interaction, we can fit a two-way model with interaction. To help see if an interaction term may be useful, we can use an interaction plot, shown below.

```
gf_line(BPSysAve ~ Race1, color = ~ Gender, data = NHANES, group = ~ Gender, stat = "summary")
```



```
# then reverse roles
gf_line(BPSysAve ~ Gender, color = ~ Race1, data = NHANES, group = ~ Race1, stat = "summary")
```



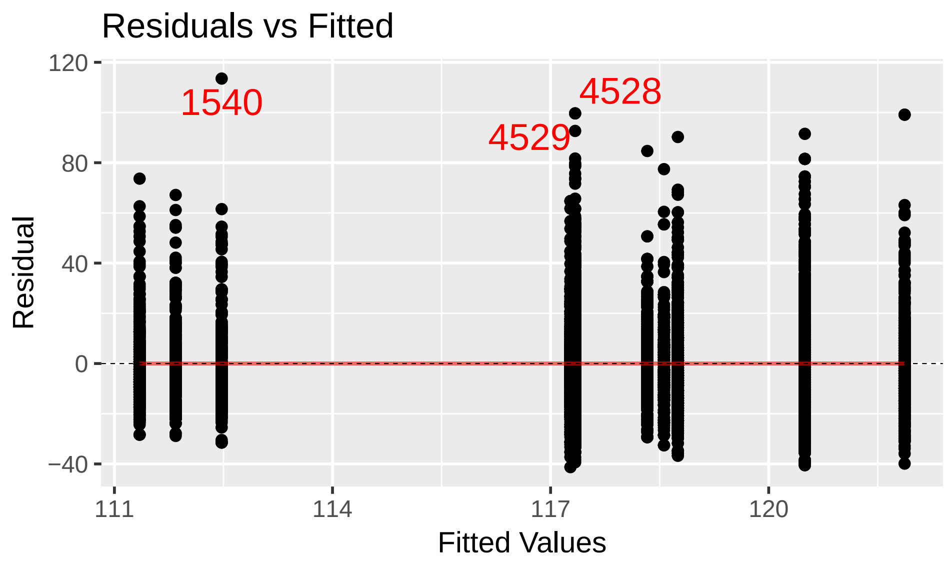
The lines do not appear to be parallel, so we may want to try an interaction term.

```
twowaymodint <- lm(BPSysAve ~ Race1 * Gender, data = NHANES)

anova(twowaymodint) #works on the lm object to get the desired table
```

```
## Analysis of Variance Table
##
## Response: BPSysAve
##                 Df  Sum Sq Mean Sq F value Pr(>F)
## Race1            4   26934    6733  23.173 <2e-16 ***
## Gender           1   31433   31433 108.172 <2e-16 ***
## Race1:Gender     4    3415     854   2.938 0.0193 *
## Residuals     8541 2481838     291
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

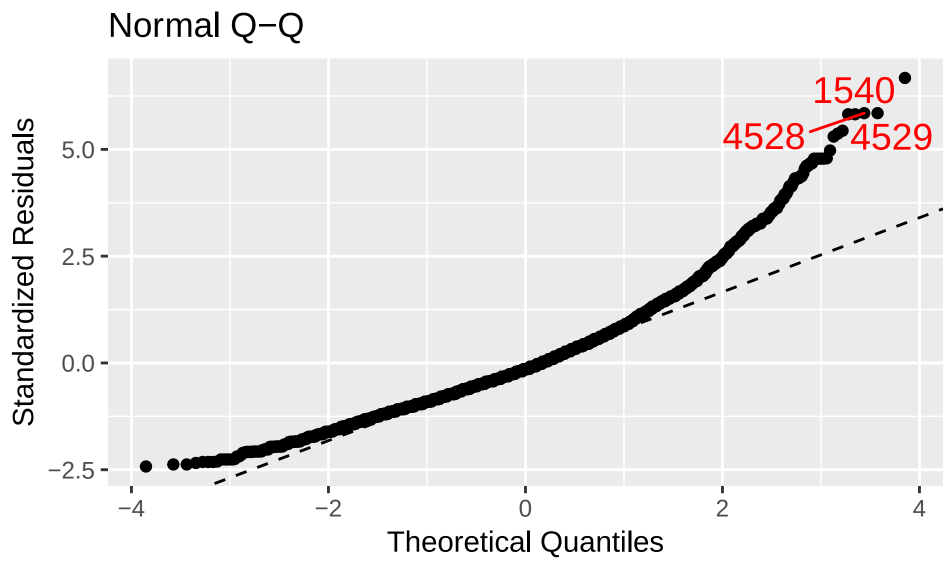
```
mplot(twowaymodint, which = 1)
```

```
## [[1]]
```



```
mplot(twowaymodint, which = 2)
```

```
## [[1]]
```



There appear to be issues with the conditions, but we demo how to perform multiple comparisons for the sake of the example.

```
TukeyHSD(twowaymodint) #just feed it the model
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = x)
##
## $Race1
##                        diff       lwr       upr    p adj
## Hispanic-Black    -4.923613 -7.47655 -2.370678 0.000001
## Mexican-Black     -5.520084 -7.72009 -3.320081 0.000000
## White-Black      -1.363302 -2.96625  0.239646 0.138415
## Other-Black      -5.136062 -7.47916 -2.792962 0.000000
## Mexican-Hispanic -0.596471 -3.24073  2.047786 0.972691
## White-Hispanic    3.560311  1.38741  5.733217 0.000077
## Other-Hispanic   -0.212449 -2.97690  2.552003 0.999571
## White-Mexican     4.156782  2.41206  5.901504 0.000000
## Other-Mexican     0.384021 -2.05826  2.826302 0.992952
## Other-White      -3.772760 -5.69478 -1.850739 0.000001
##
## $Gender
##                diff     lwr     upr p adj
## male-female 3.82977 3.10705 4.55249     0
##
## $`Race1:Gender`
##                                    diff        lwr       upr    p adj
## Hispanic:female-Black:female   -6.275501 -10.354985 -2.196017 0.000050
## Mexican:female-Black:female    -7.404268 -11.145823 -3.662713 0.000000
## White:female-Black:female      -1.413424  -3.998659  1.171810 0.778380
## Other:female-Black:female      -6.908380 -10.734944 -3.081816 0.000001
## Black:male-Black:female         3.119048  -0.310501  6.548598 0.111844
## Hispanic:male-Black:female     -0.191144  -4.445191  4.062904 1.000000
## Mexican:male-Black:female      -1.477854  -4.930280  1.974571 0.940576
## White:male-Black:female         1.746478  -0.846017  4.338973 0.504757
## Other:male-Black:female        -0.420670  -4.225836  3.384497 0.999999
## Mexican:female-Hispanic:female -1.128767  -5.524077  3.266543 0.998415
## White:female-Hispanic:female    4.862077   1.397548  8.326605 0.000386
## Other:female-Hispanic:female   -0.632879  -5.100777  3.835018 0.999989
## Black:male-Hispanic:female      9.394549   5.261593 13.527506 0.000000
## Hispanic:male-Hispanic:female   6.084357   1.245304 10.923411 0.002805
## Mexican:male-Hispanic:female    4.797647   0.645688  8.949605 0.009678
## White:male-Hispanic:female      8.021979   4.552029 11.491929 0.000000
## Other:male-Hispanic:female      5.854831   1.405246 10.304417 0.001304
## White:female-Mexican:female     5.990843   2.931432  9.050255 0.000000
## Other:female-Mexican:female     0.495888  -3.665741  4.657516 0.999998
## Black:male-Mexican:female      10.523316   6.723530 14.323102 0.000000
## Hispanic:male-Mexican:female    7.213124   2.655331 11.770918 0.000025
## Mexican:male-Mexican:female     5.926413   2.105968  9.746859 0.000041
## White:male-Mexican:female       9.150746   6.085196 12.216296 0.000000
## Other:male-Mexican:female       6.983598   2.841635 11.125561 0.000004
## Other:female-White:female      -5.494956  -8.657764 -2.332147 0.000002
## Black:male-White:female         4.532473   1.863657  7.201288 0.000004
## Hispanic:male-White:female      1.222281  -2.446191  4.890753 0.988751
## Mexican:male-White:female      -0.064430  -2.762579  2.633719 1.000000
## White:male-White:female         3.159903   1.717058  4.602747 0.000000
## Other:male-White:female         0.992755  -2.144132  4.129642 0.992243
## Black:male-Other:female        10.027428   6.143908 13.910949 0.000000
## Hispanic:male-Other:female      6.717236   2.089404 11.345069 0.000191
## Mexican:male-Other:female       5.430526   1.526789  9.334262 0.000460
## White:male-Other:female         8.654858   5.486112 11.823604 0.000000
## Other:male-Other:female         6.487710   2.268799 10.706622 0.000051
## Hispanic:male-Black:male       -3.310192  -7.615544  0.995160 0.306067
## Mexican:male-Black:male        -4.596903  -8.112352 -1.081454 0.001455
## White:male-Black:male          -1.372570  -4.048420  1.303279 0.837072
## Other:male-Black:male          -3.539718  -7.402157  0.322721 0.105552
## Mexican:male-Hispanic:male     -1.286711  -5.610308  3.036886 0.995080
## White:male-Hispanic:male        1.937622  -1.735970  5.611214 0.813005
## Other:male-Hispanic:male       -0.229526  -4.839682  4.380630 1.000000
## White:male-Mexican:male         3.224333   0.519226  5.929439 0.006303
## Other:male-Mexican:male         1.057185  -2.825580  4.939950 0.997500
## Other:male-White:male          -2.167148  -5.310022  0.975726 0.469258
```

```
PostHocTest(aov(BPSysAve ~ Race1 * Gender, data = NHANES), method = "lsd") #Fisher's LSD
```

```
##
##   Posthoc multiple comparisons of means : Fisher LSD
##     95% family-wise confidence level
##
## $Race1
##                        diff   lwr.ci    upr.ci    pval
## Hispanic-Black   -4.923613 -6.75782 -3.089407 1.5e-07 ***
## Mexican-Black    -5.520084 -7.10072 -3.939449 8.1e-12 ***
## White-Black      -1.363302 -2.51497 -0.211633  0.0203 *
## Other-Black      -5.136062 -6.81951 -3.452616 2.3e-09 ***
## Mexican-Hispanic -0.596471 -2.49629  1.303347  0.5383
## White-Hispanic    3.560311  1.99914  5.121478 7.9e-06 ***
## Other-Hispanic   -0.212449 -2.19862  1.773726  0.8339
## White-Mexican     4.156782  2.90325  5.410312 8.5e-11 ***
## Other-Mexican     0.384021 -1.37068  2.138726  0.6679
## Other-White      -3.772760 -5.15367 -2.391847 8.8e-08 ***
##
## $Gender
##                 diff  lwr.ci  upr.ci   pval
## male-female 3.82977 3.10705 4.55249 <2e-16 ***
##
## $`Race1:Gender`
##                                       diff    lwr.ci    upr.ci    pval
## Hispanic:female-Black:female   -6.275501 -8.802507 -3.748495 1.1e-06 ***
## Mexican:female-Black:female    -7.404268 -9.721946 -5.086590 4.0e-10 ***
## White:female-Black:female      -1.413424 -3.014829  0.187980 0.08364 .
## Other:female-Black:female      -6.908380 -9.278716 -4.538044 1.1e-08 ***
## Black:male-Black:female         3.119048  0.994639  5.243457 0.00401 **
## Hispanic:male-Black:female     -0.191144 -2.826281  2.443994 0.88693
## Mexican:male-Black:female      -1.477854 -3.616434  0.660725 0.17558
## White:male-Black:female         1.746478  0.140576  3.352380 0.03305 *
## Other:male-Black:female        -0.420670 -2.777752  1.936412 0.72646
## Mexican:female-Hispanic:female -1.128767 -3.851409  1.593875 0.41642
## White:female-Hispanic:female    4.862077  2.716000  7.008153 9.1e-06 ***
## Other:female-Hispanic:female   -0.632879 -3.400485  2.134727 0.65398
## Black:male-Hispanic:female      9.394549  6.834420 11.954678 6.9e-13 ***
## Hispanic:male-Hispanic:female   6.084357  3.086842  9.081873 7.0e-05 ***
## Mexican:male-Hispanic:female    4.797647  2.225747  7.369546 0.00026 ***
## White:male-Hispanic:female      8.021979  5.872545 10.171414 2.8e-13 ***
## Other:male-Hispanic:female      5.854831  3.098569  8.611094 3.2e-05 ***
## White:female-Mexican:female     5.990843  4.095714  7.885973 6.0e-10 ***
## Other:female-Mexican:female     0.495888 -2.082002  3.073777 0.70613
## Black:male-Mexican:female      10.523316  8.169567 12.877065 < 2e-16 ***
## Hispanic:male-Mexican:female    7.213124  4.389833 10.036415 5.6e-07 ***
## Mexican:male-Mexican:female     5.926413  3.559867  8.292960 9.3e-07 ***
## White:male-Mexican:female       9.150746  7.251814 11.049678 < 2e-16 ***
## Other:male-Mexican:female       6.983598  4.417890  9.549306 9.8e-08 ***
## Other:female-White:female      -5.494956 -7.454134 -3.535778 4.0e-08 ***
## Black:male-White:female         4.532473  2.879295  6.185651 7.9e-08 ***
## Hispanic:male-White:female      1.222281 -1.050127  3.494688 0.29174
## Mexican:male-White:female      -0.064430 -1.735778  1.606918 0.93977
## White:male-White:female         3.159903  2.266144  4.053662 4.5e-12 ***
## Other:male-White:female         0.992755 -0.950367  2.935876 0.31661
## Black:male-Other:female        10.027428  7.621811 12.433046 3.5e-16 ***
## Hispanic:male-Other:female      6.717236  3.850560  9.583913 4.4e-06 ***
## Mexican:male-Other:female       5.430526  3.012385  7.848666 1.1e-05 ***
## White:male-Other:female         8.654858  6.692002 10.617714 < 2e-16 ***
## Other:male-Other:female         6.487710  3.874337  9.101084 1.2e-06 ***
## Hispanic:male-Black:male       -3.310192 -5.977110 -0.643274 0.01499 *
## Mexican:male-Black:male        -4.596903 -6.774521 -2.419284 3.5e-05 ***
## White:male-Black:male          -1.372570 -3.030105  0.284965 0.10458
## Other:male-Black:male          -3.539718 -5.932277 -1.147159 0.00374 **
## Mexican:male-Hispanic:male     -1.286711 -3.964930  1.391509 0.34634
## White:male-Hispanic:male        1.937622 -0.337957  4.213201 0.09513 .
## Other:male-Hispanic:male       -0.229526 -3.085253  2.626201 0.87481
## White:male-Mexican:male         3.224333  1.548674  4.899991 0.00016 ***
## Other:male-Mexican:male         1.057185 -1.347965  3.462335 0.38892
## Other:male-White:male          -2.167148 -4.113977 -0.220318 0.02913 *
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
PostHocTest(aov(BPSysAve ~ Race1 * Gender, data = NHANES), method = "bonf")
```

```
##
##   Posthoc multiple comparisons of means : Bonferroni
##     95% family-wise confidence level
##
## $Race1
##                      diff   lwr.ci    upr.ci    pval
## Hispanic-Black   -4.923613 -7.55085 -2.296378 1.5e-06 ***
## Mexican-Black    -5.520084 -7.78412 -3.256052 8.1e-11 ***
## White-Black      -1.363302 -3.01290  0.286298  0.2034
## Other-Black      -5.136062 -7.54736 -2.724769 2.3e-08 ***
## Mexican-Hispanic -0.596471 -3.31769  2.124744  1.0000
## White-Hispanic    3.560311  1.32416  5.796457 7.9e-05 ***
## Other-Hispanic   -0.212449 -3.05736  2.632460  1.0000
## White-Mexican     4.156782  2.36128  5.952283 8.5e-10 ***
## Other-Mexican     0.384021 -2.12934  2.897382  1.0000
## Other-White      -3.772760 -5.75072 -1.794801 8.8e-07 ***
##
## $Gender
##                diff  lwr.ci  upr.ci   pval
## male-female 3.82977 3.10705 4.55249 <2e-16 ***
##
## $`Race1:Gender`
##                                    diff     lwr.ci    upr.ci    pval
## Hispanic:female-Black:female  -6.275501 -10.480485 -2.070517 5.2e-05 ***
## Mexican:female-Black:female   -7.404268 -11.260927 -3.547609 1.8e-08 ***
## White:female-Black:female     -1.413424  -4.078190  1.251341 1.00000
## Other:female-Black:female     -6.908380 -10.852663 -2.964097 5.2e-07 ***
## Black:male-Black:female         3.119048  -0.416007  6.654103 0.18053
## Hispanic:male-Black:female     -0.191144  -4.576061  4.193774 1.00000
## Mexican:male-Black:female      -1.477854  -5.036489  2.080780 1.00000
## White:male-Black:female         1.746478  -0.925772  4.418728 1.00000
## Other:male-Black:female        -0.420670  -4.342897  3.501558 1.00000
## Mexican:female-Hispanic:female -1.128767  -5.659293  3.401759 1.00000
## White:female-Hispanic:female    4.862077   1.290967  8.433187 0.00041 ***
## Other:female-Hispanic:female   -0.632879  -5.238226  3.972467 1.00000
## Black:male-Hispanic:female      9.394549   5.134448 13.654651 3.1e-11 ***
## Hispanic:male-Hispanic:female   6.084357   1.096437 11.072278 0.00314 **
## Mexican:male-Hispanic:female    4.797647   0.517958  9.077335 0.01157 *
## White:male-Hispanic:female      8.021979   4.445281 11.598677 1.3e-11 ***
## Other:male-Hispanic:female      5.854831   1.268360 10.441303 0.00142 **
## White:female-Mexican:female     5.990843   2.837313  9.144374 2.7e-08 ***
## Other:female-Mexican:female     0.495888  -3.793768  4.785543 1.00000
## Black:male-Mexican:female      10.523316   6.606635 14.439997 < 2e-16 ***
## Hispanic:male-Mexican:female    7.213124   2.515116 11.911132 2.5e-05 ***
## Mexican:male-Mexican:female     5.926413   1.988437  9.864390 4.2e-05 ***
## White:male-Mexican:female       9.150746   5.990889 12.310603 < 2e-16 ***
## Other:male-Mexican:female       6.983598   2.714213 11.252983 4.4e-06 ***
## Other:female-White:female      -5.494956  -8.755064 -2.234848 1.8e-06 ***
## Black:male-White:female         4.532473   1.781554  7.283391 3.6e-06 ***
## Hispanic:male-White:female      1.222281  -2.559047  5.003608 1.00000
## Mexican:male-White:female      -0.064430  -2.845584  2.716724 1.00000
## White:male-White:female         3.159903   1.672671  4.647134 2.0e-10 ***
## Other:male-White:female         0.992755  -2.240635  4.226144 1.00000
## Black:male-Other:female        10.027428   6.024437 14.030420 1.6e-14 ***
## Hispanic:male-Other:female      6.717236   1.947034 11.487438 0.00020 ***
## Mexican:male-Other:female       5.430526   1.406696  9.454356 0.00049 ***
## White:male-Other:female         8.654858   5.388630 11.921086 2.9e-16 ***
## Other:male-Other:female         6.487710   2.139010 10.836411 5.2e-05 ***
## Hispanic:male-Black:male       -3.310192  -7.747993  1.127609 0.67465
## Mexican:male-Black:male        -4.596903  -8.220500 -0.973306 0.00159 **
## White:male-Black:male          -1.372570  -4.130739  1.385598 1.00000
## Other:male-Black:male          -3.539718  -7.520979  0.441544 0.16829
## Mexican:male-Hispanic:male     -1.286711  -5.743317  3.169896 1.00000
## White:male-Hispanic:male        1.937622  -1.848984  5.724227 1.00000
## Other:male-Hispanic:male       -0.229526  -4.981508  4.522456 1.00000
## White:male-Mexican:male         3.224333   0.436007  6.012658 0.00734 **
## Other:male-Mexican:male         1.057185  -2.945028  5.059398 1.00000
## Other:male-White:male          -2.167148  -5.406708  1.072412 1.00000
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
PostHocTest(aov(BPSysAve ~ Race1 * Gender, data = NHANES), method = "hsd")
```

```
##
##   Posthoc multiple comparisons of means : Tukey HSD
##     95% family-wise confidence level
##
## $Race1
##                       diff   lwr.ci    upr.ci    pval
## Hispanic-Black   -4.923613 -7.47655 -2.370678 1.5e-06 ***
## Mexican-Black    -5.520084 -7.72009 -3.320081 1.0e-10 ***
## White-Black      -1.363302 -2.96625  0.239646  0.1384
## Other-Black      -5.136062 -7.47916 -2.792962 2.3e-08 ***
## Mexican-Hispanic -0.596471 -3.24073  2.047786  0.9727
## White-Hispanic    3.560311  1.38741  5.733217 7.7e-05 ***
## Other-Hispanic   -0.212449 -2.97690  2.552003  0.9996
## White-Mexican     4.156782  2.41206  5.901504 8.7e-10 ***
## Other-Mexican     0.384021 -2.05826  2.826302  0.9930
## Other-White      -3.772760 -5.69478 -1.850739 8.7e-07 ***
##
## $Gender
##                diff  lwr.ci  upr.ci    pval
## male-female 3.82977 3.10705 4.55249 2.3e-11 ***
##
## $`Race1:Gender`
##                                    diff     lwr.ci    upr.ci    pval
## Hispanic:female-Black:female   -6.275501 -10.354985 -2.196017 5.0e-05 ***
## Mexican:female-Black:female    -7.404268 -11.145823 -3.662713 1.8e-08 ***
## White:female-Black:female      -1.413424  -3.998659  1.171810 0.77838
## Other:female-Black:female      -6.908380 -10.734944 -3.081816 5.1e-07 ***
## Black:male-Black:female         3.119048  -0.310501  6.548598 0.11184
## Hispanic:male-Black:female     -0.191144  -4.445191  4.062904 1.00000
## Mexican:male-Black:female      -1.477854  -4.930280  1.974571 0.94058
## White:male-Black:female         1.746478  -0.846017  4.338973 0.50476
## Other:male-Black:female        -0.420670  -4.225836  3.384497 1.00000
## Mexican:female-Hispanic:female -1.128767  -5.524077  3.266543 0.99842
## White:female-Hispanic:female    4.862077   1.397548  8.326605 0.00039 ***
## Other:female-Hispanic:female   -0.632879  -5.100777  3.835018 0.99999
## Black:male-Hispanic:female      9.394549   5.261593 13.527506 5.4e-11 ***
## Hispanic:male-Hispanic:female   6.084357   1.245304 10.923411 0.00280 **
## Mexican:male-Hispanic:female    4.797647   0.645688  8.949605 0.00968 **
## White:male-Hispanic:female      8.021979   4.552029 11.491929 3.6e-11 ***
## Other:male-Hispanic:female      5.854831   1.405246 10.304417 0.00130 **
## White:female-Mexican:female     5.990843   2.931432  9.050255 2.7e-08 ***
## Other:female-Mexican:female     0.495888  -3.665741  4.657516 1.00000
## Black:male-Mexican:female      10.523316   6.723530 14.323102 2.3e-11 ***
## Hispanic:male-Mexican:female    7.213124   2.655331 11.770918 2.5e-05 ***
## Mexican:male-Mexican:female     5.926413   2.105968  9.746859 4.1e-05 ***
## White:male-Mexican:female       9.150746   6.085196 12.216296 2.3e-11 ***
## Other:male-Mexican:female       6.983598   2.841635 11.125561 4.3e-06 ***
## Other:female-White:female      -5.494956  -8.657764 -2.332147 1.8e-06 ***
## Black:male-White:female         4.532473   1.863657  7.201288 3.5e-06 ***
## Hispanic:male-White:female      1.222281  -2.446191  4.890753 0.98875
## Mexican:male-White:female      -0.064430  -2.762579  2.633719 1.00000
## White:male-White:female         3.159903   1.717058  4.602747 2.3e-10 ***
## Other:male-White:female         0.992755  -2.144132  4.129642 0.99224
## Black:male-Other:female        10.027428   6.143908 13.910949 2.3e-11 ***
## Hispanic:male-Other:female      6.717236   2.089404 11.345069 0.00019 ***
## Mexican:male-Other:female       5.430526   1.526789  9.334262 0.00046 ***
## White:male-Other:female         8.654858   5.486112 11.823604 2.3e-11 ***
## Other:male-Other:female         6.487710   2.268799 10.706622 5.1e-05 ***
## Hispanic:male-Black:male       -3.310192  -7.615544  0.995160 0.30607
## Mexican:male-Black:male        -4.596903  -8.112352 -1.081454 0.00146 **
## White:male-Black:male          -1.372570  -4.048420  1.303279 0.83707
## Other:male-Black:male          -3.539718  -7.402157  0.322721 0.10555
## Mexican:male-Hispanic:male     -1.286711  -5.610308  3.036886 0.99508
## White:male-Hispanic:male        1.937622  -1.735970  5.611214 0.81300
## Other:male-Hispanic:male       -0.229526  -4.839682  4.380630 1.00000
## White:male-Mexican:male         3.224333   0.519226  5.929439 0.00630 **
## Other:male-Mexican:male         1.057185  -2.825580  4.939950 0.99750
## Other:male-White:male          -2.167148  -5.310022  0.975726 0.46926
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Design (Chapter 8, 8.2)

Three of the sections here cover theoretical concepts. Only section 8.2 contains material that references R code, and the book doesn't demonstrate any of the code. Basically, we are applying randomization tests (like those from 4.5) but in an ANOVA setting, so the code will appear similar to that.

Randomization Test for Main Effect from Text

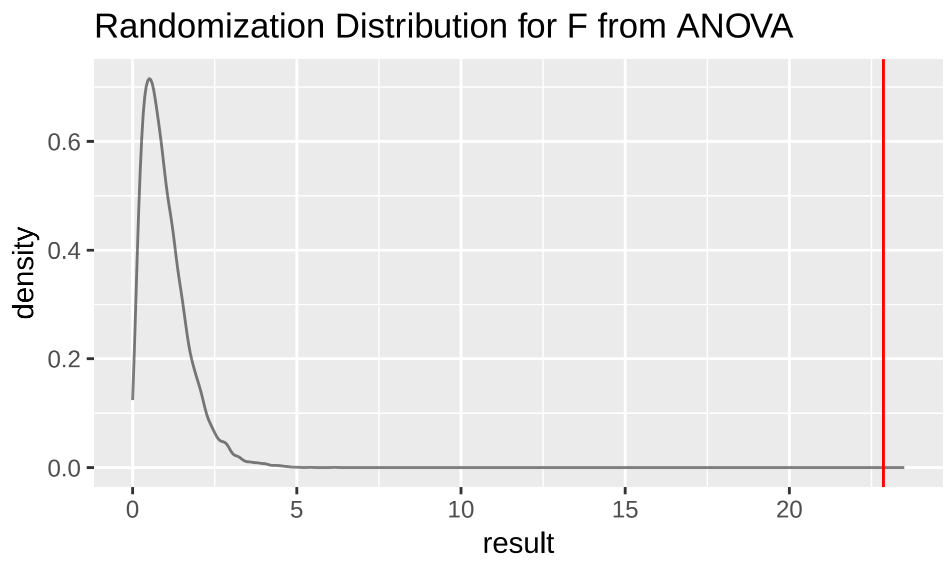
We use the one-way ANOVA model previously considered first.

```r
anovamod <- lm(BPSysAve ~ Race1, data = NHANES)
obsF <- anova(anovamod)$"F value"[1]
obsF
```

```
## [1] 22.865
```

Now, we randomize:

```r
set.seed(40) #make the results reproducible
anovaresult <-do(10000)*(anova(lm(BPSysAve ~ shuffle(Race1), data = NHANES))$"F value"[1])


gf_dens(~ result, data = anovaresult) %>%
  gf_lims(x = c(0, 23.5)) %>%
  gf_labs(title = "Randomization Distribution for F from ANOVA") %>%
  gf_vline(xintercept = ~ obsF, color = "red")
```



```
pdata(~ result, obsF, data = anovaresult, lower.tail = FALSE)
```

```
## [1] 0
```

We can see that the observed F value is well removed from the randomization distribution, with an empirical p-value of 0.

You can study two-way ANOVA in the same way, shuffling one predictor at a time. However, if you want to study an interaction, then please ask for assistance. The key is making sure that you are generating the randomization distribution correctly and that will need to be done differently if you are examining an interaction and want to leave the main effects intact. The example below illustrates one way this can be done for an interaction effect. Again, please ASK for assistance for these cases.

```
twowaymodint <- lm(BPSysAve ~ Race1 * Gender, data = NHANES)
anova(twowaymodint)
```

```
## Analysis of Variance Table
##
## Response: BPSysAve
##                Df  Sum Sq Mean Sq F value Pr(>F)
## Race1           4   26934    6733  23.173 <2e-16 ***
## Gender          1   31433   31433 108.172 <2e-16 ***
## Race1:Gender    4    3415     854   2.938 0.0193 *
## Residuals    8541 2481838     291
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
obsF <- anova(twowaymodint)$"F value"[3] # store the interaction F statistic
obsF
```

```
## [1] 2.93814
```

Here, we realize that not all observations were included in the analysis, so we focus the randomization procedure on only those observations included. Here, we wrote a function to obtain the permuted interaction F statistics, and used do to repeat the procedure. This first chunk sets up the data and interaction function.

```
newNHANES <- augment(twowaymodint) %>% select("BPSysAve", "Race1", "Gender")
twowaymod <- lm(BPSysAve ~ Race1 + Gender, data = newNHANES) #main effects only
residuals <- resid(twowaymod)
```

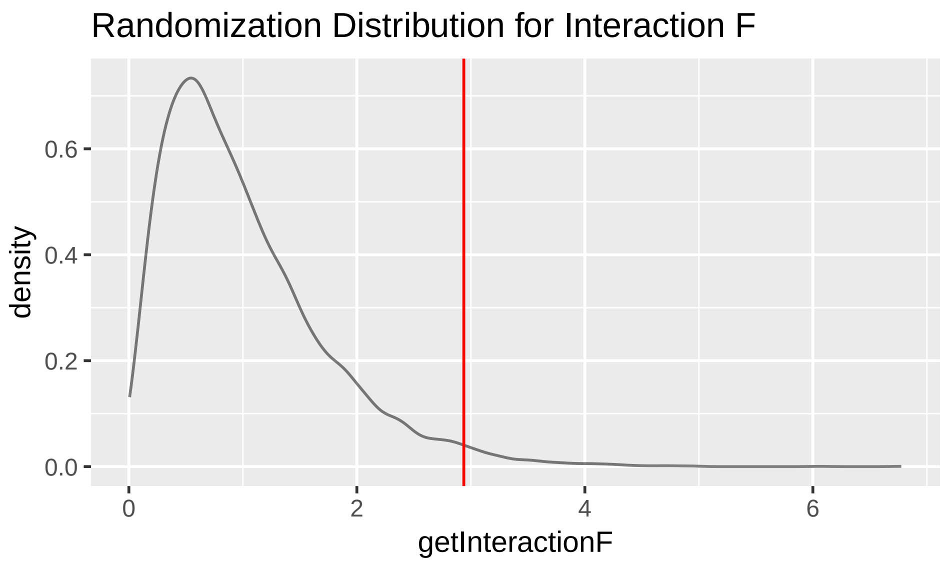
```
fitted <- fitted(twowaymod)
newAugData <- augment(twowaymod)

#run all at once to get the function
getInteractionF <- function(){
  tempres <- shuffle(newAugData$.resid)
  tempresponse <- newAugData$.fitted + tempres
  modtemp <- lm(tempresponse ~ Race1 * Gender, data = newAugData)
  anova(modtemp)$"F value"[3] #to get the interaction F
}
```

Now that we have the function, we set a seed and run it many times!

```
set.seed(230)
permF <- do(10000)*getInteractionF()
```

And now we can look at our results!

```
gf_dens(~ getInteractionF, data = permF) %>%
  gf_labs(title = "Randomization Distribution for Interaction F") %>%
  gf_vline(xintercept = ~ obsF, color = "red")
```



```
pdata(~ getInteractionF, obsF, data = permF, lower.tail = FALSE)
```

```
## [1] 0.0199
```

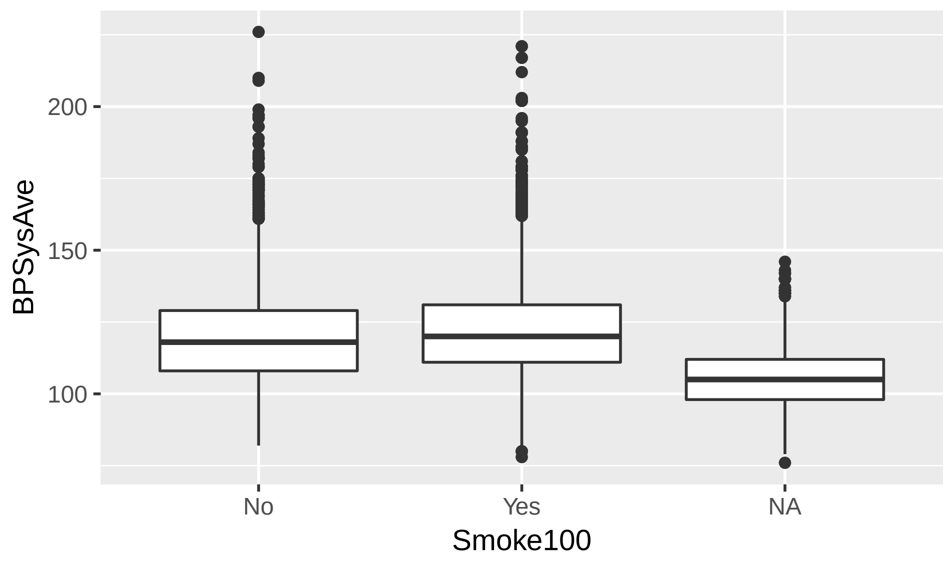
Again, for these more complicated examples, please ask for assistance!

Simple Logistic Regression (Chapter 9)

In logistic regression, we encounter a categorical (binary) response variable. In the simplest case, as in the settingl of chapter 9, we also have a quantitative predictor variable. In order to deal with the binary response, we make some major adjustments in order to get reasonable predictions. A logit function is employed, and we swap to what are called generalized linear models or glms. Please see your text and class notes for more details. Here, we illustrate code related to the setting.

First, we consider an appropriate logistic setting for NHANES. Can we predict whether a person has smoked at least 100 cigarettes in their life (Smoke100) based on average Systolic blood pressure (BPSysAve)?

```
gf_boxplot(BPSysAve ~ Smoke100, data = NHANES)
```



```
tally(~ Smoke100, data = NHANES)
```

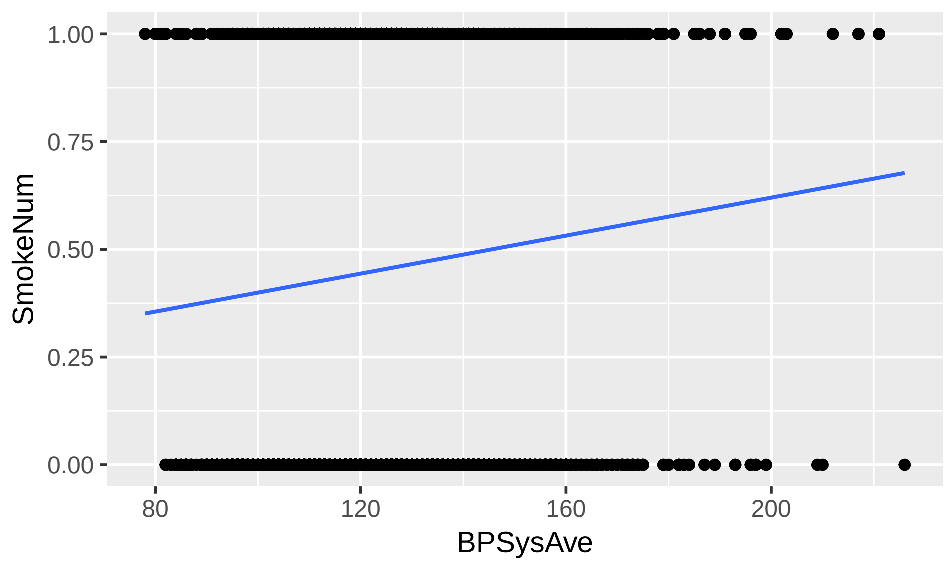
```
## Smoke100
##   No  Yes <NA>
## 4024 3211 2765
```

"Smokers" seem to have slightly higher BPSysAve values. We want to treat Smoke100 as the response, so we can filter out the NA observations, and make a numeric version of the response for plotting purposes. Note that 0 indicates a non-smoker and 1 indicates a smoker.

```
NHANES2 <- filter(NHANES, Smoke100 != "NA") %>% mutate(SmokeNum = as.numeric(Smoke100)-1)
```

Now we plot the relationship with SmokeNum (numeric version) as the response.

```
gf_point(SmokeNum ~ BPSysAve, data = NHANES2) %>%
  gf_lm()
```



Clearly, fitting a linear regression here makes no sense. The predictions are not ever 0 or 1, which are the only possible values of the response. This is why we need logistic regression.

Let's take a look at how to fit the model, then examine some more plots. The *glm* function works a lot like *lm*. It requires a numeric version of the response. The default family is binomial, so you don't actually need to specify that here, but this makes it obvious it is logistic regression.

```
logmod <- glm(SmokeNum ~ BPSysAve, data = NHANES2, family = binomial(logit))
msummary(logmod)
```

```
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.29992    0.17312   -7.51  6.0e-14 ***
## BPSysAve     0.00894    0.00141    6.32  2.6e-10 ***
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 9581.9  on 6970  degrees of freedom
## Residual deviance: 9541.6  on 6969  degrees of freedom
##   (264 observations deleted due to missingness)
## AIC: 9546
##
## Number of Fisher Scoring iterations: 4
```

You can get confidence intervals the usual way as well.

```
confint(logmod)
```

```
##                   2.5 %     97.5 %
## (Intercept) -1.64010605 -0.9613405
## BPSysAve     0.00617093  0.0117175
```

Note that for interpretation reasons, you might want:

```
exp(confint(logmod))
```

```
##                2.5 %  97.5 %
## (Intercept) 0.193959 0.38238
## BPSysAve    1.006190 1.01179
```

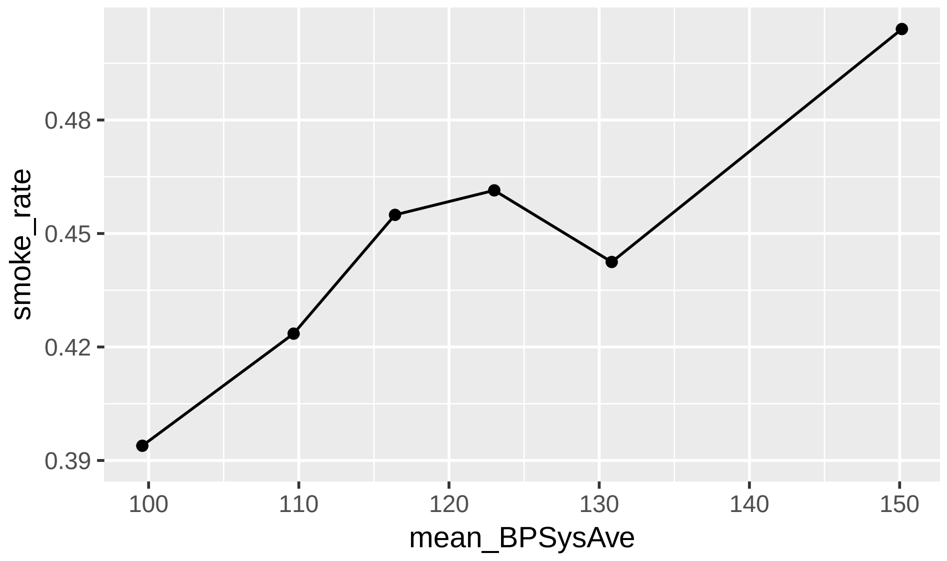
There are a number of plots for us now to explore.

First, to check conditions, we might want to make what is called an empirical logit plot. Before we introduce a function to make this plot, we explore some related options. The idea here is you want to look for a linear relationship between the predictor and logit of the probability that the response is a success.

The main way to investigate this with a continuous predictor is to "bin" the values so that you can evaluate the logit for the bin (because you may not have repeated predictor observations). We can create the bins, then plot the logit for the bin along with the original data points to examine the relationship.

We examine the proportion of smokers over the bins. The logit is a transformation of this that will give a similar shape, so you can just use that.

```
#create the bins
pred_bins <-  quantile(NHANES2$BPSysAve, probs = 0:6/6, na.rm = TRUE)
NHANES_binned <- NHANES2 %>%
  mutate(bin = cut(BPSysAve, breaks = pred_bins, include.lowest = TRUE)) %>%
  group_by(bin) %>%
  summarize(mean_BPSysAve = mean(BPSysAve),
            smoke_rate = mean(SmokeNum),
            logit_smoke_rate = logit(mean(SmokeNum)))

gf_point(smoke_rate ~ mean_BPSysAve, data = NHANES_binned) %>%
  gf_line()
```

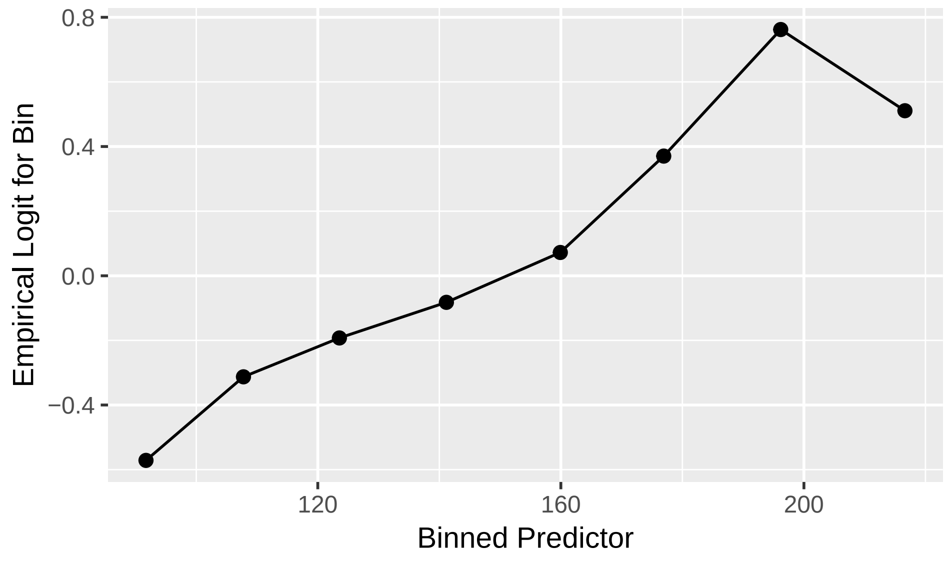


Rather than have to do this by hand though, you might want a function that will do it for you.

```
# run this entire chunk to get the function, then call as demonstrated below
emplogitplot <- function(resp, pred, numbreak = 10) {
  # assumes resp is dichotomous with values 0 and 1
  tmpGroup <- cut(pred, breaks = numbreak)
  binned.y <- mosaic::mean(~ resp | tmpGroup)
  binned.x <- mosaic::mean(~ pred | tmpGroup)
  logy <- mosaic::logit(binned.y)
  ds <- data.frame(logy, binned.x)
  gf_point(logy ~ binned.x, cex = 2, pch = 19, data=ds) %>%
    gf_line() %>%
    gf_labs(x = "Binned Predictor", y = "Empirical Logit for Bin")
}
```

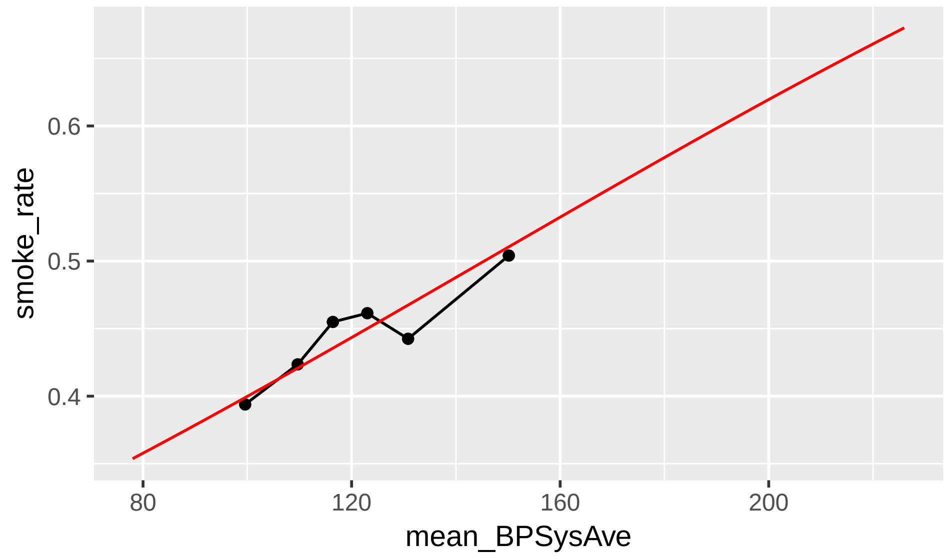
Then we can call the function, and you can change the number of breaks.

```
with(NHANES2, emplogitplot(SmokeNum, BPSysAve, 8))
```



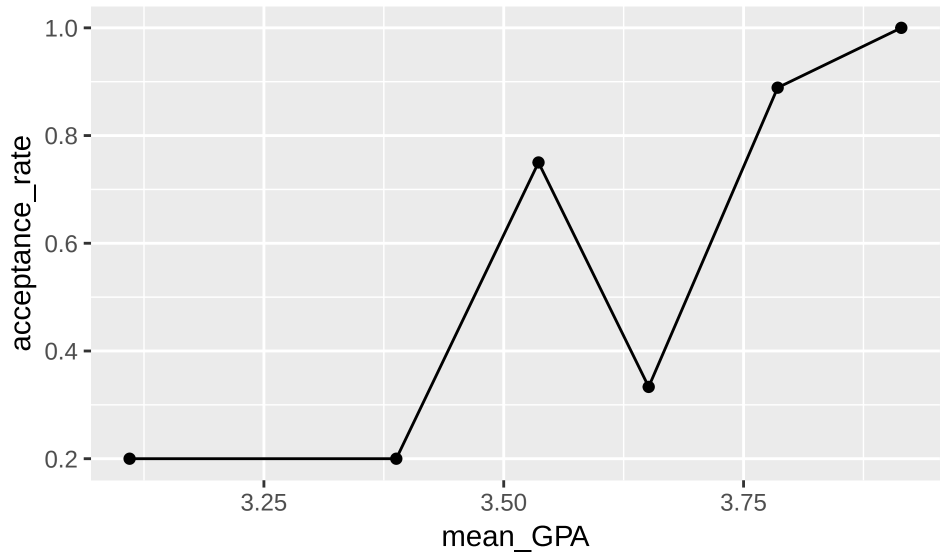
Now suppose you want to plot the fitted line on the probability scale (meaning on the binned data). You need the binned data as above, and then,

```
data_space <- gf_point(smoke_rate ~ mean_BPSysAve, data = NHANES_binned) %>%
  gf_line()
logmodaugment <- logmod %>% augment(type.predict = "response")

# logistic model on probability scale
data_space %>%
  gf_line(.fitted ~ BPSysAve, data = logmodaugment, color = "red")
```

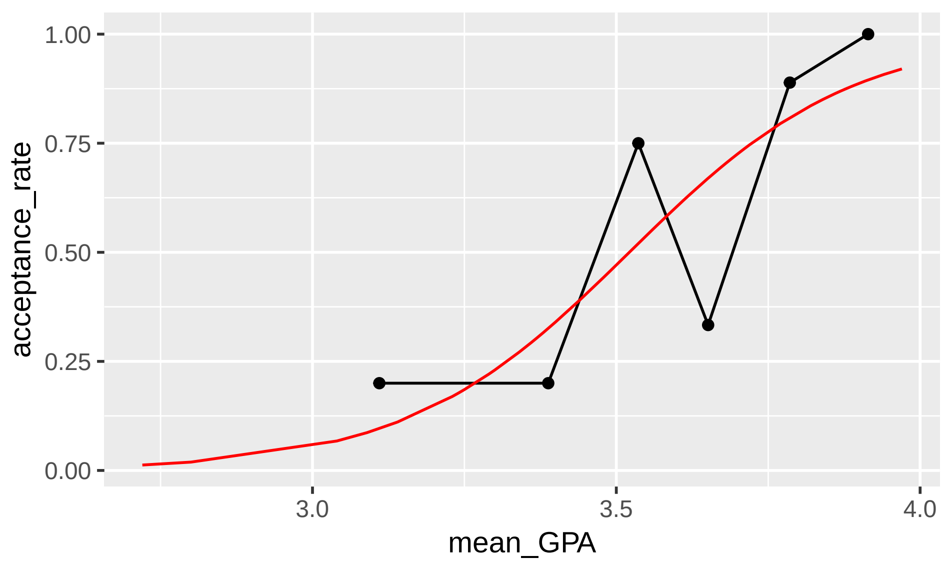


This example may not be the best, so here is another example looking at medical school applicant GPAs and probability of acceptance to medical school. Note that this code uses ggplot2 syntax, not ggformula as above.

```r
#Code from Ben Baumer
data(MedGPA)
mod <- glm(Acceptance ~ GPA, data = MedGPA, family = binomial)

gpa_bins <- quantile(MedGPA$GPA, probs = 0:6/6)
MedGPA_binned <- MedGPA %>%
  mutate(bin = cut(GPA, breaks = gpa_bins, include.lowest = TRUE)) %>%
  group_by(bin) %>%
  summarize(mean_GPA = mean(GPA),
            acceptance_rate = mean(Acceptance))

# binned points and line
data_space <- ggplot(data = MedGPA_binned, aes(x = mean_GPA, y = acceptance_rate)) +
  geom_point() + geom_line()
data_space
```



```r
# augmented model
MedGPA_plus <- mod %>% augment(type.predict = "response")

# logistic model on probability scale
data_space +
  geom_line(data = MedGPA_plus, aes(x = GPA, y = .fitted), color = "red")
```



The curve here shows the more characteristic S (or reverse S) shape.

We briefly look at code to obtain p-values and to perform the likelihood ratio test.

```
msummary(logmod)
```

```
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.29992    0.17312   -7.51  6.0e-14 ***
## BPSysAve     0.00894    0.00141    6.32  2.6e-10 ***
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 9581.9  on 6970  degrees of freedom
## Residual deviance: 9541.6  on 6969  degrees of freedom
##   (264 observations deleted due to missingness)
## AIC: 9546
##
## Number of Fisher Scoring iterations: 4
```

The summary has the null and residual deviances. The difference between them is the G statistic from your text. The difference in df is the other part needed to obtain these results.

```
G <- 9581.9 - 9541.6; G
```

```
## [1] 40.3
```

```
lrtdf <- 6970 - 6969; lrtdf
```

```
## [1] 1
```

```
pchisq(G, df = lrtdf, lower.tail = FALSE)
```

```
## [1] 2.17809e-10
```

Equivalently, the computer can do the computations with:

```
lrtest(logmod)
```

```
## Likelihood ratio test
##
```

```
## Model 1: SmokeNum ~ BPSysAve
## Model 2: SmokeNum ~ 1
##   #Df LogLik Df Chisq Pr(>Chisq)
## 1   2  -4771
## 2   1  -4791 -1 40.32   2.16e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

To wrap up this section, we examine the predictions a bit more closely.

```
logmodaugment <- logmod %>% augment(type.predict = "response")
names(logmodaugment)
```

```
##  [1] ".rownames"  "SmokeNum"   "BPSysAve"   ".fitted"    ".se.fit"
##  [6] ".resid"     ".hat"       ".sigma"     ".cooksd"    ".std.resid"
```

```
head(logmodaugment)
```

```
## # A tibble: 6 x 10
##   .rownames SmokeNum BPSysAve .fitted .se.fit .resid    .hat .sigma .cooksd
##   <chr>        <dbl>    <int>   <dbl>   <dbl>  <dbl>   <dbl>  <dbl>   <dbl>
## 1 1                1      113   0.428 0.00656   1.30 1.76e-4   1.17 1.17e-4
## 2 2                1      113   0.428 0.00656   1.30 1.76e-4   1.17 1.17e-4
## 3 3                1      113   0.428 0.00656   1.30 1.76e-4   1.17 1.17e-4
## 4 4                1      112   0.426 0.00671   1.31 1.84e-4   1.17 1.24e-4
## 5 5                0      118   0.439 0.00605  -1.08 1.49e-4   1.17 5.82e-5
## 6 6                0      118   0.439 0.00605  -1.08 1.49e-4   1.17 5.82e-5
## # ... with 1 more variable: .std.resid <dbl>
```

If you examine the augmented data set, since we asked for the correct type of prediction, our .fitted values are the probability estimated for each individual to be a smoker.

```
favstats(~ .fitted, data = logmodaugment)
```

```
##       min       Q1   median       Q3      max     mean        sd    n
##  0.353711 0.419288 0.441194 0.465554 0.672613 0.445847 0.0378163 6971
##  missing
##        0
```

If we wanted to make binary predictions for each individual, we could round to the nearest integer (i.e., anyone over 50% chance of being a smoker would be labelled a smoker).

```
logmodaugment <- mutate(logmodaugment, binprediction = round(.fitted, 0))
tally(~ binprediction, data = logmodaugment)
```

```
## binprediction
##    0    1
## 6412  559
```

```
559/6971
```

```
## [1] 0.0801894
```

We can see that using this strategy, only 559 individuals (8 percent) are predicted to be smokers. We can examine how well we did with the predictions by making a confusion matrix - looking at observed values versus predicted values.

```
with(logmodaugment, table(SmokeNum, binprediction))
```

```
##         binprediction
## SmokeNum    0    1
##        0 3602  261
##        1 2810  298
```

```
correct <- (3602 + 298)/6971; correct
```

```
## [1] 0.559461
```

We are only 55.9 percent correct on our predictions. Many folks are being labelled as non-smokers who are really smokers.

How can we adjust this? Well, we know the fraction of smokers in the original sample was closer to 45 percent than 8 percent. So, we can adjust for that some.

```
tally(~ SmokeNum, data = logmodaugment)
```

```
## SmokeNum
##    0    1
## 3863 3108
```

```
3108/6971
```

```
## [1] 0.445847
```

```
with(logmodaugment, quantile(.fitted, 1-0.45)) #need the upper cutoff because 55% are non-smokers
```

```
##      55%
## 0.445605
```

```
logmodaugment <- mutate(logmodaugment, binprediction2 = as.numeric(.fitted > 0.4456054))
tally(~ binprediction2, data = logmodaugment)
```

```
## binprediction2
##    0    1
## 3813 3158
```

(There are some other ways to work at adjusting this, but the idea was we knew the number of smokers was more than 8 percent).

Now the confusion matrix looks like:

```
with(logmodaugment, table(SmokeNum, binprediction2))
```

```
##         binprediction2
## SmokeNum    0    1
##        0 2198 1665
##        1 1615 1493
```

```
correct2 <- (2198 + 1493)/6971; correct2
```

```
## [1] 0.529479
```

Now we are 52.95 percent accurate, but have better captured the fraction of smokers in the data. What does this mean? It means our model doesn't fit very well. There doesn't appear to be much of a relationship here to capitalize on in terms of making predictions. Maybe we should include a few other predictors.

Multiple Logistic Regression (Chapters 10-11)

In our last example, we saw that there wasn't really much signal trying to predict smoking status based on BPSysAve. Now, with multiple logistic regression, we can add more predictors. We continue examining the same basic relationship, so some commands from above are still relevant.

```
NHANES2 <- filter(NHANES, Smoke100 != "NA") %>% mutate(SmokeNum = as.numeric(Smoke100)-1)
```

With our numeric version of the response, we can fit our model, now with additional predictors. You can again make empirical logit plots using the function provided above.

```
logm <- glm(SmokeNum ~ BPSysAve + Age + Gender + BMI + Pulse, data = NHANES2,
        family = binomial(logit))
msummary(logm)
```

```
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.93150    0.24765   -7.80  6.2e-15 ***
## BPSysAve     0.00344    0.00161    2.14    0.032 *
## Age          0.01264    0.00164    7.71  1.3e-14 ***
## Gendermale   0.57880    0.05065   11.43  < 2e-16 ***
## BMI         -0.02068    0.00384   -5.38  7.4e-08 ***
## Pulse        0.01386    0.00215    6.45  1.1e-10 ***
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 9511.4  on 6918  degrees of freedom
## Residual deviance: 9262.7  on 6913  degrees of freedom
##   (316 observations deleted due to missingness)
## AIC: 9275
##
## Number of Fisher Scoring iterations: 4
```

```
lrtest(logm)
```

```
## Likelihood ratio test
##
## Model 1: SmokeNum ~ BPSysAve + Age + Gender + BMI + Pulse
## Model 2: SmokeNum ~ 1
##   #Df LogLik Df Chisq Pr(>Chisq)
## 1   6  -4631
## 2   1  -4756 -5 248.7     <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
exp(confint(logm))
```

```
## Waiting for profiling to be done...
```

```
##                 2.5 %   97.5 %
## (Intercept) 0.0891113 0.235284
## BPSysAve    1.0002918 1.006617
## Age         1.0094735 1.015981
## Gendermale  1.6155376 1.970407
## BMI         0.9721492 0.986909
## Pulse       1.0097020 1.018244
```

```
logmaugment <- augment(logm, type.predict = "response")
```

For a nested drop in deviance procedure, we need a second reduced model. The challenge here is that if you fit the model on NHANES2, you will end up with more observations because the predictors dropped had some missing values. We get around this by fitting the reduced model on the augmented data set from the full model.

```r
logm2 <- glm(SmokeNum ~ Age + Gender + BMI, data = logmaugment, family = binomial(logit))
msummary(logm2)
```

```
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.59750    0.13037   -4.58  4.6e-06 ***
## Age          0.01239    0.00146    8.48  < 2e-16 ***
## Gendermale   0.54119    0.04923   10.99  < 2e-16 ***
## BMI         -0.01658    0.00377   -4.39  1.1e-05 ***
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 9511.4  on 6918  degrees of freedom
## Residual deviance: 9311.0  on 6915  degrees of freedom
## AIC: 9319
##
## Number of Fisher Scoring iterations: 4
```

```r
G <- 9311.0 - 9262.7; G
```

```
## [1] 48.3
```

```r
lrtdf <- 6915 - 6913; lrtdf
```

```
## [1] 2
```

```r
pchisq(G, df = lrtdf, lower.tail = FALSE)
```

```
## [1] 3.24929e-11
```

You can also have the computer do these computations with:

```r
anova(logm2, logm, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: SmokeNum ~ Age + Gender + BMI
## Model 2: SmokeNum ~ BPSysAve + Age + Gender + BMI + Pulse
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      6915       9311
## 2      6913       9263  2    48.31 3.24e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This suggests we should keep the larger "full" model. How well is that model doing? Well, we can look at a confusion matrix again:

```r
logmaugment <- mutate(logmaugment, binprediction = round(.fitted, 0))
tally(~ binprediction, data = logmaugment)
```

```
## binprediction
##    0    1
## 4915 2004
```

```r
2004/6971
```

```
## [1] 0.287477
```

We see that 28.7 percent of individuals are predicted to be smokers, so we may still need to adjust this.

```
with(logmaugment, table(SmokeNum, binprediction))
```

```
##          binprediction
## SmokeNum    0    1
##        0 2972  860
##        1 1943 1144
```

```
correct <- (2972 + 1144)/6971; correct
```

```
## [1] 0.590446
```

We are 59% correct. Does adjusting the number of predicted smokers up help?

```
with(logmaugment, quantile(.fitted, 1-0.45))
```

```
##      55%
## 0.458305
```

```
logmaugment <- mutate(logmaugment, binprediction2 = as.numeric(.fitted > 0.4583052))
tally(~ binprediction2, data = logmaugment)
```

```
## binprediction2
##    0    1
## 3805 3114
```

Now the confusion matrix looks like:

```
with(logmaugment, table(SmokeNum, binprediction2))
```

```
##          binprediction2
## SmokeNum    0    1
##        0 2348 1484
##        1 1457 1630
```

```
correct2 <- (2348 + 1630)/6971; correct2
```

```
## [1] 0.57065
```

Alas, this correction does not seem to indicate that our model does a good job.

How else can we check how the model is doing? We can check concordance.

```
#***FUNCTION TO CALCULATE CONCORDANCE AND DISCORDANCE***#
Association <- function(model)
{
  Con_Dis_Data <- cbind(model$y, model$fitted.values)
  ones <- Con_Dis_Data[Con_Dis_Data[, 1] == 1, ]
  zeros <- Con_Dis_Data[Con_Dis_Data[, 1] == 0, ]
  conc <- matrix(0, dim(zeros)[1], dim(ones)[1])
  disc <- matrix(0, dim(zeros)[1], dim(ones)[1])
  ties <- matrix(0, dim(zeros)[1], dim(ones)[1])
    for (j in 1:dim(zeros)[1])
    {
      for (i in 1:dim(ones)[1])
      {
        if (ones[i, 2] > zeros[j, 2])
        {conc[j, i] = 1}
        else if (ones[i, 2] < zeros[j, 2])
        {disc[j, i] = 1}
        else if (ones[i, 2] == zeros[j, 2])
        {ties[j, i] = 1}
      }
    }
  Pairs <- dim(zeros)[1]*dim(ones)[1]
  PercentConcordance <- (sum(conc)/Pairs)*100
  PercentDiscordance <- (sum(disc)/Pairs)*100
  PercentTied <- (sum(ties)/Pairs)*100
  return(list("Percent Concordance" = PercentConcordance,
              "Percent Discordance" = PercentDiscordance,
              "Percent Tied" = PercentTied, "Pairs" = Pairs))
}
#***FUNCTION TO CALCULATE CONCORDANCE AND DISCORDANCE ENDS***#
```

This function will calculate concordance and discordance values. Depending on the number of observations, this computation can take a while.

```
Association(logm)
```

```
## $`Percent Concordance`
## [1] 60.6105
##
## $`Percent Discordance`
## [1] 39.3895
##
## $`Percent Tied`
## [1] 0
##
## $Pairs
## [1] 11829384
```

The model concordance is only 60%. We'd get 50% with a coin flip. This indicates the model does not fit very well.

There are other pseudo-Rsquared type statistics out there:

```
round(PseudoR2(logm, "all"), 2)
```

```
##        McFadden     McFaddenAdj        CoxSnell      Nagelkerke
##            0.07            0.07            0.09            0.12
##   AldrichNelson VeallZimmermann          Effron McKelveyZavoina
##            0.03            0.09            0.04            0.04
##            Tjur             AIC             BIC          logLik
##            0.04         9274.71         9315.77        -4631.36
##         logLik0              G2
##        -4969.14          675.58
```

The gist here is that the model does not fit well.

Finally, a last topic from chapter 11, section 2, has to do with overdispersion. Here, I show the adjustment that should be made to code when issues with overdispersion occur.

```
logm3 <- glm(SmokeNum ~ BPSysAve + Age + Gender + BMI + Pulse, data = NHANES2,
    family = quasibinomial(logit))
msummary(logm3)
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.93150    0.24783   -7.79  7.5e-15 ***
## BPSysAve     0.00344    0.00161    2.14    0.032 *
## Age          0.01264    0.00164    7.70  1.5e-14 ***
## Gendermale   0.57880    0.05069   11.42  < 2e-16 ***
## BMI         -0.02068    0.00385   -5.38  7.8e-08 ***
## Pulse        0.01386    0.00215    6.45  1.2e-10 ***
##
## (Dispersion parameter for quasibinomial family taken to be 1.00147)
##
##     Null deviance: 9511.4  on 6918  degrees of freedom
## Residual deviance: 9262.7  on 6913  degrees of freedom
##   (316 observations deleted due to missingness)
## AIC: NA
##
## Number of Fisher Scoring iterations: 4
```