Module 3: bivariate and multivariate analysis

BIOSTAT 213: Introduction to Computing in the R Software Environment

Yea-Hung Chen, PhD, MS

Wednesday, January 30, 2019

Continuous vs continuous

Options for continuing work with the NHANES data

- one option is to simply rename your folder for Module 2 (eg, `module 2`) to something more generic (eg, `nhanes`), use that folder as the working directory, and import the data using `read.csv('nhanes.csv')`
- another option is to create a folder for Module 3 (eg, `module 3`), use that folder as the working directory, and import the data by specifying the path to the data file in `read.csv()` (eg, `read.csv('.../module 2/nhanes.csv')`)
- yet another option is to create a folder for Module 3 (eg, `module 3`), use that folder as the working directory, and (if your operating system supports it) create a symbolic link to the data file in your Module 3 folder so that you can continue to use `read.csv('nhanes.csv')`

Scatter plots

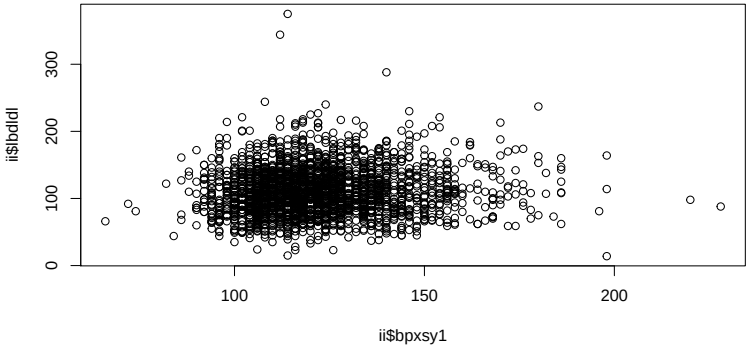
- to create a scatter plot of LDL (low-density lipoprotein) cholesterol and systolic blood pressure, with LDL cholesterol on the y axis:

```
> plot(x=ii$bpxsy1,y=ii$lbdldl)
```

- or, without argument names:

```
> plot(ii$bpxsy1,ii$lbdldl)
```

Scatter plots



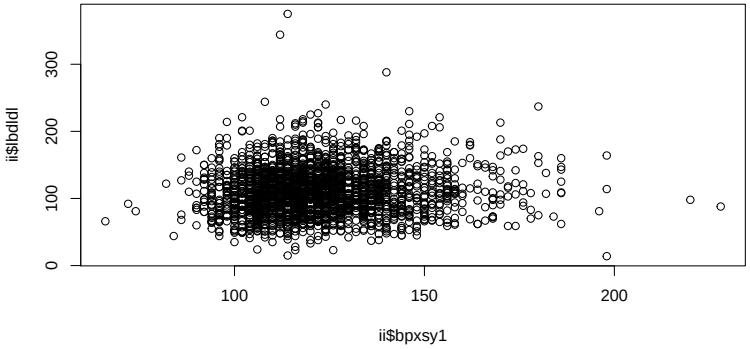
Scatter plots

- equivalently, we can use formula notation:

```
> plot(ii$lbdldl~ii$bpxsy1)
```

- the tilde (~) means versus
- formula notation is used throughout R
- we'll see it throughout the day as we discuss plotting functions, hypothesis tests, and linear regression
- formula notation, in general: `y~x`

Scatter plots



Scatter plots

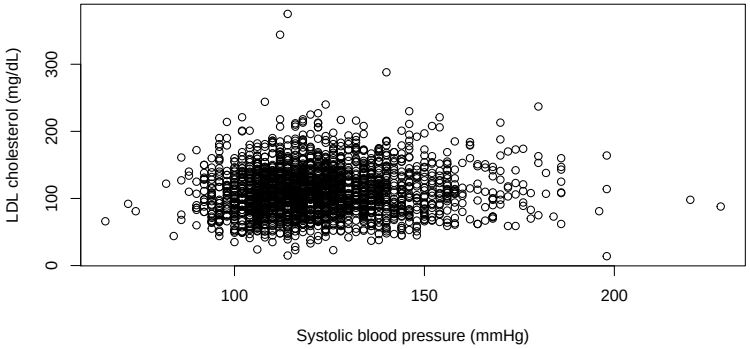
- equivalently, using the `data` argument:

```
> plot(lbdldl~bpxsy1,data=ii)
```

Scatter plots

```
> plot(ii$lbdldl~ii$bpxsy1,xlab='Systolic blood pressure (mmHg)',
+      ylab='LDL cholesterol (mg/dL)')
```

Scatter plots



LOESS

- we'll discuss LOESS (locally estimated scatterplot smoothing) when we discuss ggplot2 (I find it easier to add LOESS lines via ggplot2 than via "base" R)

Correlation coefficients

```
> cor(x=ii$bpxsy1,y=ii$lbdldl,use='complete.obs')
[1] 0.06946024
> cor(x=ii$bpxsy1,y=ii$lbdldl,use='complete.obs',
+     method='pearson')
[1] 0.06946024
> cor(x=ii$bpxsy1,y=ii$lbdldl,use='complete.obs',
+     method='spearman')
[1] 0.07451623
```

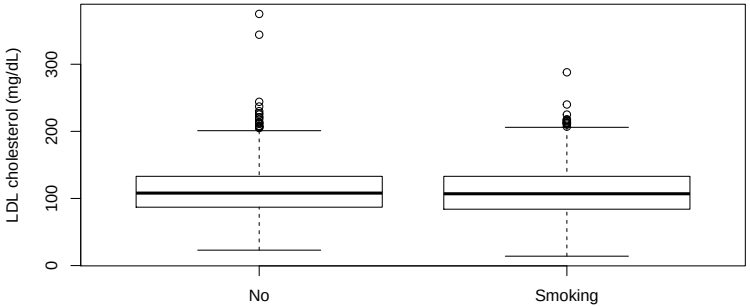
Continuous vs categorical

Box plots

- to create a box plot of LDL (low-density lipoprotein) cholesterol and history of smoking, with LDL cholesterol on the *y* axis:

```
> plot(ii$lbdldl~ii$hs,xlab='',ylab='LDL cholesterol (mg/dL)')
```

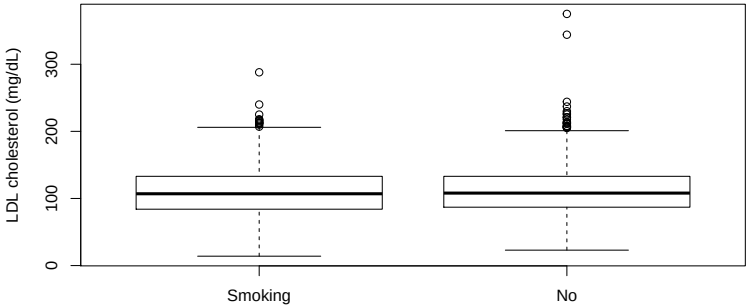
Box plots



Box plots

```
> ii$hs<-relevel(ii$hs,'Smoking')
> plot(ii$lbdldl~ii$hs,xlab='',ylab='LDL cholesterol (mg/dL)')
```

Box plots



t-tests

```
> t.test(ii$lbdldl~ii$hs)

    Welch Two Sample t-test

data:  ii$lbdldl by ii$hs
t = -0.63877, df = 2329.3, p-value = 0.523
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.692851  1.878154
sample estimates:
mean in group Smoking      mean in group No
             110.4476              111.3549
```

t-tests

```
> yetanothertest<-t.test(ii$lbdldl~ii$hs)
> names(yetanothertest)
[1] "statistic"   "parameter"   "p.value"     "conf.int"
[5] "estimate"    "null.value"  "alternative" "method"
[9] "data.name"
> yetanothertest$statistic
       t
-0.63877
> yetanothertest$p.value
[1] 0.5230354
```

t-tests

```
> t.test(ii$lbdldl~ii$hs,var.equal=TRUE)

    Two Sample t-test

data:  ii$lbdldl by ii$hs
t = -0.6417, df = 2510, p-value = 0.5211
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.680042  1.865344
sample estimates:
mean in group Smoking      mean in group No
             110.4476              111.3549
```

Wilcoxon rank-sum tests

```
> wilcox.test(ii$lbdldl~ii$hs)

    Wilcoxon rank sum test with continuity correction

data:  ii$lbdldl by ii$hs
W = 762270, p-value = 0.3983
alternative hypothesis: true location shift is not equal to 0
```

Wilcoxon rank-sum tests

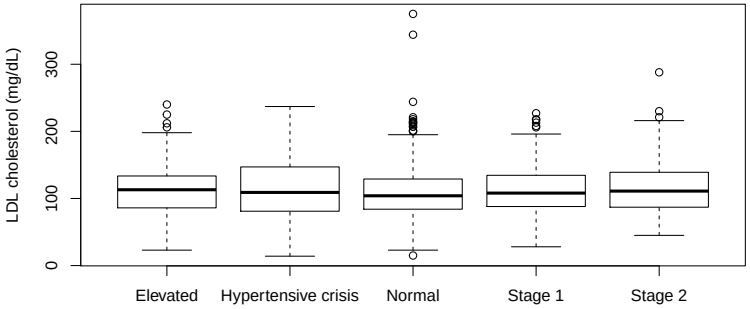
```
> tt<-wilcox.test(ii$lbdldl~ii$hs)
> names(tt)
[1] "statistic"   "parameter"   "p.value"     "null.value"
[5] "alternative" "method"      "data.name"
> tt$statistic
       W
762274.5
> tt$p.value
[1] 0.3982661
```

Box plots

- to create a box plot of LDL (low-density lipoprotein) cholesterol and categorized blood pressure, with LDL cholesterol on the *y* axis:

```
> plot(ii$lbdldl~ii$bpcat,xlab='',ylab='LDL cholesterol (mg/dL)')
```

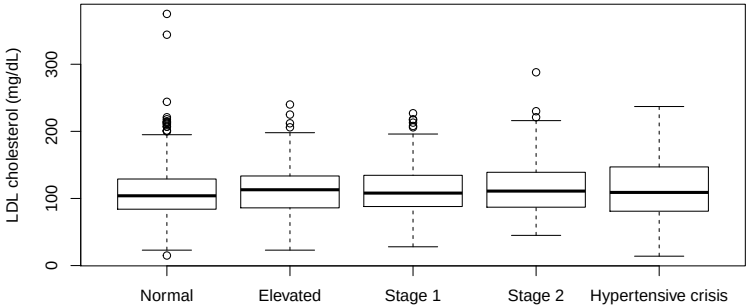
Box plots



Box plots

```
> ii$bpcat<-factor(ii$bpcat,c('Normal','Elevated','Stage 1',
+                             'Stage 2','Hypertensive crisis'))
> plot(ii$lbdldl~ii$bpcat,xlab='',ylab='LDL cholesterol (mg/dL)')
```

Box plots


LDL cholesterol (mg/dL)
0
100
200
300
Normal
Elevated
Stage 1
Stage 2
Hypertensive crisis

Analysis of variance (ANOVA)

```
> aov(ii$lbdldl~ii$bpcat)
Call:
   aov(formula = ii$lbdldl ~ ii$bpcat)

Terms:
                  ii$bpcat Residuals
Sum of Squares     17082.2 2870034.4
Deg. of Freedom          4      2310

Residual standard error: 35.24825
Estimated effects may be unbalanced
3454 observations deleted due to missingness
```

Analysis of variance (ANOVA)

```
> summary(aov(ii$lbdldl~ii$bpcat))
              Df  Sum Sq Mean Sq F value  Pr(>F)
ii$bpcat       4   17082    4271   3.437 0.00827 **
Residuals   2310 2870034    1242
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
3454 observations deleted due to missingness
```

Kruskal-Wallis tests

```
> kruskal.test(ii$lbdldl~ii$bpcat)

    Kruskal-Wallis rank sum test

data:  ii$lbdldl by ii$bpcat
Kruskal-Wallis chi-squared = 13.793, df = 4, p-value
= 0.007986
```

Kruskal-Wallis tests

```
> tt<-kruskal.test(ii$lbdldl~ii$bpcat)
> names(tt)
[1] "statistic" "parameter" "p.value"   "method"
[5] "data.name"
> tt$statistic
Kruskal-Wallis chi-squared
                  13.79292
> tt$p.value
[1] 0.00798617
```

Categorical vs categorical

Tables

- to create a table of smoking and asthma:

```
> table(ii$hs,ii$asthma)

          Asthma   No
  Smoking    430 2098
  No         448 2786
```

Row-wise percents

- for row-wise percents, use `prop.table()` with the `margin=1` argument

```
> prop.table(table(ii$hs,ii$asthma),margin=1)

            Asthma        No
Smoking 0.1700949 0.8299051
No      0.1385281 0.8614719
```

Row-wise percents

- multiplying by 100 and omitting the argument name (`margin`):

```
> prop.table(table(ii$hs,ii$asthma),1)*100

           Asthma       No
Smoking 17.00949 82.99051
No      13.85281 86.14719
```

Column-wise percents

- for column-wise percents, use `prop.table()` with the `margin=2` argument

```
> prop.table(table(ii$hs,ii$asthma),margin=2)

            Asthma        No
Smoking 0.4897494 0.4295659
No      0.5102506 0.5704341
```

Column-wise percents

- multiplying by 100 and omitting the argument name (`margin`):

```
> prop.table(table(ii$hs,ii$asthma),2)*100

           Asthma       No
Smoking 48.97494 42.95659
No      51.02506 57.04341
```

χ^2 tests

```
> chisq.test(table(ii$hs,ii$asthma))

    Pearson's Chi-squared test with Yates' continuity
    correction

data:  table(ii$hs, ii$asthma)
X-squared = 10.704, df = 1, p-value = 0.001069
```

χ^2 tests

```
> tt<-chisq.test(table(ii$hs,ii$asthma))
> names(tt)
[1] "statistic" "parameter" "p.value"   "method"
[5] "data.name" "observed"  "expected"  "residuals"
[9] "stdres"
> tt$statistic
X-squared
 10.70361
> tt$p.value
[1] 0.001069266
```

Fisher's exact test

```
> fisher.test(table(ii$hs,ii$asthma))

    Fisher's Exact Test for Count Data

data:  table(ii$hs, ii$asthma)
p-value = 0.001007
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 1.100669 1.475740
sample estimates:
odds ratio
  1.274511
```

Fisher's exact test

```
> tt<-fisher.test(table(ii$hs,ii$asthma))
> names(tt)
[1] "p.value"     "conf.int"    "estimate"    "null.value"
[5] "alternative" "method"      "data.name"
> tt$p.value
[1] 0.00100706
```

Installing and using packages

Installing packages

- to install the ggplot2 package:

```
> install.packages('ggplot2')
```

- you should only need to do this once ever

Installing packages

- to specify the repository, use the `repos` argument:

```
> install.packages('ggplot2',repos='https://cran.rstudio.com')
```

- other possible repositories include:
  - `https://cran.cnr.berkeley.edu`
  - `http://cran.stat.ucla.edu`
- a list of mirrors is available here

Using packages

- to use the ggplot2 package:

```
> library(ggplot2)
```

- you only need to do this once per session

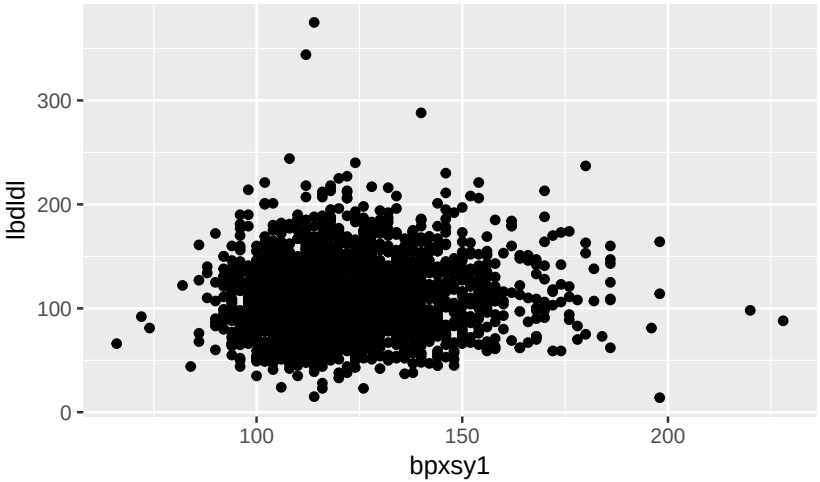
ggplot2

Scatterplots

```
> ggplot(aes(x=bpxsy1,y=lbdldl),data=ii)+
+   geom_point()
```

- the `ggplot()` function initiates the plot, but by itself it doesn't actually produce a plot
- the `aes()` function, short for aesthetics, specifies the x and y variables as well as other dimensions (as we'll see later)
- the addition of `geom_point()` indicates that we want a scatter plot

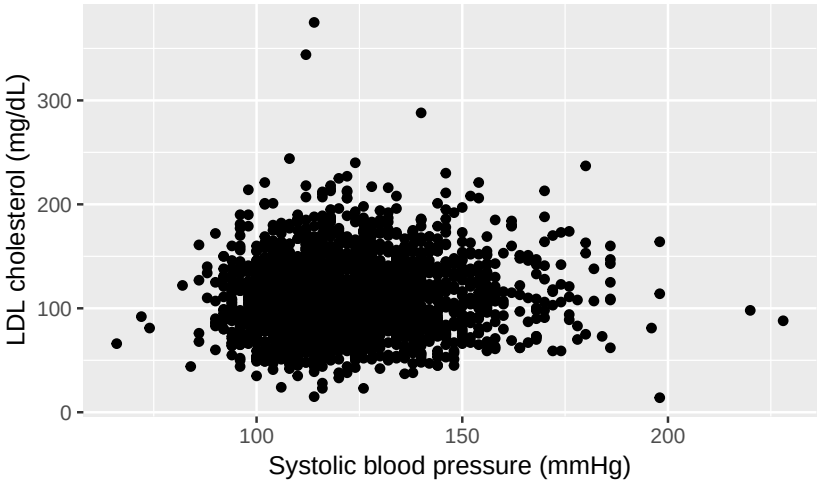
Scatterplots



Scatterplots

```
> ggplot(aes(x=bpxsy1,y=lbdldl),data=ii)+
+   geom_point()+
+   xlab('Systolic blood pressure (mmHg)')+
+   ylab('LDL cholesterol (mg/dL)')
```

Scatterplots

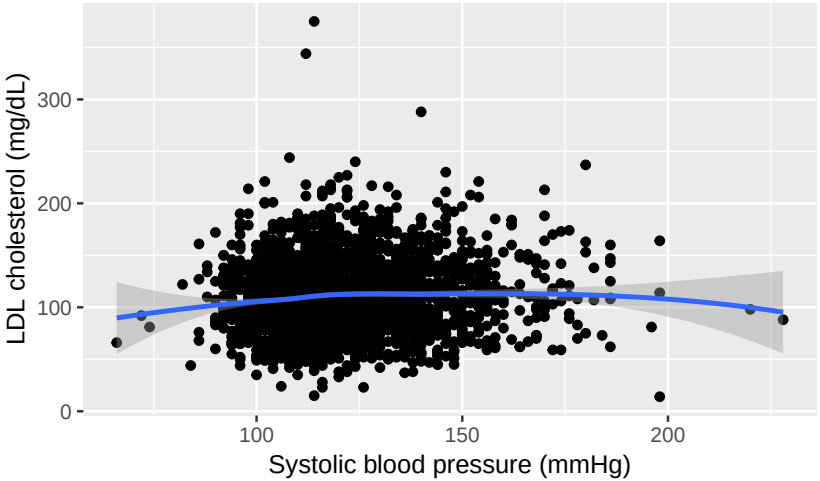


LOESS lines

- to add a LOESS line, use `geom_smooth()` with the `method='loess'` argument:

```
> ggplot(aes(x=bpxsy1,y=lbdldl),data=ii)+
+   geom_point()+
+   xlab('Systolic blood pressure (mmHg)')+
+   ylab('LDL cholesterol (mg/dL)')+
+   geom_smooth(method='loess')
```

LOESS lines

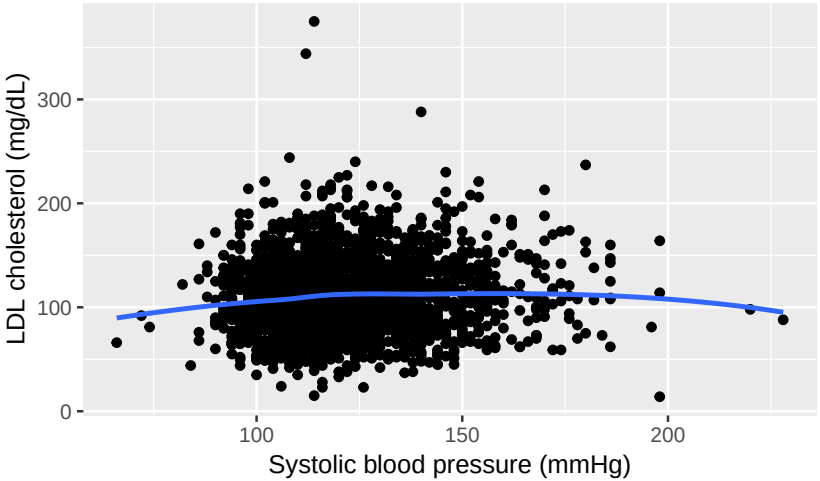


LOESS lines

- to remove the confidence interval, add the `se=FALSE` argument to `geom_smooth()`:

```
> ggplot(aes(x=bpxsy1,y=lbdldl),data=ii)+
+   geom_point()+
+   xlab('Systolic blood pressure (mmHg)')+
+   ylab('LDL cholesterol (mg/dL)')+
+   geom_smooth(method='loess',se=FALSE)
```

LOESS lines

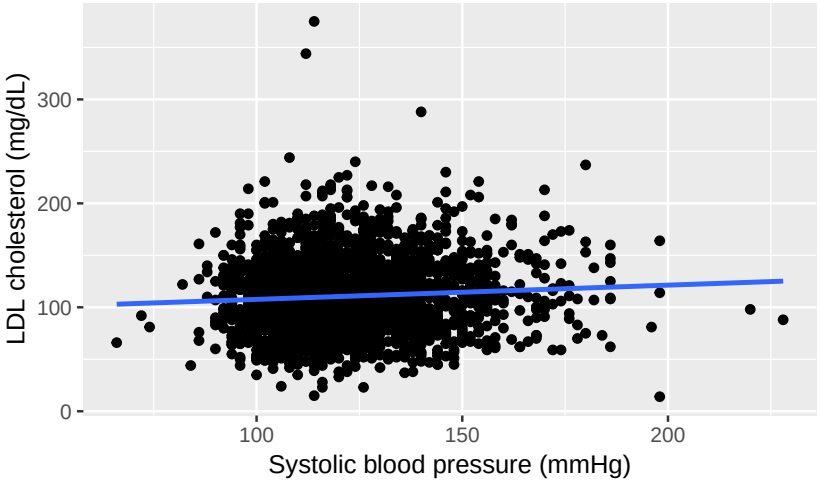


Regression lines

- to add a regression line, use `geom_smooth()` with the `method='lm'` argument and the `formula` arguments:

```
> ggplot(aes(x=bpxsy1,y=lbdldl),data=ii)+
+   geom_point()+
+   xlab('Systolic blood pressure (mmHg)')+
+   ylab('LDL cholesterol (mg/dL)')+
+   geom_smooth(method='lm',formula=y~x,se=FALSE)
```

Regression lines



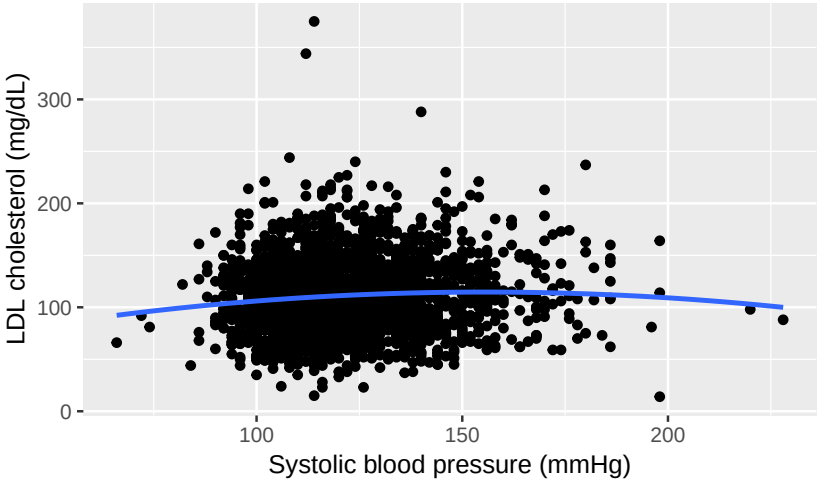
Regression lines

- for a quadratic regression line, use the `formula` argument with the `poly()` function:

```
> ggplot(aes(x=bpxsy1,y=lbdldl),data=ii)+
+   geom_point()+
+   xlab('Systolic blood pressure (mmHg)')+
+   ylab('LDL cholesterol (mg/dL)')+
+   geom_smooth(method='lm',formula=y~poly(x,2),se=FALSE)
```

- the 2 argument to `poly()` indicates that we want a quadratic formula

Regression lines

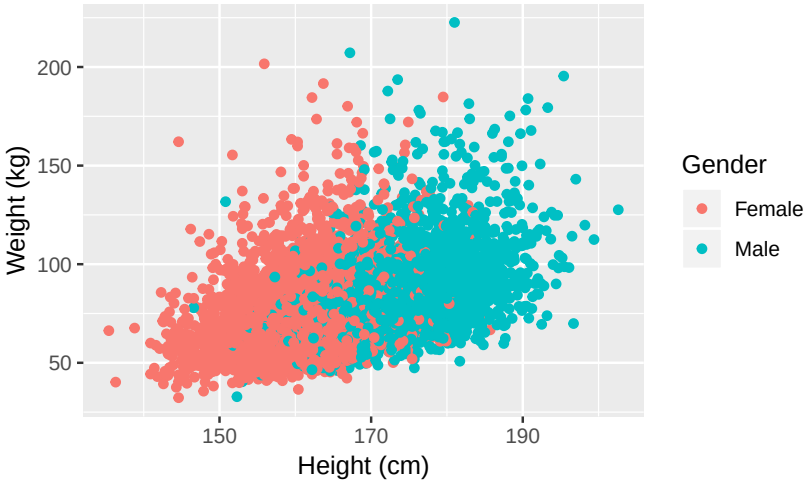


Adding color as a 3rd dimension

- to add color as a 3rd dimension, use the `color` argument:

```
> ggplot(aes(x=bmxht,y=bmxwt,color=gender),data=ii)+
+   geom_point()+
+   xlab('Height (cm)')+ylab('Weight (kg)')+
+   labs(color='Gender')
```

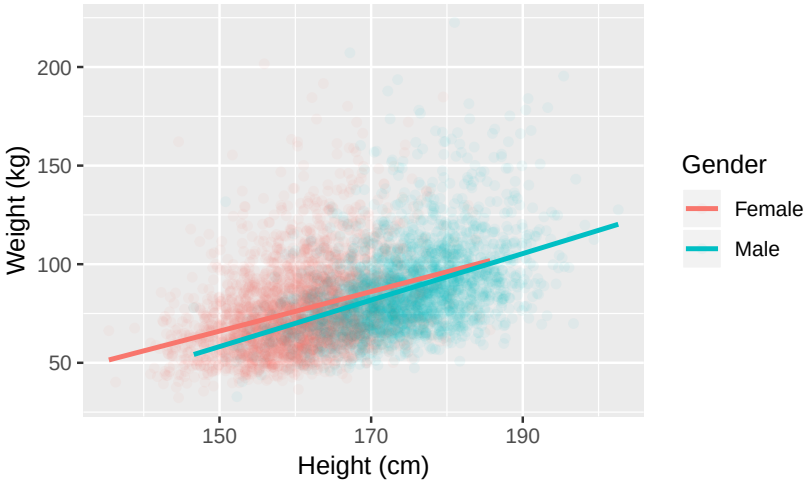
Adding color as a 3rd dimension



Adding color as a 3rd dimension

```
> ggplot(aes(x=bmxht,y=bmxwt,color=gender),data=ii)+
+   geom_point(alpha=0.05)+
+   geom_smooth(method='lm',formula=y~x,se=FALSE)+
+   xlab('Height (cm)')+ylab('Weight (kg)')+
+   labs(color='Gender')
```

Adding color as a 3rd dimension

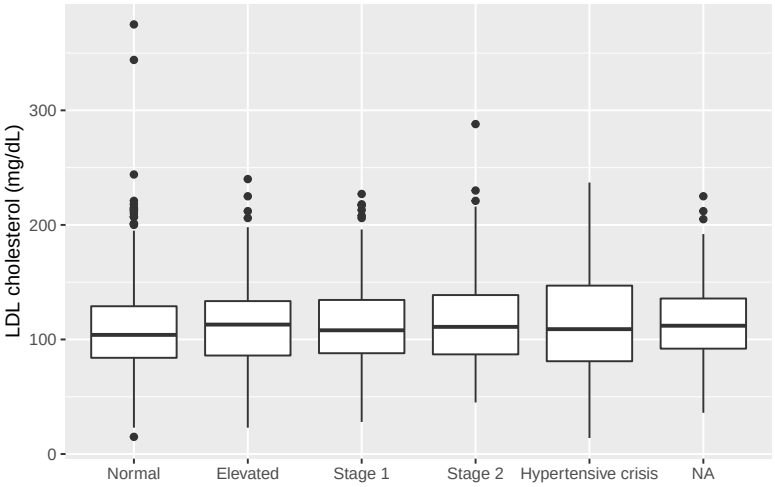


Box plots

- for box plots, use `geom_boxplot()`:

```
> ggplot(aes(x=bpcat,y=lbdldl),data=ii)+
+   geom_boxplot()+
+   ylab('LDL cholesterol (mg/dL)')+xlab('')
```

Box plots

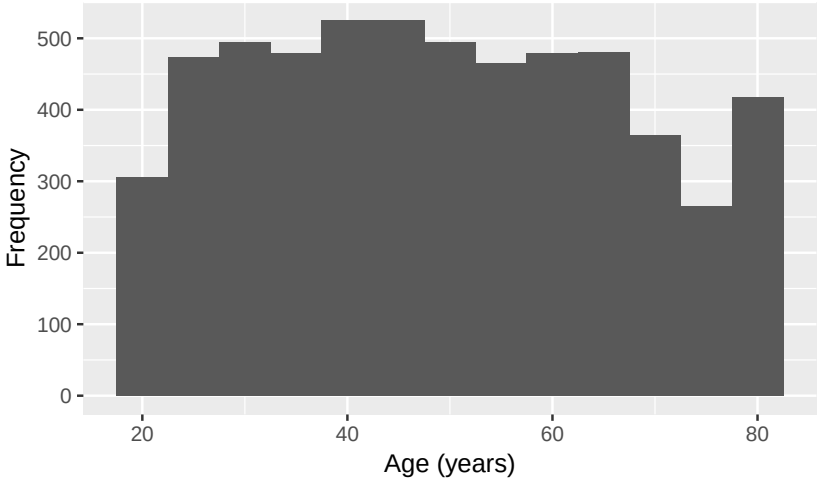

LDL cholesterol (mg/dL)
300
200
100
0
Normal
Elevated
Stage 1
Stage 2
Hypertensive crisis
NA

Histograms

- for box plots, use `geom_histogram()`:

```
> ggplot(aes(x=ridageyr),data=ii)+
+   geom_histogram(binwidth=5)+
+   ylab('Frequency')+xlab('Age (years)')
```

Histograms



More information

- for more information, see the online reference

Brief Exercise

Brief Exercise

You may work in groups of 2–3. Save your code in a script file.

1. If you have not already, import `nhanes.csv` into R, using `read.csv()`.
2. If you have not already, install and load the ggplot2 package.
3. Use `plot()` to create a scatter plot of diastolic blood pressure (`bpxdi1`) and body mass index (`bmxbmi`), with diastolic blood pressure on the y axis.
4. Use ggplot2 to create the plot.

Linear regression

Linear regression with a continuous x: simple linear regression

- to fit a linear regression model, use `lm()`:

```
> lm(ii$bpxdi1~ii$bmxbmi)

Call:
lm(formula = ii$bpxdi1 ~ ii$bmxbmi)

Coefficients:
(Intercept)    ii$bmxbmi
    64.0737       0.2022
```

Linear regression with a continuous x: simple linear regression

- equivalently, using the `data` argument:

```
> lm(bpxdi1~bmxbmi,data=ii)

Call:
lm(formula = bpxdi1 ~ bmxbmi, data = ii)

Coefficients:
(Intercept)       bmxbmi
    64.0737       0.2022
```

Linear regression with a continuous x: simple linear regression

- it is often useful to save the regression:

```
> mm<-lm(bpxdi1~bmxbmi,data=ii)
> names(mm)
 [1] "coefficients"  "residuals"     "effects"
 [4] "rank"          "fitted.values" "assign"
 [7] "qr"            "df.residual"   "na.action"
[10] "xlevels"       "call"          "terms"
[13] "model"
> mm$coefficients
(Intercept)      bmxbmi
  64.073708    0.202183
```

Linear regression with a continuous x: simple linear regression

- for a confidence interval for the coefficients, use `confint()`:

```
> confint(mm)
                 2.5 %     97.5 %
(Intercept) 62.5654171 65.5819993
bmxbmi       0.1514157  0.2529503
> confint(mm,level=0.9)
                   5 %       95 %
(Intercept) 62.8079815 65.3394349
bmxbmi       0.1595801  0.2447859
```

Linear regression with a continuous x: simple linear regression

- for more information, use `summary()`:

```
> summary(mm)
...
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  64.0737     0.7694  83.281  < 2e-16 ***
bmxbmi        0.2022     0.0259   7.808 7.04e-15 ***
...
```

Linear regression with a continuous x: simple linear regression

```
> names(summary(mm))
 [1] "call"          "terms"         "residuals"
 [4] "coefficients"  "aliased"       "sigma"
 [7] "df"            "r.squared"     "adj.r.squared"
[10] "fstatistic"    "cov.unscaled"  "na.action"
> summary(mm)$coefficients
             Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 64.073708 0.76936625 83.281152 0.000000e+00
bmxbmi       0.202183 0.02589597  7.807509 7.040358e-15
```

Linear regression with a continuous x: simple linear regression

```
> summary(mm)$r.squared
[1] 0.01190813
> summary(mm)$adj.r.squared
[1] 0.01171278
```

Linear regression with a continuous x: simple linear regression

```
> mm<-lm(bpxdi1~bmxbmi,data=ii)
```

- if `mm` is a saved `lm()` object, then:
  - `mm$coefficients` provides just the coefficient estimates
  - `confint(mm)` provides confidence intervals for the coefficients
  - `summary(mm)$coefficients` provides the coefficient estimates, plus standard errors, test statistics, and p-values (for the null hypothesis that the coefficient is equal to 0)
  - `summary(mm)$r.squared` and `summary(mm)$adj.r.squared` provide R^2
  - for more information on output, see `help(lm)` and `help(summary.lm)`

Linear regression with a continuous x: quadratic adjustment

- for quadratic adjustment, use `poly()`:

```
> mm<-lm(bpxdi1~poly(ridageyr,2),data=ii)
> mm$coefficients
      (Intercept) poly(ridageyr, 2)1 poly(ridageyr, 2)2
        69.877825          -7.570431        -300.502878
```

Linear regression with a continuous x: natural cubic splines

- for natural cubic splines, use `ns()` in the splines package:

```
> mm<-lm(bpxdi1~ns(ridageyr,df=3),data=ii)
> mm$coefficients
          (Intercept) ns(ridageyr, df = 3)1
            62.358114              9.131939
ns(ridageyr, df = 3)2 ns(ridageyr, df = 3)3
            14.599818             -6.947398
```

Linear regression with a categorical x

```
> class(ii$hs)
[1] "factor"
```

- `hs` is a factor so R will automaticaly use dummy-variable adjustment, with a coefficient for each non-reference group
- however, we might want to change the reference:

```
> levels(ii$hs)
[1] "Smoking" "No"
> ii$hs<-relevel(ii$hs,'No')
```

Linear regression with a categorical x

```
> mm<-lm(bpxdi1~hs,data=ii)
> mm$coefficients
(Intercept)   hsSmoking
 70.0843794  -0.3931701
```

Linear regression with a categorical x

```
> confint(mm)
                2.5 %     97.5 %
(Intercept) 69.615385 70.5533734
hsSmoking   -1.101302  0.3149622
```

Linear regression with a categorical x

```
> summary(mm)$coefficients
              Estimate Std. Error    t value  Pr(>|t|)
(Intercept) 70.0843794  0.2392304 292.957693 0.0000000
hsSmoking   -0.3931701  0.3612130  -1.088472 0.2764383
```

Linear regression with a categorical x

```
> class(ii$bpcat)
[1] "factor"
> levels(ii$bpcat)
[1] "Elevated"            "Hypertensive crisis"
[3] "Normal"              "Stage 1"
[5] "Stage 2"
> ii$bpcat<-factor(ii$bpcat,c('Normal','Elevated','Stage 1',
+                             'Stage 2','Hypertensive crisis'))
```

Linear regression with a categorical x

```
> mm<-lm(bmxbmi~bpcat,data=ii)
> mm$coefficients
                (Intercept)                bpcatElevated
                  27.679274                    1.798396
             bpcatStage 1                bpcatStage 2
                  2.740511                    2.201618
bpcatHypertensive crisis
                 -1.542539
```

Linear regression with a categorical x

```
> confint(mm)
                                2.5 %     97.5 %
(Intercept)                27.399817 27.9587312
bpcatElevated               1.255394  2.3413977
bpcatStage 1                2.238340  3.2426829
bpcatStage 2                1.670157  2.7330794
bpcatHypertensive crisis   -3.472078  0.3869987
```

Linear regression with a categorical x

```
> summary(mm)$coefficients
                            Estimate Std. Error    t value
(Intercept)                27.679274  0.1425486 194.174324
bpcatElevated               1.798396  0.2769806   6.492859
bpcatStage 1                2.740511  0.2561533  10.698717
bpcatStage 2                2.201618  0.2710937   8.121245
bpcatHypertensive crisis   -1.542539  0.9842406  -1.567238
                              Pr(>|t|)
(Intercept)                0.000000e+00
bpcatElevated              9.229430e-11
bpcatStage 1               1.975779e-26
bpcatStage 2               5.748109e-16
bpcatHypertensive crisis   1.171216e-01
```

Linear regression with a categorical x

```
> summary(mm)$r.squared
[1] 0.0300087
> summary(mm)$adj.r.squared
[1] 0.02924115
```

Linear regression with several x

- to fit a linear regression model with several x variables, separate the x variables with a plus sign (+):

```
> lm(bpxdi1~bmxbmi+hs,data=ii)

Call:
lm(formula = bpxdi1 ~ bmxbmi + hs, data = ii)

Coefficients:
(Intercept)       bmxbmi    hsSmoking
    64.2507       0.2023      -0.4141
```

Linear regression with several x

```
> mm<-lm(bpxdi1~bmxbmi+hs,data=ii)
> mm$coefficients
(Intercept)       bmxbmi    hsSmoking
 64.2507494   0.2022985  -0.4141349
```

Linear regression with several x

```
> confint(mm)
                 2.5 %     97.5 %
(Intercept) 62.7117359 65.7897628
bmxbmi       0.1515279  0.2530691
hsSmoking   -1.1222203  0.2939506
```

Linear regression with several x

```
> summary(mm)$coefficients
              Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 64.2507494 0.78503738 81.844191 0.000000e+00
bmxbmi       0.2022985 0.02589765  7.811462 6.826063e-15
hsSmoking   -0.4141349 0.36118823 -1.146590 2.516053e-01
```

Linear regression with several x

```
> summary(mm)$r.squared
[1] 0.01216809
> summary(mm)$adj.r.squared
[1] 0.01177733
```

Interaction

- to include interaction, use the multiplication symbol (R will automaticlaly include the main effects):

```
> lm(bpxdi1~bmxbmi*hs,data=ii)

Call:
lm(formula = bpxdi1 ~ bmxbmi * hs, data = ii)

Coefficients:
     (Intercept)            bmxbmi         hsSmoking
        63.41218           0.23134           1.55474
bmxbmi:hsSmoking
        -0.06813
```

Interaction

```
> mm<-lm(bpxdi1~bmxbmi*hs,data=ii)
> mm$coefficients
     (Intercept)           bmxbmi        hsSmoking
     63.41217864       0.23134003       1.55474166
bmxbmi:hsSmoking
     -0.06812907
```

Interaction

```
> confint(mm)
                        2.5 %      97.5 %
(Intercept)        61.4209972 65.40336009
bmxbmi              0.1643161  0.29836394
hsSmoking          -1.4952691  4.60475244
bmxbmi:hsSmoking -0.1707855  0.03452736
```

Interaction

```
> summary(mm)$coefficients
                    Estimate Std. Error    t value
(Intercept)       63.41217864 1.01568432 62.4329600
bmxbmi             0.23134003 0.03418831  6.7666405
hsSmoking          1.55474166 1.55578394  0.9993301
bmxbmi:hsSmoking -0.06812907 0.05236415 -1.3010632
                      Pr(>|t|)
(Intercept)       0.000000e+00
bmxbmi            1.467701e-11
hsSmoking         3.176827e-01
bmxbmi:hsSmoking  1.932961e-01
```

Interaction

```
> summary(mm)$r.squared
[1] 0.01249877
> summary(mm)$adj.r.squared
[1] 0.01191272
```

Model comparison

- for model comparison, use `anova()`:

```
> mm1<-lm(bpxdi1~bmxbmi,data=ii)
> mm2<-lm(bpxdi1~bmxbmi+poly(ridageyr,2),data=ii)
> anova(mm2,mm1)
Analysis of Variance Table

Model 1: bpxdi1 ~ bmxbmi + poly(ridageyr, 2)
Model 2: bpxdi1 ~ bmxbmi
  Res.Df    RSS Df Sum of Sq     F    Pr(>F)
1   5056 747752
2   5058 821464 -2    -73711 249.2 < 2.2e-16 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Prediction

- to obtain predictions for a new set of data, you must use the `data` argument when you fit the model:

```
> mm<-lm(bpxdi1~bmxbmi,data=ii)
```

- the new data must have all of the x variable names that are in the model (in this case, `bmxbmi`):

```
> dd<-data.frame(bmxbmi=seq(20,80,by=10))
```

Prediction

```
> dd
  bmxbmi
1     20
2     30
3     40
4     50
5     60
6     70
7     80
```

Prediction

- to obtain the predictions, use `predict()` with the `newdata` argument:

```
> predict(mm,newdata=dd)
       1        2        3        4        5        6
68.11737 70.13920 72.16103 74.18286 76.20469 78.22652
       7
80.24835
```

Prediction

- you may omit the argument name:

```
> predict(mm,dd)
       1        2        3        4        5        6
68.11737 70.13920 72.16103 74.18286 76.20469 78.22652
       7
80.24835
```

Prediction

- you might consider adding the predictions to the new data:

```
> dd$prediction<-predict(mm,dd)
> dd
  bmxbmi prediction
1     20   68.11737
2     30   70.13920
3     40   72.16103
4     50   74.18286
5     60   76.20469
6     70   78.22652
7     80   80.24835
```

Prediction

- an example with 2 x variables:

```
> mm<-lm(bpxdi1~bmxbmi+hs,data=ii)
> dd<-data.frame(bmxbmi=rep(seq(20,80,by=20),2),
+                hs=c(rep('Smoking',4),rep('No',4)))
```

Prediction

```
> dd
  bmxbmi      hs
1     20 Smoking
2     40 Smoking
3     60 Smoking
4     80 Smoking
5     20      No
6     40      No
7     60      No
8     80      No
```

Prediction

```
> dd$prediction<-predict(mm,dd)
> dd
  bmxbmi      hs prediction
1     20 Smoking   67.88258
2     40 Smoking   71.92855
3     60 Smoking   75.97452
4     80 Smoking   80.02049
5     20      No   68.29672
6     40      No   72.34269
7     60      No   76.38866
8     80      No   80.43463
```