PM592: Regression Analysis for Health Data Science
Lab 3 – Visualizations & Functions
Data Needed: *chs_merged (week 2), no2.sas7bdat*

Outline

- Variable Manipulation
- Functions
- Ggplot2
- Correlation
- Regression

1. Variable Manipulation: Forming Groups

1.1. Cut

1.1.1. Previously we had taken certain variables and tried to group them into categories. We accomplished this using either the "if_else" or "case_when" command. The "cut" command takes a vector and categorizes it into categories using the specified cutpoints.

1.1.2. Suppose we want to take the "chs_merged" data from last lab and cut the BMI variable into cutpoints using the specified criteria:

Standard BMI Categories

Weight Status Category	**BMI Range** (kg/m²)
Underweight	Below 18.5
Healthy weight	18.5 to 24.9
Overweight	25 to 29.9
Obese	30 or greater

1.1.3. Note the behavior of the following options: "right," "labels," "ordered_result". In the following example, we create 4 different versions of a BMI cutpoint variable. The third version adds labels to the factor variable, and the fourth version specifies that the variable is an ordinal factor. By specifying that an ordered factor is ordinal, several functions in R will behave accordingly.

```
> # Create BMI cutpoints.
> chs_merged <-
+   chs_merged %>%
+   mutate(
+     bmicat1 = cut(
+       bmi,
+       breaks = c(0, 18.5, 25, 30, 100)
+     ),
+     bmicat2 = cut(
+       bmi,
```

```
+       breaks = c(0, 18.5, 25, 30, 100),
+       right = F
+     ),
+     bmicat3 = cut(
+       bmi,
+       breaks = c(0, 18.5, 25, 30, 100),
+       right = F,
+       labels = c("Underweight", "Healthy Weight", "Overweight", "Obese")
+     ),
+     bmicat4 = cut(
+       bmi,
+       breaks = c(0, 18.5, 25, 30, 100),
+       right = F,
+       labels = c("Underweight", "Healthy Weight", "Overweight", "Obese"),
+       ordered_result = T
+     )
+   )
> chs_merged %>%
+   select(starts_with("bmicat"))
# A tibble: 1,200 x 4
   bmicat1  bmicat2  bmicat3     bmicat4
   <fct>    <fct>    <fct>       <ord>
 1 (0,18.5] [0,18.5) Underweight Underweight
 2 (0,18.5] [0,18.5) Underweight Underweight
 3 (30,100] [30,100) Obese       Obese
 4 (0,18.5] [0,18.5) Underweight Underweight
 5 (0,18.5] [0,18.5) Underweight Underweight
 6 NA       NA       NA          NA
 7 (0,18.5] [0,18.5) Underweight Underweight
 8 (0,18.5] [0,18.5) Underweight Underweight
 9 NA       NA       NA          NA
10 (0,18.5] [0,18.5) Underweight Underweight
# ... with 1,190 more rows
```

- Which version of the variable would you keep and why?

1.2. Quantile

1.2.1. Sometimes we may want to categorize a variable by quantile instead of by pre-specified cutpoints.

1.2.2. The “ntile” function in R will cut your specified variable into n quantiles.

1.2.3. In the following code, we create a 4-quantile and a 5-quantile variable for FEV, and then we look at the minimum, maximum, and mean FEV values within each quantile.

```
> chs_merged %>%
+   group_by(fev4c) %>%
+   summarise(
+     min = min(fev),
+     max = max(fev),
+     mean = mean(fev),
+     n = n()
+   )
`summarise()` ungrouping output (override with `.groups` argument)
```

```
# A tibble: 5 x 5
  fev4c   min   max  mean     n
  <int> <dbl> <dbl> <dbl> <int>
1     1  985. 1809. 1627.   277
2     2 1809. 2023. 1913.   276
3     3 2025. 2250. 2132.   276
4     4 2250  3324. 2454.   276
5    NA   NA    NA    NA     95
>
> chs_merged %>%
+   group_by(fev5c) %>%
+   summarise(
+     min = min(fev),
+     max = max(fev),
+     mean = mean(fev),
+     n = n()
+   )
`summarise()` ungrouping output (override with `.groups` argument)
# A tibble: 6 x 5
  fev5c   min   max  mean     n
  <int> <dbl> <dbl> <dbl> <int>
1     1  985. 1752. 1588.   221
2     2 1752  1933. 1845.   221
3     3 1933. 2105. 2022.   221
4     4 2109. 2296. 2201.   221
5     5 2297. 3324. 2500.   221
6    NA   NA    NA    NA     95
```

2. Functions

2.1. So far we have been using built-in functions within R. You may find that there are some operations you frequently perform that you may like to write your own functions for.

2.2. R will automatically return whichever variable is on the last line of the body of the function. However you can also return different values by explicitly specifying “return()”.

2.3. A function usually takes in one or more arguments, but this is not necessary:

```
> hello.world <- function() {
+   print("Hello world!")
+ }
> hello.world()
[1] "Hello world!"
```

2.4. Here is a function that converts temperature Fahrenheit to Celsius:

```
> # Second function: F to C
> f.to.c <- function(temp_F) {
+   temp_C <- (temp_F - 32) * 5 / 9
+   return(temp_C)
+ }
> f.to.c(32)
[1] 0
> f.to.c(100)
[1] 37.77778
```

2.5. You can create a function that modifies the behavior of existing functions. The following user-created function "mean0" will automatically remove NA values without explicitly having to state that.

```
> mean0 <- function(x) {
+   mean(x,  na.rm=T)
+ }
> mean(chs_merged$fev)
[1] NA
> mean0(chs_merged$fev)
[1] 2031.265
```

2.6. Functions can take in more than one argument as well:

```
> calc.bmi <- function(ht_inches, wt_pounds) {
+   bmi <- 703*wt_pounds/ht_inches^2
+   return(bmi)
+ }
> calc.bmi(62, 130)
[1] 23.77471
```

2.7. Functions can be quite complex. In the example in 1.1. we created BMI categories using the category definitions for adults. However since our sample is on adolescents, we should be using CDC-published BMI percentiles to study BMI. We could create these variables on our own, or we could use a user-defined function for this purpose. The following link is to a function that returns a BMI percentile for a given weight (kg), height (cm), age, and sex. Note how complex these functions can be.
https://rdrr.io/cran/PAutilities/src/R/get_BMI_percentile.R

3. **Ggplot2**

3.1. R has the capability to produce adequate figures. However, the "ggplot2" package can be used to create stunning visuals. Note that there is a *lot* to explore in ggplot, but we will go over some of the basic functions and syntax here. More in-depth guides are available online (http://r-statistics.co/Complete-Ggplot2-Tutorial-Part1-With-R-Code.html).

3.2. The basics: what you need to include

3.2.1. The name of your data set (obviously).

3.2.2. An **aesthetic**. This refers to what you want to graph and how you want it to look. Depending on the type of graph, you may want to have the following aesthetics:

3.2.2.1. x and y: which variables define your x and y axes

3.2.2.2. Fill: a variable that will define different colors, patterns, or sizes for the "fill"

3.2.2.3. Color: a variable that will define different colors, patterns, or sizes for the "color"

3.2.2.4. Group: a variable that will define groups for stratified analyses

3.2.3. A **geom**. This refers to how you want to graph your data (i.e., what will appear on the graph). We have already seen some common geoms such as:

3.2.3.1. Geom_histogram: histogram

3.2.3.2. Geom_density: a density histogram

3.2.3.3. Geom_point: scatterplot of points

3.2.3.4. Geom_boxplot: boxplots

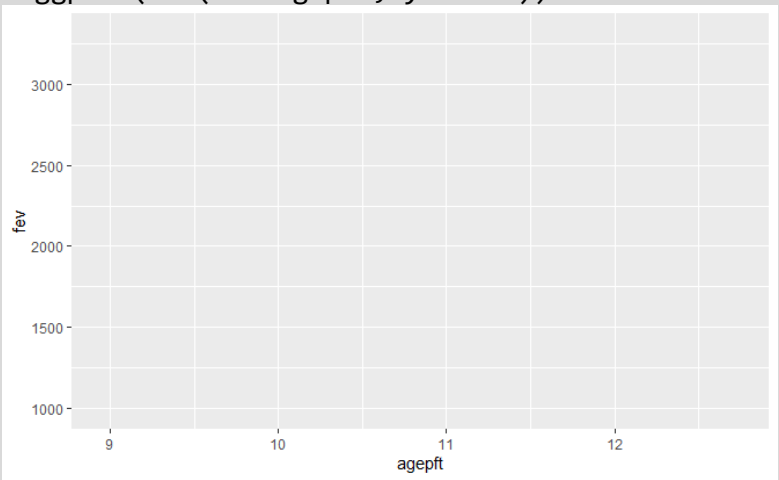
3.2.3.5. Geom_smooth: a line

3.2.4. Optionally you can specify more to enhance your graph. This includes things such as labels, titles, themes, ways to change your axes, annotations on your graph, etc.

3.3. Setting up the aesthetic.

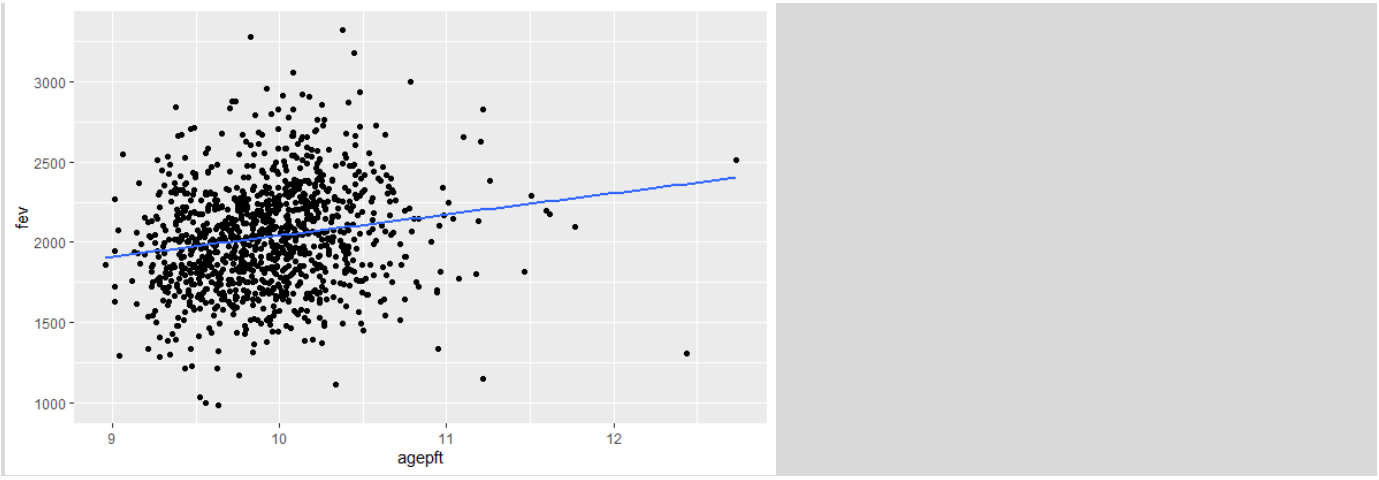
3.3.1. Suppose we want to examine how FEV changes with increasing age. In the following code note that we have set the aesthetic (and our axes are set up accordingly). However we did not specify how we wanted these two variables graphed (Do we want a scatter point? Do we want a line?).

```
chs_merged %>%
  ggplot(aes(x = agepft, y = fev))
```



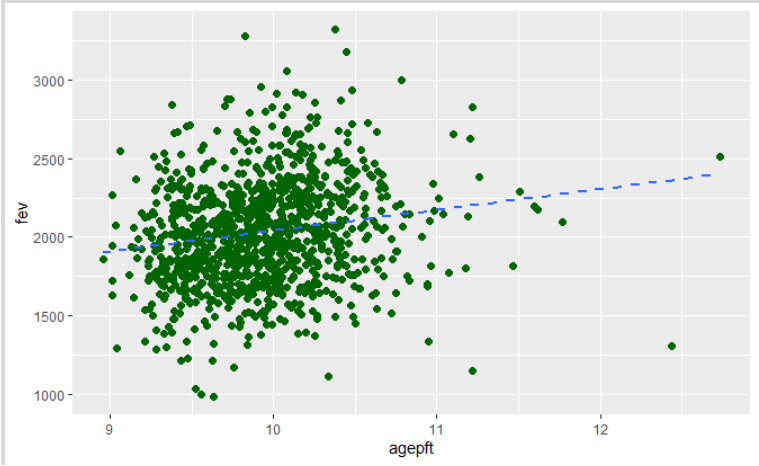
3.3.2. We will now tell ggplot that we want to add points and a least-squares line.

```
> chs_merged %>%
+   ggplot(aes(x = agepft, y = fev)) +
+   geom_point() +
+   geom_smooth(se = F, method = "lm")
`geom_smooth()` using formula 'y ~ x'
Warning messages:
1: Removed 95 rows containing non-finite values (stat_smooth).
2: Removed 95 rows containing missing values (geom_point).
```



3.3.3. Let's add some options. We'll make the points green and larger, and the line dashed.

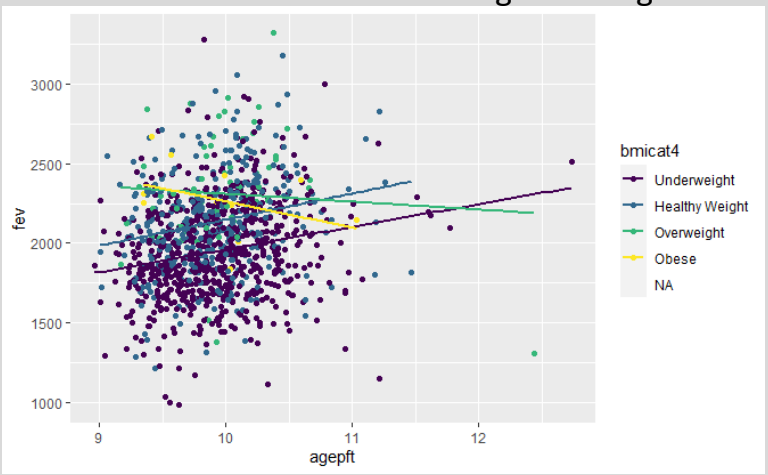
```
> chs_merged %>%
+   ggplot(aes(x = agepft, y = fev)) +
+   geom_point(size = 2, color = "darkgreen") +
+   geom_smooth(se = F, method = "lm", linetype = "dashed")
`geom_smooth()` using formula 'y ~ x'
Warning messages:
1: Removed 95 rows containing non-finite values (stat_smooth).
2: Removed 95 rows containing missing values (geom_point).
```



3.3.4. Finally, we'll color the points and line by "BMI" group. This will allow us to see how FEV relates to age within each of these BMI categories. This is interesting – it tells us that FEV increases with age for underweight and healthy weight individuals, but not for overweight and obese individuals.

```
> chs_merged %>%
+   ggplot(aes(x = agepft, y = fev, color = bmicat4)) +
+   geom_point() +
+   geom_smooth(se = F, method = "lm")
`geom_smooth()` using formula 'y ~ x'
Warning messages:
1: Removed 95 rows containing non-finite values (stat_smooth).
```

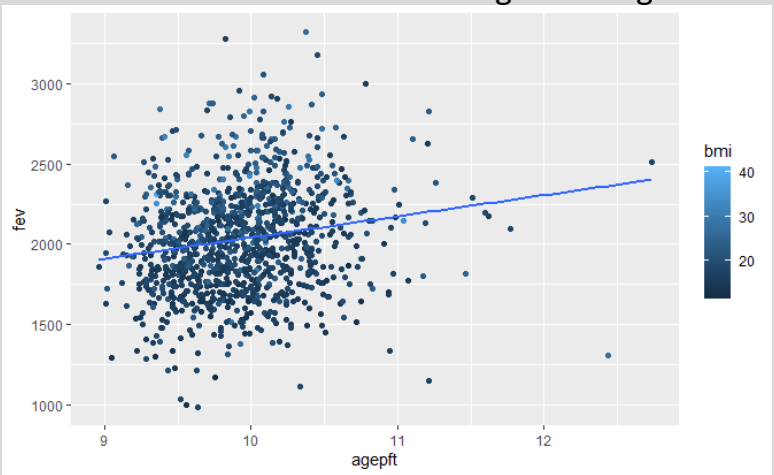
```
2: Removed 95 rows containing missing values (geom_point).
```



- What happens if you use the "bmicat3" variable for color instead of "bmicat4"? Why do you think the graph changed?

3.3.5. One of the great things about ggplot is that it recognizes what kind of variables you are using and draws its graphs accordingly. In 3.3.4 we colored variables using a categorical variable, so it created a discrete color coding system for us. Here, we color the points by BMI, a continuous variable, so it will create a continuous color coding scheme.

```
> chs_merged %>%
+   ggplot(aes(x = agepft, y = fev, color = bmi)) +
+   geom_point() +
+   geom_smooth(se = F, method = "lm")
`geom_smooth()` using formula 'y ~ x'
Warning messages:
1: Removed 95 rows containing non-finite values (stat_smooth).
2: Removed 95 rows containing missing values (geom_point).
```



4. Correlation

4.1. We can use the cor() function to arrive at correlation coefficients for variables. By default, if any data is missing then "NA" will be reported. Some common ways to fix this is to use only complete observations (use = "complete.obs") or to use only pairwise complete observations (use = "pairwise.complete.obs").

```
> chs_merged %>%
+   select(agepft, fev, fvc, mmef) %>%
+   cor()
       agepft fev fvc mmef
agepft      1  NA  NA   NA
fev        NA   1  NA   NA
fvc        NA  NA   1   NA
mmef       NA  NA  NA    1
> chs_merged %>%
+   select(agepft, fev, fvc, mmef) %>%
+   cor(use = "pairwise.complete.obs")
           agepft       fev       fvc       mmef
agepft 1.00000000 0.1714370 0.1857674 0.08282617
fev    0.17143699 1.0000000 0.9164952 0.63816935
fvc    0.18576738 0.9164952 1.0000000 0.33811832
mmef   0.08282617 0.6381694 0.3381183 1.00000000
> chs_merged %>%
+   select(agepft, fev, fvc, mmef) %>%
+   cor(use = "complete.obs")
           agepft       fev       fvc       mmef
agepft 1.00000000 0.1731801 0.1937429 0.07987482
fev    0.17318011 1.0000000 0.9180125 0.63908425
fvc    0.19374289 0.9180125 1.0000000 0.33822368
mmef   0.07987482 0.6390843 0.3382237 1.00000000
```

4.2. The cor.test() function can be used to find p-values for <u>two</u> variables.

```
> cor.test(chs_merged$fev, chs_merged$fvc)

      Pearson's product-moment correlation

data:  chs_merged$fev and chs_merged$fvc
t = 75.914, df = 1098, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.9064941 0.9254683
sample estimates:
      cor
0.9164952
```

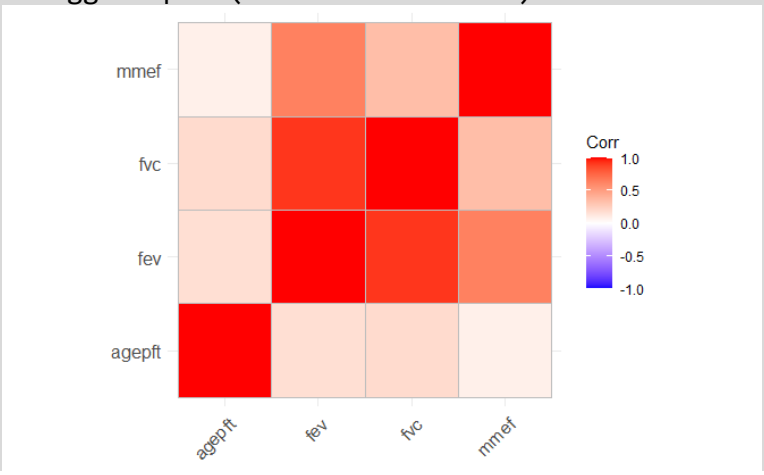
4.3. Other packages, such as Hmisc, allow for the calculation of p-values on the entire correlation matrix. You can specify in the rcorr() function whether you want Pearson's or Spearman's correlation.

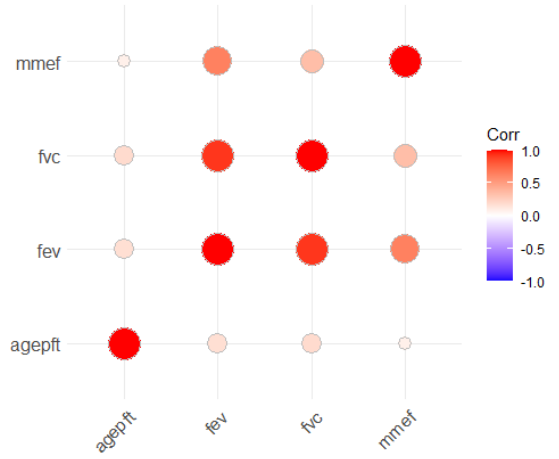
```
> chs_merged %>%
+   select(agepft, fev, fvc, mmef) %>%
+   as.matrix() %>%
+   rcorr(type = "pearson")
       agepft  fev  fvc mmef
agepft   1.00 0.17 0.19 0.08
fev      0.17 1.00 0.92 0.64
fvc      0.19 0.92 1.00 0.34
mmef     0.08 0.64 0.34 1.00
```

```
n
       agepft  fev  fvc mmef
agepft   1111 1105 1103 1094
fev      1105 1105 1100 1093
fvc      1103 1100 1103 1090
mmef     1094 1093 1090 1094

P
       agepft fev    fvc    mmef
agepft        0.0000 0.0000 0.0061
fev    0.0000        0.0000 0.0000
fvc    0.0000 0.0000        0.0000
mmef   0.0061 0.0000 0.0000
```

4.4. The package "ggcorr" interfaces with ggplot to produce visualizations of correlations (correllograms). The following code produces the default correlogram, and then demonstrates how to change the visual style. Other options, such as adding the p-values, are also available. We clearly see that FEV and FVC are the most highly related of these four variables.

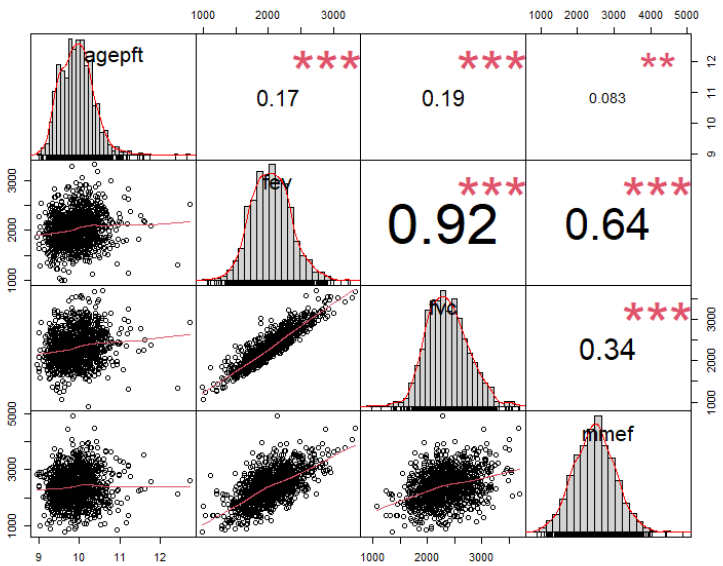
```
> chs_merged %>%
+   select(agepft, fev, fvc, mmef) %>%
+   cor(use = "pairwise.complete.obs") %>%
+   ggcorrplot()

> chs_merged %>%
+   select(agepft, fev, fvc, mmef) %>%
+   cor(use = "pairwise.complete.obs") %>%
+   ggcorrplot(method = "circle")
```





4.5. Other packages, such as PerformanceAnalytics, offer alternate ways of viewing the data. You can also specify Pearson's or Spearman's correlation in the function (though Pearson's is the default).

```
> chs_merged %>%
+   select(agepft, fev, fvc, mmef) %>%
+   chart.Correlation(histogram=TRUE, pch=19)
```



5. Single Linear Regression

5.1. Example

5.1.1. We previously saw that FEV tends to increase with age. Let's run a linear regression to examine the nature of this relationship.

5.1.2. Remember that we want to examine our data visually first, to make sure the assumptions of linear regression are met.

```
> model1 <-
+   lm(fev ~ agepft, data = chs_merged)
> model1 %>%
+   summary()

Call:
```

```
lm(formula = fev ~ agepft, data = chs_merged)

Residuals:
     Min       1Q   Median       3Q      Max
-1059.69  -220.96    -5.63   211.01  1265.43

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   721.41     226.86   3.180  0.00151 **
agepft        131.98      22.84   5.779 9.76e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 325.9 on 1103 degrees of freedom
  (95 observations deleted due to missingness)
Multiple R-squared:  0.02939, Adjusted R-squared:  0.02851
F-statistic:  33.4 on 1 and 1103 DF,  p-value: 9.758e-09
```

5.1.3. What can we tell from this output?

- The predicted FEV when agepft=0 is 721.41. This is statistically significantly different from 0 (p=.00151)
  (Is this meaningful? How would we make it more meaningful?)
- When agepft increases by 1 (1 year), FEV is expected to increase by 131.98 (p<.001).
- Our best-fit line is: $\hat{Y} = 721.4 + 132.0X_{agepft}$

5.2. Statistical vs. Practical Effects

5.2.1. Statistical significance specifically refers to the hypotheses being tested. In this case, we determine that, on average, FEV increases as age increases. A statistically significant effect, though, does not need to be a large effect. The standard error and sample size are both related to the p-value.

5.2.2. Practical significance refers to how large the effect size is. For example you may find that a new therapy reduces the risk of disease by 0.0001%, but the new therapy is expensive and time-consuming. Even if you find that p<.05, the effect size may be so small that it is not practical to implement such a program.

5.2.3. A standardized coefficient essentially refers to the relationship between the z-scores of two variables. As the variables are converted to z-scores, this gives us a standardized scale on which to measure the effect size. To make it easy, we can standardize directly within the lm() function by specifying scale().
Note: for single linear regression the standardized coefficient is equivalent to R.
Note: the standardized coefficient is equivalent to $\frac{\beta}{SD(\beta)} = \frac{\beta}{SE(\beta)\sqrt{N}}$

5.2.4. The interpretation of a standardized coefficient is “a one standard deviation increase in X is associated with a β standard deviation increase in Y.”

```
> model1_std <-
+   lm(scale(fev) ~ scale(agepft), data = chs_merged)
> model1_std %>%
+   summary()

Call:
lm(formula = scale(fev) ~ scale(agepft), data = chs_merged)
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-3.2047 -0.6682 -0.0170  0.6381  3.8269

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)   -0.0002698  0.0296509  -0.009    0.993
scale(agepft)  0.1713223  0.0296445   5.779 9.76e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.9856 on 1103 degrees of freedom
  (95 observations deleted due to missingness)
Multiple R-squared:  0.02939, Adjusted R-squared:  0.02851
F-statistic:  33.4 on 1 and 1103 DF,  p-value: 9.758e-09

> 131.98/(22.84*sqrt(1104))
[1] 0.1739112
```

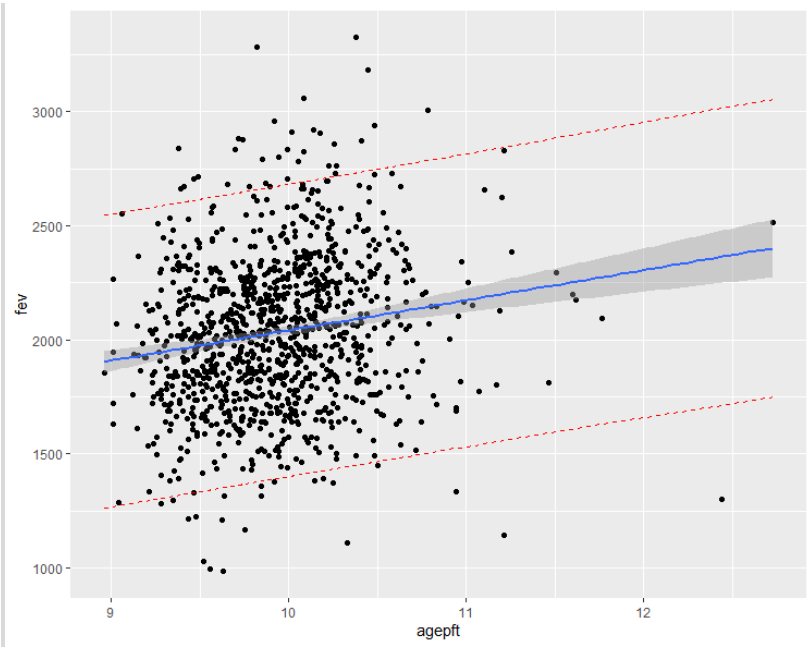
5.2.5. Another way to assess the practical effect is through the multiple R^2. This refers to the percent of the variance in the outcome that is explained by your predictors. Here, 2.9% of the variation in FEV is explained by the linear relationship with age. (This means that 97.1% of the variation in FEV is explained by other things.) The practical interpretation of R^2 varies by discipline; for example, in psychology and health behavior this value is typically low, but in “hard” sciences the expectation may be that R^2 is higher.

5.3. Confidence Intervals and Prediction Intervals

5.3.1. Recall that ggplot automatically provides 95% confidence intervals unless specified otherwise. The confidence interval tells us about our certainty for the mean value.

5.3.2. We can also obtain the 95% prediction intervals. The prediction interval tells us about our certainty for individual predictions.

```
> chs_merged2 <- cbind(
+   chs_merged,
+   predict(model1, chs_merged, interval="prediction")
+ )

> chs_merged2 %>%
+   ggplot(aes(x = agepft, y = fev))+
+   geom_point() +
+   geom_line(aes(y=lwr), color = "red", linetype = "dashed")+
+   geom_line(aes(y=upr), color = "red", linetype = "dashed")+
+   geom_smooth(method=lm, se=TRUE)
```

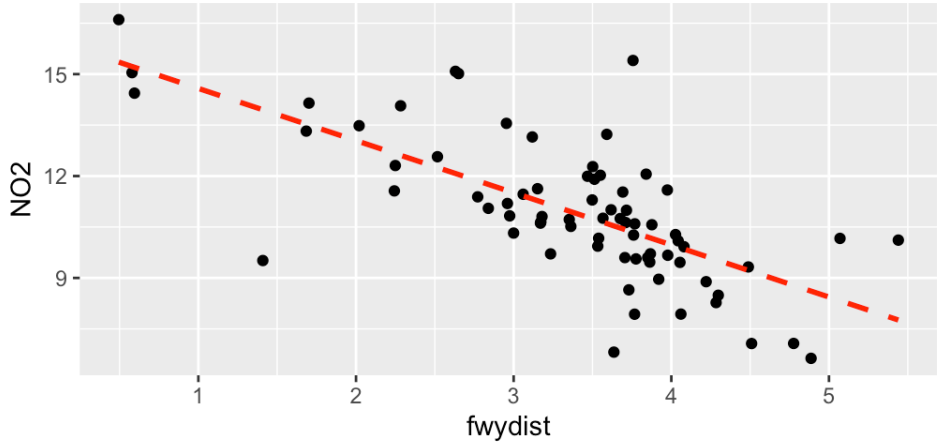

3000
2500
2000
1500
1000
fev
9
10
11
12
agepft

Lab 3 Exercises

Objective(s):	Merge two data sets, create factor variables, assess variable normality, practice variable manipulation
Datasets Required:	NO2

We will use the NO2 data for this lab. This data examines levels of nitrogen dioxide (NO_2, in ppb) at children's homes and the distance of the nearest freeway (km). You will be assigned a town.

1) Create a scatterplot for your town with X = distance to freeway and Y = NO2 exposure.
   a) From only this scatterplot, what do you think the NO_2 exposure is for children living right next to the freeway?



   It looks like NO2 exposure for children living right next to the freeway is about 15.1

2) Perform a regression on these two variables.
   a) What is your regression line?

      Y = 16.1148 + (-1.5354)X

   b) Is distance from the freeway related to NO_2 exposure?

      Yes, with p-value 8.18e-12 ***, we can reject the null hypothesis that the slope of the regression line of freeway distance on NO2 levels is not equal to 0

   c) In words, interpret your intercept value.

      The predicted NO2 exposure for a child living right next to the freeway is 16.1

   d) In words, interpret your slope value.

      For each 1 km increase in distance from freeway, the expected NO2 exposure level decreases by 1.54 ppb

   e) How much cleaner is the air 2km away from the freeway?

      Compared to being right next to the freeway, the NO2 exposure is about 3 ppb lower.

f) Do your answers make sense when compared to your scatterplot?

Yes, the answers seem to make sense when looking at the scatterplot, there is a slightly negative relationship between freeway distance and NO2 exposure, and the intercept is about correct.

3) Center the distance variable on its mean and re-run the regression.
   a) What is your regression line?

   Y = 10.9362 + (-1.5354)X

   b) Which parameter estimates changed and which didn't? Why?

   The parameter that changed is the intercept (B0), the slope did not change. This is because centering the X values doesn't change the relationship between X and Y, it only shifts the X values to have a mean of 0. The intercept is now interpreted as the predicted NO2 value for someone living at the mean distance from freeway.

4) Create a new variable that is distance to the nearest freeway in miles. Perform the regression between NO_2 levels and distance in miles.
   a) What is your regression line?

   Y = 16.1148 + (-2.4710)X

   b) Which parameter estimates changed and which didn't? Why?

   This time, the intercept did not change, but the slope did change. This is because the scaling of the X variables changed, they were all divided by 1.61 to change kilometers into miles.

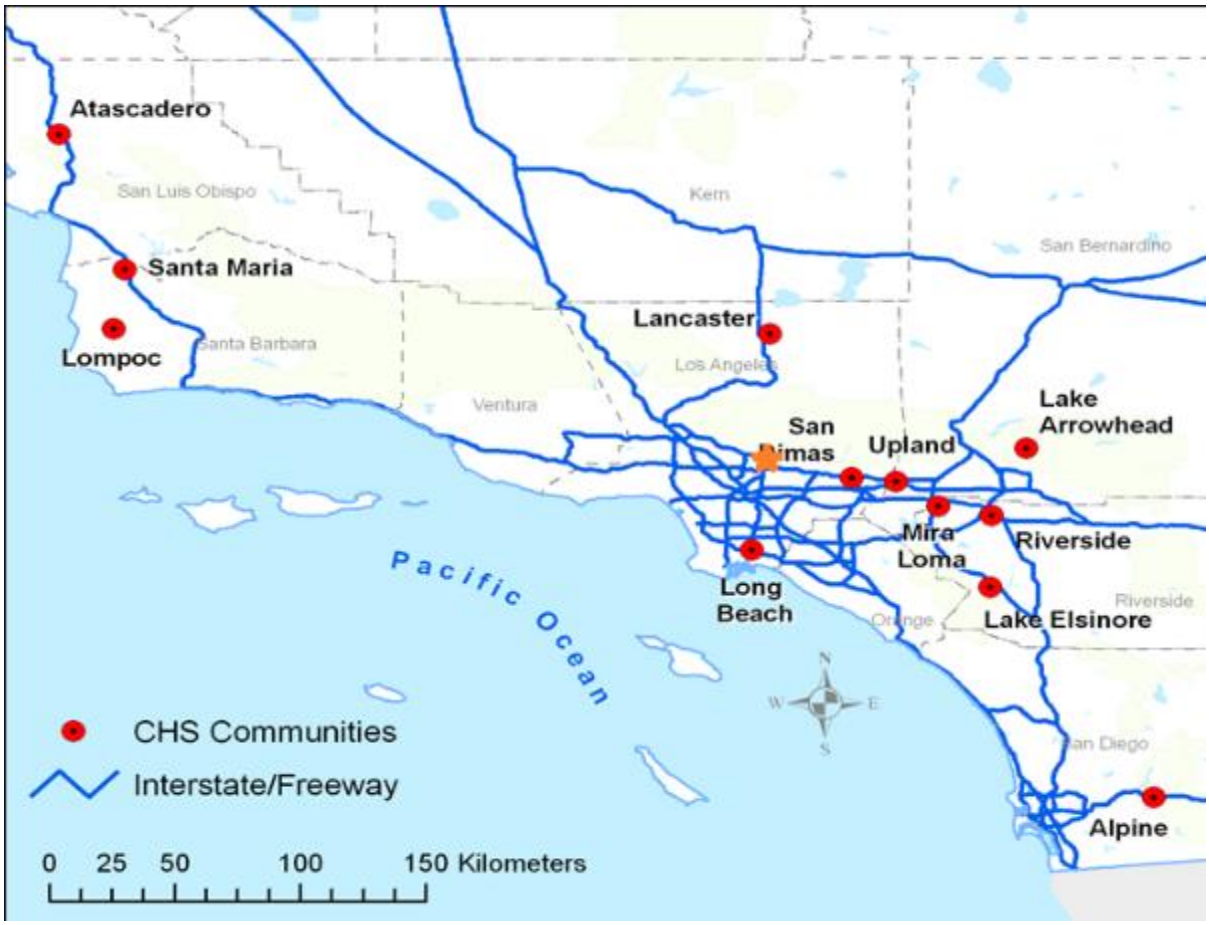
5) Using this data, how would you go about making a "best guess" as to what the NO_2 exposure is at the Soto Street Building (SSB) at USC?
   a) Provide your prediction.

USC SSB is at the orange star below. It is approximately 1.5 km from the 10 freeway.

As a best guess for USC SSB, I would use data from Long Beach, Anaheim, and Glendora. East LA seems to be in the middle of these cities. The model I got from using these three cities is:

Y = 30.3667 + (-2.6150)X

Y = 26.442 predicted NO2 ppb


Atascadero
San Luis Obispo
Kern
San Bernardino
Santa Maria
Lancaster
Lompoc
Santa Barbara
Los Angeles
Ventura
Lake Arrowhead
San Dimas
Upland
Mira Loma
Riverside
Pacific Ocean
Long Beach
Riverside
Orange
Lake Elsinore
N
W
E
S
San Diego
Alpine
CHS Communities
Interstate/Freeway
0 25 50 100 150 Kilometers