Assignment 10

Adrian Bracher (Matr. Nr. 01637180)

9.6.2021

Contents

1 Introduction to survey data

Note: Sadly I could not find the NHANES data, so a lot of the code can not be evaluated.

```
library(irr)
library(psych)
library(psychometric)
library(dplyr)
library(likert)
library(car)
library(Hmisc)
library(tidyr)
library(corrplot)
library(lavaan)
```

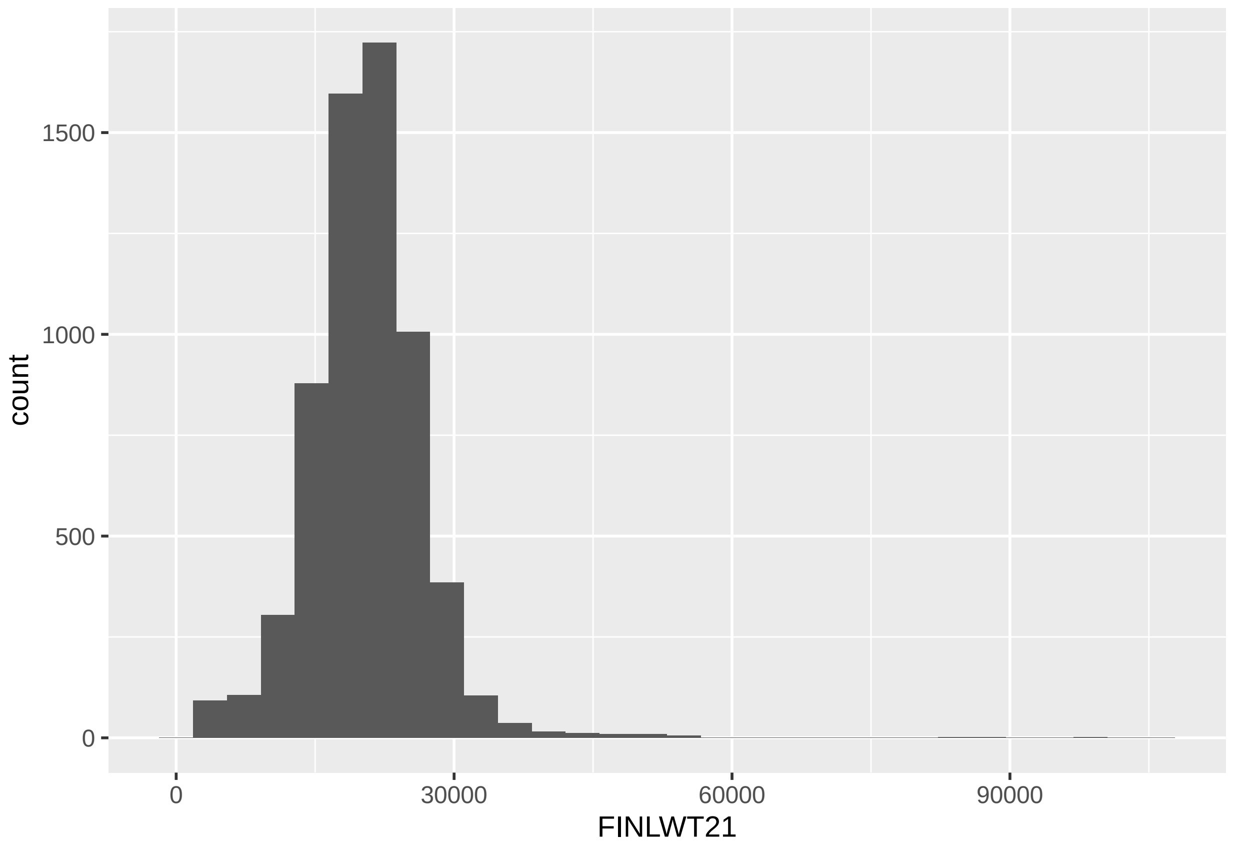
1.1 Visualizing the weights

```
# Load ggplot2
library(ggplot2)

ce = read.csv("ce.csv")

# Construct a histogram of the weights
ggplot(data = ce, mapping = aes(x = FINLWT21)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



1.2 Designs in R

In this exercise we learn how to use the function svydesign from the package survey.

```
library(survey)
```

```
## Loading required package: grid

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

##
## Attaching package: 'survey'

## The following object is masked from 'package:Hmisc':
##
##     deff

## The following object is masked from 'package:graphics':
##
##     dotchart
```

```
# Look at the apisrs dataset
data(api)

glimpse(apisrs)
```

```
## Rows: 200
## Columns: 39
## $ cds      <chr> "15739081534155", "19642126066716", "30664493030640", "196445~
## $ stype    <fct> H, E, H, E, E, E, M, E, E, E, E, H, M, E, E, E, M, M, H, E, E~
## $ name     <chr> "McFarland High", "Stowers (Cecil ", "Brea-Olinda Hig", "Alam~
## $ sname    <chr> "McFarland High", "Stowers (Cecil B.) Elementary", "Brea-Olin~
## $ snum     <dbl> 1039, 1124, 2868, 1273, 4926, 2463, 2031, 1736, 2142, 4754, 1~
## $ dname    <chr> "McFarland Unified", "ABC Unified", "Brea-Olinda Unified", "D~
## $ dnum     <int> 432, 1, 79, 187, 640, 284, 401, 401, 470, 632, 401, 753, 784,~
## $ cname    <chr> "Kern", "Los Angeles", "Orange", "Los Angeles", "San Luis Obi~
## $ cnum     <int> 14, 18, 29, 18, 39, 18, 18, 18, 18, 37, 18, 24, 14, 1, 47, 18~
## $ flag     <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ pcttest  <int> 98, 100, 98, 99, 99, 93, 98, 99, 100, 90, 95, 100, 97, 99, 98~
## $ api00    <int> 462, 878, 734, 772, 739, 835, 456, 506, 543, 649, 556, 671, 5~
## $ api99    <int> 448, 831, 742, 657, 719, 822, 472, 474, 458, 604, 575, 620, 5~
## $ target   <int> 18, NA, 3, 7, 4, NA, 16, 16, 17, 10, 11, 9, 14, 5, 15, 10, 6,~
## $ growth   <int> 14, 47, -8, 115, 20, 13, -16, 32, 85, 45, -19, 51, 4, 51, 50,~
## $ sch.wide <fct> No, Yes, No, Yes, Yes, Yes, No, Yes, Yes, Yes, No, Yes, No, Y~
## $ comp.imp <fct> Yes, Yes, No, Yes, Yes, Yes, No, Yes, Yes, No, No, Yes, No, Y~
## $ both     <fct> No, Yes, No, Yes, Yes, Yes, No, Yes, Yes, No, No, Yes, No, Ye~
## $ awards   <fct> No, Yes, No, Yes, Yes, No, No, Yes, Yes, No, No, Yes, No, Yes~
## $ meals    <int> 44, 8, 10, 70, 43, 16, 81, 98, 94, 85, 81, 67, 77, 20, 70, 75~
## $ ell      <int> 31, 25, 10, 25, 12, 19, 40, 65, 65, 57, 4, 25, 32, 16, 23, 18~
## $ yr.rnd   <fct> NA, NA, NA, NA, NA, NA, NA, No, NA, NA, NA, NA, NA, NA, No, N~
## $ mobility <int> 6, 15, 7, 23, 12, 13, 22, 43, 15, 10, 20, 12, 4, 32, 17, 9, 1~
## $ acs.k3   <int> NA, 19, NA, 23, 20, 19, NA, 18, 19, 16, 16, NA, NA, 19, 21, 2~
## $ acs.46   <int> NA, 30, NA, NA, 29, 29, 30, 29, 32, 25, 27, NA, NA, 29, 30, 2~
## $ acs.core <int> 24, NA, 28, NA, NA, NA, 27, NA, NA, 30, NA, 17, 27, NA, NA, N~
## $ pct.resp <int> 82, 97, 95, 100, 91, 71, 49, 75, 99, 49, 62, 96, 77, 96, 39, ~
## $ not.hsg  <int> 44, 4, 5, 37, 8, 1, 30, 49, 48, 23, 5, 44, 40, 4, 14, 18, 15,~
## $ hsg      <int> 34, 10, 9, 40, 21, 8, 27, 31, 34, 36, 38, 19, 34, 14, 57, 28,~
## $ some.col <int> 12, 23, 21, 14, 27, 20, 18, 15, 14, 14, 29, 17, 16, 25, 18, 2~
## $ col.grad <int> 7, 43, 41, 8, 34, 38, 22, 2, 4, 21, 24, 19, 8, 37, 10, 23, 28~
## $ grad.sch <int> 3, 21, 24, 1, 10, 34, 2, 3, 1, 6, 5, 2, 2, 19, 1, 3, 10, 32, ~
## $ avg.ed   <dbl> 1.91, 3.66, 3.71, 1.96, 3.17, 3.96, 2.39, 1.79, 1.77, 2.51, 2~
## $ full     <int> 71, 90, 83, 85, 100, 75, 72, 69, 68, 81, 84, 100, 89, 95, 96,~
## $ emer     <int> 35, 10, 18, 18, 0, 20, 25, 22, 29, 7, 16, 0, 11, 5, 6, 10, 8,~
## $ enroll   <int> 477, 478, 1410, 342, 217, 258, 1274, 566, 645, 311, 328, 210,~
## $ api.stu  <int> 429, 420, 1287, 291, 189, 211, 1090, 353, 563, 258, 253, 166,~
## $ pw       <dbl> 30.97, 30.97, 30.97, 30.97, 30.97, 30.97, 30.97, 30.97, 30.97~
## $ fpc      <dbl> 6194, 6194, 6194, 6194, 6194, 6194, 6194, 6194, 6194, 6194, 6~
```

```
# Specify a simple random sampling for apisrs
apisrs_design <- svydesign(data = apisrs, weights = ~pw, fpc = ~fpc, id = ~1)

# Produce a summary of the design
summary(apisrs_design)
```

```
## Independent Sampling design
## svydesign(data = apisrs, weights = ~pw, fpc = ~fpc, id = ~1)
## Probabilities:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.03229 0.03229 0.03229 0.03229 0.03229 0.03229
## Population size (PSUs): 6194
## Data variables:
##  [1] "cds"      "stype"    "name"     "sname"    "snum"     "dname"
##  [7] "dnum"     "cname"    "cnum"     "flag"     "pcttest"  "api00"
## [13] "api99"    "target"   "growth"   "sch.wide" "comp.imp" "both"
## [19] "awards"   "meals"    "ell"      "yr.rnd"   "mobility" "acs.k3"
## [25] "acs.46"   "acs.core" "pct.resp" "not.hsg"  "hsg"      "some.col"
## [31] "col.grad" "grad.sch" "avg.ed"   "full"     "emer"     "enroll"
## [37] "api.stu"  "pw"       "fpc"
```

1.3 Stratified designs in R

We specify stratified sampling design in the code below.

```
# Glimpse the data
glimpse(apistrat)
```

```
## Rows: 200
## Columns: 39
## $ cds      <chr> "19647336097927", "19647336016018", "19648816021505", "196473~
## $ stype    <fct> E, E, E, E, E, E, E, E, E, E, M, M, H, M, H, E, E, M, M, E, E~
## $ name     <chr> "Open Magnet: Ce", "Belvedere Eleme", "Altadena Elemen", "Sot~
## $ sname    <chr> "Open Magnet: Center for Individual (Char", "Belvedere Elemen~
## $ snum     <dbl> 2077, 1622, 2236, 1921, 6140, 6077, 6071, 904, 4637, 4311, 41~
## $ dname    <chr> "Los Angeles Unified", "Los Angeles Unified", "Pasadena Unifi~
## $ dnum     <int> 401, 401, 541, 401, 460, 689, 689, 41, 702, 135, 590, 767, 25~
## $ cname    <chr> "Los Angeles", "Los Angeles", "Los Angeles", "Los Angeles", "~
## $ cnum     <int> 18, 18, 18, 18, 55, 55, 55, 14, 36, 36, 35, 32, 9, 1, 32, 18,~
## $ flag     <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ pcttest  <int> 99, 100, 99, 100, 100, 100, 99, 98, 100, 100, 99, 99, 93, 95,~
## $ api00    <int> 840, 516, 531, 501, 720, 805, 778, 731, 592, 669, 496, 505, 4~
## $ api99    <int> 816, 476, 544, 457, 659, 780, 787, 731, 508, 658, 479, 499, 4~
## $ target   <int> NA, 16, 13, 17, 7, 1, 1, 3, 15, 7, 16, 15, 17, 20, 13, 18, 11~
## $ growth   <int> 24, 40, -13, 44, 61, 25, -9, 0, 84, 11, 17, 6, 7, 3, -10, 57,~
## $ sch.wide <fct> Yes, Yes, No, Yes, Yes, Yes, No, No, Yes, Yes, Yes, No, No, N~
## $ comp.imp <fct> No, Yes, No, Yes, Yes, Yes, No, No, Yes, No, No, No, No, No, ~
## $ both     <fct> No, Yes, No, Yes, Yes, Yes, No, No, Yes, No, No, No, No, No, ~
## $ awards   <fct> No, Yes, No, Yes, Yes, Yes, No, No, Yes, No, No, No, No, No, ~
## $ meals    <int> 33, 98, 64, 83, 26, 7, 9, 45, 75, 47, 69, 60, 66, 54, 35, 95,~
## $ ell      <int> 25, 77, 23, 63, 17, 0, 2, 2, 58, 23, 25, 10, 43, 26, 7, 66, 7~
## $ yr.rnd   <fct> No, Yes, No, No, No, No, No, No, Yes, No, No, No, No, No, No,~
## $ mobility <int> 11, 26, 17, 13, 31, 12, 10, 15, 23, 19, 26, 22, 16, 44, 18, 1~
## $ acs.k3   <int> 20, 19, 20, 17, 20, 19, 19, 19, 20, 18, NA, NA, NA, NA, NA, 1~
## $ acs.46   <int> 29, 28, 30, 30, 30, 29, 31, 31, 32, 29, 32, 32, NA, 32, NA, 3~
## $ acs.core <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 30, 32, 27, 29, 28, N~
## $ pct.resp <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 87, 67, 50, 70, 71, 2, 91, 0~
## $ not.hsg  <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 31, 49, 12, 20, 45, 9, 22, 0~
## $ hsg      <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 34, 20, 33, 20, 36, 64, 20, ~
## $ some.col <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 22, 15, 23, 31, 11, 18, 32, ~
## $ col.grad <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10, 12, 29, 23, 8, 9, 16, 10~
## $ grad.sch <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 4, 3, 6, 0, 0, 11, 0, 2, ~
## $ avg.ed   <dbl> 3.32, 1.67, 2.34, 1.86, 3.17, 3.64, 3.55, 3.10, 2.17, 2.82, 2~
## $ full     <int> 100, 57, 81, 64, 90, 95, 96, 93, 91, 96, 84, 65, 93, 55, 80, ~
## $ emer     <int> 0, 40, 26, 24, 7, 0, 0, 8, 14, 0, 18, 37, 17, 26, 19, 33, 3, ~
## $ enroll   <int> 276, 841, 441, 298, 354, 330, 385, 583, 763, 381, 1293, 1043,~
## $ api.stu  <int> 241, 631, 415, 288, 319, 315, 363, 510, 652, 322, 1035, 815, ~
## $ pw       <dbl> 44.21, 44.21, 44.21, 44.21, 44.21, 44.21, 44.21, 44.21, 44.21~
## $ fpc      <dbl> 4421, 4421, 4421, 4421, 4421, 4421, 4421, 4421, 4421, 4421, 1~
```

```
# Summarize strata sample sizes
apistrat %>%
  count(stype)
```

```
##   stype   n
## 1     E 100
## 2     H  50
## 3     M  50
```

```
# Specify the design
apistrat_design <- svydesign(data = apistrat, weights = ~pw, fpc = ~fpc, id = ~1, strata = ~stype)

# Look at the summary information stored in the design object
summary(apistrat_design)
```

```
## Stratified Independent Sampling design
## svydesign(data = apistrat, weights = ~pw, fpc = ~fpc, id = ~1,
##     strata = ~stype)
## Probabilities:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.02262 0.02262 0.03587 0.04014 0.05339 0.06623
## Stratum Sizes:
##               E  H  M
## obs         100 50 50
## design.PSU  100 50 50
## actual.PSU  100 50 50
## Population stratum sizes (PSUs):
##    E    H    M
## 4421  755 1018
## Data variables:
##  [1] "cds"      "stype"    "name"     "sname"    "snum"     "dname"
##  [7] "dnum"     "cname"    "cnum"     "flag"     "pcttest"  "api00"
## [13] "api99"    "target"   "growth"   "sch.wide" "comp.imp" "both"
## [19] "awards"   "meals"    "ell"      "yr.rnd"   "mobility" "acs.k3"
## [25] "acs.46"   "acs.core" "pct.resp" "not.hsg"  "hsg"      "some.col"
## [31] "col.grad" "grad.sch" "avg.ed"   "full"     "emer"     "enroll"
## [37] "api.stu"  "pw"       "fpc"
```

1.4 Cluster designs in R

Svydesign also supports clustered sampling designs.

```
# Glimpse the data
glimpse(apiclus2)
```

```
## Rows: 126
## Columns: 40
## $ cds      <chr> "31667796031017", "55751846054837", "41688746043517", "416887~
## $ stype    <fct> E, E, E, M, E, E, E, E, M, H, E, M, E, E, E, E, H, E, E, M, E~
## $ name     <chr> "Alta-Dutch Flat", "Tenaya Elementa", "Panorama Elemen", "Lip~
## $ sname    <chr> "Alta-Dutch Flat Elementary", "Tenaya Elementary", "Panorama ~
## $ snum     <dbl> 3269, 5979, 4958, 4957, 4956, 4915, 2548, 2550, 2549, 348, 34~
## $ dname    <chr> "Alta-Dutch Flat Elem", "Big Oak Flat-Grvlnd Unif", "Brisbane~
## $ dnum     <int> 15, 63, 83, 83, 83, 117, 132, 132, 132, 152, 152, 152, 173, 1~
## $ cname    <chr> "Placer", "Tuolumne", "San Mateo", "San Mateo", "San Mateo", ~
## $ cnum     <int> 30, 54, 40, 40, 40, 39, 19, 19, 19, 5, 5, 5, 36, 36, 36, 36, ~
## $ flag     <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ pcttest  <int> 100, 100, 98, 100, 98, 100, 100, 100, 100, 96, 98, 100, 100, ~
## $ api00    <int> 821, 773, 600, 740, 716, 811, 472, 520, 568, 591, 544, 612, 9~
## $ api99    <int> 785, 718, 632, 740, 711, 779, 432, 494, 589, 585, 554, 583, 9~
## $ target   <int> 1, 4, 8, 3, 4, 1, 18, 15, 11, 11, 12, 11, NA, NA, NA, NA, 18,~
## $ growth   <int> 36, 55, -32, 0, 5, 32, 40, 26, -21, 6, -10, 29, 14, 2, 30, -5~
## $ sch.wide <fct> Yes, Yes, No, No, Yes, Yes, Yes, Yes, No, No, No, Yes, Yes, Y~
## $ comp.imp <fct> Yes, Yes, No, No, Yes, Yes, Yes, Yes, No, No, No, Yes, Yes, Y~
## $ both     <fct> Yes, Yes, No, No, Yes, Yes, Yes, Yes, No, No, No, Yes, Yes, Y~
## $ awards   <fct> Yes, Yes, No, No, Yes, Yes, Yes, Yes, No, No, No, Yes, Yes, Y~
## $ meals    <int> 27, 43, 33, 11, 5, 25, 78, 76, 68, 42, 63, 54, 0, 4, 1, 6, 47~
## $ ell      <int> 0, 0, 5, 4, 2, 5, 38, 34, 34, 23, 42, 24, 3, 6, 2, 1, 37, 14,~
## $ yr.rnd   <fct> No, No, No, No, No, No, No, No, No, No, No, No, No, No, No, N~
## $ mobility <int> 14, 12, 9, 8, 6, 19, 13, 13, 15, 4, 15, 15, 24, 19, 14, 14, 7~
## $ acs.k3   <int> 17, 18, 19, NA, 18, 20, 19, 25, NA, NA, 20, NA, 19, 18, 19, 1~
## $ acs.46   <int> 20, 34, 29, 30, 28, 22, NA, 23, 24, NA, NA, 27, 27, 25, 27, 2~
## $ acs.core <int> NA, NA, NA, 24, NA, 31, NA, NA, 25, 21, NA, 18, NA, NA, NA, N~
## $ pct.resp <int> 89, 98, 79, 96, 98, 93, 100, 46, 91, 94, 93, 88, 90, 99, 0, 8~
## $ not.hsg  <int> 4, 8, 8, 5, 3, 5, 48, 30, 63, 20, 29, 27, 0, 1, 0, 1, 50, 24,~
## $ hsg      <int> 16, 33, 28, 27, 14, 9, 32, 27, 16, 18, 32, 25, 0, 7, 0, 5, 21~
## $ some.col <int> 53, 37, 30, 35, 22, 30, 15, 21, 13, 27, 26, 24, 4, 8, 0, 8, 1~
## $ col.grad <int> 21, 15, 32, 27, 58, 37, 4, 13, 6, 28, 7, 18, 51, 42, 0, 42, 1~
## $ grad.sch <int> 6, 7, 1, 6, 3, 19, 1, 9, 2, 7, 6, 7, 44, 41, 100, 45, 1, 6, 3~
## $ avg.ed   <dbl> 3.07, 2.79, 2.90, 3.03, 3.44, 3.56, 1.77, 2.42, 1.68, 2.84, 2~
## $ full     <int> 100, 100, 100, 82, 100, 94, 96, 86, 75, 100, 100, 97, 100, 10~
## $ emer     <int> 0, 0, 0, 18, 8, 6, 8, 24, 21, 4, 4, 3, 0, 4, 0, 4, 28, 18, 11~
## $ enroll   <int> 152, 312, 173, 201, 147, 234, 184, 512, 543, 332, 217, 520, 5~
## $ api.stu  <int> 120, 270, 151, 179, 136, 189, 158, 419, 423, 303, 182, 438, 4~
## $ pw       <dbl> 18.925, 18.925, 18.925, 18.925, 18.925, 18.925, 18.925, 18.92~
## $ fpc1     <dbl> 757, 757, 757, 757, 757, 757, 757, 757, 757, 757, 757, 757, 7~
## $ fpc2     <int> <array[26]>
```

```
# Specify the design
apiclus_design <- svydesign(id = ~dnum + snum, data = apiclus2, weights = ~pw, fpc = ~fpc1 + fpc2)

#Look at the summary information stored in the design object
summary(apiclus_design)
```

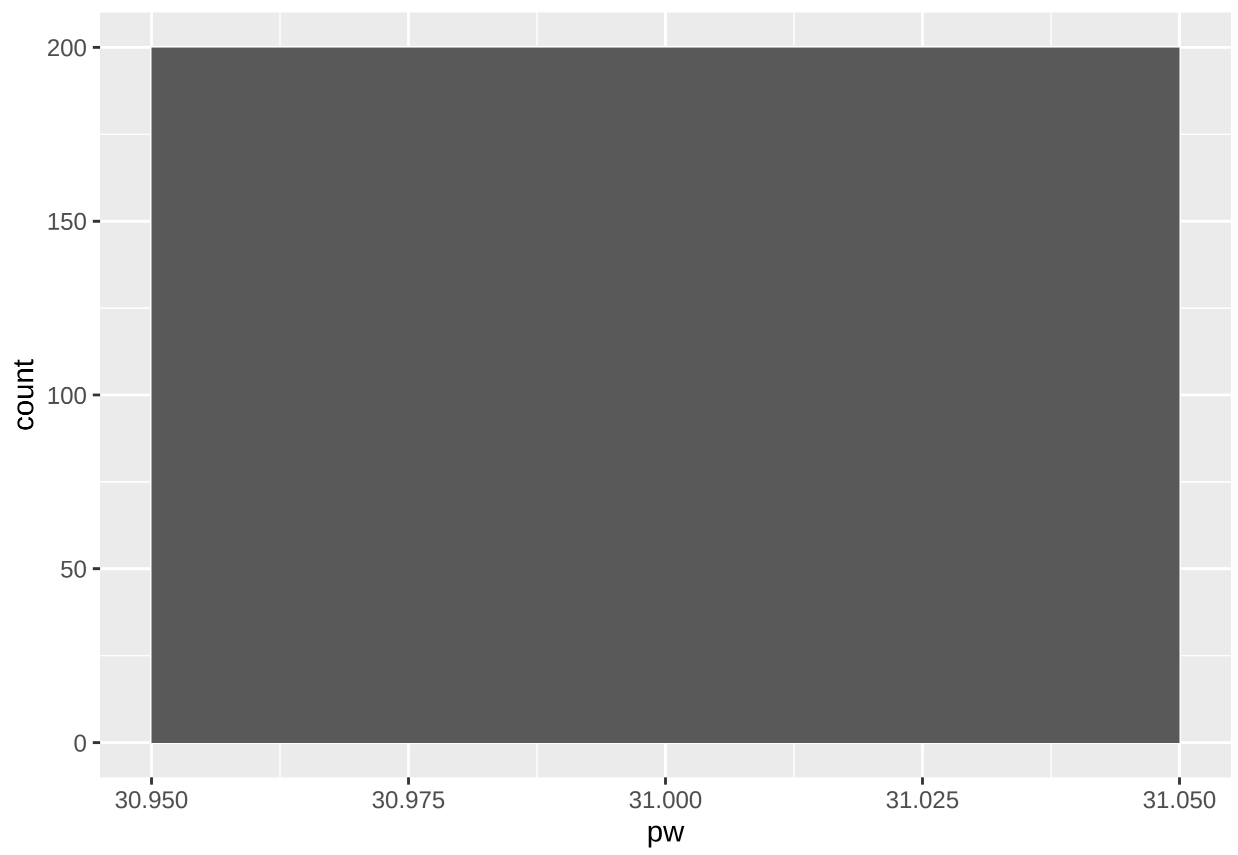
```
## 2 - level Cluster Sampling design
## With (40, 126) clusters.
## svydesign(id = ~dnum + snum, data = apiclus2, weights = ~pw,
##     fpc = ~fpc1 + fpc2)
## Probabilities:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.003669 0.037743 0.052840 0.042390 0.052840 0.052840
## Population size (PSUs): 757
## Data variables:
##  [1] "cds"      "stype"    "name"     "sname"    "snum"     "dname"
##  [7] "dnum"     "cname"    "cnum"     "flag"     "pcttest"  "api00"
## [13] "api99"    "target"   "growth"   "sch.wide" "comp.imp" "both"
## [19] "awards"   "meals"    "ell"      "yr.rnd"   "mobility" "acs.k3"
## [25] "acs.46"   "acs.core" "pct.resp" "not.hsg"  "hsg"      "some.col"
## [31] "col.grad" "grad.sch" "avg.ed"   "full"     "emer"     "enroll"
## [37] "api.stu"  "pw"       "fpc1"     "fpc2"
```

1.5 Comparing survey weights of different designs

In the code below we plot and compare histograms.

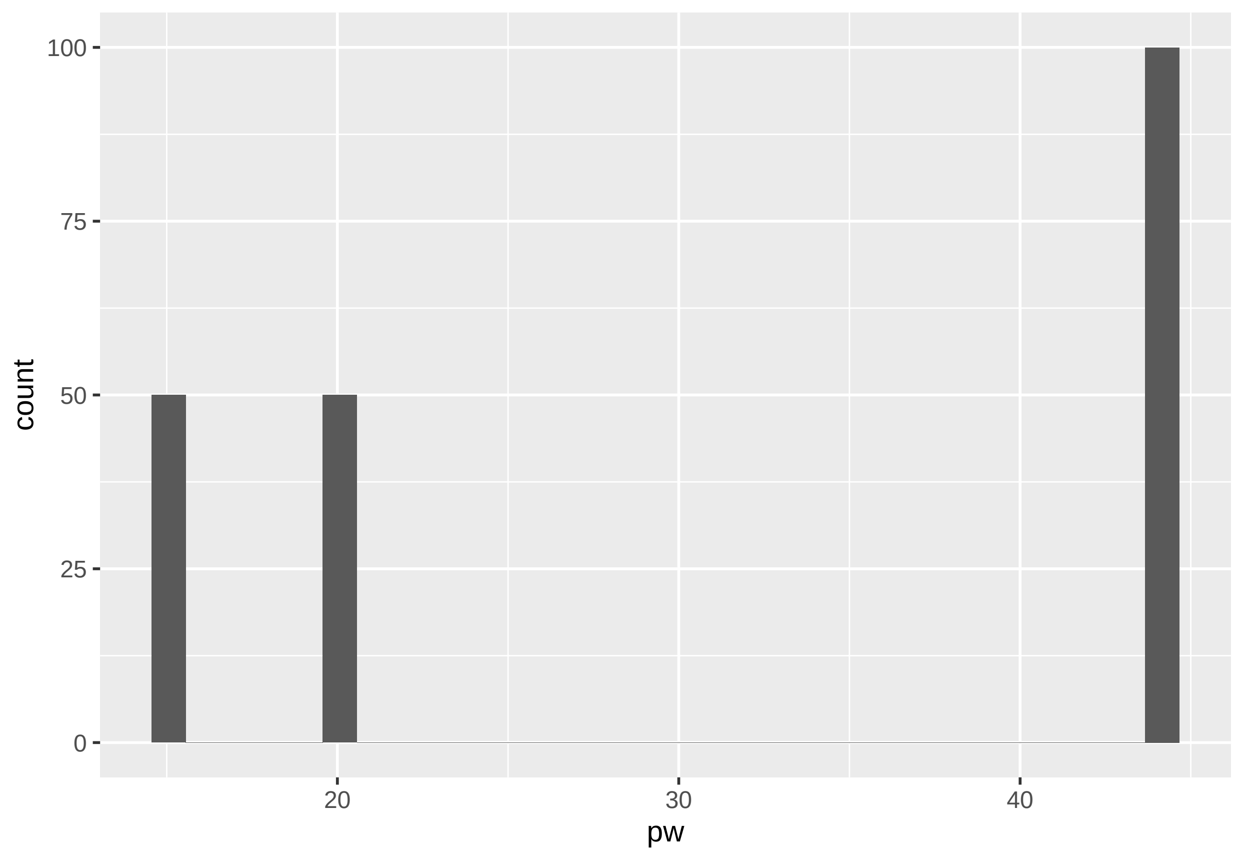
```
# Construct histogram of pw
ggplot(data = apisrs,
       mapping = aes(x = pw)) +
    geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



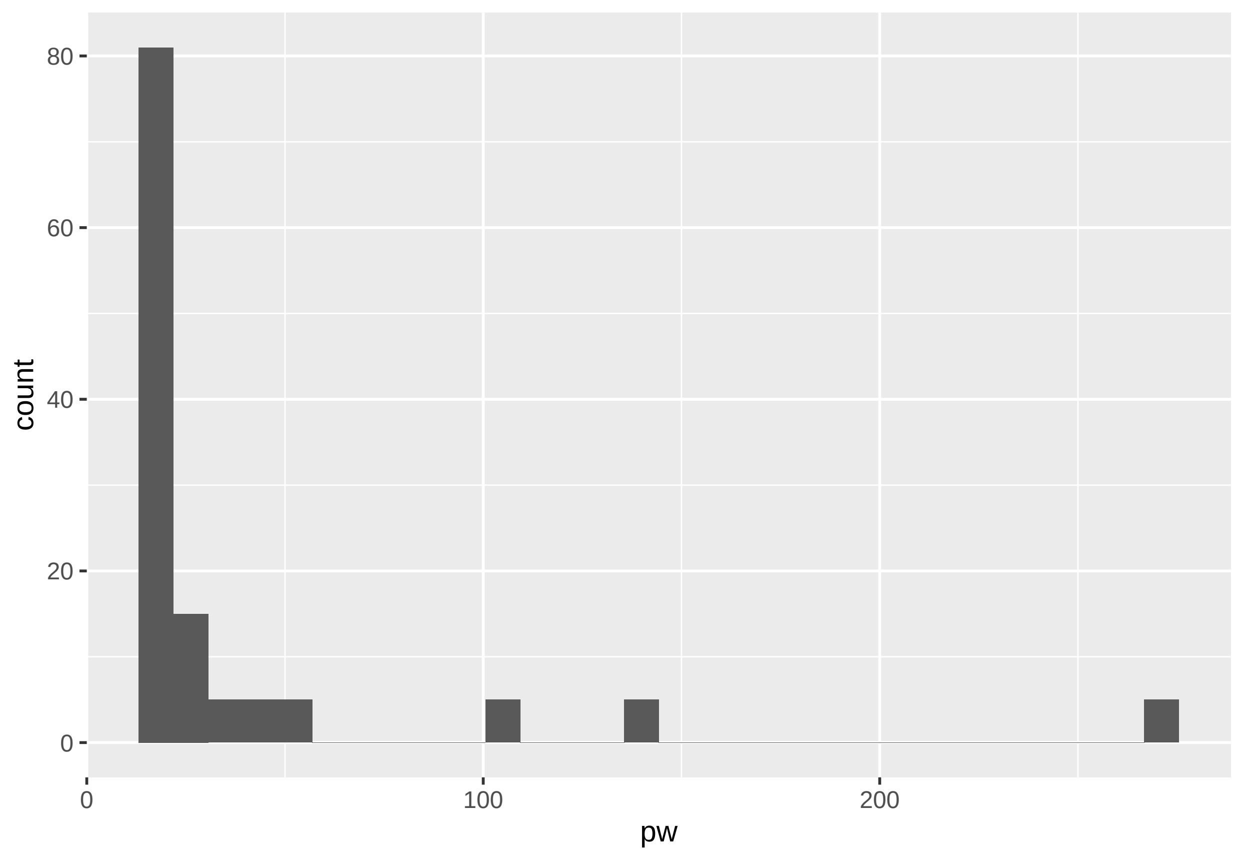
```r
# Construct histogram of pw
ggplot(data = apistrat,
       mapping = aes(x = pw)) +
    geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
# Construct histogram of pw
ggplot(data = apiclus2,
       mapping = aes(x = pw)) +
    geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



1.6 NHANES weights

Here we explore the survey weights of data from the NHANES package.

```
#Create table of average survey weights by race
tab_weights <- NHANESraw %>%
  group_by(Race1) %>%
  summarize(avg_wt = mean(WTMEC4YR))

#Print the table
tab_weights
```

1.7 Tying it all together!

We do further analyzing of the NHANES package.

```
# Specify the NHANES design
NHANES_design <- svydesign(data = NHANESraw, strata = ~SDMVSTRA, id = ~SDMVPSU, nest = TRUE, weights = 

# Print summary of design
summary(NHANES_design)

# Number of clusters
NHANESraw %>%
  summarize(n_clusters = n_distinct(SDMVSTRA, SDMVPSU))

# Sample sizes in clusters
```

```
NHANESraw %>%
  count(SDMVSTRA, SDMVPSU)
```

2 Exploring categorical data

2.1 Summarizing a categorical variable

In this exercise we create a frequency table.

```
# Specify the survey design
NHANESraw <- mutate(NHANESraw, WTMEC4YR = .5 * WTMEC2YR)
NHANES_design <- svydesign(data = NHANESraw, strata = ~SDMVSTRA, id = ~SDMVPSU, nest = TRUE, weights = 

# Determine the levels of Depressed
levels(NHANESraw$Depressed)

# Construct a frequency table of Depressed
tab_w <- svytable(~Depressed, design = NHANES_design)

# Determine class of tab_w
class(tab_w)

# Display tab_w
tab_w
```

2.2 Graphing a categorical variable

Here we compute proportions and then draw a bar plot.

```
# Add proportions to table
tab_w <- tab_w %>%
  as.data.frame() %>%
  mutate(Prop = Freq/sum(Freq))

# Create a barplot
ggplot(data = tab_w,
       mapping = aes(x = Depressed, y = Prop)) +
  geom_col()
```

2.3 Creating contingency tables

In the code below we create a contingency table with svytable and NHANES_design.

```
# Construct and display a frequency table
tab_D <- svytable(~Depressed,
            design = NHANES_design)
tab_D

# Construct and display a frequency table
tab_H <- svytable(~HealthGen,
            design = NHANES_design)
tab_H

# Construct and display a frequency table
```

```r
tab_DH <- svytable(~HealthGen+Depressed,
           design = NHANES_design)
tab_DH
```

2.4 Building segments bar graphs

Here we create a segmented bar graph of Prop_Depressed over HealthGen.

```r
# Create a segmented bar graph of the conditional proportions in tab_DH_cond
ggplot(data = tab_DH_cond,
       mapping = aes(x = HealthGen, y = Prop_Depressed, fill = Depressed)) +
  geom_col() +
  coord_flip()
```

2.5 Summarizing with svytotal()

In this exercise we learn how to use the svytotal and svymean functions.

```r
# Estimate the totals for combos of Depressed and HealthGen
tab_totals <- svytotal(x = ~interaction(Depressed, HealthGen),
                     design = NHANES_design,
                     na.rm = TRUE)

# Print table of totals
tab_totals

# Estimate the means for combos of Depressed and HealthGen
tab_means <- svymean(x = ~interaction(Depressed, HealthGen),
              design = NHANES_design,
              na.rm = TRUE)

# Print table of means
tab_means
```

2.6 Running a chi squared test

Here we run a chi squared test between Depressed and HealthGen.

```r
# Run a chi square test between Depressed and HealthGen
svychisq(~Depressed + HealthGen,
    design = NHANES_design,
    statistic = "Chisq")
```

2.7 Tying it all together!

This exercise is a repetition of everything we've learned in this chapter.

```r
# Construct a contingency table
tab <- svytable(~HomeOwn + Education, design=NHANES_design)
# Add conditional proportion of levels of HomeOwn for each educational level
tab_df <- as.data.frame(tab) %>%
  group_by(Education) %>%
  mutate(n_Education = sum(Freq), Prop_HomeOwn = Freq / n_Education) %>%
  ungroup()
# Create a segmented bar graph
```

```
ggplot(data = tab_df,
       mapping = aes(x = Education, y = Prop_HomeOwn, fill = HomeOwn)) +
  geom_col() +
  coord_flip()
# Run a chi square test
svychisq(~HomeOwn + Education,
    design = NHANES_design,
    statistic = "Chisq")
```

3 Exploring quantitative data

3.1 Survey statistics

Computing survey weighted means is easy, check the code below to see how to do it:

```
# Compute the survey-weighted mean
svymean(x = ~SleepHrsNight,
        design = NHANES_design,
        na.rm = TRUE)

# Compute the survey-weighted mean by Gender
svyby(formula = ~SleepHrsNight,
    by = ~Gender,
    design = NHANES_design,
    FUN = svymean,
    na.rm = TRUE,
    keep.names = FALSE)
```

3.2 Estimating quantiles

Here we calculate survey quantiles, once directly via svyquantile and also indirectly with svyby.

```
# Compute the survey-weighted quantiles
svyquantile(x = ~SleepHrsNight,
            design = NHANES_design,
            na.rm = TRUE,
            quantiles = c(0.01, 0.25, 0.5, 0.75, .99))

# Compute the survey-weighted quantiles by Gender
svyby(formula = ~SleepHrsNight,
      by = ~Gender,
      design = NHANES_design,
      FUN = svyquantile,
      na.rm = TRUE,
      quantiles = c(0.5),
      keep.rows = FALSE,
      keep.var = FALSE)
```

3.3 Bar plots of survey-weighted means

In this exercise we create a bar plot of survey-weighted means.

```
# Compute the survey-weighted mean by Gender
out <- svyby(formula = ~SleepHrsNight,
             by = ~Gender,
             design = NHANES_design,
             FUN = svymean,
             na.rm = TRUE,
             keep.names = FALSE)

# Construct a bar plot of average sleep by gender
ggplot(data = out, mapping = aes(x = Gender, y = SleepHrsNight)) +
  geom_col() +
  labs(y= "Average Nightly Sleep")
```

3.4 Bar plots with error

```
# Add lower and upper columns to out
out_col <- mutate(out,
                  lower = SleepHrsNight - 2*se,
                  upper = SleepHrsNight + 2*se)

# Construct a bar plot of average sleep by gender with error bars
ggplot(data = out_col,
       mapping = aes(x = Gender, y = SleepHrsNight,
                     ymin = lower, ymax = upper)) +
  geom_col(fill = "gold") +
  labs(y = "Average Nightly Sleep") +
  geom_errorbar(width = 0.7)
```

3.5 Survey-weighted histograms

In this exercise we experimented with the binwidth parameter of geom_histogram.

```
# Create a histogram with a set binwidth
ggplot(data = NHANESraw,
       mapping = aes(x = SleepHrsNight, weight = WTMEC4YR)) +
  geom_histogram(binwidth = 1,
                 color = "white") +
  labs(x = "Hours of Sleep")

# Create a histogram with a set binwidth
ggplot(data = NHANESraw,
       mapping = aes(x = SleepHrsNight, weight = WTMEC4YR)) +
  geom_histogram(binwidth = 0.5,
                 color = "white") +
  labs(x = "Hours of Sleep")

# Create a histogram with a set binwidth
ggplot(data = NHANESraw,
       mapping = aes(x = SleepHrsNight, weight = WTMEC4YR)) +
  geom_histogram(binwidth = 2,
                 color = "white") +
  labs(x = "Hours of Sleep")
```

3.6 Survey-weighted density plots

The code below creates density plots where the height represents probabilities.

```
# Density plot of sleep faceted by gender
NHANESraw %>%
    filter(!is.na(SleepHrsNight), !is.na(Gender)) %>%
    group_by(Gender) %>%
    mutate(WTMEC4YR_std = WTMEC4YR/sum(WTMEC4YR)) %>%
    ggplot(mapping = aes(x = SleepHrsNight, weight = WTMEC4YR_std)) +
        geom_density(bw = 0.6,  fill = "gold") +
        labs(x = "Hours of Sleep") +
        facet_wrap(~Gender, labeller = "label_both")
```

3.7 Survey-weighted t-test

A t-test answers the question if there's enough evidence for a given hypothesis. We compute such a t-test below.

```
# Run a survey-weighted t-test
svyttest(formula = SleepHrsNight ~ Gender,
       design = NHANES_design)
```

3.8 Tying it all together!

Repetition is the key to sustained knowledge.

```
# Find means of total cholesterol by whether or not active
out <- svyby(formula = ~TotChol,
           by = ~PhysActive,
           design = NHANES_design,
           FUN = svymean,
           na.rm = TRUE,
           keep.names = FALSE)

# Construct a bar plot of means of total cholesterol by whether or not active
ggplot(data = out,
       mapping = aes(x = PhysActive, y = TotChol)) +
  geom_col()

# Run t test for difference in means of total cholesterol by whether or not active
svyttest(formula = TotChol ~ PhysActive,
    design = NHANES_design)
```

4 Modeling quantitative data

4.1 Bubble plots

In this exercise we create a bubble plot with geom_point and the parameter size in aes().

```
# Create dataset with only 20 year olds
NHANES20 <- filter(NHANESraw,
                Age == 20)
```

4.2 Survey-weighted scatter plots

We create more scatter plots with different color and alpha points to distinguish another dimension.

```
# Construct scatter plot
ggplot(data = NHANES20,
       mapping = aes(x = Height, y = Weight)) +
    geom_point(alpha = 0.3) +
    guides(size = FALSE)

# Construct a scatter plot
ggplot(data = NHANES20,
       mapping = aes(x = Height, y = Weight, color = WTMEC4YR)) +
    geom_point() +
    guides(color = FALSE)

# Construct a scatter plot
ggplot(data = NHANES20,
       mapping = aes(x = Height, y = Weight, alpha=WTMEC4YR)) +
    geom_point() +
    guides(alpha = FALSE)
```

4.3 Use of color in scatter plots

We combine what we've learned before to construct alpha and color varying plots.

```
# Add gender to plot
ggplot(data = NHANES20,
       mapping = aes(x = Height, y = Weight, size=WTMEC4YR, color=Gender)) +
    geom_point(alpha=0.3) +
    guides(size = FALSE)

# Add gender to plot
ggplot(data = NHANES20,
       mapping = aes(x = Height, y = Weight, alpha = WTMEC4YR, color = Gender)) +
    geom_point(alpha=0.3) +
    guides(alpha = FALSE)
```

4.4 Line of best fit

In this exercise we try to regress data and find a model for our data (linear, quadratic, cubic regression).

```
# Bubble plot with linear of best fit
ggplot(data = NHANESraw, mapping = aes(x = Height, y = Weight, size = WTMEC4YR)) +
  geom_point(alpha = 0.1) +
  guides(size = FALSE) +
  geom_smooth(method = "lm", se = FALSE, mapping = aes(weight = WTMEC4YR))

# Add quadratic curve and cubic curve
ggplot(data = NHANESraw, mapping = aes(x = Height, y = Weight, size = WTMEC4YR)) +
  geom_point(alpha = 0.1) +
  guides(size = FALSE) +
  geom_smooth(method = "lm", se = FALSE, mapping = aes(weight = WTMEC4YR)) +
  geom_smooth(method = "lm", se = FALSE, mapping = aes(weight = WTMEC4YR), formula = y ~ poly(x, 2), co
  geom_smooth(method = "lm", se = FALSE, mapping = aes(weight = WTMEC4YR), formula = y ~ poly(x, 3), co
```

4.5 Trend lines

Here we're adding linear trend lines based on survey weights.

```
# Add non-survey-weighted trend lines to bubble plot
ggplot(data = NHANES20, mapping = aes(x = Height, y = Weight, size = WTMEC4YR, color = Gender)) +
  geom_point(alpha = 0.1) +
  guides(size = FALSE) +
  geom_smooth(method = "lm", se = FALSE, linetype = 2)

# Add survey-weighted trend lines
ggplot(data = NHANES20, mapping = aes(x = Height, y = Weight, size = WTMEC4YR, color = Gender)) +
  geom_point(alpha = 0.1) +
  guides(size = FALSE) +
  geom_smooth(method = "lm", se = FALSE, linetype = 2) +
  geom_smooth(method = "lm", se = FALSE, mapping = aes(weight=WTMEC4YR))
```

4.6 Regression model

We're building a linear regression model again, this time explicitly by creating a model object.

```
# Subset survey design object to only include 20 year olds
NHANES20_design <- subset(NHANES_design, Age == 20)

# Build a linear regression model
mod <- svyglm(Weight ~ Height, design = NHANES20_design)

# Print summary of the model
summary(mod)
```

4.7 Multiple linear regression

Multiple linear regression can be done with + and * operators.

```
# Build a linear regression model same slope
mod1 <- svyglm(Weight ~ Height + Gender, design = NHANES20_design)

# Print summary of the same slope model
summary(mod1)

mod2 <- svyglm(Weight ~ Height * Gender, design = NHANES20_design)

# Print summary of the same slope model
summary(mod2)
```

4.8 Tying it all together

Here we revisit everything we've learned in this chapter.

```
# Plot BPDiaAve and BPSysAve by Diabetes and include trend lines
drop_na(NHANESraw, Diabetes) %>%
ggplot(mapping = aes(x = BPDiaAve, y = BPSysAve, size = WTMEC4YR, color = Diabetes)) +
    geom_point(alpha = 0.2) +
    guides(size = FALSE) +
    geom_smooth(method = "lm", se = FALSE, mapping = aes(weight = WTMEC4YR))
```

```r
# Build simple linear regression model
mod1 <- svyglm(BPSysAve ~ BPDiaAve, design = NHANES_design)

# Build model with different slopes
mod2 <- svyglm(BPSysAve ~ BPDiaAve * Diabetes, design = NHANES_design)

# Summarize models
summarize(mod1)
summarize(mod2)
```