Dataset: Powerlifting

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.1.1

## Warning: package 'ggplot2' was built under R version 4.1.1

## Warning: package 'tibble' was built under R version 4.1.1

## Warning: package 'tidyr' was built under R version 4.1.1

## Warning: package 'readr' was built under R version 4.1.1

## Warning: package 'purrr' was built under R version 4.1.1

## Warning: package 'dplyr' was built under R version 4.1.1

## Warning: package 'stringr' was built under R version 4.1.1

## Warning: package 'forcats' was built under R version 4.1.1
```

```r
lift <- read_csv("https://uofi.box.com/shared/static/72tsz9guup6p31wc50zjw7y3lkvutqr5.csv")
```

Instructions:

- keep the following variables: Name, Sex, Event, Equipment, Age, AgeClass, BodyweightKg, WeightClassKg, Best3SquatKg, Best3BenchKg, Best3DeadliftKg, TotalKg, Wilks, IPFPoints
- filter to keep female powerlifters in AgeClass 24-34 with SBD as the Event
- remove NA values
- rename the data object as **power**

```r
clean1 <- select(lift,Name,Sex,Event,Equipment,Age,AgeClass,BodyweightKg,WeightClassKg,Best3SquatKg,Bes
clean2 <- filter(clean1,AgeClass=="24-34",Sex=="F",Event=="SBD")
miss <- unique(c(which(is.na(clean2$Name)),which(is.na(clean2$Sex)),which(is.na(clean2$Event)),which(is
power <- clean2[-miss,]
summary(power)
```

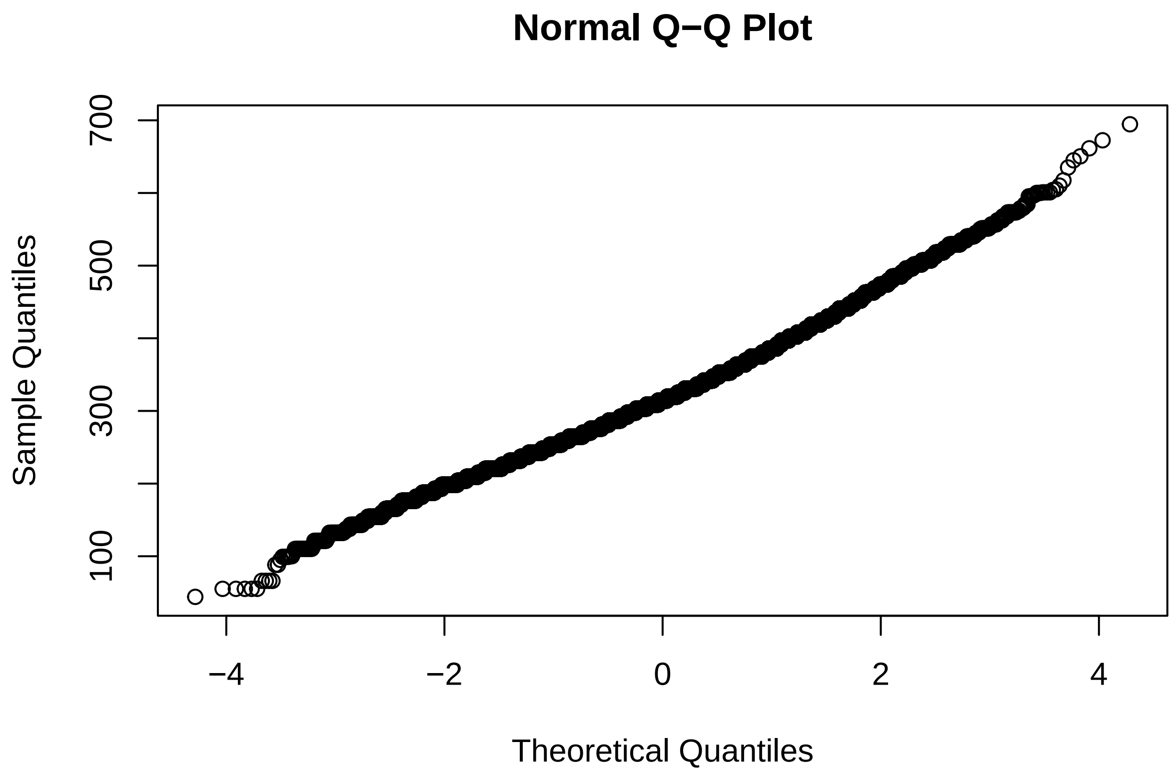
```
##      Name               Sex               Event            Equipment
##  Length:54589       Length:54589       Length:54589       Length:54589
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##       Age          AgeClass          BodyweightKg    WeightClassKg
##  Min.   :23.50   Length:54589       Min.   : 40.00   Length:54589
```

```
##  1st Qu.:25.50   Class :character   1st Qu.: 57.10   Class :character
##  Median :28.00   Mode  :character   Median : 66.04   Mode  :character
##  Mean   :28.27                      Mean   : 68.94
##  3rd Qu.:31.00                      3rd Qu.: 75.00
##  Max.   :34.50                      Max.   :184.20
##   Best3SquatKg    Best3BenchKg    Best3DeadliftKg    TotalKg
##  Min.   : 15.9   Min.   : 20.00   Min.   : 20.0   Min.   : 90.7
##  1st Qu.:100.0   1st Qu.: 55.00   1st Qu.:122.5   1st Qu.:282.5
##  Median :120.0   Median : 67.50   Median :142.5   Median :330.0
##  Mean   :126.5   Mean   : 71.47   Mean   :144.6   Mean   :342.6
##  3rd Qu.:145.0   3rd Qu.: 82.50   3rd Qu.:162.5   3rd Qu.:390.0
##  Max.   :387.5   Max.   :242.67   Max.   :315.0   Max.   :930.0
##      Wilks          IPFPoints
##  Min.   : 91.51   Min.   : 144.3
##  1st Qu.:295.99   1st Qu.: 470.2
##  Median :344.00   Median : 537.3
##  Mean   :355.54   Mean   : 542.4
##  3rd Qu.:403.16   3rd Qu.: 608.7
##  Max.   :776.17   Max.   :1145.9
```

```r
#convert bw,best squat/bench/deadlift, totalkg to pounds (lb) in new col
powerlbs <- mutate(power,bodyweightlb=(BodyweightKg*2.205),best3squatlb=(Best3SquatKg*2.205),best3benchl
powerlbs <- select(powerlbs,-c(BodyweightKg,Best3SquatKg,Best3BenchKg,Best3DeadliftKg))
```

```r
#multicollinearity, outliers
powerlbs <- select(powerlbs,-c(totallb))
powerlbs <- select(powerlbs,-c(TotalKg))
```

```r
qqnorm(powerlbs$best3deadliftlb)
```



(The following can be heavily simplified; I'll leave my original complex work as it is here.)

```
#3 sigma rule: Wilks
mn_wilks <- mean(powerlbs$Wilks)
sg_wilks <- sd(powerlbs$Wilks)
tsr_wilks <- which(abs(powerlbs$Wilks-mn_wilks) > (3*sg_wilks))
length(tsr_wilks)
```

```
## [1] 399
```

```
powerlbs$Wilks[tsr_wilks]
```

```
##   [1] 633.24 610.85 621.02 650.87 660.18 632.07 680.19 627.58 629.46 626.26
##  [11] 722.56 756.76 614.27 755.41 640.23 680.52 677.12 776.17 742.66 764.83
##  [21] 633.05 627.68 642.99 633.46 635.07 725.31 630.90 624.12 733.88 610.98
##  [31] 613.28 653.54 632.01 705.09 760.91 650.95 685.54 690.91 660.18 655.64
##  [41] 654.92 632.79  92.29 634.13 668.16 758.95 640.29 613.00  91.51 676.56
##  [51] 621.30 635.50 614.30 622.86 618.07 642.70 637.40 615.28 637.97 614.71
##  [61] 621.77 620.09 615.76 640.43 644.94 625.80 617.96 624.07 624.60 651.95
##  [71] 611.37 619.99 612.89 663.73 632.70 628.29 626.54 720.67 698.11 622.98
##  [81] 627.72 620.12 666.43 666.63 666.11 651.56 642.78 642.17 620.52 630.29
##  [91] 634.84 763.55 668.26 618.55 611.43 715.94 649.73 625.43 634.16 616.71
## [101] 623.65 630.81 654.32 648.44 613.02 630.88 618.80 629.02 644.64 644.68
## [111] 644.02 617.89 647.01 621.17 615.72 624.84 618.58 646.89 618.89 638.42
## [121] 631.41 629.23 640.44 657.08 621.90 658.42 647.13 622.03 623.29 623.19
```

```
## [131] 632.07 623.45 665.33 653.93 645.54 638.79 637.03 667.29 654.00 629.10
## [141] 631.25 652.80 654.27 623.16 633.82 628.47 617.56 632.47 629.26 645.55
## [151] 635.16 642.44 656.67 642.00 649.96 631.05 626.54 680.41 635.54 650.44
## [161] 659.07 645.79 622.78 612.26 634.20 635.16 641.46 677.82 612.95 618.09
## [171] 629.48 621.98 612.98 645.61 640.55 633.31 653.36 638.79 616.55 654.22
## [181] 653.88 625.06 654.91 625.79 632.76 661.08 667.45 634.39 613.26 628.30
## [191] 654.65 610.72 650.96 625.23 611.21 626.54 636.72 622.58 673.07 642.57
## [201] 635.34 614.52 616.55 617.09 635.49 662.88 645.33 620.23 686.34 619.70
## [211] 637.38 618.62 665.54 623.59 641.82 624.56 611.80 651.85 630.80 618.35
## [221] 635.96 642.86 635.59 634.51 627.11 620.24 638.49 613.97 641.11 613.28
## [231] 624.34 611.48 626.50 667.72 645.42 643.86 612.12 657.43 634.58 623.56
## [241] 623.58 673.90 615.79 636.38 618.06 632.29 621.39 615.00 644.61 658.39
## [251] 663.39 639.98 611.73 642.13 625.15 623.59 656.45 649.09 616.79 618.43
## [261] 629.60 629.53 619.10 681.99 628.53 614.42 618.64 636.73 611.94 637.86
## [271] 680.88 651.64 618.64 613.63 634.57 675.03 616.28 639.66 636.99 623.64
## [281] 612.58 612.27 614.60 614.88 636.66 610.98 664.38 616.61 667.05 619.85
## [291] 612.10 613.32 625.12 619.59 619.45 621.46 626.09 664.74 629.25 619.94
## [301] 629.43 622.29 630.76 631.42 614.87 620.31 622.79 633.99 627.40 612.05
## [311] 618.43 633.52 620.37 612.42 620.45 626.59 650.60 663.66 643.33 630.73
## [321] 686.51 624.40 636.80 615.19 643.98 621.60 629.91 625.57 614.48 668.28
## [331] 628.92 613.58 612.18 630.81 613.98 610.86 639.28 616.25 611.96 635.50
## [341] 618.13 627.70 634.13 621.46 635.45 611.16 626.13 740.67 685.84 643.47
## [351] 620.49 701.74 638.91 747.30 650.92 658.30 753.22 649.90 615.60 643.39
## [361] 638.22 674.63 637.05 614.91 724.23 734.21 710.17 617.90 739.51 734.21
## [371] 647.64 697.02 621.35 614.84 658.24 617.36 625.03 614.97 644.43 629.76
## [381] 622.27 621.32 626.33 615.50 619.55 622.01 620.79 621.15 765.33 679.30
## [391] 660.67 683.65 616.54 614.78 612.98 646.16 628.03 628.03 624.38
```

```r
# box plot rule: Wilks
q1<-as.vector(quantile(powerlbs$Wilks,1/4))
q3<-as.vector(quantile(powerlbs$Wilks,3/4))
iqr <- as.vector(q3-q1)
lwr_Wilks <- which(powerlbs$Wilks < q1-1.5*iqr)
upr_Wilks <- which(powerlbs$Wilks > q3+1.5*iqr)
length(lwr_Wilks);length(upr_Wilks)
```

```
## [1] 15
```

```
## [1] 1195
```

```r
# hampel identifier: Wilks
md_Wilks<-median(powerlbs$Wilks)
sg2_Wilks<-1.4826*(median(abs(powerlbs$Wilks-md_Wilks)))
hi_Wilks <- which(abs(powerlbs$Wilks-md_Wilks) > 3*sg2_Wilks)
length(hi_Wilks)
```

```
## [1] 892
```

```r
#3 sigma rule: IPFPoints
mn_ipf <- mean(powerlbs$IPFPoints)
sg_ipf <- sd(powerlbs$IPFPoints)
tsr_ipf <- which(abs(powerlbs$IPFPoints-mn_ipf) > (3*sg_ipf))
length(tsr_ipf)
```

```
## [1] 277
```

```r
# box plot rule: IPFPoints
q1<-as.vector(quantile(powerlbs$IPFPoints,1/4))
q3<-as.vector(quantile(powerlbs$IPFPoints,3/4))
iqr <- as.vector(q3-q1)
lwr_ipf <- which(powerlbs$IPFPoints < q1-1.5*iqr)
upr_ipf <- which(powerlbs$IPFPoints > q3+1.5*iqr)
length(lwr_ipf);length(upr_ipf)
```

```
## [1] 80
```

```
## [1] 577
```

```r
# hampel identifier: IPFPoints
md_ipf <- median(powerlbs$IPFPoints)
sg2_ipf <- 1.4826*(median(abs(powerlbs$IPFPoints-md_ipf)))
hi_ipf <- which(abs(powerlbs$IPFPoints-md_ipf) > 3*sg2_ipf)
length(hi_ipf)
```

```
## [1] 365
```

bodyweightlb

```r
#3 sigma rule: bodyweightlb
mn_wt <- mean(powerlbs$bodyweightlb)
sg_wt <- sd(powerlbs$bodyweightlb)
tsr_wt <- which(abs(powerlbs$bodyweightlb-mn_wt) > (3*sg_wt))
length(tsr_wt)
```

```
## [1] 1049
```

```r
# box plot rule: bodyweightlb
q1<-as.vector(quantile(powerlbs$bodyweightlb,1/4))
q3<-as.vector(quantile(powerlbs$bodyweightlb,3/4))
iqr <- as.vector(q3-q1)
lwr_wt <- which(powerlbs$bodyweightlb < q1-1.5*iqr)
upr_wt <- which(powerlbs$bodyweightlb > q3+1.5*iqr)
length(lwr_wt);length(upr_wt)
```

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

```
## [1] 2792
```

```r
# hampel identifier: bodyweightlb
md_wt <- median(powerlbs$bodyweightlb)
sg2_wt <- 1.4826*(median(abs(powerlbs$bodyweightlb-md_wt)))
hi_wt <- which(abs(powerlbs$bodyweightlb-md_wt) > 3*sg2_wt)
length(hi_wt)
```

```
## [1] 2249
```

best3squatlb

```r
#3 sigma rule: best3squatlb
mn_sqt <- mean(powerlbs$best3squatlb)
sg_sqt <- sd(powerlbs$best3squatlb)
tsr_sqt <- which(abs(powerlbs$best3squatlb-mn_sqt) > (3*sg_sqt))
length(tsr_sqt)
```

```
## [1] 613
```

```r
# box plot rule: best3squatlb
q1<-as.vector(quantile(powerlbs$best3squatlb,1/4))
q3<-as.vector(quantile(powerlbs$best3squatlb,3/4))
iqr <- as.vector(q3-q1)
lwr_sqt <- which(powerlbs$best3squatlb < q1-1.5*iqr)
upr_sqt <- which(powerlbs$best3squatlb > q3+1.5*iqr)
length(lwr_sqt);length(upr_sqt)
```

```
## [1] 26
```

```
## [1] 1720
```

```r
# hampel identifier: best3squatlb
md_sqt <- median(powerlbs$best3squatlb)
sg2_sqt <- 1.4826*(median(abs(powerlbs$best3squatlb-md_sqt)))
hi_sqt <- which(abs(powerlbs$best3squatlb-md_sqt) > 3*sg2_sqt)
length(hi_sqt)
```

```
## [1] 1296
```

best3benchlb

```r
#3 sigma rule: best3benchlb
mn_b <- mean(powerlbs$best3benchlb)
sg_b <- sd(powerlbs$best3benchlb)
tsr_b <- which(abs(powerlbs$best3benchlb-mn_b) > (3*sg_b))
length(tsr_b)
```

```
## [1] 906
```

```r
# box plot rule: best3benchlb
q1<-as.vector(quantile(powerlbs$best3benchlb,1/4))
q3<-as.vector(quantile(powerlbs$best3benchlb,3/4))
iqr <- as.vector(q3-q1)
lwr_b <- which(powerlbs$best3benchlb < q1-1.5*iqr)
upr_b <- which(powerlbs$best3benchlb > q3+1.5*iqr)
length(lwr_b);length(upr_b)
```

```
## [1] 0

## [1] 2092
```

```
# hampel identifier: best3benchlb
md_b <- median(powerlbs$best3benchlb)
sg2_b <- 1.4826*(median(abs(powerlbs$best3benchlb-md_b)))
hi_b <- which(abs(powerlbs$best3benchlb-md_b) > 3*sg2_b)
length(hi_b)
```

```
## [1] 2092
```

best3deadliftlb

```
#3 sigma rule: best3deadliftlb
mn_d <- mean(powerlbs$best3deadliftlb)
sg_d <- sd(powerlbs$best3deadliftlb)
tsr_d <- which(abs(powerlbs$best3deadliftlb-mn_d) > (3*sg_d))
length(tsr_d)
```

```
## [1] 279
```

```
# box plot rule: best3deadliftlb
q1<-as.vector(quantile(powerlbs$best3deadliftlb,1/4))
q3<-as.vector(quantile(powerlbs$best3deadliftlb,3/4))
iqr <- as.vector(q3-q1)
lwr_d <- which(powerlbs$best3deadliftlb < q1-1.5*iqr)
upr_d <- which(powerlbs$best3deadliftlb > q3+1.5*iqr)
length(lwr_d);length(upr_d)
```

```
## [1] 113

## [1] 775
```

```
# hampel identifier: best3deadliftlb
md_d <- median(powerlbs$best3deadliftlb)
sg2_d <- 1.4826*(median(abs(powerlbs$best3deadliftlb-md_d)))
hi_d <- which(abs(powerlbs$best3deadliftlb-md_d) > 3*sg2_d)
length(hi_d)
```

```
## [1] 412
```

#check for common outliers amongst all 3 methods

```
Reduce(intersect,list(tsr_wilks,lwr_Wilks,upr_Wilks,hi_Wilks))
```

```
## integer(0)
```

```r
Reduce(intersect,list(tsr_ipf,lwr_ipf,upr_ipf,hi_ipf))
```

```
## integer(0)
```

```r
Reduce(intersect,list(tsr_wt,upr_wt,hi_wt))
```

```
##   [1]    15    24    45    50    51   103   122   125   130   153   168   169
##  [13]   209   210   223   264   267   309   363   364   365   367   421   471
##  [25]   590   596   646   647   693   710   739   742   743   765   768   781
##  [37]   872   892   945   951   968   969  1010  1067  1095  1096  1108  1124
##  [49]  1153  1180  1189  3348  3379  3451  3473  3475  3562  3614  3822  3831
##  [61]  3905  3944  3972  3973  3992  4028  4062  4099  4103  4131  4163  4185
##  [73]  4231  4279  4322  4463  4470  4519  4603  4629  4692  4696  4715  4727
##  [85]  4779  4782  4819  4935  4997  5258  5403  5468  5585  5600  5748  5775
##  [97]  5949  5957  5993  6131  6162  6228  6319  6320  6565  6570  6571  6580
## [109]  6622  6870  6948  6969  7028  7983  7997  8013  8040  8109  8129  8188
## [121]  8235  8307  8368  8369  8401  8475  8476  8782  8951  8957  8966  9031
## [133]  9085  9163  9175  9225  9244  9388  9915  9933  9949 10129 10132 10133
## [145] 10223 10289 10336 10386 10387 10402 10437 10463 10472 10523 10541 10544
## [157] 10621 10668 10678 10699 10705 10720 10722 10761 10762 10777 10780 10789
## [169] 10804 10858 10870 10886 10910 10949 11005 11006 11184 11194 11211 11241
## [181] 11252 11277 11306 11316 11344 11425 11434 11443 11449 11469 11487 11572
## [193] 11594 11627 11628 11643 11650 11667 11676 11682 11690 11691 11708 11746
## [205] 11768 11912 11923 11929 12061 12080 12103 12106 12374 12455 12510 12582
## [217] 12599 12702 12781 12819 12855 12878 12902 13093 13094 13096 13099 13100
## [229] 13190 13317 13328 13417 13424 13462 13468 13482 13491 13532 13537 13538
## [241] 13584 13601 13734 13735 13740 13768 13769 13771 13772 13773 13808 13849
## [253] 13874 14037 14083 14084 14113 14184 14228 14350 14355 14454 14607 14631
## [265] 14665 14834 14835 14836 14837 14843 14845 14855 14857 14858 14859 14900
## [277] 14929 14934 14983 15032 15069 15095 15118 15193 15208 15209 15230 15236
## [289] 15325 15389 15491 15544 15623 15658 15677 15907 16012 16254 16274 16300
## [301] 16366 16382 16531 16607 16609 16655 16701 16874 16876 16979 16994 16997
## [313] 17040 17045 17072 17153 17156 17251 17259 17274 17394 17421 17422 17512
## [325] 17579 17660 17711 17728 17762 17846 17862 17886 17908 17909 18010 18044
## [337] 18232 18233 18234 18237 18239 18276 18320 18344 18394 18485 18615 18629
## [349] 18679 18713 19031 19032 19072 19106 19137 19143 19145 19179 19201 19228
## [361] 19229 19248 19255 19256 19357 19467 19480 19522 19535 19573 19632 19671
## [373] 19674 19717 19853 19880 19891 19981 19982 19992 20095 20117 20158 20179
## [385] 20233 20254 20365 20587 20588 20590 20633 20637 20694 20695 20785 20821
## [397] 20844 20928 20982 21002 21036 21212 21213 21217 21260 21276 21307 21465
## [409] 21466 21472 21540 21555 21594 21688 21707 21749 21839 21840 21881 21925
## [421] 21973 22066 22172 22379 22380 22381 22382 22387 22392 22395 22400 22401
## [433] 22468 22552 22607 22647 22648 22688 22807 22816 23045 23082 23091 23131
## [445] 23155 23192 23193 23194 23325 23365 23434 23517 23549 23593 23613 23623
## [457] 23659 23787 23820 23992 24055 24131 24233 24392 24543 24544 24545 24627
## [469] 24770 24828 24939 24945 25038 25054 25144 25155 25156 25164 25219 25253
## [481] 25346 25411 25416 25588 25621 25695 25698 25699 25700 25785 25821 25848
## [493] 25849 25938 25948 25990 26004 26093 26300 26394 26736 26773 26800 27272
## [505] 27312 27313 27335 27366 27367 27397 27415 27416 27417 27440 27442 27605
## [517] 27705 27730 27762 27828 27856 27884 28092 28093 28154 28262 28291 28392
## [529] 28417 28451 28617 28705 28708 28727 28889 28891 28977 29048 29074 29189
## [541] 29278 29285 29315 29317 29327 29348 29356 29397 29398 29399 29420 29432
```

```
##  [553] 29516 29565 29613 29614 29632 29680 29700 29746 29751 29783 29798 29806
##  [565] 29849 29943 29947 30105 30108 30112 30125 30135 30136 30153 30272 30370
##  [577] 30398 30534 30614 30719 30727 30848 30856 30857 30903 30946 30973 30992
##  [589] 31056 31147 31148 31153 31164 31175 31202 31227 31379 31396 31416 31417
##  [601] 31429 31430 31451 31475 31545 31572 31636 31651 31720 31739 31743 31772
##  [613] 31944 31946 31947 32047 32121 32151 32156 32229 32402 32495 32554 32555
##  [625] 32557 32638 32639 32652 32735 32750 32769 32773 32845 32846 32850 32912
##  [637] 33009 33066 33084 33117 33125 33150 33224 33281 33360 33419 33443 33512
##  [649] 33517 33560 33601 33635 33643 33664 33769 33788 33807 33842 33895 33919
##  [661] 33966 33999 34007 34033 34078 34102 34115 34201 34221 34259 34335 34392
##  [673] 34415 34446 34461 34467 34501 34503 34504 34591 34618 34647 34826 34913
##  [685] 34928 34931 34978 34979 35161 35162 35163 35177 35206 35207 35291 35365
##  [697] 35398 35575 35576 35643 35651 35836 35868 35874 35947 35974 35975 35980
##  [709] 35989 36030 36032 36166 36206 36268 36426 36472 36484 36607 36623 36624
##  [721] 36641 36654 36685 36701 36823 36859 36892 36937 37008 37070 37071 37072
##  [733] 37494 37496 37526 37551 37572 37601 37607 37615 37655 37656 37658 37716
##  [745] 37749 37751 37799 37893 37932 37951 37956 37968 37992 38000 38015 38024
##  [757] 38035 38050 38054 38059 38061 38079 38126 38127 38128 38129 38130 38152
##  [769] 38193 38194 38195 38199 38206 38225 38246 38271 38274 38282 38284 38296
##  [781] 38410 38434 38437 38442 38452 38460 38472 38474 38508 38552 38553 38554
##  [793] 38556 38618 38623 38626 38661 38683 38704 38719 38746 38790 38791 38814
##  [805] 38827 38905 38917 38957 38958 39174 39175 39212 39261 39621 39685 39747
##  [817] 39794 39796 39856 39908 40172 40173 40265 40266 40370 40458 41016 41064
##  [829] 41201 41203 41329 41783 41794 41811 41817 41832 41850 42370 42694 42843
##  [841] 42863 42950 42967 43105 43188 43475 43486 43575 43621 43624 43663 43665
##  [853] 43760 43799 43800 43802 43845 43889 43925 43975 43976 43977 44037 44138
##  [865] 44139 44140 44142 44143 44253 44286 44287 44387 44388 44390 44391 44467
##  [877] 44512 44533 44666 44695 44743 44744 44782 44879 44881 44882 44883 44884
##  [889] 44927 44928 44975 44976 44977 44978 45086 45164 45286 45440 45555 45582
##  [901] 45598 45618 45620 45771 46551 46566 46587 46601 46701 47108 47375 47415
##  [913] 47429 47445 47448 47567 47604 47659 47660 47683 47685 47687 47720 47722
##  [925] 47723 47742 47756 47757 47866 47894 47915 47959 47964 48016 48137 48195
##  [937] 48197 48198 48548 48582 48617 48623 48631 48870 48907 49050 49131 49156
##  [949] 49209 49297 49328 49414 49457 49590 49609 49782 49810 49847 49853 49895
##  [961] 49919 49939 49970 50015 50191 50212 50235 50373 50381 50414 50462 50494
##  [973] 50500 50507 50510 50511 50519 50550 50568 50593 50614 50659 50739 50782
##  [985] 50798 50810 50827 50843 50894 51025 51096 51201 51202 51279 51365 51557
##  [997] 51665 51666 51736 51737 51756 51757 51766 51776 51802 51837 51851 51979
## [1009] 51983 51999 52013 52079 52162 52237 52238 52241 52291 53407 53425 53441
## [1021] 53455 53523 53525 53570 53603 53604 53611 53645 53665 53666 53711 53829
## [1033] 53956 53983 53984 53997 54079 54088 54136 54150 54169 54179 54199 54221
## [1045] 54388 54398 54415 54506 54510
```

```
Reduce(intersect,list(tsr_sqt,lwr_sqt,upr_sqt,hi_sqt))
```

```
## integer(0)
```

```
Reduce(intersect,list(tsr_b,upr_b,hi_b))
```

```
##   [1]  1437  1511  1552  1646  1750  1859  2126  2197  2268  2271  2272  2294
##  [13]  2303  2311  2316  2317  2330  2362  2523  2566  2615  2659  2707  2750
##  [25]  2751  2753  2858  3225  3226  3511  3740  4149  4152  4171  4172  4185
##  [37]  4186  4191  4193  4206  4289  4358  4463  4480  4481  4484  4514  4538
```

```
##  [49]  4591  4592  4593  4638  4691  4692  4711  4712  4713  4714  4782  4869
##  [61]  4962  4980  5011  5012  5013  5047  7603  7766  7997  8331  8842  8861
##  [73]  9190  9514  9829 10276 10351 10379 10385 10386 10387 10388 10488 10541
##  [85] 10574 10634 10699 10764 10778 10803 10840 10852 10853 11005 11006 11210
##  [97] 11211 11213 11214 11215 11217 11218 11219 11220 11221 11281 11414 11457
## [109] 11478 11497 11516 11517 11519 11576 11577 11631 11636 11649 11650 11668
## [121] 11701 11729 11758 11767 11807 11913 11914 11916 11917 12254 13093 13328
## [133] 13329 13734 13794 14594 15594 15898 15900 16607 18232 18813 19028 19031
## [145] 19208 19220 19223 19225 19228 19229 20587 20667 20689 20690 20692 20694
## [157] 20695 23320 23323 23325 23644 23645 23953 24128 25155 25164 25253 25343
## [169] 25346 25348 25356 25357 25370 25371 25377 25378 25379 25382 25396 25403
## [181] 25409 25434 25435 25443 25449 25478 25497 25522 25523 25531 25584 25585
## [193] 25588 25637 25638 25774 25781 25785 25804 25806 25811 25812 25813 25815
## [205] 25819 25820 25821 25833 25835 25841 25845 25846 25885 25890 25892 25894
## [217] 25896 25897 25927 25928 25930 25934 25935 25938 25946 25948 25988 25990
## [229] 25991 26004 26013 26034 26041 26045 26049 26050 26051 26068 26151 26159
## [241] 26161 26162 26165 26199 26203 26206 26207 26208 26211 26212 26263 26266
## [253] 26280 26281 26348 26381 26387 26390 26391 26394 26406 26436 26609 26616
## [265] 26617 26618 26620 26621 26692 26697 26698 26732 26736 26760 26764 26767
## [277] 26768 26954 27269 27272 27517 27521 29652 30424 30425 30660 31631 32577
## [289] 33069 33193 34815 35778 37148 37171 37214 37367 37395 37496 37551 37811
## [301] 37973 37975 38002 38004 38825 38869 38870 38917 39160 39161 39163 39164
## [313] 39169 39171 39174 39175 39193 39207 39208 39212 39238 39245 39252 39254
## [325] 39257 39260 39290 39293 39298 39299 39323 39396 39402 39403 39405 39408
## [337] 39409 39412 39413 39428 39432 39436 39438 39439 39443 39532 39578 39611
## [349] 39618 39621 39658 39664 39665 39676 39679 39684 39685 39708 39713 39717
## [361] 39718 39720 39722 39726 39727 39730 39777 39778 39784 39788 39789 39794
## [373] 39796 39803 39851 39852 39856 39857 39875 39902 39908 39942 39989 39993
## [385] 40001 40003 40031 40036 40057 40062 40070 40071 40075 40077 40078 40079
## [397] 40111 40164 40165 40172 40173 40213 40217 40219 40220 40223 40250 40251
## [409] 40255 40256 40260 40261 40263 40265 40266 40293 40298 40304 40305 40310
## [421] 40312 40313 40432 40442 40448 40449 40452 40453 40458 40505 40586 40612
## [433] 40623 40636 40642 40646 40649 40650 40652 40653 40770 40774 40794 40883
## [445] 40890 40891 40899 40906 40910 40918 40923 40924 40928 40929 40981 41003
## [457] 41009 41013 41015 41047 41051 41059 41063 41166 41189 41190 41192 41197
## [469] 41198 41199 41200 41201 41202 41203 41235 41242 41243 41247 41249 41250
## [481] 41255 41256 41257 41312 41313 41318 41322 41323 41324 41325 41327 41328
## [493] 41329 41398 41422 41427 41428 41430 41438 41440 41441 41444 41531 41563
## [505] 41564 41566 41575 41577 41578 41628 41717 41727 41728 41730 41740 41743
## [517] 41773 42493 42532 42591 42681 43312 43397 43398 43400 43403 43404 43405
## [529] 43410 43464 43467 43470 43471 43472 43473 43475 43476 43479 43480 43481
## [541] 43486 43514 43523 43524 43525 43552 43559 43562 43573 43574 43575 43576
## [553] 43621 43643 43646 43647 43648 43649 43650 43654 43655 43656 43660 43663
## [565] 43664 43666 43739 43745 43755 43756 43757 43758 43760 43765 43772 43775
## [577] 43780 43782 43784 43785 43786 43792 43793 43794 43795 43799 43800 43801
## [589] 43802 43820 43826 43836 43838 43843 43844 43952 43959 43961 43963 43964
## [601] 43966 43967 43968 43969 43971 43972 43975 43976 44060 44061 44074 44075
## [613] 44079 44080 44083 44138 44168 44170 44171 44177 44178 44179 44180 44181
## [625] 44188 44191 44194 44195 44196 44213 44230 44232 44233 44234 44236 44243
## [637] 44244 44245 44249 44253 44254 44255 44272 44273 44278 44279 44281 44282
## [649] 44284 44285 44286 44287 44288 44311 44318 44324 44327 44330 44331 44332
## [661] 44387 44490 44493 44495 44496 44502 44504 44505 44509 44510 44511 44512
## [673] 44523 44528 44529 44531 44532 44537 44613 44673 44677 44687 44690 44693
## [685] 44695 44696 44731 44737 44741 44743 44744 44779 44781 44782 44785 44791
```

```
## [697] 44879 44907 44908 44910 44915 44916 44917 44918 44924 44927 44928 44929
## [709] 44930 44931 44975 45001 45011 45019 45021 45023 45024 45025 45036 45039
## [721] 45040 45042 45043 45072 45078 45079 45080 45084 45086 45087 45158 45162
## [733] 45165 45178 45266 45267 45276 45282 45283 45311 45320 45321 45330 45333
## [745] 45334 45336 45338 45339 45341 45342 45384 45396 45401 45425 45432 45438
## [757] 45439 45588 45589 45628 45643 45652 45694 45713 45744 46024 46028 46066
## [769] 46369 46738 46992 47050 47115 47297 47298 47299 47309 47328 47333 47340
## [781] 47341 47342 47343 47399 47400 47401 47454 47455 47456 47483 47495 47518
## [793] 47519 47678 47682 47713 47740 47741 48124 48822 48903 49045 49069 49089
## [805] 49167 49206 49454 49521 49607 49748 49780 49782 49808 49809 49810 49819
## [817] 49845 49847 49854 49919 49957 49970 49979 50009 50022 50096 50097 50120
## [829] 50124 50133 50150 50152 50156 50197 50199 50361 50373 50478 50481 50574
## [841] 50866 50966 50967 51027 51032 51040 51144 51188 51201 51213 51252 51291
## [853] 51323 51337 51360 51384 51386 51417 51429 51466 51483 51484 51493 51507
## [865] 51527 51543 51557 51566 51612 51641 51665 51756 51765 51817 51818 51843
## [877] 51857 51905 51915 51926 51932 51999 52331 52335 52413 52488 52497 52739
## [889] 52747 52767 52783 52835 52886 52887 52989 53206 53556 53601 53680 53681
## [901] 53861 53862 53880 53945 54167 54193
```

```
Reduce(intersect,list(tsr_d,upr_d,lwr_d,hi_d))
```

```
## integer(0)
```

```
Reduce(intersect,list(tsr_b,upr_b,hi_b,tsr_wt,upr_wt,hi_wt))
```

```
##  [1]  4185  4463  4692  4782  7997 10386 10387 10541 10699 11005 11006 11211
## [13] 11650 13093 13328 13734 16607 18232 19031 19228 19229 20587 20694 20695
## [25] 23325 25155 25164 25253 25346 25588 25785 25821 25938 25948 25990 26004
## [37] 26394 26736 27272 37496 37551 38917 39174 39175 39212 39621 39685 39794
## [49] 39796 39856 39908 40172 40173 40265 40266 40458 41201 41203 41329 43475
## [61] 43486 43575 43621 43663 43760 43799 43800 43802 43975 43976 44138 44253
## [73] 44286 44287 44387 44512 44695 44743 44744 44782 44879 44927 44928 44975
## [85] 45086 49782 49810 49847 49919 49970 50373 51201 51557 51665 51756 51999
```

```
powerlbs2 <- powerlbs
powerlbs3 <- powerlbs2[-c(4185,4463,4692,4782,7997,10386,10387,10541,10699,11005,11006,11211,11650,1309
```

Instructions: Use the `set.seed` random number generator (where the seed number is 448) to select a random sample size of 100 observations of the previous dataset (after removing the 10 outliers). Then, standardize the continuous variables of this random subset.

```
set.seed(448)
rs <- sample(nrow(powerlbs3),100)
powerlbs0 <- powerlbs3[rs,]
head(powerlbs0)
```

```
## # A tibble: 6 x 13
##   Name         Sex   Event Equipment   Age AgeClass WeightClassKg Wilks IPFPoints
##   <chr>        <chr> <chr> <chr>     <dbl> <chr>    <chr>         <dbl>     <dbl>
```

```
## 1 Daniela Mi~ F     SBD   Raw         29   24-34    82.5           223.      378.
## 2 Johanna Ka~ F     SBD   Single-p~  27.5 24-34    72             544.      679.
## 3 Amanda Pate F     SBD   Raw         32   24-34    60             375.      593.
## 4 Jaxson Wea~ F     SBD   Wraps       29   24-34    60             435.      690.
## 5 Annakaisa ~ F     SBD   Raw         31.5 24-34    72             243.      398.
## 6 Heidi Van ~ F     SBD   Raw         25   24-34    63             340.      545.
## # ... with 4 more variables: bodyweightlb <dbl>, best3squatlb <dbl>,
## #   best3benchlb <dbl>, best3deadliftlb <dbl>
```

```
powerlbs000 <- select(powerlbs0,-Name,-Sex,-Event,-Equipment,-Age,-AgeClass,-WeightClassKg)
powerlbs000 <- scale(powerlbs000)
powerlbs00 <- as.data.frame(powerlbs000)
```

Instructions: Use k-means clustering and select 2 clusters. Show the cluster attributes in the form of both

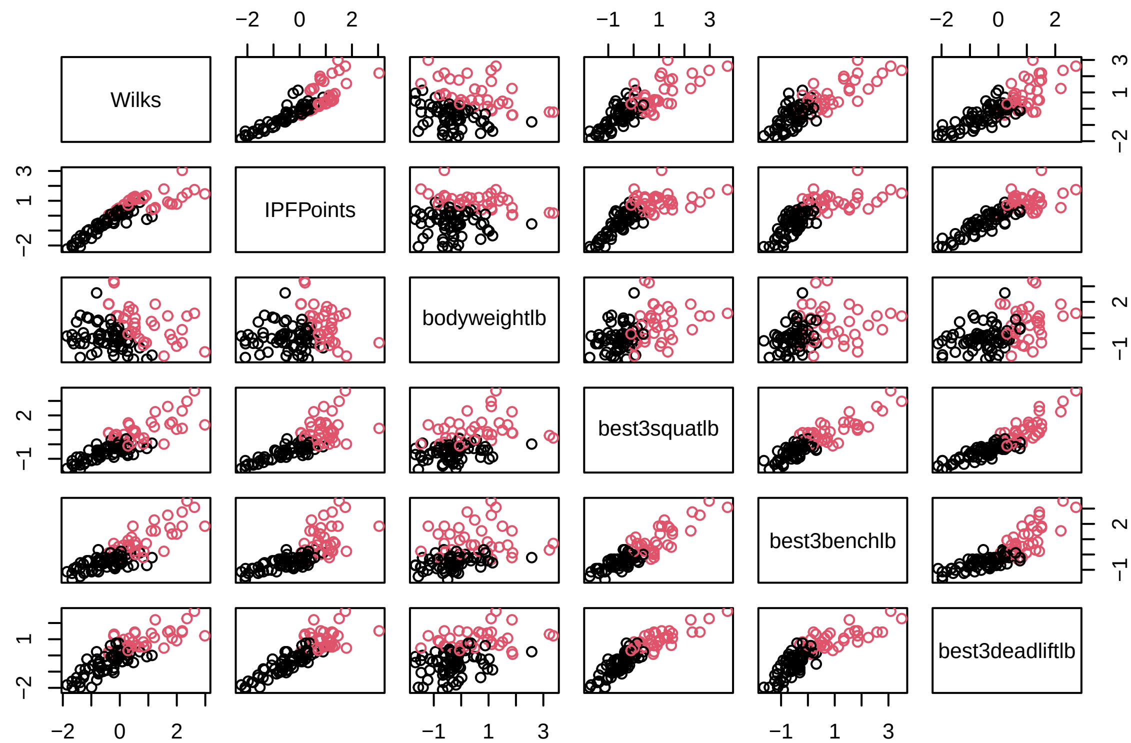
a) a scatter plot matrix of the 6 continuous variables
b) tables showing the mean and median of these 6 variables.

```
kc8 <- kmeans(powerlbs00,centers=2)$cluster
table(kc8)
```

```
## kc8
##  1  2
## 65 35
```

a)

```
pairs(powerlbs00[,1:6],col=kc8)
```



b)

```
summarise(group_by(powerlbs00,kmeans(powerlbs00,centers=2)$cluster),clustersize=length(Wilks),avgwilks=
```

```
## # A tibble: 2 x 14
##   `kmeans(powerlbs00,~ clustersize avgwilks medwilks avgipf medipf  avgwt  medwt
##                  <int>       <int>    <dbl>    <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1                    1          35    0.892    0.549  0.961  0.935  0.582  0.495
## 2                    2          65   -0.480   -0.444 -0.517 -0.443 -0.313 -0.371
## # ... with 6 more variables: avgsqt <dbl>, medsqt <dbl>, avgb <dbl>,
## #   medb <dbl>, avgd <dbl>, medd <dbl>
```

Instructions: Use hierarchical clustering with single linkage and select 2 clusters. Show the cluster attributes in the form of both

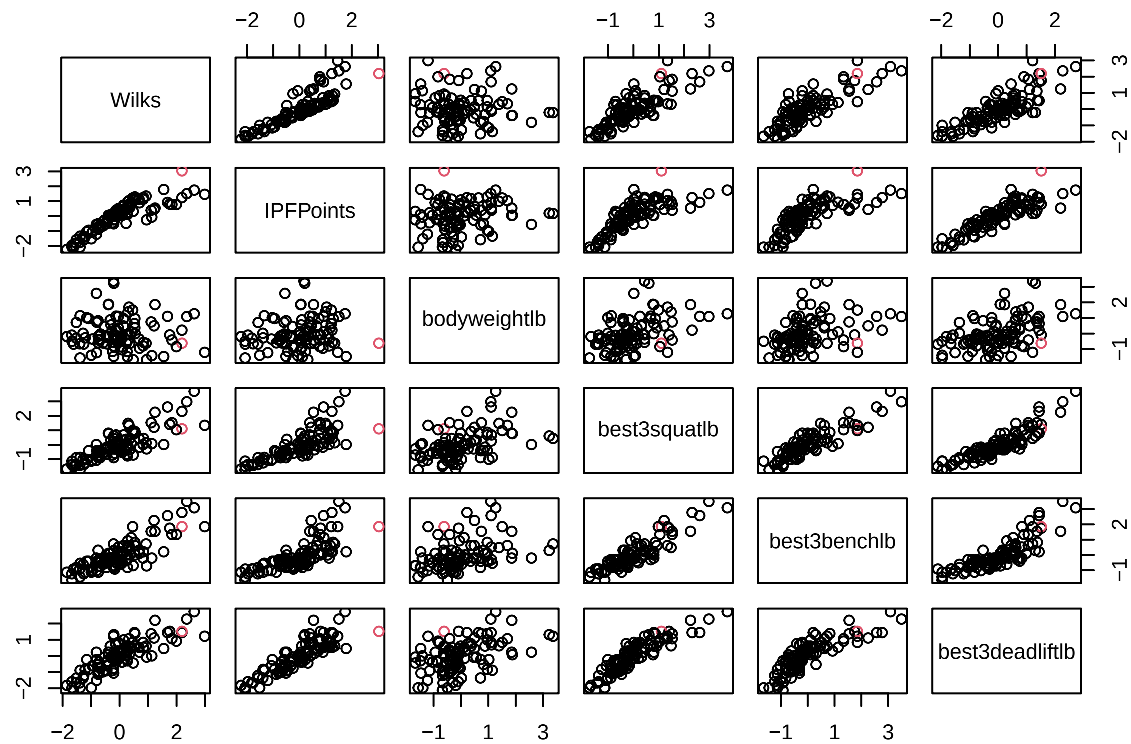
a) a scatter plot matrix of the 6 continuous variables

b) tables showing the mean and median of these 6 variables.

```
d <- dist(powerlbs00, method="euclidean")
fit9 <- hclust(d,method="single")
kc9 <- cutree(fit9,k=2)
```

a)

```
pairs(powerlbs00[,1:6],col=kc9)
```



b)

```
summarise(group_by(powerlbs00,cutree(fit9,k=2)),clustersize=length(Wilks),avgwilks=mean(Wilks),medwilks=
```

```
## # A tibble: 2 x 14
##   `cutree(fit9, k ~ clustersize avgwilks medwilks  avgipf medipf    avgwt  medwt
##                <int>       <int>     <dbl>     <dbl>   <dbl>  <dbl>    <dbl>  <dbl>
## 1                  1          99   -0.0221    -0.103 -0.0307 0.0677  0.00621 -0.172
## 2                  2           1    2.19       2.19   3.04    3.04   -0.615   -0.615
## # ... with 6 more variables: avgsqt <dbl>, medsqt <dbl>, avgb <dbl>,
## #   medb <dbl>, avgd <dbl>, medd <dbl>
```