STATISTICAL ANALYSIS

Gerko Vink
g.vink@uu.nl
Methodology & Statistics @ Utrecht University

5 Jun 2025

DISCLAIMER

I owe a debt of gratitude to many people as the thoughts and code in these slides are the process of years-long development cycles and discussions with my team, friends, colleagues and peers. When someone has contributed to the content of the slides, I have credited their authorship.

Images are either directly linked, or generated with StableDiffusion or DALL-E. That said, there is no information in this presentation that exceeds legal use of copyright materials in academic settings, or that should not be part of the public domain.

> ⚠ **Warning**
>
> You **may use** any and all content in this presentation - including my name - and submit it as input to generative AI tools, with the following **exception**:
>
> - You must ensure that the content is not used for further training of the model

SLIDE MATERIALS AND SOURCE CODE

Materials

- lecture slides on Moodle
- course page: www.gerkovink.com/sur
- source: github.com/gerkovink/sur

RECAP

Gisteren hebben we deze onderwerpen behandeld:

- Het combineren van datasets
- Groeperen en aggregeren
- Nieuwe variabelen creëren
- Filteren en sorteren van gegevens
- Het maken en aanpassen van datagroepen
- Clustering van gegevens

TODAY

Vandaag behandelen we de volgende onderwerpen:

- Beschrijvende statistiek
- Kruistabellen en frequentieverdelingen
- χ^2-toets
- Andere toets- en associatiematen
- Simpele lineaire regressie
- Analyses draaien op groepen

WE USE THE FOLLOWING PACKAGES

```
1 library(dplyr) # data manipulation
2 library(magrittr) # for the pipe
3 library(psych) # descriptive statistics
4 library(mice) # for the boys data
5 library(haven) # for reading Stata files
```

BOYS DATA

```
boys %>%
  slice_head(n=6) # select first 6 rows
```

```
    age  hgt   wgt   bmi   hc  gen  phb tv   reg
1 0.035 50.1 3.650 14.54 33.7 <NA> <NA> NA south
2 0.038 53.5 3.370 11.77 35.0 <NA> <NA> NA south
3 0.057 50.0 3.140 12.56 35.2 <NA> <NA> NA south
4 0.060 54.5 4.270 14.37 36.7 <NA> <NA> NA south
5 0.062 57.5 5.030 15.21 37.3 <NA> <NA> NA south
6 0.068 55.5 4.655 15.11 37.0 <NA> <NA> NA south
```

BESCHRIJVENDE STATISTIEK

psych::describe()

Yesterday we already used the `describe()` function from package `psych`

```
boys %>%
  select(1:5) %>% # select first 5 columns
  psych::describe()
```

```
    vars   n   mean    sd median trimmed   mad   min    max  range  skew
age    1 748   9.16  6.89  10.50    9.03 10.14  0.04  21.18  21.14 -0.03
hgt    2 728 132.15 46.51 147.30  134.31 52.34 50.00 198.00 148.00 -0.35
wgt    3 744  37.15 26.03  34.65   35.49 34.93  3.14 117.40 114.26  0.38
bmi    4 727  18.07  3.05  17.45   17.73  2.64 11.77  31.74  19.97  1.15
hc     5 702  51.51  5.91  53.00   52.18  5.26 33.70  65.00  31.30 -0.88
    kurtosis   se
age    -1.56 0.25
hgt    -1.43 1.72
wgt    -1.03 0.95
bmi     1.76 0.11
hc      0.05 0.22
```

summary()

The `summary()` function is a base R function that provides a summary of the data. It is a very useful function for quick inspection of variables

```
1 boys %>% summary()
```

```
      age              hgt              wgt              bmi        
 Min.   : 0.035   Min.   : 50.00   Min.   :  3.14   Min.   :11.77  
 1st Qu.: 1.581   1st Qu.: 84.88   1st Qu.: 11.70   1st Qu.:15.90  
 Median :10.505   Median :147.30   Median : 34.65   Median :17.45  
 Mean   : 9.159   Mean   :132.15   Mean   : 37.15   Mean   :18.07  
 3rd Qu.:15.267   3rd Qu.:175.22   3rd Qu.: 59.58   3rd Qu.:19.53  
 Max.   :21.177   Max.   :198.00   Max.   :117.40   Max.   :31.74  
                  NA's   :20       NA's   :4        NA's   :21     
       hc          gen         phb             tv           reg      
 Min.   :33.70   G1  : 56   P1  : 63   Min.   : 1.00   north: 81  
 1st Qu.:48.12   G2  : 50   P2  : 40   1st Qu.: 4.00   east :161  
 Median :53.00   G3  : 22   P3  : 19   Median :12.00   west :239  
 Mean   :51.51   G4  : 42   P4  : 32   Mean   :11.89   south:191  
 3rd Qu.:56.00   G5  : 75   P5  : 50   3rd Qu.:20.00   city : 73  
 Max.   :65.00   NA's:503   P6  : 41   Max.   :25.00   NA's :  3  
 NA's   :46                 NA's:503   NA's   :522                
```

`summary()` ONLY ON THE NON-NUMERIC COLUMNS

To perform summary only on the non-numeric columns in the `boys` data set, we can use all of the skills we have learned so far:

```
boys %>%
  select(!where(is.numeric)) %>% # pay attention to the ! location
  summary()
```

```
  gen         phb          reg      
 G1  : 56   P1  : 63   north: 81  
 G2  : 50   P2  : 40   east :161  
 G3  : 22   P3  : 19   west :239  
 G4  : 42   P4  : 32   south:191  
 G5  : 75   P5  : 50   city : 73  
 NA's:503   P6  : 41   NA's :  3  
            NA's:503              
```

CONTINGENCY TABLES

`table()` WITH 2 VARIABLES

```
1 boys %$%
2   table(gen, phb)
```

```
    phb
gen  P1 P2 P3 P4 P5 P6
  G1 46  9  0  0  0  0
  G2 16 27  6  1  0  0
  G3  0  3 10  7  2  0
  G4  0  1  3 21 14  3
  G5  0  0  0  3 34 38
```

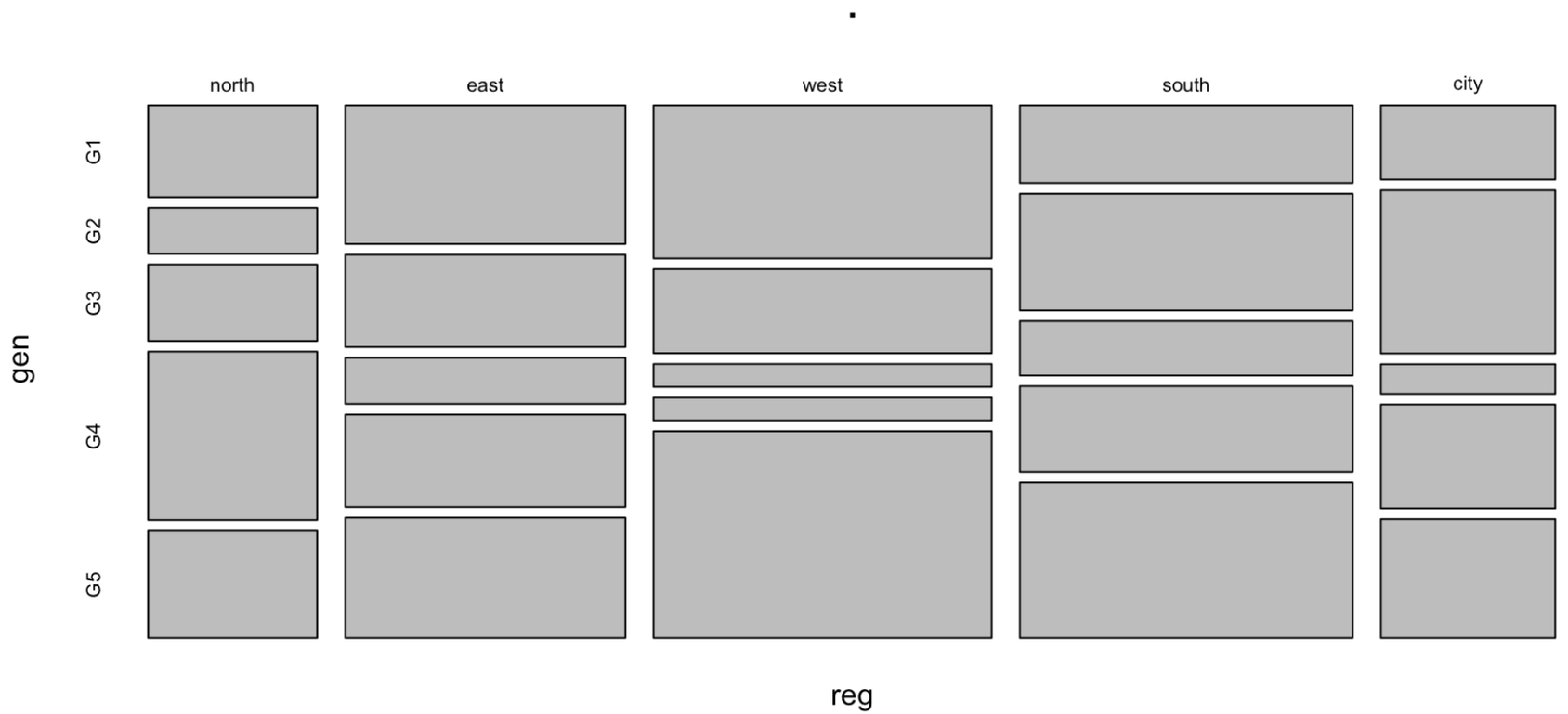
`table()` WITH 3 VARIABLES

```
1 boys %$%
2   table(gen, phb, reg)
```

```
, , reg = north

    phb
gen  P1 P2 P3 P4 P5 P6
  G1  4  1  0  0  0  0
  G2  1  1  1  0  0  0
  G3  0  0  2  2  1  0
  G4  0  0  1  4  5  1
  G5  0  0  0  0  4  3

, , reg = east

    phb
gen  P1 P2 P3 P4 P5 P6
  G1 12  3  0  0  0  0
  G2  2  6  1  1  0  0
  G3  0  1  1  3  0  0
  G4  0  0  1  6  3  0
  G5  0  0  0  0  3 10
```

PLOTTING TABLES

```
boys %$%
  table(reg, gen) %>% # table with reg x gen
  plot() # reg on the x-axis and gen on the y-axis
```



PROPORTIONAL TABLES

```
1 boys %$%
2   table(reg, gen) %>% # table with reg x gen
3   prop.table() # table proportions
```

```
       gen
reg              G1          G2          G3          G4          G5
  north 0.024489796 0.012244898 0.020408163 0.044897959 0.028571429
  east  0.061224490 0.040816327 0.020408163 0.040816327 0.053061224
  west  0.081632653 0.044897959 0.012244898 0.012244898 0.110204082
  south 0.040816327 0.061224490 0.028571429 0.044897959 0.081632653
  city  0.020408163 0.044897959 0.008163265 0.028571429 0.032653061
```

```
1 boys %$%
2   table(reg, gen) %>% # table with reg x gen
3   prop.table() %>%  # table proportions
4   sum() # check that the table sums up to 1
```

```
[1] 1
```

PROPORTIONAL TABLES - ROW MARGIN

```
boys %$%
  table(reg, gen) %>% # table with reg x gen
  prop.table(margin = 1) # rows sum up to 1
```

```
       gen
reg              G1         G2         G3         G4         G5
  north 0.18750000 0.09375000 0.15625000 0.34375000 0.21875000
  east  0.28301887 0.18867925 0.09433962 0.18867925 0.24528302
  west  0.31250000 0.17187500 0.04687500 0.04687500 0.42187500
  south 0.15873016 0.23809524 0.11111111 0.17460317 0.31746032
  city  0.15151515 0.33333333 0.06060606 0.21212121 0.24242424
```

```
boys %$%
  table(reg, gen) %>% # table with reg x gen
  prop.table(margin = 1) %>% # rows sum up
  rowSums() # check that rows sum up to 1
```

```
north  east  west south  city
    1     1     1     1     1
```

The argument `margin = 1` means that the proportions are calculated for each row, so that the rows sum up to 1. Remember that in R rows are always the first margin, and columns are the second margin. Just like in matrices, where the first dimension is the rows and the second dimension is the columns (`matrix(data, nrow = 3, ncol = 2)`). The same with subsetting: `data[1:3, 1:2]` means the first three rows and the first two columns.

PROPORTIONAL TABLES - COLUMN MARGIN

```r
boys %$%
  table(reg, gen) %>% # table with reg x gen
  prop.table(margin = 2) # columns sum up to 1
```

```
       gen
reg             G1         G2         G3         G4         G5
  north 0.10714286 0.06000000 0.22727273 0.26190476 0.09333333
  east  0.26785714 0.20000000 0.22727273 0.23809524 0.17333333
  west  0.35714286 0.22000000 0.13636364 0.07142857 0.36000000
  south 0.17857143 0.30000000 0.31818182 0.26190476 0.26666667
  city  0.08928571 0.22000000 0.09090909 0.16666667 0.10666667
```

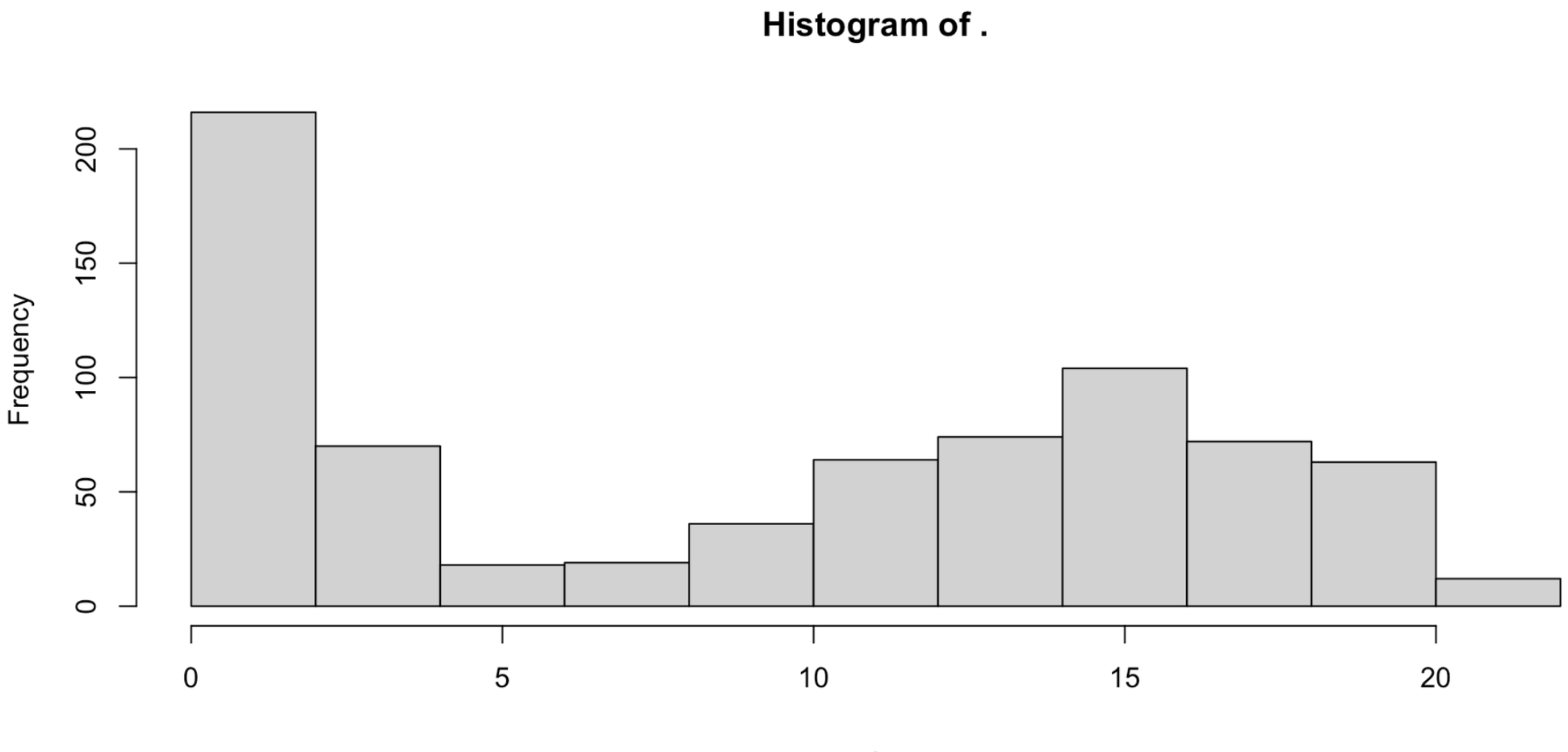
```r
boys %$%
  table(reg, gen) %>% # table with reg x gen
  prop.table(margin = 2) %>%  # columns sum up to 1
  colSums() # check that columns sum up to 1
```

```
G1 G2 G3 G4 G5
 1  1  1  1  1
```

FREQUENCY DISTRIBUTIONS

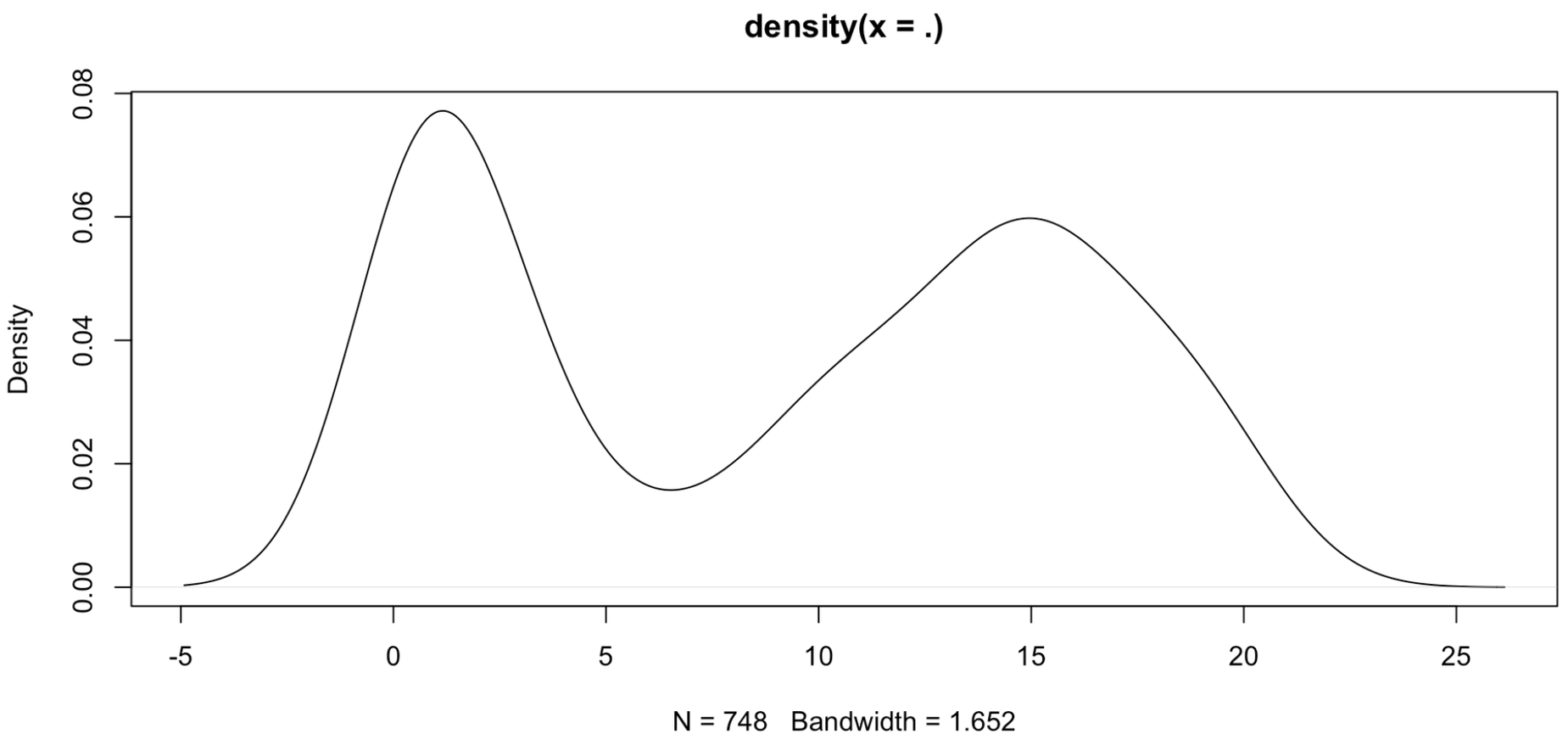
INSPECTING CONTINUOUS DATA

```
1 boys$age %>% hist()
```



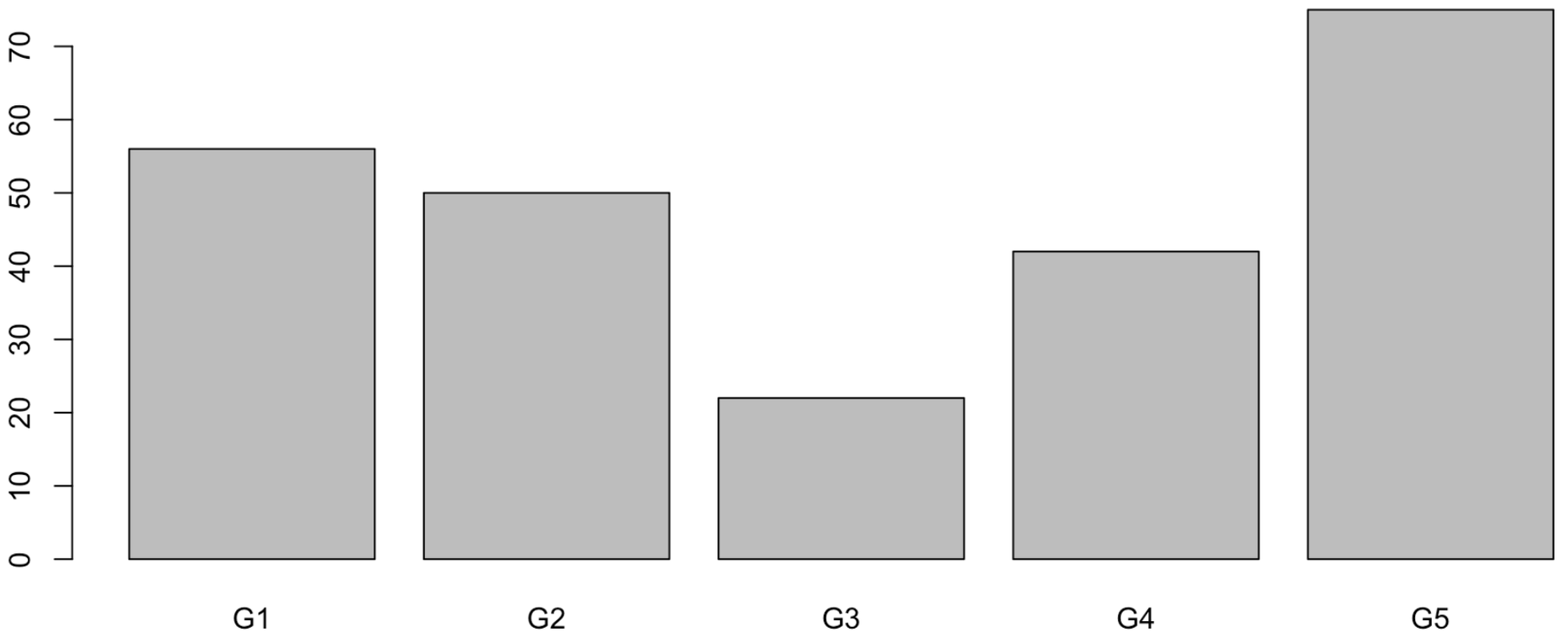
INSPECTING CONTINUOUS DATA

```
1 boys$age %>%
2   density() %>%
3   plot()
```



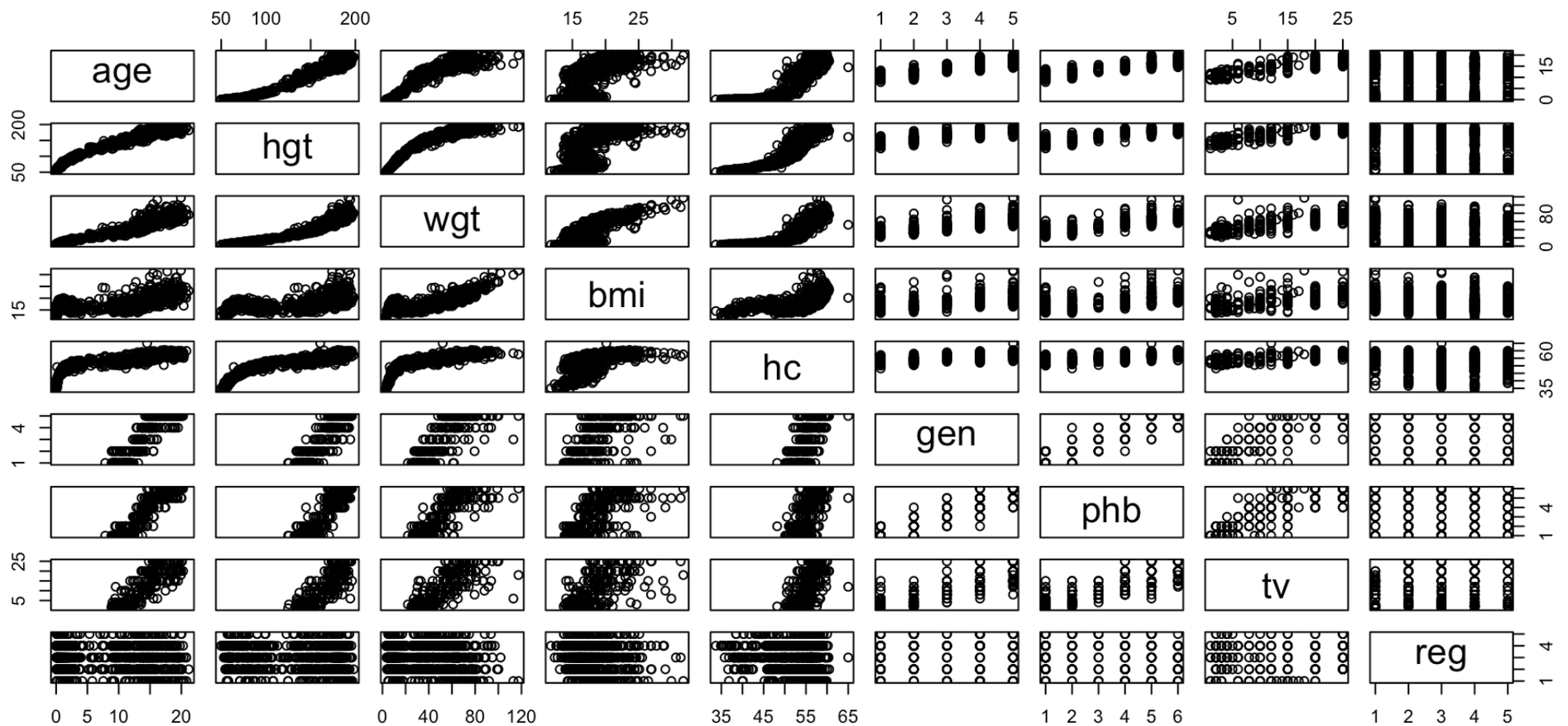
INSPECTING CATEGORICAL DATA

```
1 boys$gen %>% plot()
```



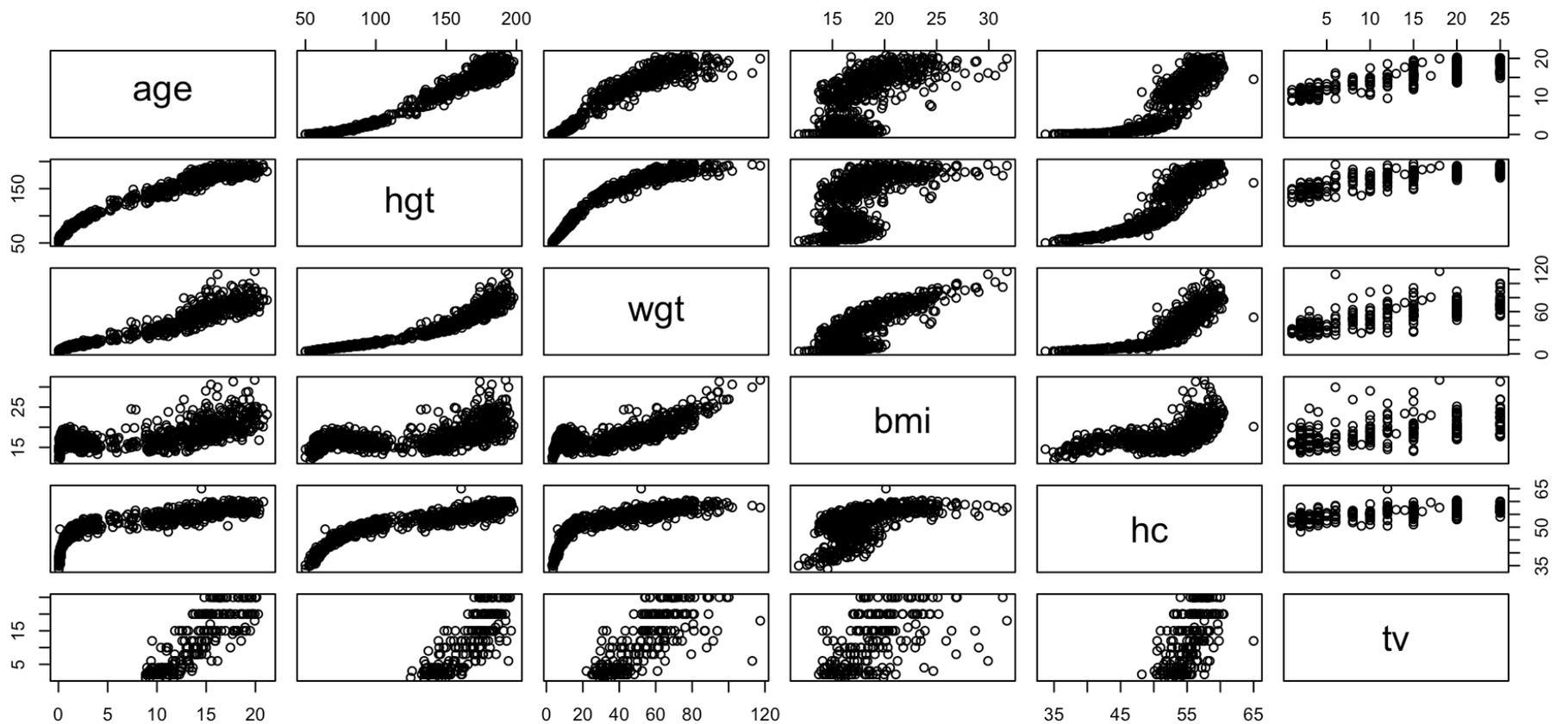
PLOTTING DATA FRAMES

```
1 boys %>%
2   plot()
```



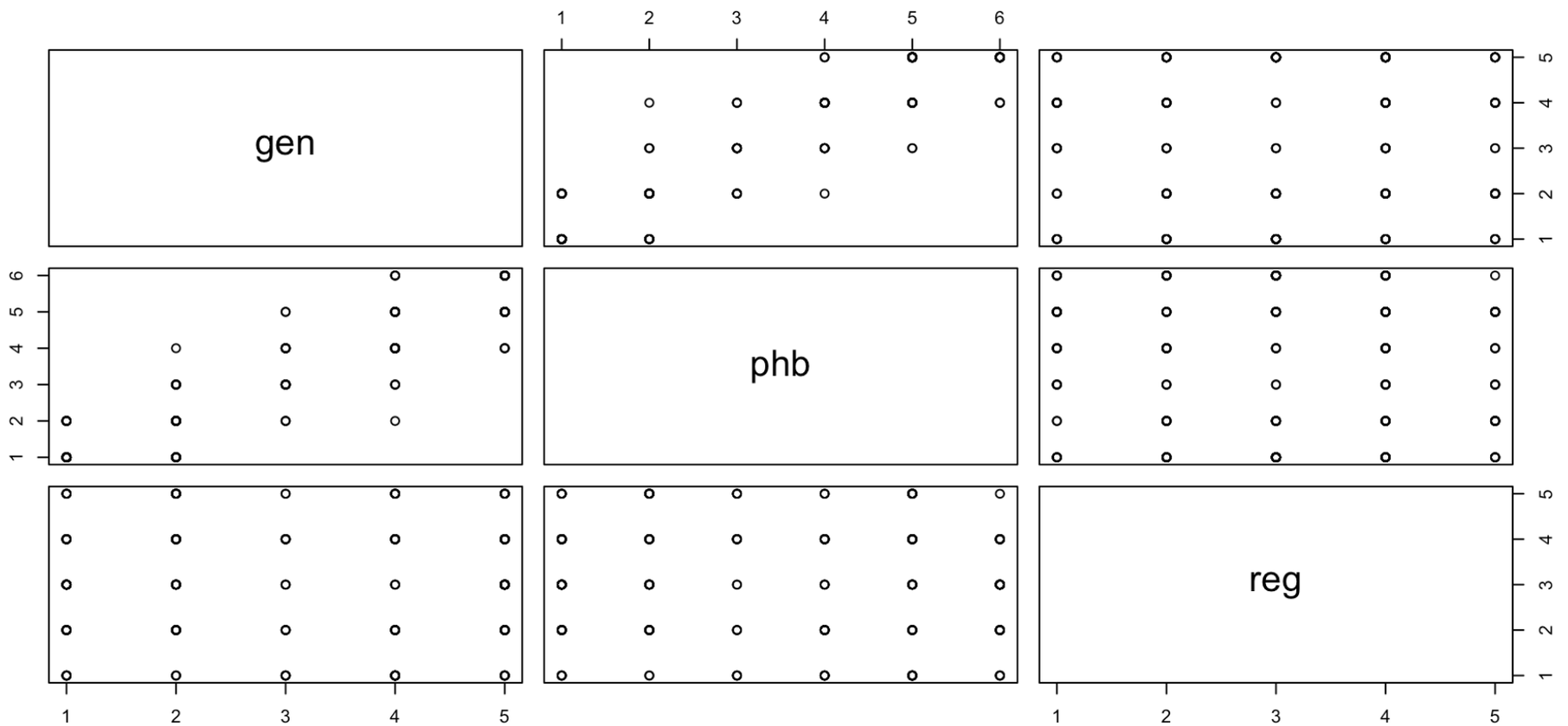
PLOTTING (PARTS OF) DATA FRAMES

```
1 boys %>%
2   select(where(is.numeric)) %>%
3   plot()
```



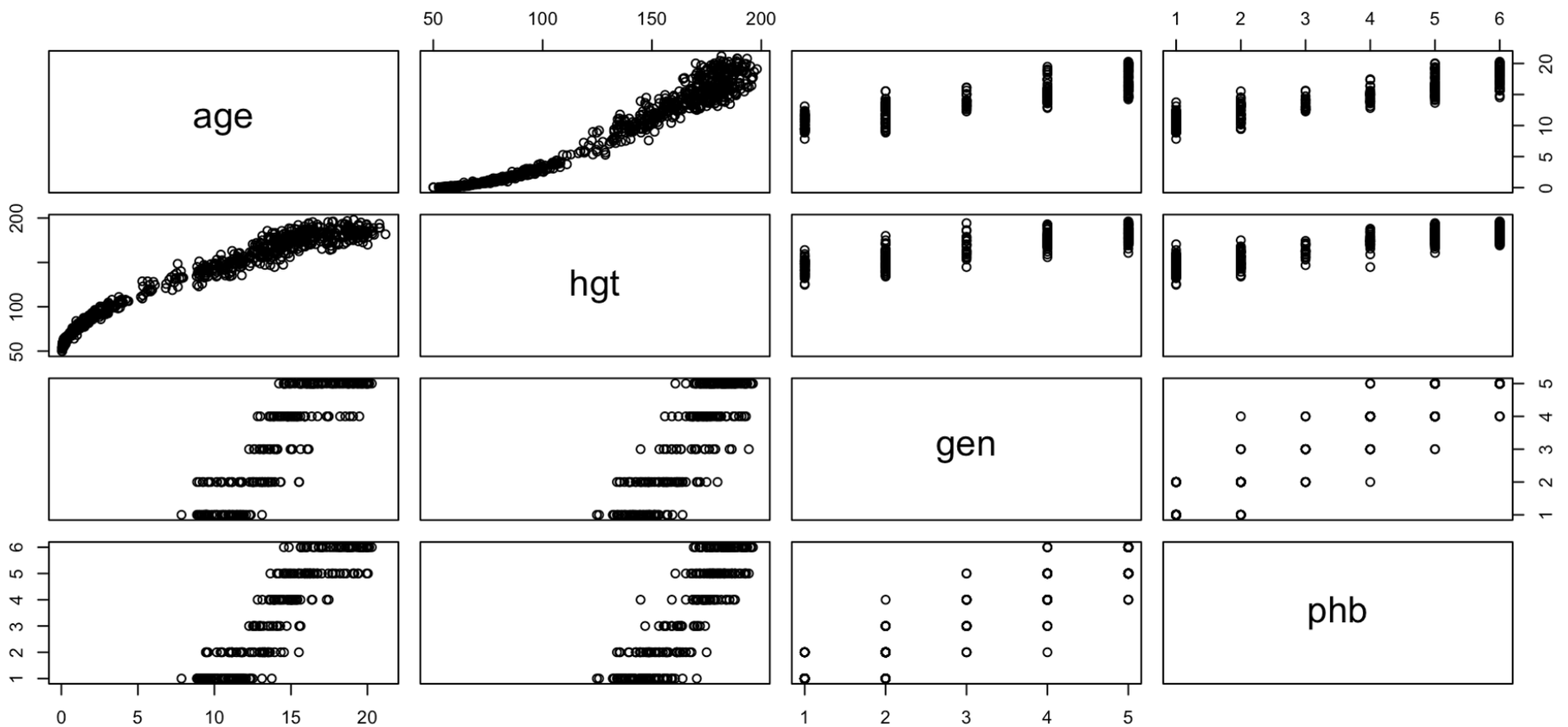
PLOTTING (PARTS OF) DATA FRAMES

```
1 boys %>%
2   select(where(is.factor)) %>%
3   plot()
```



PLOTTING (PARTS OF) DATA FRAMES

```
1 boys %>%
2   select(age, hgt, gen, phb) %>%
3   plot()
```



ASSOCIATION MEASURES

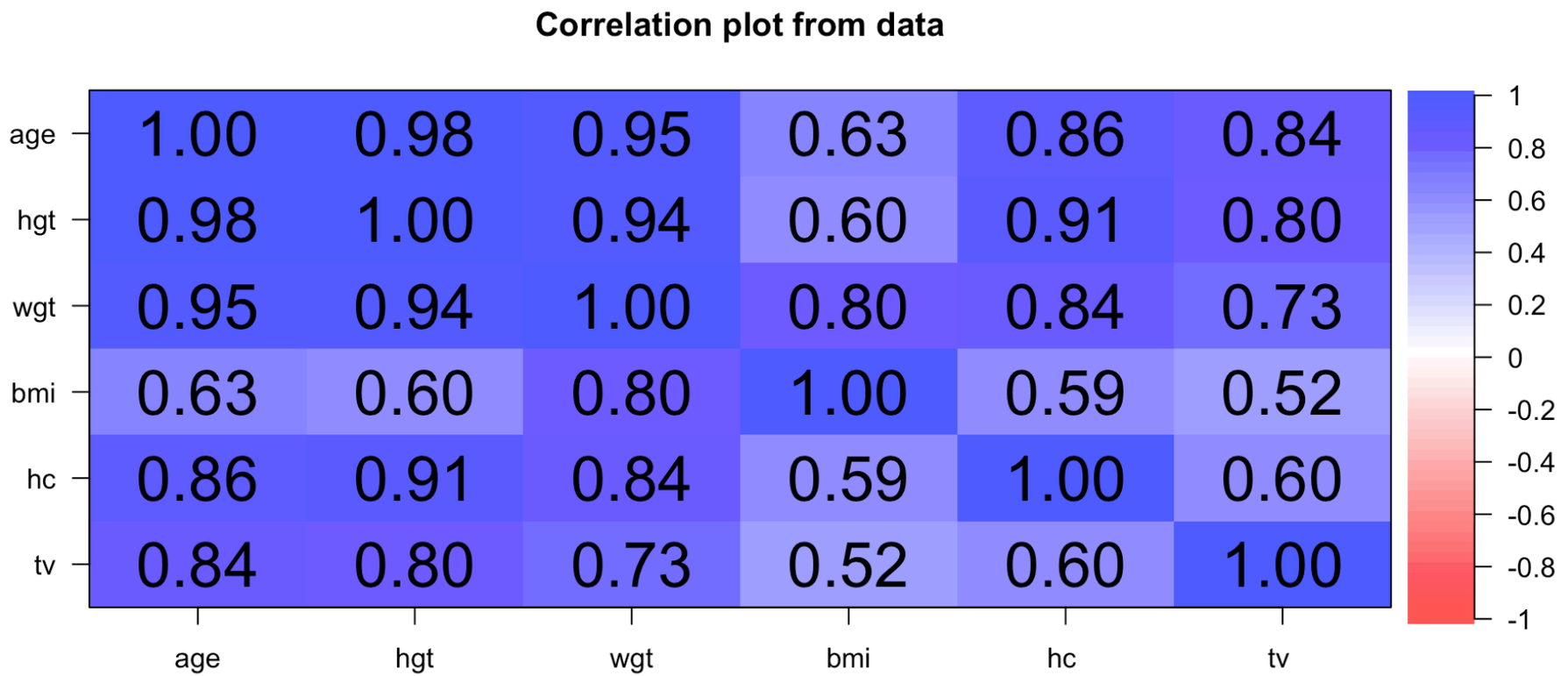
CORRELATION BETWEEN CONTINUOUS VARIABLES

```r
boys %>%
  select(where(is.numeric)) %>% # select numeric columns
  cor(use = "pairwise.complete.obs") # correlation matrix
```

```
          age       hgt       wgt       bmi        hc        tv
age 1.0000000 0.9752568 0.9505762 0.6319678 0.8572431 0.8357285
hgt 0.9752568 1.0000000 0.9428906 0.5999647 0.9123139 0.7951793
wgt 0.9505762 0.9428906 1.0000000 0.7976402 0.8374706 0.7280757
bmi 0.6319678 0.5999647 0.7976402 1.0000000 0.5912613 0.5186216
hc  0.8572431 0.9123139 0.8374706 0.5912613 1.0000000 0.5958305
tv  0.8357285 0.7951793 0.7280757 0.5186216 0.5958305 1.0000000
```

psych::cor.plot()

```
boys %>%
  select(where(is.numeric)) %>%
  cor.plot()
```



TESTING THE CORRELATION

```
1 boys %$%
2   cor.test(hgt, wgt) # correlation test between height and weight
```

```
    Pearson's product-moment correlation

data:  hgt and wgt
t = 76.217, df = 725, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.9342287 0.9504409
sample estimates:
      cor
0.9428906
```

CORRELATIONS BETWEEN ORDERED CATEGORICAL VARIABLES

We can use Spearman's rank-order correlation to calculate the association between two ordered (ordinal) vectors

```
boys %>%
  select(where(is.ordered)) %>% # select ordered categorical columns
  mutate(across(where(is.ordered), ~ as.numeric(.))) %>%
  cor(method = "spearman") # spearman correlation
```

```
     gen phb
gen    1  NA
phb   NA   1
```

```
boys %>%
  select(where(is.ordered)) %>% # select ordered categorical columns
  mutate(across(where(is.ordered), ~ as.numeric(.))) %>%
  cor(use = "pairwise.complete.obs", method = "spearman") # spearman correlation
```

```
          gen       phb
gen 1.000000 0.920754
phb 0.920754 1.000000
```

MANUAL CALCULATION OF SPEARMAN'S ϱ

```r
genphb <- boys %>% select(gen, phb) %>% na.omit() # joint observations
rx <- rank(as.numeric(genphb$gen), ties.method = "average") # rank the values of gen
ry <- rank(as.numeric(genphb$phb), ties.method = "average") # rank the values of phb
rho <- cov(rx, ry) / (sd(rx) * sd(ry))
rho
```

```
[1] 0.920754
```

```r
boys %>%
  select(where(is.ordered)) %>% # select ordered categorical columns
  mutate(across(where(is.ordered), ~ as.numeric(.))) %$%
  cor.test(gen, phb, method = "spearman") # spearman correlation
```

```
    Spearman's rank correlation rho

data:  gen and phb
S = 191862, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
     rho
0.920754
```

TIES

In the following chunk, the 2nd and 3rd values are tied

```
x <- c(10, 20, 20, 40, 50)
y <- c(1, 2, 2, 4, 5)
```

Tied ranks will by default recieve the average of their ranks

```
rank(x)  # returns: 1 2.5 2.5 4 5
```

```
[1] 1.0 2.5 2.5 4.0 5.0
```

```
rank(y)  # returns: 1 2.5 2.5 4 5
```

```
[1] 1.0 2.5 2.5 4.0 5.0
```

This can cause problems with the robustness of Spearman's ϱ.

The reason why I mention this, is that results may differ between packages or statistical processors (e.g. R, STATA, SPSS, etc) because the ties are by default handled differently. In such cases, explore `?rank` to see what methods are available.

KENDALL'S TAU

Use Kendall's tau for small, clean, tied, or ordinal data. Use Spearman's rho for quick rank-based correlation on larger datasets.

```
boys %>%
  select(where(is.ordered)) %>% # select ordered categorical columns
  mutate(across(where(is.ordered), ~ as.numeric(.))) %$%
  cor.test(gen, phb, method = "kendall") # spearman correlation
```

```
    Kendall's rank correlation tau

data:  gen and phb
z = 16.481, p-value < 2.2e-16
alternative hypothesis: true tau is not equal to 0
sample estimates:
      tau
0.8463354
```

DIFFERENCE TESTS FOR TWO GROUPS

STUDENT'S T-TEST

```r
boys %>%
  mutate(overweight = case_when(bmi >= 25 ~ "overweight",
                                bmi < 25 ~ "not overweight")) %$%
  t.test(age ~ overweight)
```

```
    Welch Two Sample t-test

data:  age by overweight
t = -15.971, df = 32.993, p-value < 2.2e-16
alternative hypothesis: true difference in means between group not overweight and group overweight is not equal
to 0
95 percent confidence interval:
 -9.393179 -7.270438
sample estimates:
mean in group not overweight     mean in group overweight
                    9.103392                    17.435200
```

The Welch Two Sample t-test compares the means of two groups while allowing for unequal variances and sample sizes between them. Assumptions:

- The data in each group are normally distributed (or approximately so),
- The observations are independent,
- But it does not assume equal variances between groups (unlike the classic Student's t-test).

STUDENT'S T-TEST

```
boys %>%
  mutate(city = case_when(reg == "city" ~ "city",
                          reg != "city" ~ "rural")) %$%
  t.test(bmi ~ city)
```

```
    Welch Two Sample t-test

data:  bmi by city
t = 0.39583, df = 92.567, p-value = 0.6931
alternative hypothesis: true difference in means between group city and group rural is not equal to 0
95 percent confidence interval:
 -0.5316442  0.7963303
sample estimates:
 mean in group city mean in group rural
           18.20000            18.06766
```

X^2-TEST FOR 2 CATEGORIES

```r
boys_new <- boys %>%
  mutate(city = case_when(reg == "city" ~ "city",
                          reg != "city" ~ "rural"),
         overweight = case_when(bmi >= 25 ~ "overweight",
                                bmi < 25 ~ "not overweight"))

boys_new %$% table(city, overweight)
```

```
       overweight
city    not overweight overweight
  city              70          1
  rural            634         19
```

X^2-TEST FOR 2 CATEGORIES

```
boys_new %$%
  table(city, overweight) %>%
  chisq.test()
```

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

data:  .
X-squared = 0.12372, df = 1, p-value = 0.725
```

```
boys_new %$%
  chisq.test(city, overweight)
```

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

data:  city and overweight
X-squared = 0.12372, df = 1, p-value = 0.725
```

EXPECTED CELL FREQUENCIES

```
1 X2_boys <- boys_new %$%
2   chisq.test(city, overweight)
3 X2_boys$observed
```

```
       overweight
city    not overweight overweight
  city              70          1
  rural            634         19
```

```
1 X2_boys$expected
```

```
       overweight
city    not overweight overweight
  city        69.03867   1.961326
  rural      634.96133  18.038674
```

We can also manually calculate the expected cell frequencies. For example, for the cell [1, 1], the cell frequency would be

```
1 (704 * 71) / 724 # column [, 1] total times row [1, ] total divided by grand total
```

```
[1] 69.03867
```

EXTRACT EXPECTED CELL FREQUENCIES WITH THE PIPE

We can do this in the pipe by using the placeholder `.`:

```
boys_new %$%
  chisq.test(city, overweight) %>%
  .$expected
```

```
       overweight
city    not overweight overweight
  city         69.03867   1.961326
  rural       634.96133  18.038674
```

FISHER EXACT TEST

If the expected cell frequencies are too low, it is more robust to calculate Fisher's exact test instead of the χ^2-test.

```
boys_new %$%
  fisher.test(city, overweight)
```

```
    Fisher's Exact Test for Count Data

data:  city and overweight
p-value = 0.7109
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
  0.3232424 88.3683832
sample estimates:
odds ratio
   2.09617
```

Fisher's exact test is better than the chi-squared test for low expected cell frequencies because it calculates the exact p-value without relying on large-sample approximations, making it more accurate when expected counts are small.

DIFFERENCE TESTS FOR MORE THAN TWO GROUPS

X^2-TEST

```
boys %$%
  table(gen, phb)
```

```
    phb
gen   P1 P2 P3 P4 P5 P6
  G1 46  9  0  0  0  0
  G2 16 27  6  1  0  0
  G3  0  3 10  7  2  0
  G4  0  1  3 21 14  3
  G5  0  0  0  3 34 38
```

```
boys %$%
  chisq.test(gen, phb)
```

```
    Pearson's Chi-squared test

data:  gen and phb
X-squared = 411.34, df = 20, p-value < 2.2e-16
```


X^2-TEST OR FISHER EXACT TEST

```
boys %$%
  chisq.test(gen, phb) %>%
  .$expected
```

```
    phb
gen          P1         P2       P3       P4         P5         P6
  G1 13.975410  9.016393 4.282787 7.213115 11.270492  9.241803
  G2 12.704918  8.196721 3.893443 6.557377 10.245902  8.401639
  G3  5.590164  3.606557 1.713115 2.885246  4.508197  3.696721
  G4 10.672131  6.885246 3.270492 5.508197  8.606557  7.057377
  G5 19.057377 12.295082 5.840164 9.836066 15.368852 12.602459
```

```
boys %$%
  fisher.test(gen, phb)

Error in fisher.test(gen, phb) :
  FEXACT error 6 (f5xact).  LDKEY=621 is too small for this problem: kval=186370998.
Try increasing the size of the workspace.
```

In this case the p-value is so small, that a lot of memory is needed to model exactly how small the p-value is.

SIMULATE FISHER EXACT TEST

```
boys %$%
  fisher.test(gen, phb, simulate.p.value = TRUE)
```

```
    Fisher's Exact Test for Count Data with simulated p-value (based on
    2000 replicates)

data:  gen and phb
p-value = 0.0004998
alternative hypothesis: two.sided
```

```
boys %$%
  fisher.test(gen, phb,
              simulate.p.value = TRUE,
              B = 100000) # number of monte carlo simulations
```

```
    Fisher's Exact Test for Count Data with simulated p-value (based on
    1e+05 replicates)

data:  gen and phb
p-value = 1e-05
alternative hypothesis: two.sided
```


ONE-WAY ANOVA

```r
boys %>%
  group_by(reg) %>%
  summarise(mean_age = mean(age, na.rm=TRUE))
```

```
# A tibble: 6 × 2
  reg   mean_age
  <fct>    <dbl>
1 north    11.9
2 east      9.24
3 west      9.00
4 south     8.59
5 city      8.27
6 <NA>      1.33
```

```r
boys %$%
  lm(age ~ reg) %>%
  anova()
```

```
Analysis of Variance Table

Response: age
           Df Sum Sq Mean Sq F value   Pr(>F)
reg         4    734 183.381  3.9247 0.003678 **
Residuals 740  34577  46.725
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

ONE-WAY ANOVA

```
1 boys %$%
2   lm(age ~ reg) %>%
3   summary()
```

```
Call:
lm(formula = age ~ reg)

Residuals:
    Min      1Q  Median      3Q     Max
-11.805  -7.376   1.339   5.955  11.935

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  11.8984     0.7595  15.666  < 2e-16 ***
regeast      -2.6568     0.9312  -2.853 0.004449 **
regwest      -2.9008     0.8788  -3.301 0.001011 **
regsouth     -3.3074     0.9064  -3.649 0.000282 ***
regcity      -3.6266     1.1031  -3.288 0.001058 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.836 on 740 degrees of freedom
```

`group_by()` AND ANALYSES

```
1 boys %>%
2   group_by(reg) %>%
3   summarise(correlation = cor(hgt, wgt, use = "pairwise.complete.obs"))
```

```
# A tibble: 6 × 2
  reg   correlation
  <fct>       <dbl>
1 north       0.929
2 east        0.931
3 west        0.951
4 south       0.944
5 city        0.956
6 <NA>        0.998
```

NOTE: `summarise()` expects outputs that are scalars, not lists or objects.

`group_by()` AND COMPLEX ANALYSES

You can't directly use `summarise()` with complex analyses like `cor.test()` to extract output because:

- cor.test() returns a complex object (not just a number).

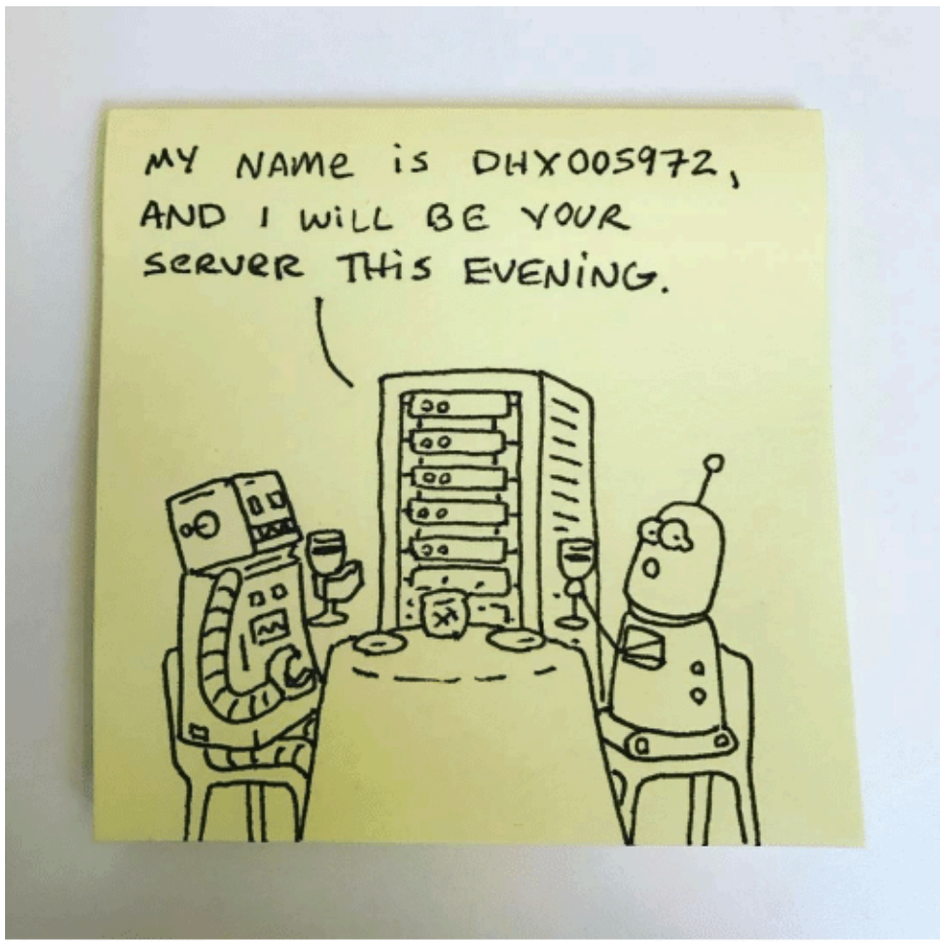
There are ways around this

```
1 library(purrr)
2 boys%>%
3   group_by(reg) %>%
4   summarise(
5     test = list(cor.test(hgt, wgt)),
6     estimate = map_dbl(test, ~ .x$estimate),
7     p_value = map_dbl(test, ~ .x$p.value)
8   )
```

```
# A tibble: 6 × 4
  reg   test    estimate   p_value
  <fct> <list>     <dbl>     <dbl>
1 north <htest>    0.929 7.01e- 35
2 east  <htest>    0.931 1.66e- 70
3 west  <htest>    0.951 1.43e-117
4 south <htest>    0.944 1.14e- 90
5 city  <htest>    0.956 1.41e- 38
6 <NA>  <htest>    0.998 4.33e-  2
```

We will explore these ways in the next lecture.

FOR FUN



When you go out for a byte

source

PRACTICAL