DATA MANAGEMENT IN R

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

2 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

Yesterday we learned

1. How to use `R` and `RStudio`
2. How to install packages
3. How to use simple data containers
4. How to do subsetting in base `R` with `[ ]` and `$`
5. How to use logical operators to subset data
6. How to adhere to code conventions en style

TODAY

- Importeren en bestuderen van datasets
- Begrijpen en toepassen van verschillende datatypes en database formats
- Variabelen labelen en (her)coderen
- De blauwdruk van R: frames en environments
- Pipes
- Formules gebruiken in functies

NEW PACKAGES WE USE

```
1 library(tibble)   # tibbles variation on data frames
2 library(dplyr)    # data manipulation
3 library(haven)    # in/exporting data
4 library(magrittr) # pipes
5 library(labelled) # labelled data manipulation
6 library(tidyr)    # data tidying
7 library(broom)    # tidying model outputs
```

IMPORTING DATA: STATA

```
1 stata_data <- read_dta("files/03-poverty-analysis-data-2022-rt001-housing-plus.dta")
2 head(stata_data)
```

```
# A tibble: 6 × 114
        hhid domain2     psu domain  gp_subdom district fortnight panel   hhid16
       <dbl> <dbl+lbl> <dbl> <dbl+l> <dbl+lbl> <chr>        <dbl> <dbl+l>  <dbl>
1 1102500401 1.1       10250 1 [Gre… 1 [Param… Paramar…          1 1 [Pan… 6.02e6
2 1102500501 1.1       10250 1 [Gre… 1 [Param… Paramar…          1 1 [Pan… 6.02e6
3 1102500502 1.1       10250 1 [Gre… 1 [Param… Paramar…          1 1 [Pan… 6.02e6
4 1102501202 1.1       10250 1 [Gre… 1 [Param… Paramar…          1 1 [Pan… 6.02e6
5 1107430501 1.1       10743 1 [Gre… 1 [Param… Paramar…          1 1 [Pan… 6.04e6
6 1107430801 1.1       10743 1 [Gre… 1 [Param… Paramar…          1 1 [Pan… 6.04e6
# ℹ 105 more variables: lat_cen <dbl>, long_cen <dbl>, result <dbl+lbl>,
#   end_date_n <date>, Year_s <dbl>, Month_s <dbl>, Day <dbl>, stratum <dbl>,
#   hhid_text <chr>, HHsize <dbl>, HHsize2 <dbl>, interv <dbl>, end_date <chr>,
#   q17_02 <dbl+lbl>, q17_03a <dbl+lbl>, q17_03b <dbl+lbl>, q17_04 <dbl+lbl>,
#   q12a <dbl+lbl>, q12_01a <dbl>, q12_01b <dbl>, q12_02a <dbl>, q12_02b <dbl>,
#   q12_03a <dbl>, q12_03b <dbl>, q12_04a <dbl>, q12_04b <dbl>,
#   q12_05 <dbl+lbl>, q13_01 <dbl+lbl>, q13_01_ot <chr>, q13_02 <dbl+lbl>, …
```


IMPORTING DATA: SPSS

```r
spss_data <- read_sav("files/SUR_2023_LAPOP_AmericasBarometer_v1.0_w_orginal.sav",
                      user_na = TRUE)
spss_data2 <- read_sav("files/SUR_2023_LAPOP_AmericasBarometer_v1.0_w_orginal.sav")
head(spss_data)
```

```
# A tibble: 6 × 162
  idnum     pais          nationality     estratopri          estratosec  strata
  <dbl+lbl> <dbl+lbl>     <dbl+lbl>       <dbl+lbl>           <dbl+lbl>   <dbl+>
1 5581      27 [Suriname] 27 [Surinamese] 2702 [Wanica / Par… 2 [Medium … 2702
2 5642      27 [Suriname] 27 [Surinamese] 2701 [Paramaribo]   1 [Large (… 2701
3 4622      27 [Suriname] 27 [Surinamese] 2701 [Paramaribo]   1 [Large (… 2701
4 4034      27 [Suriname] 27 [Surinamese] 2702 [Wanica / Par… 2 [Medium … 2702
5 9206      27 [Suriname] 27 [Surinamese] 2701 [Paramaribo]   1 [Large (… 2701
6 2101      27 [Suriname] 27 [Surinamese] 2702 [Wanica / Par… 2 [Medium … 2702
# ℹ 156 more variables: prov <dbl+lbl>, municipio <dbl+lbl>, upm <dbl+lbl>,
#   ur <dbl+lbl>, cluster <dbl+lbl>, year <dbl+lbl>, wave <dbl+lbl>,
#   wt <dbl+lbl>, q1tc_r <dbl+lbl>, q2 <dbl+lbl>, a4n <dbl+lbl>,
#   soct2 <dbl+lbl>, idio2 <dbl+lbl>, mesfut1 <dbl+lbl>, cp8 <dbl+lbl>,
#   it1 <dbl+lbl>, jc10 <dbl+lbl>, jc13 <dbl+lbl>, jc15a <dbl+lbl>,
#   jc16a <dbl+lbl>, vic1ext <dbl+lbl>, aoj11 <dbl+lbl>, aoj12 <dbl+lbl>,
#   pese1 <dbl+lbl>, pese2 <dbl+lbl>, aoj17 <dbl+lbl>, ivol24 <dbl+lbl>, …
```


QUICK INSPECTION: MISSINGNESS

```
1 sum(is.na(stata_data)) # total number of NAs
```

```
[1] 31355
```

```
1 sum(is.na(spss_data)) # total number of NAs
```

```
[1] 28174
```

```
1 sum(is.na(spss_data2)) # total number of NAs
```

```
[1] 28174
```

LET'S LOOK AT GENDER

```
1 head(spss_data$q1tc_r, n = 20)
```

```
<labelled_spss<double>[20]>: Gender
 [1] 888888      2      1      2      1      2      2      1      1      2
[11]      1      2      1      2      1      2      1      1      1      2
Missing values: 888888, 988888, 999999

Labels:
  value                                  label
      1                                Man/male
      2                              Woman/female
      3 Does not identify as either man or woman
 888888                                     DK
 988888                                     NR
 999999                                    N/A
```

```
1 head(spss_data2$q1tc_r, n = 20)
```

```
<labelled<double>[20]>: Gender
 [1] NA  2  1  2  1  2  2  1  1  2  1  2  1  2  1  2  1  1  1  2

Labels:
  value                                  label
      1                                Man/male
      2                              Woman/female
      3 Does not identify as either man or woman
 888888                                     DK
 988888                                     NR
 999999                                    N/A
```

LET'S LOOK AT GENDER

```
1 table(spss_data$q1tc_r)
```

```
     1      2      3 888888 988888
   743    746      1     13     36
```

```
1 table(spss_data2$q1tc_r)
```

```
  1   2   3
743 746   1
```

CORRECTING THE MISSINGS

In base R, there is only one type of missing value: NA. In SPSS and Stata, there are multiple types of missing values. In R, we can use the haven package to convert these to NA.

```
spss_data <- zap_missing(spss_data) # set all special missing values to NA
```

LET'S LOOK AT GENDER AGAIN

```
table(spss_data$q1tc_r)
```

```
  1   2   3
743 746   1
```

```
table(spss_data2$q1tc_r)
```

```
  1   2   3
743 746   1
```

TAGGED MISSING VALUES

Alternatively, if you'd still like to use *special NAs*, you can use `haven::tag_na()`

```
x <- c(1:5, tagged_na("a"), tagged_na("z"), NA)

# Tagged NA's work identically to regular NAs
x
```

```
[1]  1  2  3  4  5 NA NA NA
```

```
is.na(x)
```

```
[1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
```


EXPLORING DATA SETS

```
1 glimpse(spss_data)
```

```
Rows: 1,539
Columns: 162
$ idnum       <dbl+lbl> 5581, 5642, 4622, 4034, 9206, 2101, 3574,  709, 8666, …
$ pais        <dbl+lbl> 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27…
$ nationality <dbl+lbl> 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27…
$ estratopri  <dbl+lbl> 2702, 2701, 2701, 2702, 2701, 2702, 2701, 2701, 2701, …
$ estratosec  <dbl+lbl> 2, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, …
$ strata      <dbl+lbl> 2702, 2701, 2701, 2702, 2701, 2702, 2701, 2701, 2701, …
$ prov        <dbl+lbl> 2702, 2701, 2701, 2702, 2701, 2702, 2701, 2701, 2701, …
$ municipio   <dbl+lbl> 270214, 270109, 270109, 270214, 270109, 270214, 270109…
$ upm         <dbl+lbl> 43, 21, 21, 43, 21, 43, 21, 21, 21, 21, 21, 21, 43, 21…
$ ur          <dbl+lbl> 2, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, …
$ cluster     <dbl+lbl>  87,  67,   5, 234,  67, 234, 233, 233,   5,  92,  67,…
$ year        <dbl+lbl> 2023, 2023, 2023, 2023, 2023, 2023, 2023, 2023, 2023, …
$ wave        <dbl+lbl> 2023, 2023, 2023, 2023, 2023, 2023, 2023, 2023, 2023, …
$ wt          <dbl+lbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
$ q1tc_r      <dbl+lbl> NA,  2,  1,  2,  1,  2,  2,  1,  1,  2,  1,  2,  1,  2…
$ q2          <dbl+lbl> 59, 61, 30, 36, 34, 53, 75, 27, 27, 50, 60, 34, 50, 30…
$ a4n         <dbl+lbl>  1,  1,  1,  1,  1,  1,  1,  1, 77,  1,  1,  1,  1, 77…
```

EXPLORING DATA SETS

```
1 glimpse(stata_data)
```

```
Rows: 2,502
Columns: 114
$ hhid          <dbl> 1102500401, 1102500501, 1102500502, 1102501202, 11074305…
$ domain2       <dbl+lbl> 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 2.0, 2.…
$ psu           <dbl> 10250, 10250, 10250, 10250, 10743, 10743, 10743, 10743, …
$ domain        <dbl+lbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2…
$ gp_subdom     <dbl+lbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0…
$ district      <chr> "Paramaribo", "Paramaribo", "Paramaribo", "Paramaribo", …
$ fortnight     <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,…
$ panel         <dbl+lbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
$ hhid16        <dbl> 6015041, 6015051, 6015052, 6015122, 6039051, 6039081, 60…
$ lat_cen       <dbl> 5.847621, 5.847621, 5.847621, 5.847621, 5.819147, 5.8191…
$ long_cen      <dbl> -55.17032, -55.17032, -55.17032, -55.17032, -55.21745, -…
$ result        <dbl+lbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
$ end_date_n    <date> 2022-01-04, 2022-01-05, 2022-01-10, 2022-01-04, 2022-01…
$ Year_s        <dbl> 2022, 2022, 2022, 2022, 2022, 2022, 2022, 2022, 2022, 20…
$ Month_s       <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,…
$ Day           <dbl> 4, 5, 10, 4, 5, 5, 5, 5, 5, 7, 7, 7, 7, 7, 7, 15, 9, 14,…
$ stratum       <dbl> 2, 2, 2, 2, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 6, 6, 6, 6,…
```

LABELS AND FACTORS

Currently, the `q1tc_r` variable is a numeric vector - even though the SPSS labels are still recorded. In R, we often use factors to represent categorical data. Factors are stored as integers with labels attached.

```
1 is.factor(spss_data$q1tc_r)
```

```
[1] FALSE
```

We can easily convert the `q1tc_r` variable to a factor using the `haven` package's `as_factor()` function, which will also preserve the labels.

```
1 spss_data <- as_factor(spss_data)
2 is.factor(spss_data$q1tc_r)
```

```
[1] TRUE
```


EXPLORING AGAIN - FACTORED

```
1 glimpse(as_factor(spss_data))
```

```
Rows: 1,539
Columns: 162
$ idnum       <fct> 5581, 5642, 4622, 4034, 9206, 2101, 3574, 709, 8666, 1566,…
$ pais        <fct> Suriname, Suriname, Suriname, Suriname, Suriname, Suriname…
$ nationality <fct> Surinamese, Surinamese, Surinamese, Surinamese, Surinamese…
$ estratopri  <fct> Wanica / Para, Paramaribo, Paramaribo, Wanica / Para, Para…
$ estratosec  <fct> "Medium (Between 3,000 and 10,000 inhabitants)", "Large (M…
$ strata      <fct> 2702, 2701, 2701, 2702, 2701, 2702, 2701, 2701, 2701, 2701…
$ prov        <fct> Wanica, Paramaribo, Paramaribo, Wanica, Paramaribo, Wanica…
$ municipio   <fct> Saramacca Polder, Flora, Flora, Saramacca Polder, Flora, S…
$ upm         <fct> 43, 21, 21, 43, 21, 43, 21, 21, 21, 21, 21, 21, 43, 21, 21…
$ ur          <fct> Rural, Urban, Urban, Rural, Urban, Rural, Urban, Urban, Ur…
$ cluster     <fct> 87, 67, 5, 234, 67, 234, 233, 233, 5, 92, 67, 233, 234, 5,…
$ year        <fct> 2023, 2023, 2023, 2023, 2023, 2023, 2023, 2023, 2023, 2023…
$ wave        <fct> 2023, 2023, 2023, 2023, 2023, 2023, 2023, 2023, 2023, 2023…
$ wt          <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
$ q1tc_r      <fct> NA, Woman/female, Man/male, Woman/female, Man/male, Woman/…
$ q2          <fct> 59, 61, 30, 36, 34, 53, 75, 27, 27, 50, 60, 34, 50, 30, 18…
$ a4n         <fct> "Economic issues", "Economic issues", "Economic issues", "…
```


EXPLORING AGAIN - FACTORED

```
1 glimpse(as_factor(stata_data))
```

```
Rows: 2,502
Columns: 114
$ hhid          <dbl> 1102500401, 1102500501, 1102500502, 1102501202, 11074305…
$ domain2       <fct> 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, Rest of the…
$ psu           <dbl> 10250, 10250, 10250, 10250, 10743, 10743, 10743, 10743, …
$ domain        <fct> Great Paramaribo, Great Paramaribo, Great Paramaribo, Gr…
$ gp_subdom     <fct> Paramaribo, Paramaribo, Paramaribo, Paramaribo, Paramari…
$ district      <chr> "Paramaribo", "Paramaribo", "Paramaribo", "Paramaribo", …
$ fortnight     <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,…
$ panel         <fct> Panel, Panel, Panel, Panel, Panel, Panel, Panel, Panel, …
$ hhid16        <dbl> 6015041, 6015051, 6015052, 6015122, 6039051, 6039081, 60…
$ lat_cen       <dbl> 5.847621, 5.847621, 5.847621, 5.847621, 5.819147, 5.8191…
$ long_cen      <dbl> -55.17032, -55.17032, -55.17032, -55.17032, -55.21745, -…
$ result        <fct> Interview finalized - Fully completed, Interview finaliz…
$ end_date_n    <date> 2022-01-04, 2022-01-05, 2022-01-10, 2022-01-04, 2022-01…
$ Year_s        <dbl> 2022, 2022, 2022, 2022, 2022, 2022, 2022, 2022, 2022, 20…
$ Month_s       <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,…
$ Day           <dbl> 4, 5, 10, 4, 5, 5, 5, 5, 5, 7, 7, 7, 7, 7, 7, 15, 9, 14,…
$ stratum       <dbl> 2, 2, 2, 2, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 6, 6, 6, 6,…
```


PIPES

Pipes are a way to chain together multiple operations in a more readable way. The pipe operator %>% takes the output of the left-hand side and passes it as the first argument to the function on the right-hand side. In R, there is now also the |> operator, which is a base R pipe.

Remember

```
glimpse(as_factor(stata_data))
```

With a pipe this would be

```
spss_data %>%
  as_factor() %>%
  glimpse()
```

and with the base R pipe |> this would be

```
spss_data |>
  as_factor() |>
  glimpse()
```

```
Rows: 1,539
Columns: 162
$ idnum       <fct> 5581, 5642, 4622, 4034, 9206, 2101, 3574, 709, 8666, 1566,…
$ pais        <fct> Suriname, Suriname, Suriname, Suriname, Suriname, Suriname…
$ nationality <fct> Surinamese, Surinamese, Surinamese, Surinamese, Surinamese…
$ estratopri  <fct> Wanica / Para, Paramaribo, Paramaribo, Wanica / Para, Para…
```


```
$ estratosec  <fct> "Medium (Between 3,000 and 10,000 inhabitants)", "Large (M…
$ strata      <fct> 2702, 2701, 2701, 2702, 2701, 2702, 2701, 2701, 2701, 2701…
$ prov        <fct> Wanica, Paramaribo, Paramaribo, Wanica, Paramaribo, Wanica…
$ municipio   <fct> Saramacca Polder, Flora, Flora, Saramacca Polder, Flora, S…
$ upm         <fct> 43, 21, 21, 43, 21, 43, 21, 21, 21, 21, 21, 21, 43, 21, 21…
$ ur          <fct> Rural, Urban, Urban, Rural, Urban, Rural, Urban, Urban, Ur…
$ cluster     <fct> 87, 67, 5, 234, 67, 234, 233, 233, 5, 92, 67, 233, 234, 5,…
$ year        <fct> 2023, 2023, 2023, 2023, 2023, 2023, 2023, 2023, 2023, 2023…
$ wave        <fct> 2023, 2023, 2023, 2023, 2023, 2023, 2023, 2023, 2023, 2023…
$ wt          <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
$ q1tc_r      <fct> NA, Woman/female, Man/male, Woman/female, Man/male, Woman/…
$ q2          <fct> 59, 61, 30, 36, 34, 53, 75, 27, 27, 50, 60, 34, 50, 30, 18…
$ a4n         <fct> "Economic issues", "Economic issues", "Economic issues", "…
```


PIPES IN DETAIL

- `%>%` is the pipe operator from the `magrittr` package
- `|>` is the base R pipe operator introduced in R 4.1.0

Both operators allow you to chain together multiple operations in a more readable way

A pipe is a way to pass the output of one function as the input to another function, without having to create intermediate variables.

`A %>% B` is equivalent to `B(A)`, where `A` is the output of the left-hand side and `B` is the function on the right-hand side. `A` is expected to be the first argument in function `B`.

> ⚠ **Warning**
>
> The next step in a pipe is always expected to be a function! If the next step is not a function, you need to be clever, otherwise you'll get an error.

WHAT IF THE NEXT STEP IS NOT THE FIRST ARGUMENT?

You can use the **placeholder** `.` (or `_` with the native R pipe) to indicate where the output of the previous step should go in the next function.

For example,

```
stata_data %>%
  filter(q13_05 == 1 | q13_05 == 2) %>%
  t.test(Year_s ~ q13_05, data = .)
```

```
    Welch Two Sample t-test

data:  Year_s by q13_05
t = -1, df = 1429, p-value = 0.3175
alternative hypothesis: true difference in means between group 1 and group 2 is not equal to 0
95 percent confidence interval:
 -0.0020710668  0.0006724654
sample estimates:
mean in group 1 mean in group 2
       2022.000        2022.001
```


WHAT IF THE NEXT STEP IS NOT THE FIRST ARGUMENT?

You can use the **placeholder** `.` (or `_` with the native R pipe) to indicate where the output of the previous step should go in the next function.

For example,

```
stata_data %>%
  filter(q13_05 == 1 | q13_05 == 2) |>
  t.test(Year_s ~ q13_05, data = _)
```

```
    Welch Two Sample t-test

data:  Year_s by q13_05
t = -1, df = 1429, p-value = 0.3175
alternative hypothesis: true difference in means between group 1 and group 2 is not equal to 0
95 percent confidence interval:
 -0.0020710668  0.0006724654
sample estimates:
mean in group 1 mean in group 2
       2022.000        2022.001
```

OTHER PIPES

- `%$%` is the exposition pipe from the `magrittr` package, which allows you to use the names of the variables in the data frame directly without having to use the `$` operator.
- `%<>%` is the assignment pipe from the `magrittr` package, which allows you to modify the data frame in place, without the need for calling `assign()` or `<-` again.

There are more pipes (like the `%T>%` pipe), but they can be very confusing and we therefore skip them in this course.

```
1 stata_data %>%
2   filter(q13_05 == 1 | q13_05 == 2) %$% # Note the exposition pipe
3   t.test(Year_s ~ q13_05)
```

```
    Welch Two Sample t-test

data:  Year_s by q13_05
t = -1, df = 1429, p-value = 0.3175
alternative hypothesis: true difference in means between group 1 and group 2 is not equal to 0
95 percent confidence interval:
 -0.0020710668  0.0006724654
sample estimates:
mean in group 1 mean in group 2
       2022.000        2022.001
```

RENAMING VARIABLES WITH A PIPE

```
spss_data <- spss_data %>%
  rename(gender = q1tc_r)
```

```
spss_data <- spss_data %<>%
  rename(gender = q1tc_r)

spss_data$gender %>% head()
```

```
[1] <NA>          Woman/female Man/male     Woman/female Man/male
[6] Woman/female
6 Levels: Man/male Woman/female ... N/A
```

RECODING WITH `mutate()` EN `recode()`

```
spss_data <- spss_data %>%
  mutate(gender_rec = recode(gender,
                             "Man/male" = "male",
                             "Woman/female" = "female"))

spss_data$gender %>% head()
```

```
[1] <NA>         Woman/female Man/male     Woman/female Man/male
[6] Woman/female
6 Levels: Man/male Woman/female ... N/A
```

```
spss_data$gender_rec %>% head()
```

```
[1] <NA>   female male   female male   female
Levels: male female Does not identify as either man or woman DK NR N/A
```

LABELING VARIABLES

With labeled variables, we can add an additional layer of description to variable, very similar to what SPSS and STATA do.

```
spss_data <- set_variable_labels(spss_data, gender_rec = "Gerecodeerd geslacht")
spss_data$gender_rec %>% glimpse()
```

```
Factor w/ 6 levels "male","female",..: NA 2 1 2 1 2 2 1 1 2 ...
- attr(*, "label")= chr "Gerecodeerd geslacht"
```

SELECTING AND FILTERING

```
spss_data %<>%
  rename(age = q2)

spss_data %>%
  filter(age > 18) %>%
  select(age, gender) %>%
  summary()
```

```
      age                                              gender
 18     :0   Man/male                                    :0
 19     :0   Woman/female                                :0
 20     :0   Does not identify as either man or woman:0
 21     :0   DK                                          :0
 22     :0   NR                                          :0
 23     :0   N/A                                         :0
 (Other):0
```


CALCULATIONS AND SUMMARISING

```
spss_data %>%
  filter(age > 18) %>%
  summarise(mean_age = mean(age, na.rm = TRUE))
```

```
# A tibble: 1 × 1
  mean_age
     <dbl>
1       NA
```

`age` is also a factor, hence `mean()` is meaningless. We have to convert `age` to numeric:

```
spss_data %>%
  mutate(age = as.numeric(age)) %>%
  filter(age > 18) %>%
  summarise(mean_age = mean(age, na.rm = TRUE))
```

```
# A tibble: 1 × 1
  mean_age
     <dbl>
1     35.6
```

MODELING IN R

To model objects based on other objects, we use ~ (tilde)

For example, to model body mass index (BMI) on weight, we would type

```
1  BMI ~ weight
```

The ~ is used to separate the left- and right-hand sides in a model **formula**.

For functions (or models), within models we use `I()` - For example, to model body mass index (BMI) on its deterministic function of weight and height, we would type

```
1  BMI ~ I(weight / height^2)
```


MODELING CONTINUED

We already saw the use of the `~` operator in the `t.test()` function, where we specified the outcome variable on the left-hand side and the grouping variable on the right-hand side.

```
stata_data %>%
  filter(q13_05 == 1 | q13_05 == 2) %$%
  t.test(Year_s ~ q13_05)
```

USING FORMULAS

```
1 # Use a formula in function lm() in a pipe
2 mtcars %>%
3   lm(mpg ~ wt + hp, data = .)
```

```
Call:
lm(formula = mpg ~ wt + hp, data = .)

Coefficients:
(Intercept)           wt           hp
   37.22727     -3.87783     -0.03177
```

USING FORMULAS WITH broom

With the broom package, we can easily tidy up the output of models and other functions that return complex objects.

```
mtcars %>%
  lm(mpg ~ wt + hp, data = .) %>%
  tidy()
```

```
# A tibble: 3 × 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)  37.2      1.60         23.3  2.57e-20
2 wt           -3.88     0.633        -6.13 1.12e- 6
3 hp           -0.0318   0.00903      -3.52 1.45e- 3
```


TABLES AND CONTINGENCY TABLES

```
stata_data %$%
  table(district)
```

```
district
Brokopondo Commewijne    Coronie  Marowijne   Nickerie       Para Paramaribo
         7        170         33         75        213         27        945
 Saramacca Sipaliwini     Wanica
       233        118        681
```

```
stata_data %$%
  table(district, stratum)
```

```
            stratum
district       1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
  Brokopondo   0   0   0   0   0   0   0   0   4   0   0   0   0   0   0   3
  Commewijne   0   0   0   0   0  46   0   0   0 124   0   0   0   0   0   0
  Coronie      0   0   0  33   0   0   0   0   0   0   0   0   0   0   0   0
  Marowijne    0   0   0   0   0  47   0   0   0   0   0   0   0   0   0  28
  Nickerie     0   0   0 213   0   0   0   0   0   0   0   0   0   0   0   0
  Para         0   0   0   0   0   0   0   0  22   0   0   0   0   0   0   5
  Paramaribo 150 147 112   0 165   0   0   0   0   0 120 129 122   0   0   0
  Saramacca    0   0   0   0   0   0   0 182  51   0   0   0   0   0   0   0
  Sipaliwini   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 118
  Wanica       0   0   0   0   0   0 233   0  57   0   0   0   0 164 227   0
```

TABLES

```
stata_data %$%
  table(district, stratum) %>% # calculate table
  prop.table() %>% # convert to proportions
  round(3) # round to 3 decimals
```

```
            stratum
district          1     2     3     4     5     6     7     8     9    10    11
  Brokopondo  0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.002 0.000 0.000
  Commewijne  0.000 0.000 0.000 0.000 0.000 0.018 0.000 0.000 0.000 0.050 0.000
  Coronie     0.000 0.000 0.000 0.013 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  Marowijne   0.000 0.000 0.000 0.000 0.000 0.019 0.000 0.000 0.000 0.000 0.000
  Nickerie    0.000 0.000 0.000 0.085 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  Para        0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.009 0.000 0.000
  Paramaribo  0.060 0.059 0.045 0.000 0.066 0.000 0.000 0.000 0.000 0.000 0.048
  Saramacca   0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.073 0.020 0.000 0.000
  Sipaliwini  0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
  Wanica      0.000 0.000 0.000 0.000 0.000 0.000 0.093 0.000 0.023 0.000 0.000
            stratum
district         12    13    14    15    16
  Brokopondo  0.000 0.000 0.000 0.000 0.001
  Commewijne  0.000 0.000 0.000 0.000 0.000
  Coronie     0.000 0.000 0.000 0.000 0.000
  Marowijne   0.000 0.000 0.000 0.000 0.011
  Nickerie    0.000 0.000 0.000 0.000 0.000
```

TIBBLES VS DATA.FRAMES

Tibbles are a modern re-imagining of data frames in R. They are part of the `tidyverse` and provide a more user-friendly interface for working with data.

```
1 is_tibble(band_members)
```

```
[1] TRUE
```

```
1 band_members
```

```
# A tibble: 3 × 2
  name  band
  <chr> <chr>
1 Mick  Stones
2 John  Beatles
3 Paul  Beatles
```

```
1 band_members %>%
2   as.data.frame()
```

```
  name    band
1 Mick  Stones
2 John Beatles
3 Paul Beatles
```

LAYERS IN R

There are several ‘layers’ in R. Some layers you are allowed to fiddle around in, some are forbidden. In general there is the following distinction:

- The global environment.
- User environments
- Functions
- Packages
- Namespaces

ENVIRONMENTS

The global environment can be seen as an olympic-size swimming pool. Everything you do has its place there.

If you'd like, you may create another, separate environment to work in.

- A user environment would by default not have access to other environments

FUNCTIONS

- If you create a function, it is positioned in the global environment.
- Everything that happens in a function, stays in a function. Unless you specifically tell the function to share the information with the global environment.
- See functions as a shampoo bottle in a swimming pool to which you add some water. If you'd like to see the color of the mixture, you'd have to squeeze the bottle for it to come out.

PACKAGE AND NAMESPACES

Namespaces are a way to organize functions and data in R. Every package has its own namespace, which means that functions and data in one package do not interfere with functions and data in another package.

- Everything needed to run the functions in a package is neatly contained within its own space
- See packages as separate (mini) pools that are connected to the main pool (the global environment)

%IN%

The `%in%` operator is used to check if elements of one vector are present in another vector. It returns a logical vector indicating whether each element of the first vector is found in the second vector.

```
x <- c(1, 2, 3, 4, 5)
y <- c(3, 4, 5, 6, 7)
x %in% y
```

```
[1] FALSE FALSE  TRUE  TRUE  TRUE
```

grepl()

The `grepl()` function is used to search for a pattern in a character vector. It returns a logical vector indicating whether the pattern is found in each element of the character vector.

```
1 x <- c("apple", "banana", "cherry", "date")
2 pattern <- "a"
3 grepl(pattern, x)
```

```
[1]  TRUE  TRUE FALSE  TRUE
```

```
1 grepl("^a", x) # starts with a
```

```
[1]  TRUE FALSE FALSE FALSE
```

```
1 grepl("e$", x) # ends with e
```

```
[1]  TRUE FALSE FALSE  TRUE
```

```
1 grepl("cherry", x)
```

```
[1] FALSE FALSE  TRUE FALSE
```

ANSCOMBE DATA

```
anscombe |>
  as_tibble()
```

```
# A tibble: 11 × 8
      x1    x2    x3    x4    y1    y2    y3    y4
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1    10    10    10     8  8.04  9.14  7.46  6.58
 2     8     8     8     8  6.95  8.14  6.77  5.76
 3    13    13    13     8  7.58  8.74 12.7   7.71
 4     9     9     9     8  8.81  8.77  7.11  8.84
 5    11    11    11     8  8.33  9.26  7.81  8.47
 6    14    14    14     8  9.96  8.1   8.84  7.04
 7     6     6     6     8  7.24  6.13  6.08  5.25
 8     4     4     4    19  4.26  3.1   5.39 12.5
 9    12    12    12     8 10.8   9.13  8.15  5.56
10     7     7     7     8  4.82  7.26  6.42  7.91
11     5     5     5     8  5.68  4.74  5.73  6.89
```


ANSCOMBE DATA PROPERTIES

```
anscombe |>
  cor() |>
  round(2)
```

```
      x1    x2    x3    x4    y1    y2    y3    y4
x1  1.00  1.00  1.00 -0.50  0.82  0.82  0.82 -0.31
x2  1.00  1.00  1.00 -0.50  0.82  0.82  0.82 -0.31
x3  1.00  1.00  1.00 -0.50  0.82  0.82  0.82 -0.31
x4 -0.50 -0.50 -0.50  1.00 -0.53 -0.72 -0.34  0.82
y1  0.82  0.82  0.82 -0.53  1.00  0.75  0.47 -0.49
y2  0.82  0.82  0.82 -0.72  0.75  1.00  0.59 -0.48
y3  0.82  0.82  0.82 -0.34  0.47  0.59  1.00 -0.16
y4 -0.31 -0.31 -0.31  0.82 -0.49 -0.48 -0.16  1.00
```


ADDING SOME RANDOM NUMBERS

```
anscombe_new <-
  anscombe |>
  mutate(x1 = x1 + rnorm(nrow(anscombe), mean = 0, sd = 0.1),
         x2 = x2 + rnorm(nrow(anscombe), mean = 0, sd = 0.1),
         x3 = x3 + rnorm(nrow(anscombe), mean = 0, sd = 0.1),
         x4 = x4 + rnorm(nrow(anscombe), mean = 0, sd = 0.1)) |>
  as_tibble()
```

anscombe_new DATA PROPERTIES

```
1 anscombe_new |>
2   cor() |>
3   round(2)
```

```
      x1    x2    x3    x4    y1    y2    y3    y4
x1  1.00  1.00  1.00 -0.51  0.82  0.82  0.82 -0.32
x2  1.00  1.00  1.00 -0.50  0.81  0.81  0.81 -0.31
x3  1.00  1.00  1.00 -0.50  0.81  0.82  0.81 -0.31
x4 -0.51 -0.50 -0.50  1.00 -0.53 -0.72 -0.35  0.81
y1  0.82  0.81  0.81 -0.53  1.00  0.75  0.47 -0.49
y2  0.82  0.81  0.82 -0.72  0.75  1.00  0.59 -0.48
y3  0.82  0.81  0.81 -0.35  0.47  0.59  1.00 -0.16
y4 -0.32 -0.31 -0.31  0.81 -0.49 -0.48 -0.16  1.00
```


PRACTICAL