Tidy Cooking

Recipes for Data Management in R

Frank Loesche

22 February 2018

Tidy what?

- tidyverse is set of R packages
- Wickham (2014) suggests data format that acknowledges "both statistical and cognitive factors"
- each observational unit: 1 table
- each observation: 1 row
- each variable: 1 column

Goal

- every object is a data frame (`tibble`)
- common set of matching tools to manipulate
- Reduce "mundane data manipulation chores" (Wickham, 2014)

Advantages

Think about languages:

- similar structure allows quick and easy communication (S-V-O)
- Grammar (English): S-V-O
- Example: "Frank uses tidyverse."

Think about tools:

- same structure allows easy maintenance
- "Philosophy": One type of screw head per machine

Figure 1: Simple tool

Tidyverse is a tool, R a language

- based on Wilkinson (2005, p. 23): `[Source] → (make a graph) → [Renderer]`
- Philosophy / Grammar: **input** %>% **verb** -> **result**
- or **result** <- **input** %>% **verb**
- Example:

```
task_one <-
  df %>%
  filter(task == 1)
```

- **input**: `df`, **verb**: `filter()`, **result**: `task_one`

Suggestion

- make `library(tidyverse)` a habit
- use tidyverse to describe you analysis
  - don't think of it as "programming"
  - think in sentences: "From dataset df filter the rows where task is equal to 1 and store the result in task_one"

Elements to construct descriptions

Tibble

- more comfortable `data.frame`

Pipe

- `%>%`: the pipe. Take output from the left side, use it as input on the right side.
  - Example:

```
task_one %>%
  filter(reaction_time < 22) %>%
  print()
```

```
## # A tibble: 1 x 3
##   participant_id  task reaction_time
##   <chr>          <dbl>         <dbl>
## 1 pid-02          1.00          21.5
```

Figure 2: Magritte

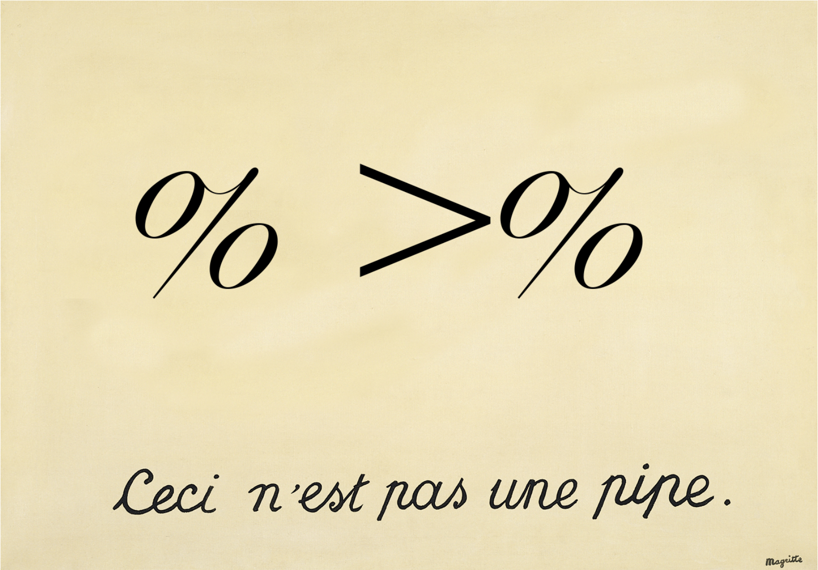


Figure 3: tidyverse magrittr::

Verbs

- every tidyverse packages has them
- readr:: `write_csv()`, `read_csv()`,`read_rds()`
- examples for dplyr:: `filter()`, `select()`, `mutate()`, `slice()`, `distinct()`, `summarise()`, `group_by()`, `left_join()`, `rownames_to_columns()`
- ggplot2:: `ggplot()`
- modelr:: `add_predictors()`, `add_residuals()`
- broom:: `glance()`, `tidy()`
- write your own

Construction of descriptions

- tidyverse offers many different **verbs**
- any number of **verbs** in one description

```
task_one <-
  df %>%
  filter(task == 1) %>%
  filter(participant_id == 'pid-01') %>%
  filter(reaction_time < 25)
```

- but: try to construct descriptions that
  - make sense (“statistically”) and
  - are readable (“cognitively”)

Workflow

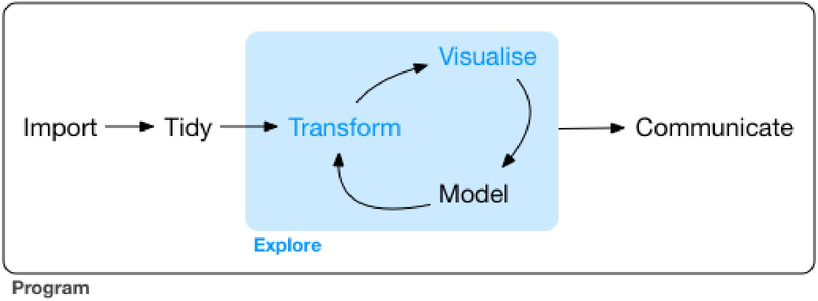


Figure 4: Workflow for data exploration[1]

- Data from
  - 'Just enough R'
  - Stuart's robot_club

[1] from *R for Data Science*, http://r4ds.had.co.nz

Import

```
fn_e13 <- 'e13.csv'
# other sources, eg https://osf.io/66fvm/download
# https://zenodo.org/record/...
url_exp13test %>% str_trunc(35)
```

```
## [1] "https://github.com/sspicer/robot..."
```

download file

```
url_exp13test %>%
  download.file(fn_e13)
```

open file

```
e13 <-
  fn_e13 %>%
    read_csv()
```

open URL

```
e13 <-
  url_exp13test %>%
    read_csv()
```

```r
e13
```

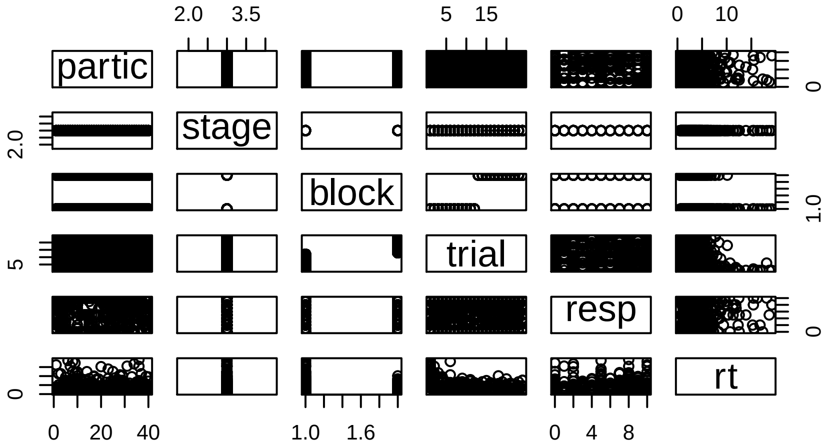
```
## # A tibble: 960 x 11
##   partic stage block trial stim   resp    rt
##    <int> <int> <int> <int> <chr> <int> <dbl>
## 1      1     3     1     1 Z        10 16.2
## 2      1     3     1     2 WC        7  3.78
## # ... with 958 more rows, and 4 more variables:
## #   fruit1 <chr>, fruit2 <chr>, outcome <chr>,
## #   date <chr>
```

```r
# e13 %>% print()
```

```r
e13 %>% glimpse()
```

```
## Observations: 960
## Variables: 11
## $ partic  <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
## $ stage   <int> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,...
## $ block   <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
## $ trial   <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10...
## $ stim    <chr> "Z", "WC", "XF", "F", "Y", "B...
## $ resp    <int> 10, 7, 5, 1, 1, 8, 8, 2, 10, ...
## $ rt      <dbl> 16.209, 3.777, 2.943, 1.638, ...
## $ fruit1  <chr> "apple", "banana", "orange", ...
## $ fruit2  <chr> NA, "kiwi", "plum", NA, NA, N...
## $ outcome <chr> NA, NA, NA, NA, NA, NA, NA, N...
## $ date    <chr> "2017_Sep_27_0935", "2017_Sep...
```

```
e13 %>%
  select(partic, stage, block, trial, resp, rt) %>%
  pairs()
```



Verbs for file types

- tabular data (with `readr::`)
  - csv: `read_csv()`, tsv: `read_tsv()`, fixed width: `read_fwf()`, webserver log files: `read_log()`
- Microsoft Excel (with `library(readxl)`)
  - xls and xlsx: `read_excel()`
  - select sheet: `read_excel(sheet="Raw Data")`, or `read_excel(sheet=3_)`
- Other (with `library(haven)`)
  - **SPSS** sav: `read_sav()`, por: `read_por()`
  - **SAS** xpt: `read_xpt()`, cat+bat: `read_sas()`
  - **Stat** dta: `read_dta()`

Recipe 1: Open a file

Ingredients

- file location `fn`:
  - URL or file name
- file type
  - select *verb* `read_*()`, eg `read_csv()`

Expected outcome

tibble `raw_content` with raw file content

Method

```
raw_content <-
  fn %>%
    read_csv()
```

string manipulation

- mutate(): create new variable for each observation

```
all_files <-
  tibble(
    fn = paste0("person", 1:10, ".csv")
  )
url_jer <- "https://github.com/benwhalley/just-enough-r/"
path_mf <- "raw/master/data/multiple-file-example/"
path_local <- "data/"

all_files <-
  all_files %>%
    mutate(
      url = paste0(url_jer, path_mf, fn),
      local = paste0(path_local, fn)
    )
```

Download all files

- `library(fs)`: interact with filesystem
- `rowwise()`: group data by row
- `do()`: apply function (most generic verb)
- `.`: current observation, `$`: access variable

```
fs::dir_create(path_local)

all_files %>%
  rowwise() %>%
  do(., download.file(.$url, .$local))
```

Recipe 2: Open many files

Ingredients

- tibble `all_files`
  - 1 file per line
  - file names `$local` or URLs `$url`
- files with the same structure
  - define column `person` as factorial data

Expected outcome

- tibble `rt_data` with all observations

Method

```
col_def <- list(person = col_factor(c(1:10)))
rt_data <-
  all_files %>%
    rowwise() %>%
    do(., read_csv(.$local, col_types = col_def))
```

Create toy data

- `select()`: select variable(s)
- `distinct()`: get unique observations
- `n_distinct()`: count unique observations

```
demographics <-
  rt_data %>%
  select(person) %>%
  distinct() %>%
  mutate(
    age = sample(21:25,1),
    handednenss = sample(c("Left", "Right"),1)
  )
```

Merge data

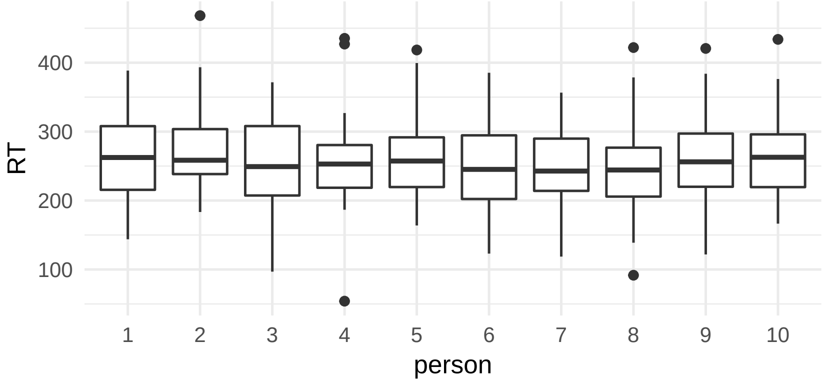
```
rt_dem_data <-
  rt_data %>%
    left_join(demographics, by = c("person"))

rt_dem_data %>%
  glimpse()
```

```
## Observations: 500
## Variables: 7
## $ Condition   <int> 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ trial       <int> 1, 2, 3, 4, 5, 6, 7, 8, 9...
## $ time        <int> 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ person      <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ RT          <dbl> 284.5, 309.3, 346.7, 291....
## $ age         <int> 21, 21, 21, 21, 21, 21, 2...
## $ handednenss <chr> "Left", "Left", "Left", "...
```

Plot data

- ggplot(): tidy way of plotting data
  - described in Wickham (2010)

```
rt_data %>%
  ggplot(aes(person, RT)) +
  geom_boxplot() +
  theme_minimal()
```



Transform

Sorting

- `arrange()`: sort ascending or `desc()`ending

```
rt_data %>%
  arrange(desc(time), trial, person)
```

```
## # A tibble: 500 x 5
##   Condition trial  time person     RT
##       <int> <int> <int> <fct>   <dbl>
## 1         1     1     2 1         368
## 2         1     1     2 2         247
## # ... with 498 more rows
```

extract observations

- `slice()`: by row position
- `sample_frac()`: sample a subset

```
rt_data %>%
  sample_frac(.3) %>%
  arrange(RT) %>%
  slice(1:3)
```

```
## # A tibble: 3 x 5
##   Condition trial  time person    RT
##       <int> <int> <int> <fct>  <dbl>
## 1         1    12     2 7        119
## 2         1     8     2 3        129
## 3         1     2     2 7        132
```

Group

- `group_by()`: manipulate each group separately
- use `ungroup()` to remove all groups

```
rt_dem_data %>%
  group_by(trial) %>%
  summarise(
    mean_rt = mean(RT),
    count = n())
```

```
## # A tibble: 25 x 3
##   trial mean_rt count
##   <int>   <dbl> <int>
## 1     1     276    20
## 2     2     272    20
## # ... with 23 more rows
```

Structural changes: spreading

Lets assume the RT for 1st and 2nd time are considered to be part of the same observation.

- `spread()`: spread key & value into columns
- `rename()`: change column names

```
rt12_dem <-
  rt_dem_data %>%
    spread(key = time, value = RT) %>%
    rename(RT1 = `1`, RT2 = `2`)
rt12_dem
```

```
## # A tibble: 250 x 7
##   Condition trial person   age handedness    RT1
##       <int> <int> <fct>  <int> <chr>       <dbl>
## 1         1     1 1         21 Left          285
## 2         1     1 2         21 Right         235
## # ... with 248 more rows, and 1 more variable:
## #   RT2 <dbl>
```

Structural changes: gathering

- `gather()`: columns to key-value pairs
- `parse_number()`: extract numbers from strings

```
rt12_dem %>%
  gather(repetition, reaction_time, RT1:RT2) %>%
  mutate(rep = parse_number(repetition)) %>%
  glimpse()
```

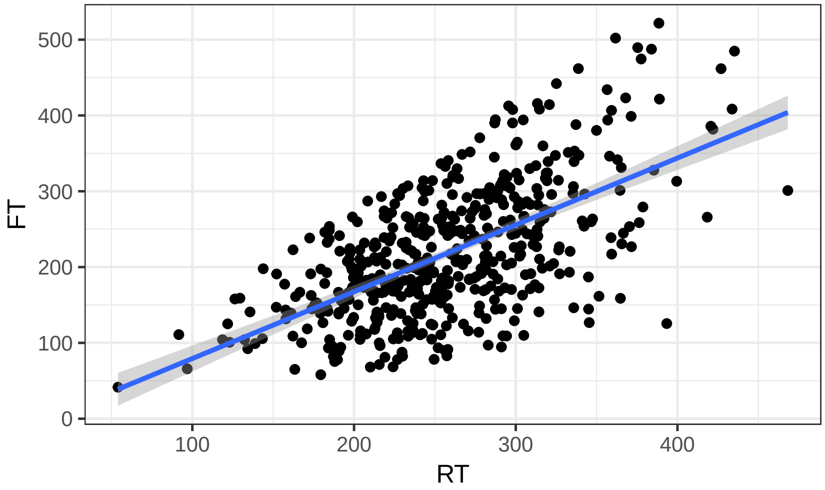
```
## Observations: 500
## Variables: 8
## $ Condition     <int> 1, 1, 1, 1, 1, 1, 1, 1,...
## $ trial         <int> 1, 1, 1, 1, 1, 1, 1, 1,...
## $ person        <fct> 1, 2, 3, 4, 5, 6, 7, 8,...
## $ age           <int> 21, 21, 21, 24, 25, 25,...
## $ handednenss   <chr> "Left", "Right", "Left"...
## $ repetition    <chr> "RT1", "RT1", "RT1", "R...
## $ reaction_time <dbl> 284.5, 235.0, 358.9, 23...
## $ rep           <dbl> 1, 1, 1, 1, 1, 1, 1, 1,...
```

Create toy data

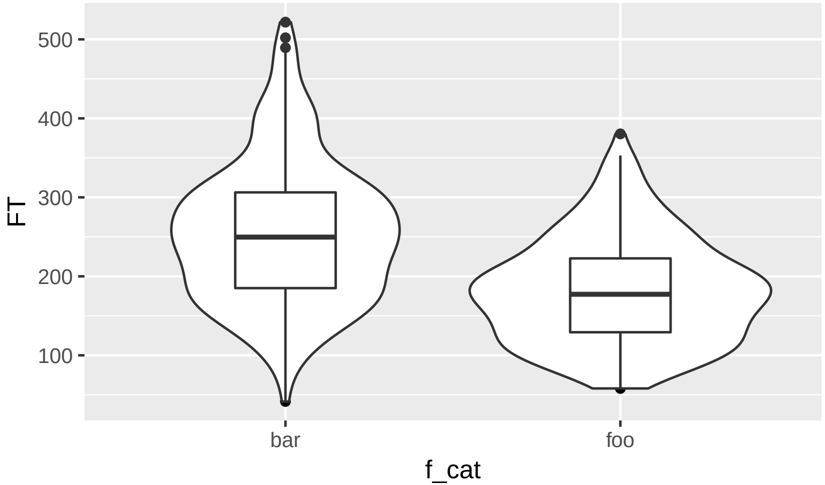
- `case_when()`: vectorized if else

```
s_dat <-
  rt_data %>%
    mutate(FT = (RT * .3 * time) + runif(1, 0, RT*.8),
           f_cat = case_when(
             time == 1 ~ "foo",
             TRUE      ~ "bar" ))
```

```r
s_dat %>%
  ggplot(aes(RT, FT)) +
    geom_point() +
    geom_smooth(method = 'lm') + theme_bw()
```



```r
s_dat %>%
  ggplot(aes(f_cat, FT)) +
    geom_point() +
    geom_violin() +
    geom_boxplot(width = .3)
```



Inferential statistics as data

- `library(broom)`: convert analysis objects to tibbles
- `tidy()`: test to summary table

```
attach(s_dat)
s_stat <-
  bind_rows(
    t.test(FT ~ f_cat) %>% broom::tidy(),
    t.test(RT ~ f_cat) %>% broom::tidy(),
    wilcox.test(FT ~ f_cat) %>% broom::tidy(),
    wilcox.test(RT ~ f_cat) %>% broom::tidy(),
    cor.test(FT, time) %>% broom::tidy(),
    cor.test(RT, time) %>% broom::tidy()
  )
detach(s_dat)
```

- `select()` to reorder
- `everything()` selects all variables

```
s_stat %>%
  select(method, p.value, everything()) %>%
  filter(p.value < 0.01) %>% glimpse()
```

```
## Observations: 3
## Variables: 10
## $ method      <chr> "Welch Two Sample t-test"...
## $ p.value     <dbl> 8.240e-23, 1.174e-20, 5.7...
## $ estimate    <dbl> 73.4111, NA, 0.4216
## $ estimate1   <dbl> 255.2, NA, NA
## $ estimate2   <dbl> 181.8, NA, NA
## $ statistic   <dbl> 10.38, 46304.00, 10.38
## $ parameter   <dbl> 460.8, NA, 498.0
## $ conf.low    <dbl> 59.5083, NA, 0.3468
## $ conf.high   <dbl> 87.3139, NA, 0.4912
## $ alternative <fct> two.sided, two.sided, two...
```

define toy models

```r
s_model1 <- lm(FT ~ RT, data = s_dat)
s_model2 <- lm(FT ~ RT + f_cat, data = s_dat)
s_model3 <- lm(FT ~ RT * f_cat, data = s_dat)
```

- library(modelr): modelling for the pipe
- add_predictions() and add_residuals() per observation

```
s_dat %>%
  add_predictions(s_model1, "pred1") %>%
  add_residuals(s_model1, "res1")
```

```
## Source: local data frame [500 x 5]
## Groups: <by row>
##
## # A tibble: 500 x 5
##   trial    RT    FT pred1  res1
##   <int> <dbl> <dbl> <dbl> <dbl>
## 1     1   285   175   242 -67.0
## 2     2   309   191   264 -72.5
## # ... with 498 more rows
```

- `augment()`: create predictions, residuals etc
- `augement_columns()`: add to existing data

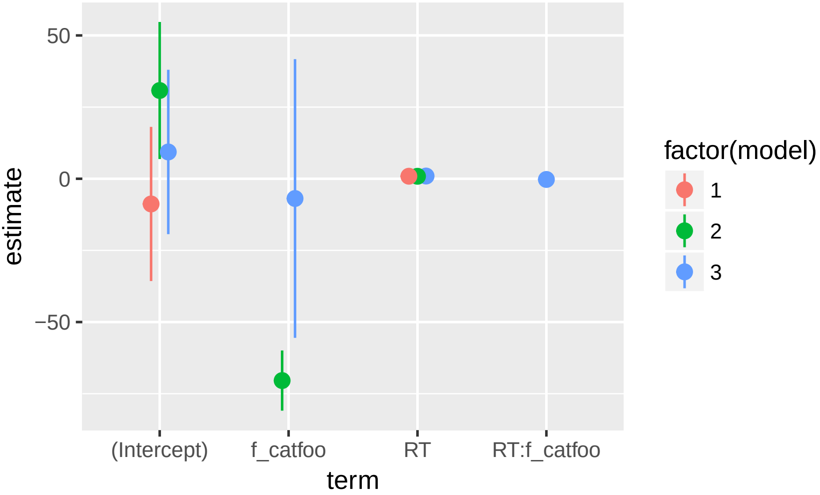
```
s_model1 %>%
  augment_columns(rt_data)
```

```
## # A tibble: 500 x 12
##   Condition person .fitted .resid trial  time
##       <int> <fct>    <dbl>  <dbl> <int> <int>
## 1         1 1          242  -67.0     1     1
## 2         1 1          264  -72.5     2     1
## # ... with 498 more rows, and 6 more variables:
## #   RT <dbl>, .se.fit <dbl>, .hat <dbl>,
## #   .sigma <dbl>, .cooksd <dbl>, .std.resid <dbl>
```

Plot model parameters

```
model_comp <-
  bind_rows(
    s_model1 %>% tidy(conf.int = T) %>%
      mutate(model = 1),
    s_model2 %>% tidy(conf.int = T) %>%
      mutate(model = 2),
    s_model3 %>% tidy(conf.int = T) %>%
      mutate(model = 3))
```

```r
model_comp %>%
  ggplot(aes(term, estimate,
             ymin = conf.low, ymax = conf.high,
             colour = factor(model))) +
  geom_pointrange(position = position_dodge(width = .2))
```



Prepare data for model comparison

```r
bind_rows(
  s_model1 %>% glance(),
  s_model2 %>% glance(),
  s_model3 %>% glance()
)
```

```
## # A tibble: 3 x 11
##   r.squared   AIC sigma adj.r.squared statistic
##       <dbl> <dbl> <dbl>         <dbl>     <dbl>
## 1     0.369  5662  69.3         0.367       291
## 2     0.532  5514  59.7         0.530       282
## 3     0.538  5509  59.4         0.536       193
## # ... with 6 more variables: p.value <dbl>,
## #   df <int>, logLik <dbl>, BIC <dbl>,
## #   deviance <dbl>, df.residual <int>
```

Flexibility: write your own verb

In `function.R`:

```
please_clean_dataset <- function(df) {
  df %>%
    janitor::clean_names() %>%
    filter(!is.na()) %>%
    filter(x < 1)
}
```

in main document:

```
df %>% please_clean_dataset() %>% please_remove_outliers()
```

- more details on 'polite programming' at doc/DataWorkflow.pdf

Communication

- Seamless integration in publications
- see session on literate programming: Marks Ups and Downs
- reproducible science

Summary

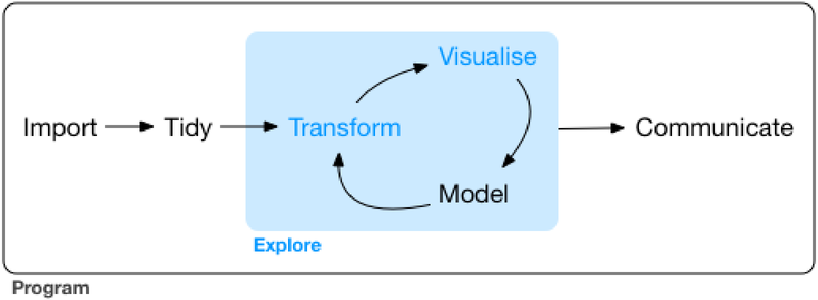


Figure 5: Workflow for data exploration

- tidyverse offers consistent syntax from data import to communication
- KISS & WORE principle
- you can focus on what, not on how
- easily extensible
- many different extensions exist

Conclusion

- tidyverse makes life easier, focus on the science not on data wrangling
- teach the tidyverse to beginners

References

Wickham, H. (2010). A layered grammar of graphics. *Journal of Computational and Graphical Statistics*, *19*(1), 3–28. https://doi.org/10.1198/jcgs.2009.07098

Wickham, H. (2014). Tidy data. *Journal of Statistical Software*, *59*(10), 1–23. https://doi.org/10.18637/jss.v059.i10

Wilkinson, L. (2005). *The grammar of graphics (statistics and computing)*. Springer.