Using a custom R package to facilitate teaching

Mark Andrews
Psychology Department, Nottingham Trent University

mark.andrews@ntu.ac.uk

Background: R in NTU Psychology

- For approximately 10 years, we have used R as the software in undergraduate and MSc advanced statistics modules ($\approx$ 70 students per year).
- In 2019/20, we began the department wide replacement of SPSS by R as the software used in all statistics and research methods modules.
- In 2019/20, 1st year UG statistics and research methods modules were taught using R ($\approx$ 1000 students)
- In 2020/21, 1st and 2nd year UG statistics and research methods modules were taught using R ($\approx$ 2000 students)
- In 2021/22, in addition to 1st and 2nd year UG statistics and research methods modules, 3rd year research project data analysis supervision will use R ($\approx$ 2500 students)

Initial motivation for using a custom R package

- By their second workshop in their 1st year, students were being asked to write the likes of the following:

```
Df %>%
  group_by(identified_as) %>%
  summarise(mean_correct = mean(percent_corr, na.rm = T),
            iqr = IQR(percent_corr, na.rm = T))



ggplot(data = Df,
       aes(x = Faithful,
           fill = FaceSex, colour = FaceSex)) +
  geom_histogram(binwidth = .25,
                 position = "identity",
                 alpha = .35) +
  labs(x = "Faithfulness rating",
       y = "Frequency")
```

Initial motivation for using a custom R package

- Opaque multiline R code in the first few classes lead many students to conclude that R code is opaque and inscrutable.
- Using R was perceived as a matter of copying and pasting inscrutable code.
- Not much is gained if students replace blindly clicking (in SPSS) by blindly copy-n-pasting (in R).
- Therefore, the objective of a custom R package was primarily to provide custom functions to replace all opaque multiline R code used throughout all workshops and labs.

Arguments against a custom R package

- Arguments against using a custom R package for teaching were usually variants of the following related arguments:
  - students won't learn powerful general tools like `dplyr` and `ggplot`, and instead just learn some limited functions.
  - using custom function in place of general R code means students will not learn R code generally, nor will they understand what calculations the functions implement.
  - students will become dependent on a specific package

Rebuttal to arguments against a custom R package

- The second argument is an argument against all R packages, and even against R (or all high level languages) itself.
- Likewise, the third argument is an argument against all R packages.
- In response to the first argument, while teaching powerful general tools like `dplyr` and `ggplot` are excellent and well worth learning, do we have the resources to properly teach them?

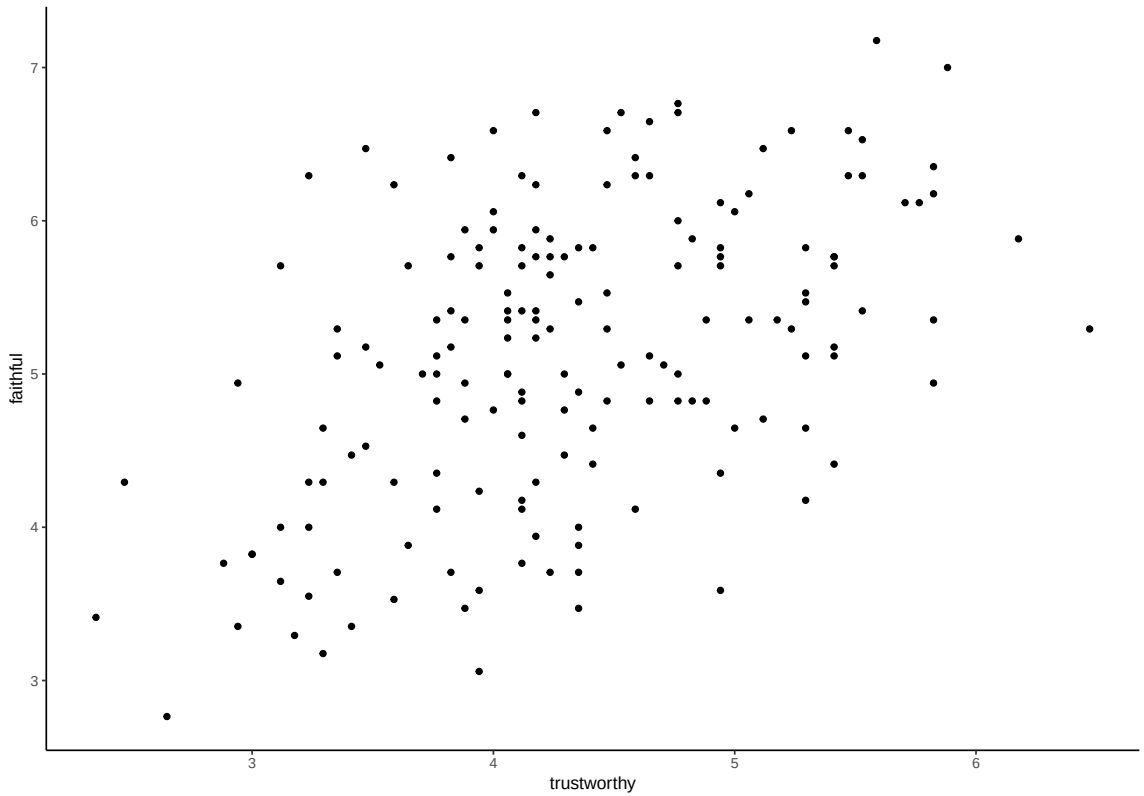
Overview of `psyntur` (0.0.2)

```
library(psyntur)
packageVersion("psyntur")
#> [1] '0.0.2'
```

- The `psyntur`, in its current state, provides
  - Some methods for data visualization
  - Some methods for data exploration
  - Some helper functions for statistical analysis
- Available at `https://github.com/mark-andrews/psyntur`
- Package webpage at `https://mark-andrews.github.io/psyntur`

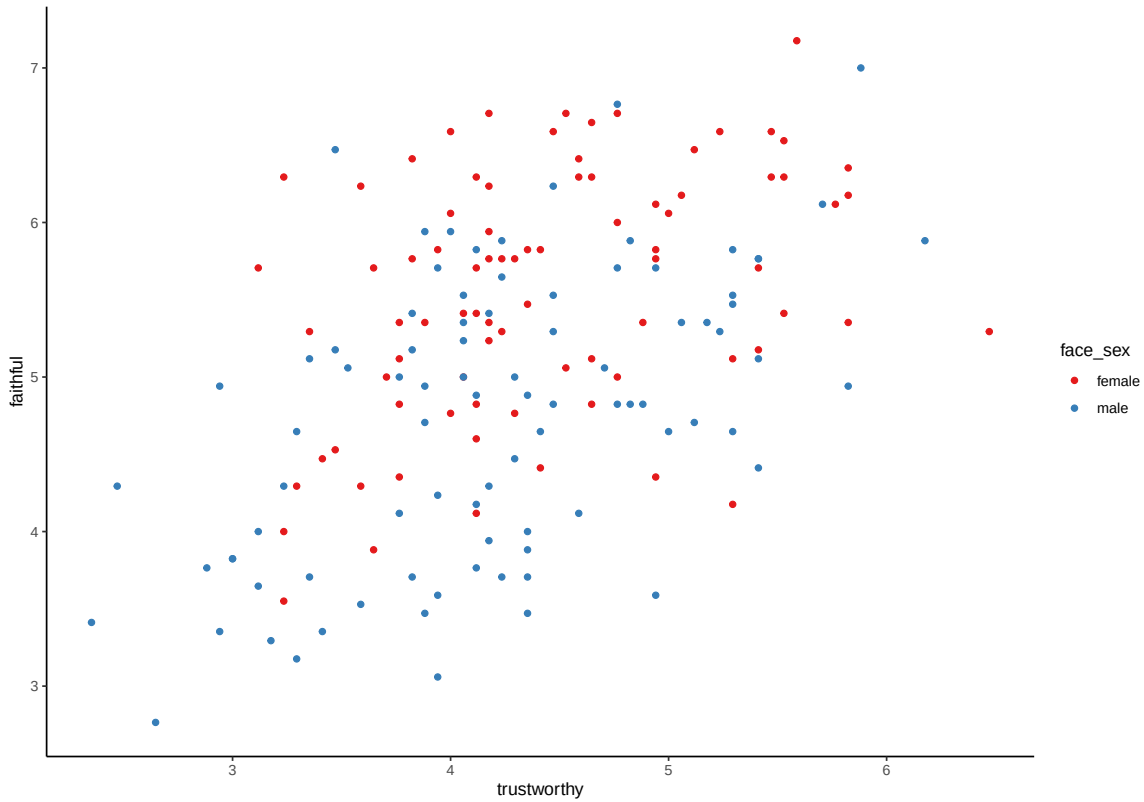
Data visualization: Scatterplot

```
scatterplot(x = trustworthy, y = faithful,
            data = faithfulfaces)
```



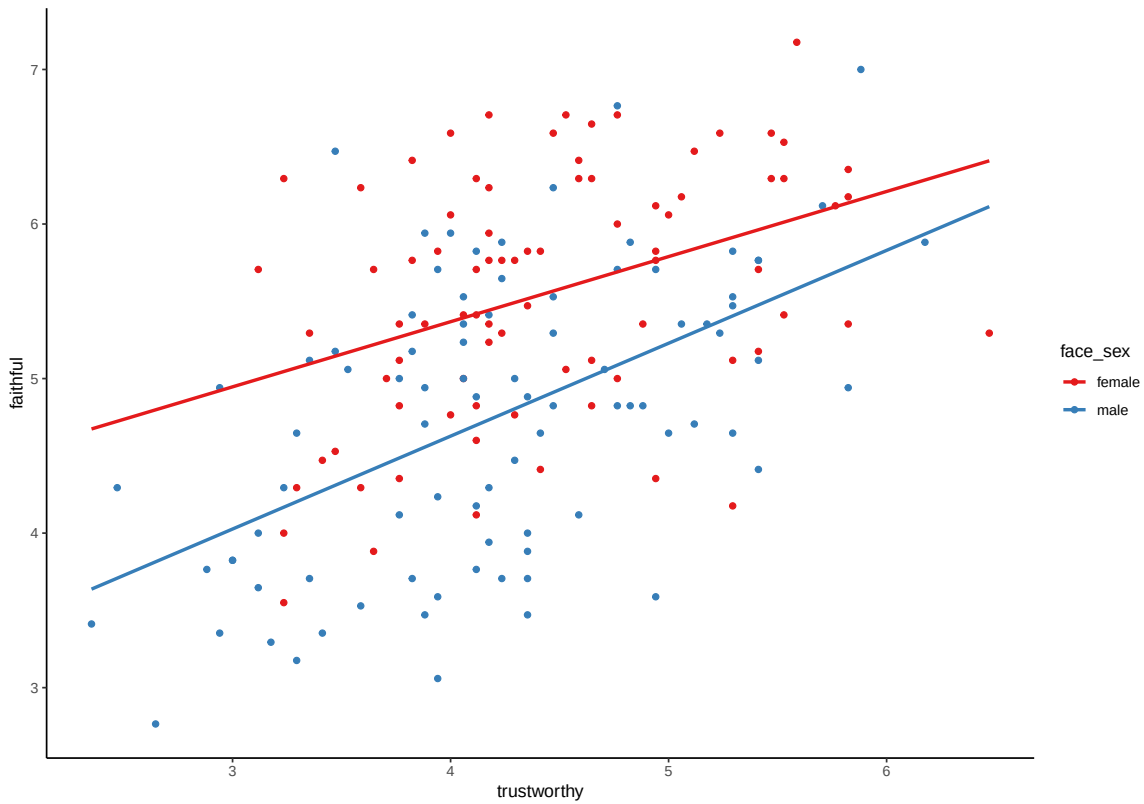
Data visualization: Coloured scatterplot

```
scatterplot(x = trustworthy, y = faithful,
            by = face_sex,
            data = faithfulfaces)
```



Data visualization: Coloured scatterplot, with line of best fit

```
scatterplot(x = trustworthy, y = faithful,
            by = face_sex,
            best_fit_line = TRUE,
            data = faithfulfaces)
```



Comparing `ggplot` and `psyntur`

- This `scatterplot` function

```
scatterplot(x = trustworthy, y = faithful,
            by = face_sex,
            best_fit_line = TRUE,
            data = faithfulfaces)
```

implements exactly this:

```
ggplot(faithfulfaces,
       aes(x = trustworthy, y = faithful, colour = face_sex)
) + geom_point() +
  stat_smooth(method = 'lm', se = FALSE, fullrange = TRUE,
              formula = 'y ~ x') +
  theme_classic() +
  scale_colour_brewer(palette = "Set1")
```

Exploratory data analysis: Summary statistics

- The `describe` function is a wrapper to `dplyr`'s `summarize` (and `group_by` etc):

```
describe(data = faithfulfaces,
         avg = mean(faithful), stdev = sd(faithful))
#> # A tibble: 1 x 2
#>     avg stdev
#>   <dbl> <dbl>
#> 1  5.14 0.957
```

```
describe(data = faithfulfaces,
         by = face_sex,
         avg = mean(faithful), stdev = sd(faithful))
#> # A tibble: 2 x 3
#>   face_sex   avg stdev
#>   <chr>    <dbl> <dbl>
#> 1 female    5.55 0.802
#> 2 male      4.75 0.932
```

Exploratory data analysis: Multiple variables, multiple functions

The `describe_across` is an interface to `summarize`, `across`, `pivot_wider`:

```
describe_across(faithfulfaces,
                variables = c(trustworthy, faithful),
                functions = list(avg = mean,
                                 median = median,
                                 stdev = sd),
                pivot = TRUE
)
#> # A tibble: 2 x 4
#>   variable      avg median stdev
#>   <chr>       <dbl>  <dbl> <dbl>
#> 1 trustworthy  4.32   4.24 0.791
#> 2 faithful     5.14   5.24 0.957
```

Statistics helper functions

- We also implemented a few helper functions like the following:

```
shapiro_test(faithful, by = face_sex,
             data = faithfulfaces)
#> # A tibble: 2 x 3
#>   face_sex statistic p_value
#>   <chr>        <dbl>   <dbl>
#> 1 female       0.972  0.0688
#> 2 male         0.979  0.168
```

which implements

```
get_shapiro_test_results <- function(x){
    results <- stats::shapiro.test(x)
    tibble(statistic = results$statistic, p_value = results$p.value)
  }
faithfulfaces %>%
  group_by(face_sex) %>%
  summarise(get_shapiro_test_results(faithful),
            .groups = 'drop')
#> # A tibble: 2 x 3
#>   face_sex statistic p_value
#>   <chr>        <dbl>   <dbl>
#> 1 female       0.972  0.0688
#> 2 male         0.979  0.168
```

Other advantages to using a custom R package

- Data files with help files (e.g. `?faithfulfaces`)
- Custom help pages for functions
- Vignettes, see
  `https://mark-andrews.github.io/psyntur/articles/`

The great `install_github` gotcha

- We did have any urgency getting `psyntur` onto CRAN.
- `devtools::install_github` seemed like a perfectly acceptable alternative
- However, hundreds of students hit GitHub's `API rate limit exceeded` restriction in workshops.
- `install_github`, by defaults, asks if all dependencies should be updated to their latest version. This often lead students down the `Do you want to install from sources the package which needs compilation?` rabbit hole.