The finafit package brings together the day-to-day functions we use to generate final results tables and plots when modelling. I spent many years repeatedly manually copying results from R analyses and built these functions to automate our standard healthcare data workflow. It is particularly useful when undertaking a large study involving multiple different regression analyses. When combined with RMarkdown, the reporting becomes entirely automated.

**Installation and Documentation**

You can install finalfit from github with:

library("finalfit")

It is recommended that this package is used together with dplyr, which is a dependent.

Some of the functions require rstan and boot. These have been left as Suggests rather than Depends to avoid unnecessary installation. If needed, they can be installed in the normal way:

install.packages("rstan")

install.packages("boot")

To install off-line (or in a Safe Haven), use devtools::install\_local().

**Main Features**

**1. Summarise variables/factors by a categorical variable**

summary\_factorlist() is a wrapper used to aggregate any number of explanatory variables by a single **variable of interest**. This is often “Table 1” of a published study. When categorical, the variable of interest can have a maximum of five levels. It uses Hmisc::summary.formula().

library(finalfit)

library(dplyr)

# Load example dataset, modified version of survival::colon

data(colon\_s)

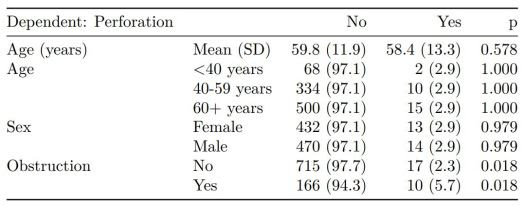
# Table 1 - Patient demographics by variable of interest ----

explanatory = c("age", "age.factor", "sex.factor", "obstruct.factor")

dependent = "perfor.factor" # Bowel perforation

colon\_s %>%

summary\_factorlist(dependent, explanatory, p=TRUE, add\_dependent\_label=TRUE)

[](https://i1.wp.com/www.datasurg.net/wp-content/uploads/2018/05/table1.jpg)

See other options relating to inclusion of missing data, mean vs. median for continuous variables, column vs. row proportions, include a total column etc.

summary\_factorlist() is also commonly used to summarise any number of variables by an **outcome variable** (say dead yes/no).

# Table 2 - 5 yr mortality ----

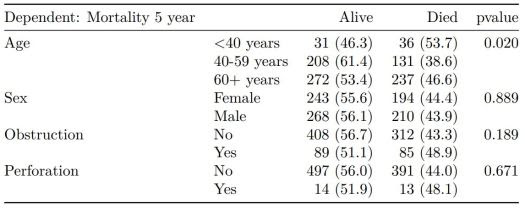
explanatory = c("age", "age.factor", "sex.factor",

"obstruct.factor")

dependent = 'mort\_5yr'

colon\_s %>%

summary\_factorlist(dependent, explanatory, p=TRUE, add\_dependent\_label=TRUE)

[](https://i0.wp.com/www.datasurg.net/wp-content/uploads/2018/05/table2.jpg)

Tables can be knitted to PDF, Word or html documents. We do this in RStudio from a .Rmd document. Example chunk:

```{r, echo = FALSE, results='asis'}

knitr::kable(example\_table, row.names=FALSE,

align=c("l", "l", "r", "r", "r", "r"))

```

**2. Summarise regression model results in final table format**

The second main feature is the ability to create final tables for linear (lm()), logistic (glm()), hierarchical logistic (lme4::glmer()) and  
Cox proportional hazards (survival::coxph()) regression models.

The finalfit() “all-in-one” function takes a single dependent variable with a vector of explanatory variable names (continuous or categorical variables) to produce a final table for publication including summary statistics, univariable and multivariable regression analyses. The first columns are those produced by summary\_factorist(). The appropriate regression model is chosen on the basis of the dependent variable type and other arguments passed.

**Logistic regression: glm()**

Of the form: glm(depdendent ~ explanatory, family="binomial")

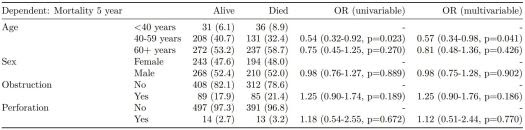
explanatory = c("age.factor", "sex.factor",

"obstruct.factor", "perfor.factor")

dependent = 'mort\_5yr'

colon\_s %>%

finalfit(dependent, explanatory)

[](https://i0.wp.com/www.datasurg.net/wp-content/uploads/2018/05/table3.jpg)

**Logistic regression with reduced model: glm()**

Where a multivariable model contains a subset of the variables included specified in the full univariable set, this can be specified.

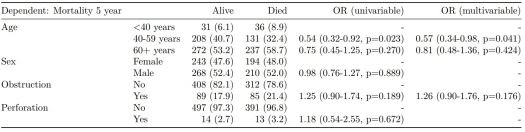
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")

explanatory\_multi = c("age.factor", "obstruct.factor")

dependent = 'mort\_5yr'

colon\_s %>%

finalfit(dependent, explanatory, explanatory\_multi)

[](https://i2.wp.com/www.datasurg.net/wp-content/uploads/2018/05/table4.jpg)

**Mixed effects logistic regression: lme4::glmer()**

Of the form: lme4::glmer(dependent ~ explanatory + (1 | random\_effect), family="binomial")

Hierarchical/mixed effects/multilevel logistic regression models can be specified using the argument random\_effect. At the moment it is just set up for random intercepts (i.e. (1 | random\_effect), but in the future I’ll adjust this to accommodate random gradients if needed (i.e. (variable1 | variable2).

explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")

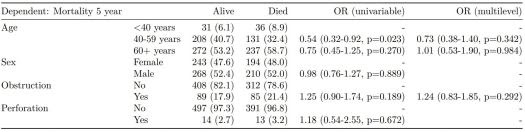
explanatory\_multi = c("age.factor", "obstruct.factor")

random\_effect = "hospital"

dependent = 'mort\_5yr'

colon\_s %>%

finalfit(dependent, explanatory, explanatory\_multi, random\_effect)

[](https://i0.wp.com/www.datasurg.net/wp-content/uploads/2018/05/table5.jpg)

**Cox proportional hazards: survival::coxph()**

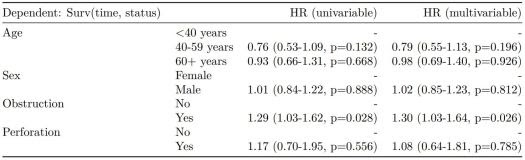
Of the form: survival::coxph(dependent ~ explanatory)

explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")

dependent = "Surv(time, status)"

colon\_s %>%

finalfit(dependent, explanatory)

[](https://i1.wp.com/www.datasurg.net/wp-content/uploads/2018/05/table6.jpg)

**Add common model metrics to output**

metrics=TRUE provides common model metrics. The output is a list of two dataframes. Note chunk specification for output below.

explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")

dependent = 'mort\_5yr'

colon\_s %>%

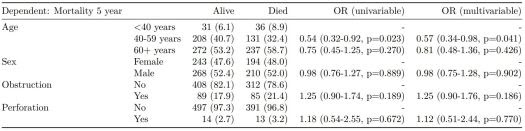
finalfit(dependent, explanatory, metrics=TRUE)

```{r, echo=FALSE, results="asis"}

knitr::kable(table7[[1]], row.names=FALSE, align=c("l", "l", "r", "r", "r"))

knitr::kable(table7[[2]], row.names=FALSE)

```

[](https://i1.wp.com/www.datasurg.net/wp-content/uploads/2018/05/table7a.jpg)

[](https://i1.wp.com/www.datasurg.net/wp-content/uploads/2018/05/table7b.jpg)

Rather than going all-in-one, any number of subset models can be manually added on to a summary\_factorlist() table using finalfit\_merge(). This is particularly useful when models take a long-time to run or are complicated.

Note the requirement for fit\_id=TRUE in summary\_factorlist(). fit2df extracts, condenses, and add metrics to supported models.

explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")

explanatory\_multi = c("age.factor", "obstruct.factor")

random\_effect = "hospital"

dependent = 'mort\_5yr'

# Separate tables

colon\_s %>%

summary\_factorlist(dependent, explanatory, fit\_id=TRUE) -> example.summary

colon\_s %>%

glmuni(dependent, explanatory) %>%

fit2df(estimate\_suffix=" (univariable)") -> example.univariable

colon\_s %>%

glmmulti(dependent, explanatory) %>%

fit2df(estimate\_suffix=" (multivariable)") -> example.multivariable

colon\_s %>%

glmmixed(dependent, explanatory, random\_effect) %>%

fit2df(estimate\_suffix=" (multilevel)") -> example.multilevel

# Pipe together

example.summary %>%

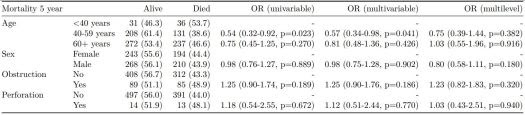
finalfit\_merge(example.univariable) %>%

finalfit\_merge(example.multivariable) %>%

finalfit\_merge(example.multilevel) %>%

select(-c(fit\_id, index)) %>% # remove unnecessary columns

dependent\_label(colon\_s, dependent, prefix="") # place dependent variable label

[](https://i0.wp.com/www.datasurg.net/wp-content/uploads/2018/05/table8.jpg)

**Bayesian logistic regression: with stan**

Our own particular rstan models are supported and will be documented in the future. Broadly, if you are running (hierarchical) logistic regression models in [Stan](http://mc-stan.org/users/interfaces/rstan) with coefficients specified as a vector labelled beta, then fit2df() will work directly on the stanfit object in a similar manner to if it was a glm or glmerMod object.

**3. Summarise regression model results in plot**

Models can be summarized with odds ratio/hazard ratio plots using or\_plot, hr\_plot and surv\_plot.

**OR plot**

# OR plot

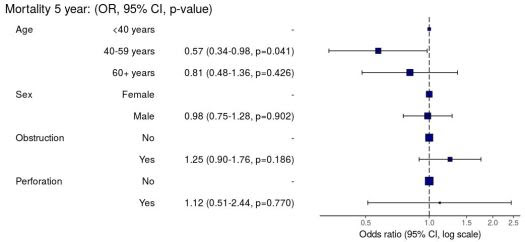
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")

dependent = 'mort\_5yr'

colon\_s %>%

or\_plot(dependent, explanatory)

# Previously fitted models (`glmmulti()` or `glmmixed()`) can be provided directly to `glmfit`

[](https://i2.wp.com/www.datasurg.net/wp-content/uploads/2018/05/plot1.jpg)

**HR plot**

# HR plot

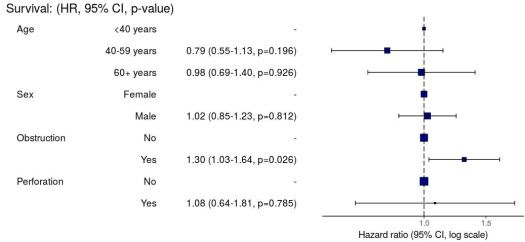
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")

dependent = "Surv(time, status)"

colon\_s %>%

hr\_plot(dependent, explanatory, dependent\_label = "Survival")

# Previously fitted models (`coxphmulti`) can be provided directly using `coxfit`

[](https://i2.wp.com/www.datasurg.net/wp-content/uploads/2018/05/plot2.jpg)

**Kaplan-Meier survival plots**

KM plots can be produced using the library(survminer)

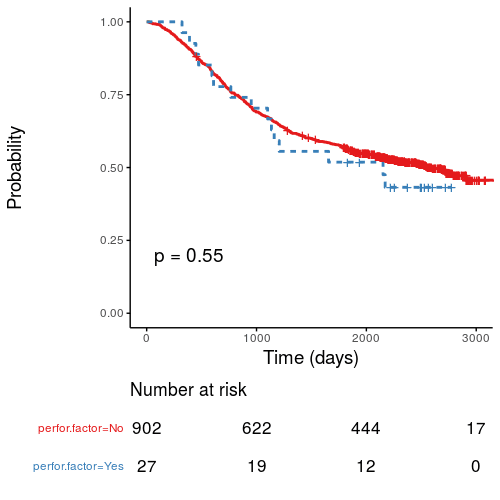
# KM plot

explanatory = c("perfor.factor")

dependent = "Surv(time, status)"

colon\_s %>%

surv\_plot(dependent, explanatory, xlab="Time (days)", pval=TRUE, legend="none")

[](https://i1.wp.com/www.datasurg.net/wp-content/uploads/2018/05/Rplot.png)

**Notes**

Use Hmisc::label() to assign labels to variables for tables and plots.

label(colon\_s$age.factor) = "Age (years)"

Export dataframe tables directly or to [R Markdown](http://rmarkdown.rstudio.com/) knitr::kable().

Note wrapper summary\_missing() is also useful. Wraps mice::md.pattern.

colon\_s %>%

summary\_missing(dependent, explanatory)

Development will be on-going, but any input appreciated.