

My R Codes for Data Analysis

Articles per journals per country

Serdar Balci

```
{r # Sys.Date() }
```


Contents

Chapter 1

Prerequisites

Chapter 2

Introduction

You can label chapter and section titles using `{#label}` after them, e.g., we can reference Chapter `??`. If you do not manually label them, there will be automatic labels anyway, e.g., Chapter `??`.

Figures and tables with captions will be placed in `figure` and `table` environments, respectively.

```
par(mar = c(4, 4, .1, .1))  
plot(pressure, type = 'b', pch = 19)
```

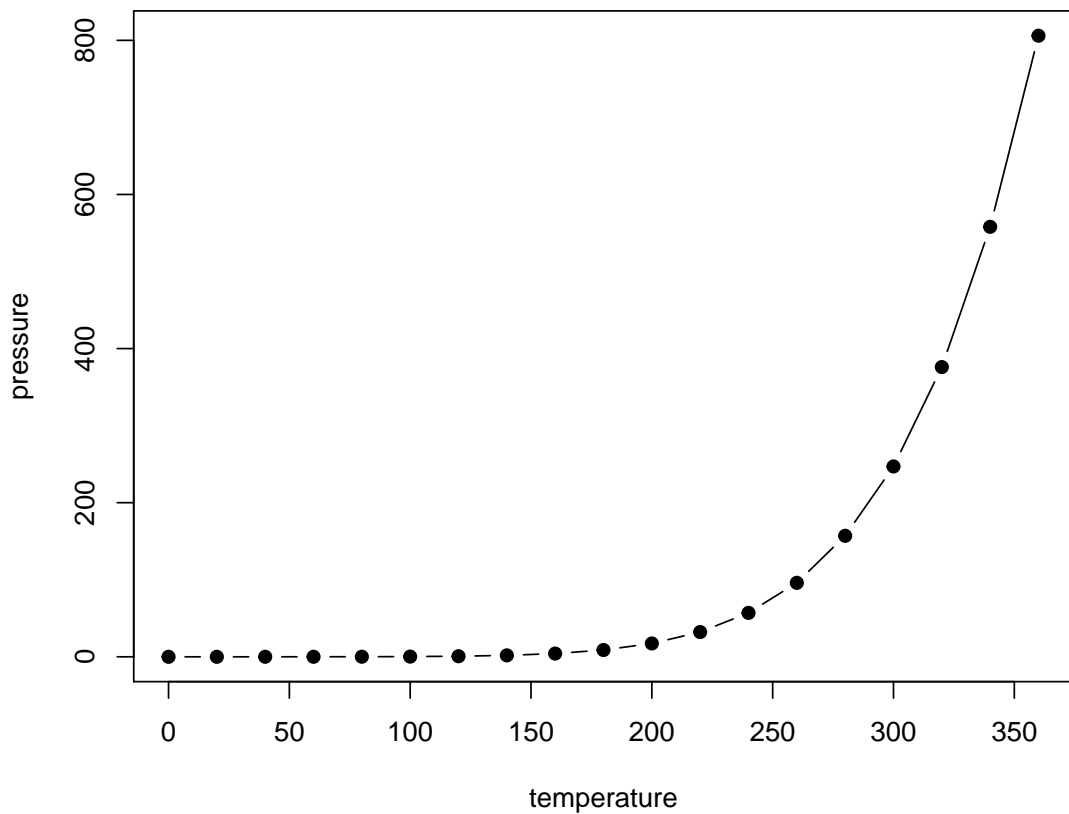


Figure 2.1: Here is a nice figure!

Reference a figure by its code chunk label with the `fig:` prefix, e.g., see Figure `??`. Similarly, you can reference tables generated from `knitr::kable()`, e.g., see Table `??`.

Table 2.1: Here is a nice table!

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa
5.8	4.0	1.2	0.2	setosa
5.7	4.4	1.5	0.4	setosa
5.4	3.9	1.3	0.4	setosa
5.1	3.5	1.4	0.3	setosa
5.7	3.8	1.7	0.3	setosa
5.1	3.8	1.5	0.3	setosa

```
knitr::kable(
  head(iris, 20), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```

You can write citations, too. For example, we are using the **bookdown** package (?) in this sample book, which was built on top of R Markdown and **knitr** (?).

Chapter 3

Literature

Here is a review of existing methods.

Chapter 4

Literature

Here is a review of existing methods.

Chapter 5

Methods

We describe our methods in this chapter.

Chapter 6

Applications

Some significant applications are demonstrated in this chapter.

6.1 Example one

6.2 Example two

Chapter 7

Final Words

We have finished a nice book.

```
{r # if (knitr::is_html_output()) ' # References {-} ' }
```

output:

```
html_notebook:
  code_folding: hide
  fig_caption: yes
  highlight: kate
  number_sections: yes
  theme: cerulean
  toc: yes
  toc_float: yes
html_document:
  code_folding: hide
  df_print: kable
  keep_md: yes
  number_sections: yes
  theme: cerulean
  toc: yes
  toc_float: yes
  highlight: kate
```

If you want to see the code used in the analysis please click the code button on the right upper corner or throughout the page.

Chapter 8

Analysis

8.1 Articles per journals per country

Aim:

In the previous analysis we have observed that Japanese researchers have much more articles than German and Turkish researchers.

Here we will look at the distribution of articles per journals per country.

Methods:

Pathology Journal ISSN List was retrieved from In Cites Clarivate, and Journal Data Filtered as follows: JCR Year: 2016 Selected Editions: SCIE,SSCI Selected Categories: 'PATHOLOGY' Selected Category Scheme: WoS

Data is retrieved from PubMed via RISmed package. PubMed collection from National Library of Medicine (<https://www.ncbi.nlm.nih.gov/pubmed/>), has the most comprehensive information about peer reviewed articles in medicine. The API (<https://dataguide.nlm.nih.gov/>), and R packages are available for getting and fetching data from the server.

The search formula for PubMed is generated as “ISSN List AND Country[Affiliation]” like done in advanced search of PubMed.

Articles from Japan, German and Turkey are retrieved limiting the search with pathology journals, affiliation and last 10 years.

The retrieved information was compiled in a table.

Result:

In this graph x-axis is the list of journals with decreasing impact factor, and y-axis is the number of articles published in that journal. The colors and shapes are showing the country of affiliation. We see that in one journal articles from Japan is more than 800.

Comment:

It is seen that one of the journals ISSN: 1440-1827 has more than 800 articles from Japan. This journal is also from Japan. Here we wonder if there is an editorial preference for articles from their home country.

We sometimes observe this situation if there is a conference in that country, and the conference abstracts are indexed.

This may also be a clue that if a country has a journal listed in indexes, than it is more easy for the researchers in that country to publish their results.

Future Work:

Whether this observation is a unique situation, or there is a tendency in the journals to publish article from their country of origin, merits further investigation.

Chapter 9

Feedback

Serdar Balcı, MD, Pathologist would like to hear your feedback: <https://goo.gl/forms/YjGZ5DHgtPlR1RnB3>

This document will be continiously updated and the last update was on {r # Sys.Date()}.

Chapter 10

Back to Main Menu

Main Page for Bibliographic Analysis

```
output:
  html_notebook:
    code_folding: hide
    fig_caption: yes
    highlight: kate
    number_sections: yes
    theme: cerulean
    toc: yes
    toc_float: yes
  html_document:
    code_folding: hide
    df_print: kable
    keep_md: yes
    number_sections: yes
    theme: cerulean
    toc: yes
    toc_float: yes
    highlight: kate
```


Chapter 11

Analysis

11.1 PubMed Indexed Peer Reviewed Articles in Pathology Journals: A country based comparison

Aim:

Here, we are going to compare 3 countries (German, Japan and Turkey), in terms of number of articles in pathology journals during the last decade.

Methods:

If you want to see the code used in the analysis please click the code button on the right upper corner or throughout the page.

Pathology Journal ISSN List was retrieved from In Cites Clarivate, and Journal Data Filtered as follows: JCR Year: 2016 Selected Editions: SCIE,SSCI Selected Categories: 'PATHOLOGY' Selected Category Scheme: WoS

Data is retrieved from PubMed via RISmed package. PubMed collection from National Library of Medicine (<https://www.ncbi.nlm.nih.gov/pubmed/>), has the most comprehensive information about peer reviewed articles in medicine. The API (<https://dataguide.nlm.nih.gov/>), and R packages are available for getting and fetching data from the server.

The search formula for PubMed is generated as “ISSN List AND Country[Affiliation]” like done in advanced search of PubMed.

From the fetched data the year of articles are grouped and counted by country.

Result:

In the below table we see the number of articles per country in the last decade.

And the figure below shows this data in a line graph.

Comment:

We see that Japan has much more articles than German and Turkey. Turkey has a small increase in number of articles.

Future Work:

- Identify why Japan has too much articles.
- Compare Japan with other countries.
- Compare Turkey with neighbours, EU, OECD & Middle East countries.
- Analyse multinational studies.
- Analyse adding journal impact as a factor.

Chapter 12

Feedback

Serdar Balci, MD, Pathologist would like to hear your feedback: <https://goo.gl/forms/YjGZ5DHgtPlR1RnB3>

This document will be continuously updated and the last update was on {r # Sys.Date()}.

Chapter 13

Back to Main Menu

Main Page for Bibliographic Analysis

```
output:
  html_notebook:
    code_folding: hide
    highlight: kate
    number_sections: yes
    theme: cerulean
    toc: yes
    toc_float: yes
    fig_caption: yes
  html_document:
    df_print: kable
    number_sections: yes
    toc: yes
```


Chapter 14

Analysis of Recent Pancreas Related Articles

Pancreas Journals <https://www.ncbi.nlm.nih.gov/nlmcatalog/?term=pancreas>

Pathology Journals

Member List

DOI Link PubMed Link Journal Link Altmetric API Dimensions API

USCAP abstracts vs publication

Member list vs worldmap

Pathology Journal ISSN List was retrieved from In Cites Clarivate, and Journal Data Filtered as follows: JCR Year: 2016 Selected Editions: SCIE,SSCI Selected Categories: 'PATHOLOGY' Selected Category Scheme: WoS

Data is retrieved from PubMed via E-direct.

PubMed collection from National Library of Medicine (<https://www.ncbi.nlm.nih.gov/pubmed/>), has the most comprehensive information about peer reviewed articles in medicine. The API (<https://dataguide.nlm.nih.gov/>) is available for getting and fetching data from the server.

The query for PubMed is generated as “ISSN List AND keywords” like done in advanced search of PubMed.

From the fetched data articles are grouped by country and keywords.

Result:

mapgraph

And the figure below shows this data in a line graph.

Chapter 15

Feedback

Serdar Balcı, MD, Pathologist would like to hear your feedback: <https://goo.gl/forms/YjGZ5DHgtPlR1RnB3>

This document will be continiously updated and the last update was on {r # Sys.Date()}.

Chapter 16

Back to Main Menu

Main Page for Bibliographic Analysis

```
output:
  html_notebook:
    code_folding: hide
    fig_caption: yes
    highlight: kate
    number_sections: yes
    theme: cerulean
    toc: yes
    toc_float: yes
  html_document:
    code_folding: hide
    df_print: kable
    keep_md: yes
    number_sections: yes
    theme: cerulean
    toc: yes
    toc_float: yes
    highlight: kate
```

If you want to see the code used in the analysis please click the code button on the right upper corner or throughout the page.

Chapter 17

Analysis

17.1 MeSH Terms In Pathology Articles From Turkey

Background

PubMed collection from National Library of Medicine, has the most comprehensive information about peer reviewed articles in medicine.

MeSH Terms is a controlled vocabulary that is used to label PubMed articles according to their content. It is done by experts in National Library of Medicine. Keywords are labels that are given by authors of the article. Both are included in a PubMed record of an article.

Aim:

In this analysis we aimed to identify the common research topics Turkish pathologists are interested. We extracted most common MeSH terms and keywords from PubMed articles using EDirect: MeSH Terms Pathology Articles From Turkey

Methods:

Packages used for analysis. Tidyverse is used for data manipulation, and rstudioapi to run e-utilities commands from RStudio.

Pathology Journal ISSN List was retrieved from In Cites Clarivate, and Journal Data Filtered as follows:

JCR Year: 2016 Selected Editions: SCIE,SSCI Selected Categories: 'PATHOLOGY' Selected Category Scheme: WoS

Data is retrieved from PubMed via e-Utilities.

The search formula for PubMed is generated as “ISSN List AND Country[Affiliation]” like done in advanced search of PubMed.

Articles are downloaded as xml.

MeSH terms are extracted from xml. Common terms are excluded and major topics are selected.

Keywords are extracted from xml.

Result:

The retrieved information was compiled in a table.

Comment:

Future Work:

Chapter 18

Feedback

Serdar Balcı, MD, Pathologist would like to hear your feedback: <https://goo.gl/forms/YjGZ5DHgtPlR1RnB3>

This document will be continiously updated and the last update was on {r # Sys.Date()}.

Chapter 19

Back to Main Menu

Main Page for Bibliographic Analysis

Several packages support making beautiful tables with R, such as

- xtable
- stargazer
- pander
- tables
- ascii
- etc.

It is also very easy to make tables with knitr's `kable` function:

```
library(knitr)
kable(mtcars[1:5, ], caption = "A knitr kable.")
```

- Analysing the HIV pandemic

<https://rviews.rstudio.com/2019/04/30/analysing-hiv-pandemic-part-1/>

Table 19.1: A knitr kable.

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2

Chapter 20

arsenal

20.1 The compare function

<https://cran.r-project.org/web/packages/arsenal/vignettes/compare.html>

```
Summary of data.frames
version arg ncol  nrow
x  mockstudy  14  1499
y  mockstudy2  13  1495
Variables not shared
version variable  position  class
x  age 2  integer
x  arm 3  character
x  fu.time 6  integer
x  fu.stat 7  integer
y  fu_time 11 integer
y  fu stat 12 integer
y  Arm 13  character
Other variables not compared
var.x  pos.x  class.x var.y  pos.y  class.y
race   5  character race   3  factor
ast 12  integer ast 8  numeric
Observations not shared
version case  observation
x  88989  9
x  90158  8
x  99508  7
x  112263 5
Differences detected by variable
var.x  var.y  n  NAs
sex sex 1495  0
ps ps 1 1
hgb hgb 266 266
bmi bmi 0 0
alk.phos  alk.phos  0 0
mdquality.s mdquality.s 0 0
age.ord age.ord 0 0
First 10 differences detected per variable (1741 differences not shown)
var.x  var.y  case  values.x  values.y  row.x  row.y
sex sex 76170  Male  Male  26  20
sex sex 76240  Male  Male  27  21
```

```
sex sex 76431 Female Female 28 22
sex sex 76712 Male Male 29 23
sex sex 76780 Female Female 30 24
sex sex 77066 Female Female 31 25
sex sex 77316 Male Male 32 26
sex sex 77355 Male Male 33 27
sex sex 77591 Male Male 34 28
sex sex 77851 Male Male 35 29
```

```
ps ps 86205 0 NA 6 3
hgb hgb 88714 NA -9 192 186
hgb hgb 88955 NA -9 204 198
hgb hgb 89549 NA -9 229 223
hgb hgb 89563 NA -9 231 225
hgb hgb 89584 NA -9 237 231
hgb hgb 89591 NA -9 238 232
hgb hgb 89595 NA -9 239 233
hgb hgb 89647 NA -9 243 237
hgb hgb 89665 NA -9 244 238
hgb hgb 89827 NA -9 255 249
```

Non-identical attributes

```
var.x var.y name
sex sex label
sex sex levels
race race class
race race label
race race levels
bmi bmi label
```

Column name comparison options

It is possible to change which column names are considered "the same variable".

Ignoring case

For example, to ignore case in variable names (so that Arm and arm are considered the same), pass `tol.vars`

You can do this using `comparison.control()`

```
summary(compare(mockstudy, mockstudy2, by = "case", control = comparison.control(tol.vars = "case")))
or pass it through the ... arguments.
```

```
summary(compare(mockstudy, mockstudy2, by = "case", tol.vars = "case"))
```

Summary of data.frames

```
version arg ncol nrow
x mockstudy 14 1499
y mockstudy2 13 1495
```

Variables not shared

```
version variable position class
x age 2 integer
x fu.time 6 integer
x fu.stat 7 integer
y fu_time 11 integer
y fu stat 12 integer
```

Other variables not compared

```
var.x pos.x class.x var.y pos.y class.y
race 5 character race 3 factor
ast 12 integer ast 8 numeric
```

Observations not shared

```
version case observation
x 88989 9
```

```

x  90158  8
x  99508  7
x  112263 5
Differences detected by variable
var.x  var.y  n  NAs
arm Arm 0  0
sex sex 1495  0
ps ps 1  1
hgb hgb 266 266
bmi bmi 0  0
alk.phos  alk.phos  0  0
mdquality.s mdquality.s 0  0
age.ord age.ord 0  0
First 10 differences detected per variable (1741 differences not shown)
var.x  var.y  case  values.x  values.y  row.x  row.y
sex sex 76170  Male  Male  26  20
sex sex 76240  Male  Male  27  21
sex sex 76431  Female Female 28  22
sex sex 76712  Male  Male  29  23
sex sex 76780  Female Female 30  24
sex sex 77066  Female Female 31  25
sex sex 77316  Male  Male  32  26
sex sex 77355  Male  Male  33  27
sex sex 77591  Male  Male  34  28
sex sex 77851  Male  Male  35  29
ps ps 86205  0  NA  6  3
hgb hgb 88714  NA  -9 192 186
hgb hgb 88955  NA  -9 204 198
hgb hgb 89549  NA  -9 229 223
hgb hgb 89563  NA  -9 231 225
hgb hgb 89584  NA  -9 237 231
hgb hgb 89591  NA  -9 238 232
hgb hgb 89595  NA  -9 239 233
hgb hgb 89647  NA  -9 243 237
hgb hgb 89665  NA  -9 244 238
hgb hgb 89827  NA  -9 255 249
Non-identical attributes
var.x  var.y  name
arm Arm label
sex sex label
sex sex levels
race  race  class
race  race  label
race  race  levels
bmi bmi label
Treating dots and underscores the same (equivalence classes)
It is possible to treat certain characters or sets of characters as the same by passing a character vector

In short, each string in the vector is split into single characters, and the resulting set of characters is

Passing a single character as an element this vector will replace that character with the empty string. For

For mockstudy, let's treat dots, underscores, and spaces as the same, and ignore case:

summary(compare(mockstudy, mockstudy2, by = "case",
               tol.vars = c("._ ", "case") # dots=underscores=spaces, ignore case
))

```

```

Summary of data.frames
version arg ncol      nrow
x  mockstudy  14  1499
y  mockstudy2 13  1495
Variables not shared
version variable      position      class
x  age 2      integer
Other variables not compared
var.x  pos.x  class.x var.y  pos.y  class.y
race   5      character race   3      factor
ast 12 integer ast 8      numeric
Observations not shared
version case      observation
x  88989  9
x  90158  8
x  99508  7
x  112263 5
Differences detected by variable
var.x  var.y  n  NAs
arm Arm 0  0
sex sex 1495  0
fu.time fu_time 0  0
fu.stat fu_stat 0  0
ps ps 1  1
hgb hgb 266 266
bmi bmi 0  0
alk.phos alk.phos 0  0
mdquality.s mdquality.s 0  0
age.ord age.ord 0  0
First 10 differences detected per variable (1741 differences not shown)
var.x  var.y  case  values.x  values.y  row.x  row.y
sex sex 76170  Male  Male  26  20
sex sex 76240  Male  Male  27  21
sex sex 76431  Female Female 28  22
sex sex 76712  Male  Male  29  23
sex sex 76780  Female Female 30  24
sex sex 77066  Female Female 31  25
sex sex 77316  Male  Male  32  26
sex sex 77355  Male  Male  33  27
sex sex 77591  Male  Male  34  28
sex sex 77851  Male  Male  35  29
ps ps 86205  0  NA  6  3
hgb hgb 88714  NA  -9  192 186
hgb hgb 88955  NA  -9  204 198
hgb hgb 89549  NA  -9  229 223
hgb hgb 89563  NA  -9  231 225
hgb hgb 89584  NA  -9  237 231
hgb hgb 89591  NA  -9  238 232
hgb hgb 89595  NA  -9  239 233
hgb hgb 89647  NA  -9  243 237
hgb hgb 89665  NA  -9  244 238
hgb hgb 89827  NA  -9  255 249
Non-identical attributes
var.x  var.y  name
arm Arm label
sex sex label
sex sex levels

```

```

race    race    class
race    race    label
race    race    levels
bmi bmi label

```

Column comparison options

Logical tolerance

Use the `tol.logical=` argument to change how logicals are compared. By default, they're expected to be equal

Numeric tolerance

To allow numeric differences of a certain tolerance, use the `tol.num=` and `tol.num.val=` options. `tol.num.val`

Also note the option `int.as.num=`, which determines whether integers and numerics should be compared despite

```

summary(compare(mockstudy, mockstudy2, by = "case",
               tol.vars = c("._ ", "case"), # dots=underscores=spaces, ignore case
               int.as.num = TRUE             # compare integers and numerics
))

```

Summary of data.frames

```

version arg ncol    nrow
x  mockstudy  14  1499
y  mockstudy2 13  1495

```

Variables not shared

```

version variable    position    class
x  age 2    integer

```

Other variables not compared

```

var.x  pos.x  class.x var.y  pos.y  class.y
race   5    character race   3    factor

```

Observations not shared

```

version case    observation
x  88989  9
x  90158  8
x  99508  7
x  112263 5

```

Differences detected by variable

```

var.x  var.y  n  NAs
arm Arm 0  0
sex sex 1495  0
fu.time fu_time 0  0
fu.stat fu_stat 0  0
ps  ps 1  1
hgb hgb 266 266
bmi bmi 0  0
alk.phos  alk.phos  0  0
ast ast 3  0
mdquality.s mdquality.s 0  0
age.ord age.ord 0  0

```

First 10 differences detected per variable (1741 differences not shown)

```

var.x  var.y  case  values.x  values.y  row.x  row.y
sex sex 76170  Male  Male    26  20
sex sex 76240  Male  Male    27  21
sex sex 76431  Female Female  28  22
sex sex 76712  Male  Male    29  23
sex sex 76780  Female Female  30  24
sex sex 77066  Female Female  31  25
sex sex 77316  Male  Male    32  26
sex sex 77355  Male  Male    33  27
sex sex 77591  Male  Male    34  28

```

```

sex sex 77851   Male   Male   35  29
ps  ps  86205   0     NA    6    3
hgb hgb 88714   NA     -9    192 186
hgb hgb 88955   NA     -9    204 198
hgb hgb 89549   NA     -9    229 223
hgb hgb 89563   NA     -9    231 225
hgb hgb 89584   NA     -9    237 231
hgb hgb 89591   NA     -9    238 232
hgb hgb 89595   NA     -9    239 233
hgb hgb 89647   NA     -9    243 237
hgb hgb 89665   NA     -9    244 238
hgb hgb 89827   NA     -9    255 249
ast ast 86205   27    36    6    3
ast ast 105271  100   36    3    2
ast ast 110754  35    36    1    1

```

Non-identical attributes

```

var.x  var.y  name
arm Arm label
sex sex label
sex sex levels
race  race   class
race  race   label
race  race   levels
bmi bmi label

```

Suppose a tolerance of up to 10 is allowed for ast:

```

summary(compare(mockstudy, mockstudy2, by = "case",
               tol.vars = c("._ ", "case"), # dots=underscores=spaces, ignore case
               int.as.num = TRUE,           # compare integers and numerics
               tol.num.val = 10             # allow absolute differences <= 10
))

```

Summary of data.frames

```

version arg ncol  nrow
x  mockstudy  14 1499
y  mockstudy2 13 1495

```

Variables not shared

```

version variable position  class
x  age 2 integer

```

Other variables not compared

```

var.x  pos.x  class.x var.y  pos.y  class.y
race   5  character race   3  factor

```

Observations not shared

```

version case  observation
x  88989  9
x  90158  8
x  99508  7
x  112263 5

```

Differences detected by variable

```

var.x  var.y  n  NAs
arm Arm 0  0
sex sex 1495  0
fu.time fu_time 0  0
fu.stat fu_stat 0  0
ps  ps 1  1
hgb hgb 266 266
bmi bmi 0  0
alk.phos  alk.phos  0  0

```



```

ast ast 1 0
mdquality.s mdquality.s 0 0
age.ord age.ord 0 0
First 10 differences detected per variable (1741 differences not shown)
var.x var.y case values.x values.y row.x row.y
sex sex 76170 Male Male 26 20
sex sex 76240 Male Male 27 21
sex sex 76431 Female Female 28 22
sex sex 76712 Male Male 29 23
sex sex 76780 Female Female 30 24
sex sex 77066 Female Female 31 25
sex sex 77316 Male Male 32 26
sex sex 77355 Male Male 33 27
sex sex 77591 Male Male 34 28
sex sex 77851 Male Male 35 29
ps ps 86205 0 NA 6 3
hgb hgb 88714 NA -9 192 186
hgb hgb 88955 NA -9 204 198
hgb hgb 89549 NA -9 229 223
hgb hgb 89563 NA -9 231 225
hgb hgb 89584 NA -9 237 231
hgb hgb 89591 NA -9 238 232
hgb hgb 89595 NA -9 239 233
hgb hgb 89647 NA -9 243 237
hgb hgb 89665 NA -9 244 238
hgb hgb 89827 NA -9 255 249
ast ast 105271 100 36 3 2
Non-identical attributes
var.x var.y name
arm Arm label
sex sex label
sex sex levels
race race class
race race label
race race levels
bmi bmi label
Factor tolerance
By default, factors are compared to each other based on both the labels and the underlying numeric levels.

summary(compare(mockstudy, mockstudy2, by = "case",
               tol.vars = c(".", "case"), # dots=underscores=spaces, ignore case
               int.as.num = TRUE,         # compare integers and numerics
               tol.num.val = 10,          # allow absolute differences <= 10
               tol.factor = "labels"      # match only factor labels
))
Summary of data.frames
version arg ncol nrow
x mockstudy 14 1499
y mockstudy2 13 1495
Variables not shared
version variable position class
x age 2 integer
Other variables not compared
var.x pos.x class.x var.y pos.y class.y
race 5 character race 3 factor
Observations not shared
version case observation

```

```

x 88989 9
x 90158 8
x 99508 7
x 112263 5
Differences detected by variable
var.x var.y n NAs
arm Arm 0 0
sex sex 0 0
fu.time fu_time 0 0
fu.stat fu_stat 0 0
ps ps 1 1
hgb hgb 266 266
bmi bmi 0 0
alk.phos alk.phos 0 0
ast ast 1 0
mdquality.s mdquality.s 0 0
age.ord age.ord 0 0
First 10 differences detected per variable (256 differences not shown)
var.x var.y case values.x values.y row.x row.y
ps ps 86205 0 NA 6 3
hgb hgb 88714 NA -9 192 186
hgb hgb 88955 NA -9 204 198
hgb hgb 89549 NA -9 229 223
hgb hgb 89563 NA -9 231 225
hgb hgb 89584 NA -9 237 231
hgb hgb 89591 NA -9 238 232
hgb hgb 89595 NA -9 239 233
hgb hgb 89647 NA -9 243 237
hgb hgb 89665 NA -9 244 238
hgb hgb 89827 NA -9 255 249
ast ast 105271 100 36 3 2
Non-identical attributes
var.x var.y name
arm Arm label
sex sex label
sex sex levels
race race class
race race label
race race levels
bmi bmi label
Also note the option factor.as.char=, which determines whether factors and characters should be compared de

summary(compare(mockstudy, mockstudy2, by = "case",
               tol.vars = c("._ ", "case"), # dots=underscores=spaces, ignore case
               int.as.num = TRUE,           # compare integers and numerics
               tol.num.val = 10,             # allow absolute differences <= 10
               tol.factor = "labels",        # match only factor labels
               factor.as.char = TRUE         # compare factors and characters
))
Summary of data.frames
version arg ncol nrow
x mockstudy 14 1499
y mockstudy2 13 1495
Variables not shared
version variable position class
x age 2 integer
Other variables not compared

```

No other variables not compared

Observations not shared

version case observation

x 88989 9

x 90158 8

x 99508 7

x 112263 5

Differences detected by variable

var.x var.y n NAs

arm Arm 0 0

sex sex 0 0

race race 1285 0

fu.time fu_time 0 0

fu.stat fu_stat 0 0

ps ps 1 1

hgb hgb 266 266

bmi bmi 0 0

alk.phos alk.phos 0 0

ast ast 1 0

mdquality.s mdquality.s 0 0

age.ord age.ord 0 0

First 10 differences detected per variable (1531 differences not shown)

var.x	var.y	case	values.x	values.y	row.x	row.y
race	race	76170	Caucasian	caucasian	26	20
race	race	76240	Caucasian	caucasian	27	21
race	race	76431	Caucasian	caucasian	28	22
race	race	76712	Caucasian	caucasian	29	23
race	race	76780	Caucasian	caucasian	30	24
race	race	77066	Caucasian	caucasian	31	25
race	race	77316	Caucasian	caucasian	32	26
race	race	77591	Caucasian	caucasian	34	28
race	race	77851	Caucasian	caucasian	35	29
race	race	77956	Caucasian	caucasian	36	30
ps	ps	86205	0 NA	6 3		
hgb	hgb	88714	NA	-9 192 186		
hgb	hgb	88955	NA	-9 204 198		
hgb	hgb	89549	NA	-9 229 223		
hgb	hgb	89563	NA	-9 231 225		
hgb	hgb	89584	NA	-9 237 231		
hgb	hgb	89591	NA	-9 238 232		
hgb	hgb	89595	NA	-9 239 233		
hgb	hgb	89647	NA	-9 243 237		
hgb	hgb	89665	NA	-9 244 238		
hgb	hgb	89827	NA	-9 255 249		
ast	ast	105271	100 36	3 2		

Non-identical attributes

var.x	var.y	name
arm	Arm	label
sex	sex	label
sex	sex	levels
race	race	class
race	race	label
race	race	levels
bmi	bmi	label

Character tolerance

Use the tol.char= argument to change how character variables are compared. By default, they are compared as

```

summary(compare(mockstudy, mockstudy2, by = "case",
               tol.vars = c(".", "case"), # dots=underscores=spaces, ignore case
               int.as.num = TRUE,         # compare integers and numerics
               tol.num.val = 10,          # allow absolute differences <= 10
               tol.factor = "labels",     # match only factor labels
               factor.as.char = TRUE,     # compare factors and characters
               tol.char = "case"         # ignore case in character vectors
))
Summary of data.frames
version arg ncol  nrow
x  mockstudy  14 1499
y  mockstudy2 13 1495
Variables not shared
version variable  position  class
x  age 2 integer
Other variables not compared
No other variables not compared
Observations not shared
version case  observation
x  88989 9
x  90158 8
x  99508 7
x  112263 5
Differences detected by variable
var.x  var.y  n  NAs
arm Arm 0 0
sex sex 0 0
race race 0 0
fu.time fu_time 0 0
fu.stat fu_stat 0 0
ps ps 1 1
hgb hgb 266 266
bmi bmi 0 0
alk.phos alk.phos 0 0
ast ast 1 0
mdquality.s mdquality.s 0 0
age.ord age.ord 0 0
First 10 differences detected per variable (256 differences not shown)
var.x  var.y  case  values.x  values.y  row.x  row.y
ps ps 86205 0 NA 6 3
hgb hgb 88714 NA -9 192 186
hgb hgb 88955 NA -9 204 198
hgb hgb 89549 NA -9 229 223
hgb hgb 89563 NA -9 231 225
hgb hgb 89584 NA -9 237 231
hgb hgb 89591 NA -9 238 232
hgb hgb 89595 NA -9 239 233
hgb hgb 89647 NA -9 243 237
hgb hgb 89665 NA -9 244 238
hgb hgb 89827 NA -9 255 249
ast ast 105271 100 36 3 2
Non-identical attributes
var.x  var.y  name
arm Arm label
sex sex label
sex sex levels
race race class

```

```

race    race    label
race    race    levels
bmi bmi label
Date tolerance

```

Use the `tol.date=` argument to change how dates are compared. By default, they're expected to be equal to each other.

Other data type tolerances

Use the `tol.other=` argument to change how other objects are compared. By default, they're expected to be identical.

User-defined tolerance functions

Details

The `comparison.control()` function accepts functions for any of the tolerance arguments in addition to the standard ones.

Any custom tolerance function must accept two vectors as arguments and return a logical vector of the same length.

CAUTION: the results should not include NAs, since the logical vector is used to subset the input data.frame.

`tol.NA`

```

function(x, y, idx)
{
  (is.na(x) & !is.na(y)) | (is.na(y) & !is.na(x)) | (!is.na(x) &
    !is.na(y) & idx)
}
<environment: namespace:arsenal>

```

The `tol.NA()` function is used in all default tolerance functions to help handle NAs.

Example 1

Suppose we want to ignore any dates which are later in the second dataset than the first. We define a custom tolerance function.

```

my.tol <- function(x, y, tol)
{
  tol.NA(x, y, x > y)
}

```

```

date.df1 <- data.frame(dt = as.Date(c("2017-09-07", "2017-08-08", "2017-07-09", NA)))
date.df2 <- data.frame(dt = as.Date(c("2017-10-01", "2017-08-08", "2017-07-10", "2017-01-01")))
n.diffs(compare(date.df1, date.df2)) # default finds any differences
[1] 3
n.diffs(compare(date.df1, date.df2, tol.date = my.tol)) # our function identifies only the NA as different.
[1] 1
n.diffs(compare(date.df2, date.df1, tol.date = my.tol)) # ... until we change the argument order
[1] 3

```

Example 2

(Continuing our mockstudy example)

Suppose we're okay with NAs getting replaced by -9.

```

tol.minus9 <- function(x, y, tol)
{
  idx1 <- is.na(x) & !is.na(y) & y == -9
  idx2 <- tol.num.absolute(x, y, tol) # find other absolute differences
  return(!idx1 & idx2)
}

```

```

summary(compare(mockstudy, mockstudy2, by = "case",
  tol.vars = c(".", "case"), # dots=underscores=spaces, ignore case
  int.as.num = TRUE,         # compare integers and numerics

```

```

        tol.num.val = 10,          # allow absolute differences <= 10
        tol.factor = "labels",    # match only factor labels
        factor.as.char = TRUE,    # compare factors and characters
        tol.char = "case",        # ignore case in character vectors
        tol.num = tol.minus9      # ignore NA -> -9 changes
    ))
Summary of data.frames
version arg ncol   nrow
x  mockstudy  14  1499
y  mockstudy2 13  1495
Variables not shared
version variable   position   class
x  age 2 integer
Other variables not compared
No other variables not compared
Observations not shared
version case      observation
x  88989 9
x  90158 8
x  99508 7
x  112263 5
Differences detected by variable
var.x var.y  n  NAs
arm Arm 0 0
sex sex 0 0
race race 0 0
fu.time fu_time 0 0
fu.stat fu_stat 0 0
ps ps 1 1
hgb hgb 0 0
bmi bmi 0 0
alk.phos alk.phos 0 0
ast ast 1 0
mdquality.s mdquality.s 0 0
age.ord age.ord 0 0
First 10 differences detected per variable
var.x var.y case values.x values.y row.x row.y
ps ps 86205 0 NA 6 3
ast ast 105271 100 36 3 2
Non-identical attributes
var.x var.y name
arm Arm label
sex sex label
sex sex levels
race race class
race race label
race race levels
bmi bmi label
Extract Differences
Differences can be easily extracted using the diffs() function. If you only want to determine how many diff

cmp <- compare(mockstudy, mockstudy2, by = "case", tol.vars = c("._ ", "case"), int.as.num = TRUE)
n.diffs(cmp)
[1] 1765
head(diffs(cmp))
  var.x var.y case values.x values.y row.x row.y
1  sex  sex 76170 Male Male 26 20

```

```

2  sex  sex 76240    Male    Male    27    21
3  sex  sex 76431    Female  Female    28    22
4  sex  sex 76712    Male    Male    29    23
5  sex  sex 76780    Female  Female    30    24
6  sex  sex 77066    Female  Female    31    25

```

Differences can also be summarized by variable.

```

diffs(cmp, by.var = TRUE)
      var.x      var.y      n NAs
1      arm      Arm      0  0
2      sex      sex 1495  0
3  fu.time  fu_time      0  0
4  fu.stat  fu_stat      0  0
5      ps      ps      1  1
6      hgb      hgb    266 266
7      bmi      bmi      0  0
8  alk.phos  alk.phos      0  0
9      ast      ast      3  0
10 mdquality.s mdquality.s      0  0
11  age.ord  age.ord      0  0

```

To report differences from only a few variables, one can pass a list of variable names to `diffs()`.

```

diffs(cmp, vars = c("ps", "ast"), by.var = TRUE)
      var.x var.y n NAs
5      ps   ps 1  1
9      ast  ast 3  0
diffs(cmp, vars = c("ps", "ast"))
      var.x var.y case values.x values.y row.x row.y
1496   ps   ps 86205      0      NA      6      3
1763   ast  ast 86205     27     36      6      3
1764   ast  ast 105271    100     36      3      2
1765   ast  ast 110754     35     36      1      1

```

Appendix

Structure of the Object

(This section is just as much for my use as for yours!)

```
obj <- compare(mockstudy, mockstudy2, by = "case")
```

There are two main objects in the "compare.data.frame" object, each with its own print method.

The `frame.summary` contains:

the substituted-deparsed arguments

information about the number of columns and rows in each dataset

the by-variables for each dataset (which may not be the same)

the attributes for each dataset (which get counted in the print method)

a data.frame of by-variables and row numbers of observations not shared between datasets

the number of shared observations

```

print(obj$frame.summary)
      version      arg ncol nrow  by      attrs      unique n.shared
1          x mockstudy   14 1499 case 3 attributes 4 unique obs    1495
2          y mockstudy2   13 1495 case 3 attributes 0 unique obs    1495

```

The `vars.summary` contains:

variable name, column number, and class vector (with possibly more than one element) for each x and y. These

values, a list-column of the text string "by-variable" for the by-variables, NULL for columns that aren't c

The by-variables for differences found

The values which are different for x and y

The row numbers for differences found

attrs, a list-column of NULL if there are no attributes, or a data.frame containing:

The name of the attributes

The attributes for x and y, set to NA if non-existent

The actual attributes (if `show.attr=TRUE`).

```
print(obj$vars.summary)
```

	var.x	pos.x	class.x	var.y	pos.y	class.y	values	attrs
8	case	1	integer	case	1	integer	by-variable 0	attributes
17	sex	4	factor	sex	2	factor	1495 differences 2	attributes
16	race	5	character	race	3	factor	Not compared 3	attributes
15	ps	8	integer	ps	4	integer	1 differences 0	attributes
13	hgb	9	numeric	hgb	5	numeric	266 differences 0	attributes
7	bmi	10	numeric	bmi	6	numeric	0 differences 1	attributes
4	alk.phos	11	integer	alk.phos	7	integer	0 differences 0	attributes
6	ast	12	integer	ast	8	numeric	Not compared 0	attributes
14	mdquality.s	13	integer	mdquality.s	9	integer	0 differences 0	attributes
3	age.ord	14	ordered, factor	age.ord	10	ordered, factor	0 differences 0	attributes
2	age	2	integer	<NA>	NA	NA	Not compared 0	attributes
5	arm	3	character	<NA>	NA	NA	Not compared 0	attributes
11	fu.time	6	integer	<NA>	NA	NA	Not compared 0	attributes
10	fu.stat	7	integer	<NA>	NA	NA	Not compared 0	attributes
12	<NA>	NA	NA	fu_time	11	integer	Not compared 0	attributes
9	<NA>	NA	NA	fu stat	12	integer	Not compared 0	attributes
1	<NA>	NA	NA	Arm	13	character	Not compared 0	attributes

The `freqlist` function

<https://cran.r-project.org/web/packages/arsenal/vignettes/freqlist.html>

The `freqlist` function

Tina Gunderson and Ethan Heinzen

09 November, 2018

Overview

Sample dataset

The `freqlist` object

Basic output using `summary()`

Using a formula with `freqlist`

Rounding percentage digits or changing variable names for printing

Additional examples

Including combinations with frequencies of zero

Options for NA handling

Frequency counts and percentages subset by factor levels

Change labels on the fly

Using xtable() to format and print freqlist() results

Use freqlist in bookdown

Appendix: Notes regarding table options in R

NAs

Table dimname names (dnn)

Overview

freqlist() is a function meant to produce output similar to SAS's PROC FREQ procedure when using the /list

```
require(arsenal)
```

Sample dataset

For our examples, we'll load the mockstudy data included with this package and use it to create a basic table

```
# load the data
```

```
data(mockstudy)
```

```
# retain NAs when creating the table using the useNA argument
```

```
tab.ex <- table(mockstudy[, c("arm", "sex", "mdquality.s")], useNA = "ifany")
```

The freqlist object

The freqlist() function returns an object of class "freqlist", which has three parts: freqlist, byVar, and

freqlist is a single data frame containing all contingency tables with calculated frequencies, cumulative frequencies,

byVar and labels are used in the summary method for subgroups and variable names, which will be covered in

Note that freqlist() is an S3 generic, with methods for tables and formulas.

```
noby <- freqlist(tab.ex)
```

```
str(noby)
```

List of 3

```
$ freqlist:'data.frame': 18 obs. of 7 variables:
```

```
..$ arm : Factor w/ 3 levels "A: IFL","F: FOLFOX",...: 1 1 1 1 1 1 2 2 2 2 ...
```

```
..$ sex : Factor w/ 2 levels "Male","Female": 1 1 1 2 2 2 1 1 1 2 ...
```

```
..$ mdquality.s: Factor w/ 2 levels "0","1": 1 2 NA 1 2 NA 1 2 NA 1 ...
```

```
..$ Freq : int [1:18] 29 214 34 12 118 21 31 285 95 21 ...
```

```
..$ cumFreq : int [1:18] 29 243 277 289 407 428 459 744 839 860 ...
```

```
..$ freqPercent: num [1:18] 1.93 14.28 2.27 0.8 7.87 ...
```

```
..$ cumPercent : num [1:18] 1.93 16.21 18.48 19.28 27.15 ...
```

```
$ byVar : NULL
```

```
$ labels : NULL
```

```
- attr(*, "class")= chr "freqlist"
```

view the data frame portion of freqlist output

```
head(noby[["freqlist"]]) ## or use as.data.frame(noby)
```

	arm	sex	mdquality.s	Freq	cumFreq	freqPercent	cumPercent
1	A: IFL	Male	0	29	29	1.93	1.93
2	A: IFL	Male	1	214	243	14.28	16.21
3	A: IFL	Male	<NA>	34	277	2.27	18.48
4	A: IFL	Female	0	12	289	0.80	19.28
5	A: IFL	Female	1	118	407	7.87	27.15
6	A: IFL	Female	<NA>	21	428	1.40	28.55

Basic output using summary()

The summary method for freqlist() relies on the kable() function (in the knitr package) for printing. knitr

Note that you must supply `results="asis"` to properly format the markdown output.

```
summary(noby)
arm sex mdquality.s Freq cumFreq freqPercent cumPercent
A: IFL Male 0 29 29 1.93 1.93
1 214 243 14.28 16.21
NA 34 277 2.27 18.48
Female 0 12 289 0.80 19.28
1 118 407 7.87 27.15
NA 21 428 1.40 28.55
F: FOLFOX Male 0 31 459 2.07 30.62
1 285 744 19.01 49.63
NA 95 839 6.34 55.97
Female 0 21 860 1.40 57.37
1 198 1058 13.21 70.58
NA 61 1119 4.07 74.65
G: IROX Male 0 17 1136 1.13 75.78
1 187 1323 12.47 88.26
NA 24 1347 1.60 89.86
Female 0 14 1361 0.93 90.79
1 121 1482 8.07 98.87
NA 17 1499 1.13 100.00
You can print a title for the table using the title= argument.
```

```
summary(noby, title = "Basic freqlist output")
Basic freqlist output
arm sex mdquality.s Freq cumFreq freqPercent cumPercent
A: IFL Male 0 29 29 1.93 1.93
1 214 243 14.28 16.21
NA 34 277 2.27 18.48
Female 0 12 289 0.80 19.28
1 118 407 7.87 27.15
NA 21 428 1.40 28.55
F: FOLFOX Male 0 31 459 2.07 30.62
1 285 744 19.01 49.63
NA 95 839 6.34 55.97
Female 0 21 860 1.40 57.37
1 198 1058 13.21 70.58
NA 61 1119 4.07 74.65
G: IROX Male 0 17 1136 1.13 75.78
1 187 1323 12.47 88.26
NA 24 1347 1.60 89.86
Female 0 14 1361 0.93 90.79
1 121 1482 8.07 98.87
NA 17 1499 1.13 100.00
```

You can also easily pull out the `freqlist` data frame for more complicated formatting or manipulation (e.g.

```
head(as.data.frame(noby))
  arm sex mdquality.s Freq cumFreq freqPercent cumPercent
1 A: IFL Male 0 29 29 1.93 1.93
2 A: IFL Male 1 214 243 14.28 16.21
3 A: IFL Male <NA> 34 277 2.27 18.48
4 A: IFL Female 0 12 289 0.80 19.28
5 A: IFL Female 1 118 407 7.87 27.15
6 A: IFL Female <NA> 21 428 1.40 28.55
```

Using a formula with `freqlist`

Instead of passing a pre-computed table to `freqlist()`, you can instead pass a formula, which will be in turn

Note that the `addNA=` argument was added to `xtabs()` in R 3.4.0. In previous versions, NAs have to be added to

```
### this works in R >= 3.4.0 summary(freqlist(~ arm + sex + mdquality.s, data =
### mockstudy, addNA = TRUE))
```

```
### This one is backwards-compatible
summary(freqlist(~arm + sex + addNA(mdquality.s), data = mockstudy))
```

arm	sex	addNA.mdquality.s.	Freq	cumFreq	freqPercent	cumPercent
:-----	:-----	:-----	----:	-----:	-----:	-----:
A: IFL	Male	0	29	29	1.93	1.93
		1	214	243	14.28	16.21
		NA	34	277	2.27	18.48
	Female	0	12	289	0.80	19.28
		1	118	407	7.87	27.15
		NA	21	428	1.40	28.55
F: FOLFOX	Male	0	31	459	2.07	30.62
		1	285	744	19.01	49.63
		NA	95	839	6.34	55.97
	Female	0	21	860	1.40	57.37
		1	198	1058	13.21	70.58
		NA	61	1119	4.07	74.65
G: IROX	Male	0	17	1136	1.13	75.78
		1	187	1323	12.47	88.26
		NA	24	1347	1.60	89.86
	Female	0	14	1361	0.93	90.79
		1	121	1482	8.07	98.87
		NA	17	1499	1.13	100.00

One can also set NAs to an explicit value using `includeNA()`.

```
summary(freqlist(~arm + sex + includeNA(mdquality.s, "Missing"), data = mockstudy))
```

arm	sex	includeNA.mdquality.s...Missing..	Freq	cumFreq	freqPercent	cumPercent
:-----	:-----	:-----	----:	-----:	-----:	-----:
A: IFL	Male	0	29	29	1.93	1.93
		1	214	243	14.28	16.21
		Missing	34	277	2.27	18.48
	Female	0	12	289	0.80	19.28
		1	118	407	7.87	27.15
		Missing	21	428	1.40	28.55
F: FOLFOX	Male	0	31	459	2.07	30.62
		1	285	744	19.01	49.63
		Missing	95	839	6.34	55.97
	Female	0	21	860	1.40	57.37
		1	198	1058	13.21	70.58
		Missing	61	1119	4.07	74.65
G: IROX	Male	0	17	1136	1.13	75.78
		1	187	1323	12.47	88.26
		Missing	24	1347	1.60	89.86
	Female	0	14	1361	0.93	90.79
		1	121	1482	8.07	98.87
		Missing	17	1499	1.13	100.00

Rounding percentage digits or changing variable names for printing

The `digits=` argument takes a single numeric value and controls the rounding of percentages in the output. The

```
withnames <- freqlist(tab.ex, labelTranslations = c("Treatment Arm", "Gender", "LASA QOL"),
  digits = 0)
```

```
summary(withnames)
```

```
Treatment Arm  Gender  LASA QOL  Freq  cumFreq freqPercent cumPercent
```

```
A: IFL Male 0 29 29 2 2
```

```
1 214 243 14 16
```

```
NA 34 277 2 18
```

```
Female 0 12 289 1 19
```

```
1 118 407 8 27
```

```
NA 21 428 1 29
```

```
F: FOLFOX Male 0 31 459 2 31
```

```
1 285 744 19 50
```

```
NA 95 839 6 56
```

```
Female 0 21 860 1 57
```

```
1 198 1058 13 71
```

```
NA 61 1119 4 75
```

```
G: IROX Male 0 17 1136 1 76
```

```
1 187 1323 12 88
```

```
NA 24 1347 2 90
```

```
Female 0 14 1361 1 91
```

```
1 121 1482 8 99
```

```
NA 17 1499 1 100
```

Additional examples

Including combinations with frequencies of zero

The `sparse=` argument takes a single logical value as input. The default option is FALSE. If set to TRUE, the

```
summary(freqlist(~race + sex + arm, data = mockstudy, sparse = TRUE, digits = 1))
```

```
race sex arm Freq cumFreq freqPercent cumPercent
```

```
African-Am Male A: IFL 25 25 1.7 1.7
```

```
F: FOLFOX 24 49 1.6 3.3
```

```
G: IROX 16 65 1.1 4.4
```

```
Female A: IFL 14 79 0.9 5.3
```

```
F: FOLFOX 25 104 1.7 7.0
```

```
G: IROX 11 115 0.7 7.7
```

```
Asian Male A: IFL 0 115 0.0 7.7
```

```
F: FOLFOX 10 125 0.7 8.4
```

```
G: IROX 1 126 0.1 8.4
```

```
Female A: IFL 1 127 0.1 8.5
```

```
F: FOLFOX 4 131 0.3 8.8
```

```
G: IROX 2 133 0.1 8.9
```

```
Caucasian Male A: IFL 240 373 16.1 25.0
```

```
F: FOLFOX 352 725 23.6 48.6
```

```
G: IROX 195 920 13.1 61.7
```

```
Female A: IFL 131 1051 8.8 70.4
```

```
F: FOLFOX 234 1285 15.7 86.1
```

```
G: IROX 136 1421 9.1 95.2
```

```
Hawaii/Pacific Male A: IFL 1 1422 0.1 95.3
```

```
F: FOLFOX 1 1423 0.1 95.4
```

```
G: IROX 0 1423 0.0 95.4
```

```
Female A: IFL 0 1423 0.0 95.4
```

```
F: FOLFOX 2 1425 0.1 95.5
```

```
G: IROX 1 1426 0.1 95.6
```

```
Hispanic Male A: IFL 8 1434 0.5 96.1
```

```
F: FOLFOX 17 1451 1.1 97.3
```

```
G: IROX 12 1463 0.8 98.1
```

```

Female A: IFL 4 1467 0.3 98.3
F: FOLFOX 11 1478 0.7 99.1
G: IROX 2 1480 0.1 99.2
Native-Am/Alaska Male A: IFL 1 1481 0.1 99.3
F: FOLFOX 0 1481 0.0 99.3
G: IROX 2 1483 0.1 99.4
Female A: IFL 1 1484 0.1 99.5
F: FOLFOX 1 1485 0.1 99.5
G: IROX 0 1485 0.0 99.5
Other Male A: IFL 2 1487 0.1 99.7
F: FOLFOX 2 1489 0.1 99.8
G: IROX 1 1490 0.1 99.9
Female A: IFL 0 1490 0.0 99.9
F: FOLFOX 2 1492 0.1 100.0
G: IROX 0 1492 0.0 100.0

```

Options for NA handling

The various na.options= allow you to include or exclude data with missing values for one or more factor levels

```

summary(freqlist(tab.ex, na.options = "include"))
arm sex mdquality.s Freq cumFreq freqPercent cumPercent
A: IFL Male 0 29 29 1.93 1.93
1 214 243 14.28 16.21
NA 34 277 2.27 18.48
Female 0 12 289 0.80 19.28
1 118 407 7.87 27.15
NA 21 428 1.40 28.55
F: FOLFOX Male 0 31 459 2.07 30.62
1 285 744 19.01 49.63
NA 95 839 6.34 55.97
Female 0 21 860 1.40 57.37
1 198 1058 13.21 70.58
NA 61 1119 4.07 74.65
G: IROX Male 0 17 1136 1.13 75.78
1 187 1323 12.47 88.26
NA 24 1347 1.60 89.86
Female 0 14 1361 0.93 90.79
1 121 1482 8.07 98.87
NA 17 1499 1.13 100.00
summary(freqlist(tab.ex, na.options = "showexclude"))
arm sex mdquality.s Freq cumFreq freqPercent cumPercent
A: IFL Male 0 29 29 2.33 2.33
1 214 243 17.16 19.49
NA 34 NA NA NA
Female 0 12 255 0.96 20.45
1 118 373 9.46 29.91
NA 21 NA NA NA
F: FOLFOX Male 0 31 404 2.49 32.40
1 285 689 22.85 55.25
NA 95 NA NA NA
Female 0 21 710 1.68 56.94
1 198 908 15.88 72.81
NA 61 NA NA NA
G: IROX Male 0 17 925 1.36 74.18
1 187 1112 15.00 89.17
NA 24 NA NA NA
Female 0 14 1126 1.12 90.30
1 121 1247 9.70 100.00

```

```

NA 17 NA NA NA
summary(freqlist(tab.ex, na.options = "remove"))
arm sex mdquality.s Freq cumFreq freqPercent cumPercent
A: IFL Male 0 29 29 2.33 2.33
1 214 243 17.16 19.49
Female 0 12 255 0.96 20.45
1 118 373 9.46 29.91
F: FOLFOX Male 0 31 404 2.49 32.40
1 285 689 22.85 55.25
Female 0 21 710 1.68 56.94
1 198 908 15.88 72.81
G: IROX Male 0 17 925 1.36 74.18
1 187 1112 15.00 89.17
Female 0 14 1126 1.12 90.30
1 121 1247 9.70 100.00
Frequency counts and percentages subset by factor levels
The groupBy= argument internally subsets the data by the specified factor prior to calculating cumulative counts

withby <- freqlist(tab.ex, groupBy = c("arm", "sex"))
summary(withby)
arm sex mdquality.s Freq cumFreq freqPercent cumPercent
A: IFL Male 0 29 29 10.47 10.47
1 214 243 77.26 87.73
NA 34 277 12.27 100.00
arm sex mdquality.s Freq cumFreq freqPercent cumPercent
A: IFL Female 0 12 12 7.95 7.95
1 118 130 78.15 86.09
NA 21 151 13.91 100.00
arm sex mdquality.s Freq cumFreq freqPercent cumPercent
F: FOLFOX Male 0 31 31 7.54 7.54
1 285 316 69.34 76.89
NA 95 411 23.11 100.00
arm sex mdquality.s Freq cumFreq freqPercent cumPercent
F: FOLFOX Female 0 21 21 7.50 7.50
1 198 219 70.71 78.21
NA 61 280 21.79 100.00
arm sex mdquality.s Freq cumFreq freqPercent cumPercent
G: IROX Male 0 17 17 7.46 7.46
1 187 204 82.02 89.47
NA 24 228 10.53 100.00
arm sex mdquality.s Freq cumFreq freqPercent cumPercent
G: IROX Female 0 14 14 9.21 9.21
1 121 135 79.61 88.82
NA 17 152 11.18 100.00
# using the single = TRUE argument will collapse results into a single table for
# printing
summary(withby, single = TRUE)
arm sex mdquality.s Freq cumFreq freqPercent cumPercent
A: IFL Male 0 29 29 10.47 10.47
1 214 243 77.26 87.73
NA 34 277 12.27 100.00
Female 0 12 12 7.95 7.95
1 118 130 78.15 86.09
NA 21 151 13.91 100.00
F: FOLFOX Male 0 31 31 7.54 7.54
1 285 316 69.34 76.89
NA 95 411 23.11 100.00

```

```
Female 0 21 21 7.50 7.50
1 198 219 70.71 78.21
NA 61 280 21.79 100.00
G: IROX Male 0 17 17 7.46 7.46
1 187 204 82.02 89.47
NA 24 228 10.53 100.00
Female 0 14 14 9.21 9.21
1 121 135 79.61 88.82
NA 17 152 11.18 100.00
```

Change labels on the fly

At this time, the labels can be changed just for the variables (e.g. not the frequency columns).

```
labels(noby) <- c("Arm", "Sex", "QOL")
summary(noby)
Arm Sex QOL Freq cumFreq freqPercent cumPercent
A: IFL Male 0 29 29 1.93 1.93
1 214 243 14.28 16.21
NA 34 277 2.27 18.48
Female 0 12 289 0.80 19.28
1 118 407 7.87 27.15
NA 21 428 1.40 28.55
F: FOLFOX Male 0 31 459 2.07 30.62
1 285 744 19.01 49.63
NA 95 839 6.34 55.97
Female 0 21 860 1.40 57.37
1 198 1058 13.21 70.58
NA 61 1119 4.07 74.65
G: IROX Male 0 17 1136 1.13 75.78
1 187 1323 12.47 88.26
NA 24 1347 1.60 89.86
Female 0 14 1361 0.93 90.79
1 121 1482 8.07 98.87
NA 17 1499 1.13 100.00
You can also supply labelTranslations= to summary().
```

```
summary(noby, labelTranslations = c("Arm", "Sex", "QOL"))
Arm Sex QOL Freq cumFreq freqPercent cumPercent
A: IFL Male 0 29 29 1.93 1.93
1 214 243 14.28 16.21
NA 34 277 2.27 18.48
Female 0 12 289 0.80 19.28
1 118 407 7.87 27.15
NA 21 428 1.40 28.55
F: FOLFOX Male 0 31 459 2.07 30.62
1 285 744 19.01 49.63
NA 95 839 6.34 55.97
Female 0 21 860 1.40 57.37
1 198 1058 13.21 70.58
NA 61 1119 4.07 74.65
G: IROX Male 0 17 1136 1.13 75.78
1 187 1323 12.47 88.26
NA 24 1347 1.60 89.86
Female 0 14 1361 0.93 90.79
1 121 1482 8.07 98.87
NA 17 1499 1.13 100.00
```

Using `xtable()` to format and print `freqlist()` results

Fair warning: `xtable()` has kind of a steep learning curve. These examples are given without explanation, for

```
require(xtable)
Loading required package: xtable
# set up custom function for xtable text
italic <- function(x) {
  paste0("<i>", x, "</i>")
}
xftbl <- xtable(noby[["freqlist"]], caption = "xtable formatted output of freqlist data frame",
  align = "|r|r|r|r|c|c|c|r|")
```

```
# change the column names
names(xftbl)[1:3] <- c("Arm", "Gender", "LASA QOL")
```

```
print(xftbl, sanitize.colnames.function = italic, include.rownames = FALSE, type = "html",
  comment = FALSE)
```

```
xtable formatted output of freqlist data frame
```

Arm	Gender	LASA QOL	Freq	cumFreq	freqPercent	cumPercent
A: IFL	Male	0	29	29	1.93	1.93
A: IFL	Male	1	214	243	14.28	16.21
A: IFL	Male		34	277	2.27	18.48
A: IFL	Female	0	12	289	0.80	19.28
A: IFL	Female	1	118	407	7.87	27.15
A: IFL	Female		21	428	1.40	28.55
F: FOLFOX	Male	0	31	459	2.07	30.62
F: FOLFOX	Male	1	285	744	19.01	49.63
F: FOLFOX	Male		95	839	6.34	55.97
F: FOLFOX	Female	0	21	860	1.40	57.37
F: FOLFOX	Female	1	198	1058	13.21	70.58
F: FOLFOX	Female		61	1119	4.07	74.65
G: IROX	Male	0	17	1136	1.13	75.78
G: IROX	Male	1	187	1323	12.47	88.26
G: IROX	Male		24	1347	1.60	89.86
G: IROX	Female	0	14	1361	0.93	90.79
G: IROX	Female	1	121	1482	8.07	98.87
G: IROX	Female		17	1499	1.13	100.00

```
Use freqlist in bookdown
```

```
Since the backbone of freqlist() is knitr::kable(), tables still render well in bookdown. However, print.su
```

```
summary(freqlist(~sex + age, data = mockstudy), title = "(\\#tab:mytableby) Caption here")
```

```
Appendix: Notes regarding table options in R
```

```
NAs
```

```
There are several widely used options for basic tables in R. The table() function in base R is probably the
```

```
# base table default removes NAs
```

```
tab.d1 <- base::table(mockstudy[, c("arm", "sex", "mdquality.s")], useNA = "ifany")
```

```
tab.d1
```

```
, , mdquality.s = 0
```

	sex	
arm	Male	Female
A: IFL	29	12
F: FOLFOX	31	21
G: IROX	17	14

```
, , mdquality.s = 1
```

```
sex
```


arm		Male	Female
A: IFL		214	118
F: FOLFOX		285	198
G: IROX		187	121

```
, , mdquality.s = NA
```

	sex		
arm		Male	Female
A: IFL		34	21
F: FOLFOX		95	61
G: IROX		24	17

xtabs() is similar to table(), but uses a formula-based syntax. However, there is not an option for retaining

```
# without specifying addNA
```

```
tab.d2 <- xtabs(formula = ~arm + sex + mdquality.s, data = mockstudy)
```

```
tab.d2
```

```
, , mdquality.s = 0
```

	sex		
arm		Male	Female
A: IFL		29	12
F: FOLFOX		31	21
G: IROX		17	14

```
, , mdquality.s = 1
```

	sex		
arm		Male	Female
A: IFL		214	118
F: FOLFOX		285	198
G: IROX		187	121

```
# now with addNA
```

```
tab.d3 <- xtabs(~arm + sex + addNA(mdquality.s), data = mockstudy)
```

```
tab.d3
```

```
, , addNA(mdquality.s) = 0
```

	sex		
arm		Male	Female
A: IFL		29	12
F: FOLFOX		31	21
G: IROX		17	14

```
, , addNA(mdquality.s) = 1
```

	sex		
arm		Male	Female
A: IFL		214	118
F: FOLFOX		285	198
G: IROX		187	121

```
, , addNA(mdquality.s) = NA
```

	sex		
arm		Male	Female
A: IFL		34	21
F: FOLFOX		95	61

```
G: IROX      24      17
```

Since the formula method of `freqlist()` uses `xtabs()`, NAs should be treated in the same way. `includeNA()` can

Table `dimname` names (`dnn`)

Supplying a `data.frame` to the `table()` function without giving columns individually will create a contingency

However, if the columns of a `data.frame` or `matrix` are supplied separately (i.e., as vectors), column names

```
# providing variables separately (as vectors) drops column names
```

```
tab.d4 <- base::table(mockstudy$arm, mockstudy$sex, mockstudy$mdquality.s)
```

```
tab.d4
```

```
, , = 0
```

```
      Male Female
```

```
A: IFL      29      12
```

```
F: FOLFOX   31      21
```

```
G: IROX     17      14
```

```
, , = 1
```

```
      Male Female
```

```
A: IFL     214     118
```

```
F: FOLFOX  285     198
```

```
G: IROX    187     121
```

If desired, you can use the `dnn=` argument to pass variable names.

```
# add the column name labels back using dnn option in base::table
```

```
tab.dnn <- base::table(mockstudy$arm, mockstudy$sex, mockstudy$mdquality.s, dnn = c("Arm",
      "Sex", "QOL"))
```

```
tab.dnn
```

```
, , QOL = 0
```

```
      Sex
```

```
Arm      Male Female
```

```
A: IFL      29      12
```

```
F: FOLFOX   31      21
```

```
G: IROX     17      14
```

```
, , QOL = 1
```

```
      Sex
```

```
Arm      Male Female
```

```
A: IFL     214     118
```

```
F: FOLFOX  285     198
```

```
G: IROX    187     121
```

If using `freqlist()`, you can provide the labels directly to `freqlist()` or to `summary()` using `labelTranslation`

```
---
```

```
## A Few Notes on Labels
```

<https://cran.r-project.org/web/packages/arsenal/vignettes/labels.html>

A Few Notes on Labels

Ethan Heinzen

09 November, 2018

Introduction

Examples

Set labels in the function call

Modify labels after the fact

Add labels to a data.frame

Introduction

The arsenal package relies somewhat heavily on variable labels to make output more "pretty". A label here is

We'll use the mockstudy dataset for all examples here:

```
library(arsenal)
```

```
data(mockstudy)
```

```
library(magrittr)
```

```
# for 'freqlist' examples
```

```
tab.ex <- table(mockstudy[, c("arm", "sex", "mdquality.s")], useNA="ifany")
```

Examples

Set labels in the function call

The summary() method for tableby(), modelsum(), and freqlist() objects contains a labelTranslations = argument

```
summary(freqlist(tab.ex),
```

```
  labelTranslations = c("Treatment Arm", "Gender", "LASA QOL"))
```

```
Treatment Arm  Gender  LASA QOL  Freq  cumFreq freqPercent cumPercent
```

```
A: IFL Male 0 29 29 1.93 1.93
```

```
1 214 243 14.28 16.21
```

```
NA 34 277 2.27 18.48
```

```
Female 0 12 289 0.80 19.28
```

```
1 118 407 7.87 27.15
```

```
NA 21 428 1.40 28.55
```

```
F: FOLFOX Male 0 31 459 2.07 30.62
```

```
1 285 744 19.01 49.63
```

```
NA 95 839 6.34 55.97
```

```
Female 0 21 860 1.40 57.37
```

```
1 198 1058 13.21 70.58
```

```
NA 61 1119 4.07 74.65
```

```
G: IROX Male 0 17 1136 1.13 75.78
```

```
1 187 1323 12.47 88.26
```

```
NA 24 1347 1.60 89.86
```

```
Female 0 14 1361 0.93 90.79
```

```
1 121 1482 8.07 98.87
```

```
NA 17 1499 1.13 100.00
```

```
summary(tableby(arm ~ sex + age, data = mockstudy),
```

```
  labelTranslations = c(sex = "SEX", age = "Age, yrs"))
```

```
A: IFL (N=428) F: FOLFOX (N=691) G: IROX (N=380) Total (N=1499) p value
```

```
SEX 0.190
```

```
Male 277 (64.7%) 411 (59.5%) 228 (60.0%) 916 (61.1%)
```

```
Female 151 (35.3%) 280 (40.5%) 152 (40.0%) 583 (38.9%)
```

```
Age, yrs 0.614
```

```
Mean (SD) 59.673 (11.365) 60.301 (11.632) 59.763 (11.499) 59.985 (11.519)
```

```
Range 27.000 - 88.000 19.000 - 88.000 26.000 - 85.000 19.000 - 88.000
```

```
summary(modelsum(bmi ~ age, adjust = ~sex, data = mockstudy),
```

```
  labelTranslations = list(sexFemale = "Female", age = "Age, yrs"))
```

```
estimate std.error p.value adj.r.squared
```

```
(Intercept) 26.793 0.766 < 0.001 0.004
```

```
Age, yrs 0.012 0.012 0.348
```

```
Female -0.718 0.291 0.014
```

```
Modify labels after the fact
```

Another option is to add labels after you have created the object. To do this, you can use the form `labels()`

```
# the non-pipe version; somewhat clunky
```

```
tmp <- freqlist(tab.ex)
```

```
labels(tmp) <- c("Treatment Arm", "Gender", "LASA QOL")
```

```
summary(tmp)
```

```
Treatment Arm Gender LASA QOL Freq cumFreq freqPercent cumPercent
```

```
A: IFL Male 0 29 29 1.93 1.93
```

```
1 214 243 14.28 16.21
```

```
NA 34 277 2.27 18.48
```

```
Female 0 12 289 0.80 19.28
```

```
1 118 407 7.87 27.15
```

```
NA 21 428 1.40 28.55
```

```
F: FOLFOX Male 0 31 459 2.07 30.62
```

```
1 285 744 19.01 49.63
```

```
NA 95 839 6.34 55.97
```

```
Female 0 21 860 1.40 57.37
```

```
1 198 1058 13.21 70.58
```

```
NA 61 1119 4.07 74.65
```

```
G: IROX Male 0 17 1136 1.13 75.78
```

```
1 187 1323 12.47 88.26
```

```
NA 24 1347 1.60 89.86
```

```
Female 0 14 1361 0.93 90.79
```

```
1 121 1482 8.07 98.87
```

```
NA 17 1499 1.13 100.00
```

```
# piped--much cleaner
```

```
mockstudy %>%
```

```
  tableby(arm ~ sex + age, data = .) %>%
```

```
  set_labels(c(sex = "SEX", age = "Age, yrs")) %>%
```

```
  summary()
```

```
A: IFL (N=428) F: FOLFOX (N=691) G: IROX (N=380) Total (N=1499) p value
```

```
SEX 0.190
```

```
Male 277 (64.7%) 411 (59.5%) 228 (60.0%) 916 (61.1%)
```

```
Female 151 (35.3%) 280 (40.5%) 152 (40.0%) 583 (38.9%)
```

```
Age, yrs 0.614
```

```
Mean (SD) 59.673 (11.365) 60.301 (11.632) 59.763 (11.499) 59.985 (11.519)
```

```
Range 27.000 - 88.000 19.000 - 88.000 26.000 - 85.000 19.000 - 88.000
```

```
mockstudy %>%
```

```
  modelsum(bmi ~ age, adjust = ~ sex, data = .) %>%
```

```
  set_labels(list(sexFemale = "Female", age = "Age, yrs")) %>%
```

```
  summary()
```

```
estimate std.error p.value adj.r.squared
```

```
(Intercept) 26.793 0.766 < 0.001 0.004
```

```
Age, yrs 0.012 0.012 0.348
```

```
Female -0.718 0.291 0.014
```

```
Add labels to a data.frame
```

`tableby()` and `modelsum()` also allow you to have label attributes on the data. Note that by default these at

```
mockstudy.lab <- keep.labels(mockstudy)
```

You can set attributes one at a time in two ways:

```
attr(mockstudy.lab$sex, "label") <- "Sex"
```

```
labels(mockstudy.lab$age) <- "Age, yrs"
```

...or all at once:

```
labels(mockstudy.lab) <- list(sex = "Sex", age = "Age, yrs")
summary(tableby(arm ~ sex + age, data = mockstudy.lab))
A: IFL (N=428)  F: FOLFOX (N=691)  G: IROX (N=380) Total (N=1499)  p value
Sex           0.190
  Male 277 (64.7%) 411 (59.5%) 228 (60.0%) 916 (61.1%)
  Female 151 (35.3%) 280 (40.5%) 152 (40.0%) 583 (38.9%)
Age, yrs      0.614
  Mean (SD) 59.673 (11.365) 60.301 (11.632) 59.763 (11.499) 59.985 (11.519)
  Range 27.000 - 88.000 19.000 - 88.000 26.000 - 85.000 19.000 - 88.000
You can pipe this, too.
```

```
mockstudy %>%
  set_labels(list(sex = "SEX", age = "Age, yrs")) %>%
  modelsum(bmi ~ age, adjust = ~ sex, data = .) %>%
  summary()
estimate      std.error    p.value adj.r.squared
(Intercept) 26.793    0.766    < 0.001 0.004
Age, yrs     0.012     0.012    0.348
SEX Female   -0.718    0.291    0.014
To extract labels from a data.frame, simply use the labels() function:
```

```
labels(mockstudy.lab)
## $case
## NULL
##
## $age
## [1] "Age, yrs"
##
## $arm
## [1] "Treatment Arm"
##
## $sex
## [1] "Sex"
##
## $race
## [1] "Race"
##
## $fu.time
## NULL
##
## $fu.stat
## NULL
##
## $ps
## NULL
##
## $hgb
## NULL
##
## $bmi
## [1] "Body Mass Index (kg/m^2)"
##
## $alk.phos
## NULL
##
```

```
## $ast
## NULL
##
## $mdquality.s
## NULL
##
## $age.ord
## NULL
```

```
---
```

```
## The modelsum function
```

```
https://cran.r-project.org/web/packages/arsenal/vignettes/modelsum.html
```

```
The modelsum function
```

```
Beth Atkinson, Ethan Heinzen, Pat Votruba, Jason Sinnwell, Shannon McDonnell and Greg Dougherty  
09 November, 2018
```

```
Introduction
```

```
Simple Example
```

```
Pretty text version of table
```

```
Pretty Rmarkdown version of table
```

```
Data frame version of table
```

```
Add an adjustor to the model
```

```
Models for each endpoint type
```

```
Gaussian
```

```
Fit and summarize linear regression model
```

```
Extract data using the broom package
```

```
Create a summary table using modelsum
```

```
Binomial
```

```
Fit and summarize logistic regression model
```

```
Extract data using broom package
```

```
Create a summary table using modelsum
```

```
Survival
```

```
Fit and summarize a Cox regression model
```

```
Extract data using broom package
```

```
Create a summary table using modelsum
```

```
Poisson
```

```
Example 1: fit and summarize a Poisson regression model
```

```
Extract data using broom package
```

```
Create a summary table using modelsum
```

```
Example 2: fit and summarize a Poisson regression model
```

```
Extract data using broom package
```

```
Create a summary table using modelsum
```

```
Additional Examples
```

1. Change summary statistics globally
2. Add labels to independent variables
3. Don't show intercept values
4. Don't show results for adjustment variables
5. Summarize multiple variables without typing them out
6. Subset the dataset used in the analysis
7. Create combinations of variables on the fly
8. Transform variables on the fly
9. Change the ordering of the variables or delete a variable
10. Merge two modelsum objects together
11. Add a title to the table

12. Modify how missing values are treated
13. Modify the number of digits used
14. Use case-weights in the models
15. Use modelsum within an Sweave document
16. Export modelsum results to a .CSV file
17. Write modelsum object to a separate Word or HTML file
18. Use modelsum in R Shiny
23. Use modelsum in bookdown

Available Function Options

Summary statistics

modelsum.control settings

summary.modelsum settings

Introduction

Very often we are asked to summarize model results from multiple fits into a nice table. The endpoint might

In developing the modelsum function, the goal was to bring the best features of these macros into an R function

This report provides step-by-step directions for using the functions associated with modelsum. All functions

Simple Example

The first step when using the modelsum function is to load the arsenal package. All the examples in this re

```
> require(arsenal)
> data(mockstudy) # load data
> dim(mockstudy) # look at how many subjects and variables are in the dataset
[1] 1499 14
> # help(mockstudy) # learn more about the dataset and variables
> str(mockstudy) # quick look at the data
'data.frame': 1499 obs. of 14 variables:
 $ case      : int 110754 99706 105271 105001 112263 86205 99508 90158 88989 90515 ...
 $ age       : atomic 67 74 50 71 69 56 50 57 51 63 ...
 ..- attr(*, "label")= chr "Age in Years"
 $ arm       : atomic F: FOLFOX A: IFL A: IFL G: IROX ...
 ..- attr(*, "label")= chr "Treatment Arm"
 $ sex       : Factor w/ 2 levels "Male","Female": 1 2 2 2 2 1 1 1 2 1 ...
 $ race      : atomic Caucasian Caucasian Caucasian Caucasian ...
 ..- attr(*, "label")= chr "Race"
 $ fu.time   : int 922 270 175 128 233 120 369 421 387 363 ...
 $ fu.stat   : int 2 2 2 2 2 2 2 2 2 2 ...
 $ ps       : int 0 1 1 1 0 0 0 0 1 1 ...
 $ hgb       : num 11.5 10.7 11.1 12.6 13 10.2 13.3 12.1 13.8 12.1 ...
 $ bmi       : atomic 25.1 19.5 NA 29.4 26.4 ...
 ..- attr(*, "label")= chr "Body Mass Index (kg/m^2)"
 $ alk.phos  : int 160 290 700 771 350 569 162 152 231 492 ...
 $ ast       : int 35 52 100 68 35 27 16 12 25 18 ...
 $ mdquality.s: int NA 1 1 1 NA 1 1 1 1 1 ...
 $ age.ord   : Ord.factor w/ 8 levels "10-19"<"20-29"<...: 6 7 4 7 6 5 4 5 5 6 ...
```

To create a simple linear regression table (the default), use a formula statement to specify the variables

```
> tab1 <- modelsum(bmi ~ sex + age, data=mockstudy)
```

If you want to take a quick look at the table, you can use summary on your modelsum object and the table wi

Pretty text version of table

If you want a nicer version in your console window then adding the text=TRUE option.

```
> summary(tab1, text=TRUE)
```

```
|
|:-----|:-----|:-----|:-----|:-----|
|(Intercept) |27.491 |0.181 |< 0.001 |0.004 |
|sex Female | -0.731 |0.290 |0.012 | |
|(Intercept) |26.424 |0.752 |< 0.001 |0.000 |
|Age in Years |0.013 |0.012 |0.290 | |
```

Pretty Rmarkdown version of table

In order for the report to look nice within an R markdown (knitr) report, you just need to specify results=

```
> summary(tab1)
```

```
estimate      std.error    p.value adj.r.squared
(Intercept) 27.491 0.181 < 0.001 0.004
sex Female -0.731 0.290 0.012
(Intercept) 26.424 0.752 < 0.001 0.000
Age in Years 0.013 0.012 0.290
```

Data frame version of table

If you want a data.frame version, simply use as.data.frame.

```
> as.data.frame(tab1)
```

```
model      term      label term.type      estimate      std.error
1      1 (Intercept) (Intercept) Intercept 27.49147713 0.18134740
2      1 sexFemale   sex Female      Term -0.73105055 0.29032223
3      2 (Intercept) (Intercept) Intercept 26.42372272 0.75211474
4      2 age Age in Years      Term 0.01304859 0.01231653
      p.value adj.r.squared
1 0.000000e+00 3.632258e-03
2 1.190605e-02 3.632258e-03
3 1.279109e-196 8.354809e-05
4 2.895753e-01 8.354809e-05
```

Add an adjustor to the model

The argument adjust allows the user to indicate that all the variables should be adjusted for these terms.

```
> tab2 <- modelsum(alk.phos ~ arm + ps + hgb, adjust= ~age + sex, data=mockstudy)
```

```
> summary(tab2)
```

```
estimate      std.error    p.value adj.r.squared    Nmiss
(Intercept) 175.548 20.587 < 0.001 -0.001 0
Treatment Arm F: FOLFOX -13.701 8.730 0.117
Treatment Arm G: IROX -2.245 9.860 0.820
Age in Years -0.017 0.319 0.956
sex Female 3.016 7.521 0.688
(Intercept) 148.391 19.585 < 0.001 0.045 266
ps 46.721 5.987 < 0.001
Age in Years -0.084 0.311 0.787
sex Female 1.169 7.343 0.874
(Intercept) 336.554 32.239 < 0.001 0.031 266
hgb -13.845 2.137 < 0.001
Age in Years 0.095 0.314 0.763
sex Female -5.980 7.516 0.426
```

Models for each endpoint type

To make sure the correct model is run you need to specify "family". The options available right now are : g

Gaussian

Fit and summarize linear regression model

Look at whether there is any evidence that AlkPhos values vary by study arm after adjusting for sex and age

```
> fit <- lm(alk.phos ~ arm + age + sex, data=mockstudy)
```



```
> summary(fit)
```

```
Call:
```

```
lm(formula = alk.phos ~ arm + age + sex, data = mockstudy)
```

```
Residuals:
```

	Min	1Q	Median	3Q	Max
	-168.80	-81.45	-47.17	37.39	853.56

```
Coefficients:
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	175.54808	20.58665	8.527	<2e-16 ***
armF: FOLFOX	-13.70062	8.72963	-1.569	0.117
armG: IROX	-2.24498	9.86004	-0.228	0.820
age	-0.01741	0.31878	-0.055	0.956
sexFemale	3.01598	7.52097	0.401	0.688

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 128.5 on 1228 degrees of freedom
```

```
(266 observations deleted due to missingness)
```

```
Multiple R-squared:  0.002552, Adjusted R-squared:  -0.0006969
```

```
F-statistic: 0.7855 on 4 and 1228 DF, p-value: 0.5346
```

```
> plot(fit)
```

The results suggest that the endpoint may need to be transformed. Calculating the Box-Cox transformation su

```
> require(MASS)
```

```
> boxcox(fit)
```

```
> fit2 <- lm(log(alk.phos) ~ arm + age + sex, data=mockstudy)
```

```
> summary(fit2)
```

```
Call:
```

```
lm(formula = log(alk.phos) ~ arm + age + sex, data = mockstudy)
```

```
Residuals:
```

	Min	1Q	Median	3Q	Max
	-3.0098	-0.4470	-0.1065	0.4205	2.0620

```
Coefficients:
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	4.9692474	0.1025239	48.469	<2e-16 ***
armF: FOLFOX	-0.0766798	0.0434746	-1.764	0.078 .
armG: IROX	-0.0192828	0.0491041	-0.393	0.695
age	-0.0004058	0.0015876	-0.256	0.798
sexFemale	0.0179253	0.0374553	0.479	0.632

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.6401 on 1228 degrees of freedom
```

```
(266 observations deleted due to missingness)
```

```
Multiple R-squared:  0.003121, Adjusted R-squared:  -0.0001258
```

```
F-statistic: 0.9613 on 4 and 1228 DF, p-value: 0.4278
```

```
> plot(fit2)
```

Finally, look to see whether there there is a non-linear relationship with age.

```

> require(gam)
> fit3 <- lm(log(alk.phos) ~ arm + ns(age, df=2) + sex, data=mockstudy)
>
> # test whether there is a difference between models
> stats::anova(fit2,fit3)
Analysis of Variance Table

Model 1: log(alk.phos) ~ arm + age + sex
Model 2: log(alk.phos) ~ arm + ns(age, df = 2) + sex
  Res.Df    RSS Df Sum of Sq    F Pr(>F)
1     1228 503.19
2     1227 502.07  1     1.1137 2.7218 0.09924 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> # look at functional form of age
> termplot(fit3, term=2, se=T, rug=T)

```

In this instance it looks like there isn't enough evidence to say that the relationship is non-linear.

Extract data using the broom package

The broom package makes it easy to extract information from the fit.

```

> tmp <- tidy(fit3) # coefficients, p-values
> class(tmp)
[1] "tbl_df"      "tbl"        "data.frame"
> tmp
# A tibble: 6 x 5
  term                estimate std.error statistic  p.value
<chr>                <dbl>    <dbl>    <dbl>    <dbl>
1 (Intercept)         4.76      0.141     33.8 1.93e-177
2 armF: FOLFOX        -0.0767   0.0434    -1.77 7.78e- 2
3 armG: IROX          -0.0195   0.0491    -0.396 6.92e- 1
4 ns(age, df = 2)1     0.330     0.260     1.27 2.04e- 1
5 ns(age, df = 2)2    -0.101     0.0935    -1.08 2.82e- 1
6 sexFemale           0.0183   0.0374     0.489 6.25e- 1
>
> glance(fit3)
# A tibble: 1 x 11
  r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
*   <dbl>         <dbl> <dbl>    <dbl>    <dbl> <int>  <dbl> <dbl> <dbl>
1  0.00533      0.00127 0.640     1.31   0.255     6 -1196. 2405. 2441.
# ... with 2 more variables: deviance <dbl>, df.residual <int>
Create a summary table using modelsum
> ms.logy <- modelsum(log(alk.phos) ~ arm + ps + hgb, data=mockstudy, adjust= ~age + sex,
+                      family=gaussian,
+                      gaussian.stats=c("estimate", "CI.lower.estimate", "CI.upper.estimate", "p.value"))
> summary(ms.logy)
estimate    CI.lower.estimate    CI.upper.estimate    p.value
(Intercept) 4.969      4.768      5.170      < 0.001
Treatment Arm F: FOLFOX -0.077    -0.162    0.009    0.078
Treatment Arm G: IROX  -0.019    -0.116    0.077    0.695
Age in Years  -0.000    -0.004    0.003    0.798
sex Female    0.018    -0.056    0.091    0.632
(Intercept) 4.832      4.640      5.023      < 0.001
ps 0.226      0.167      0.284      < 0.001

```

```

Age in Years    -0.001  -0.004  0.002   0.636
sex Female      0.009   -0.063  0.081   0.814
(Intercept)    5.765    5.450   6.080   < 0.001
hgb -0.069     -0.090  -0.048   < 0.001
Age in Years      0.000   -0.003  0.003   0.925
sex Female      -0.027  -0.101   0.046   0.468
Binomial
Fit and summarize logistic regression model
> boxplot(age ~ mdquality.s, data=mockstudy, ylab=attr(mockstudy$age,'label'), xlab='mdquality.s')

>
> fit <- glm(mdquality.s ~ age + sex, data=mockstudy, family=binomial)
> summary(fit)

Call:
glm(formula = mdquality.s ~ age + sex, family = binomial, data = mockstudy)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1832   0.4500   0.4569   0.4626   0.4756

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.329442   0.514684   4.526 6.01e-06 ***
age          -0.002353   0.008256  -0.285   0.776
sexFemale     0.039227   0.195330   0.201   0.841
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 807.68  on 1246  degrees of freedom
Residual deviance: 807.55  on 1244  degrees of freedom
(252 observations deleted due to missingness)
AIC: 813.55

Number of Fisher Scoring iterations: 4
>
> # create Odd's ratio w/ confidence intervals
> tmp <- data.frame(summary(fit)$coef)
> tmp
              Estimate Std..Error   z.value   Pr...z..
(Intercept)  2.329441734 0.514683688  4.5259677 6.011977e-06
age          -0.002353404 0.008255814 -0.2850602 7.755980e-01
sexFemale     0.039227292 0.195330166  0.2008256 8.408350e-01
>
> tmp$OR <- round(exp(tmp[,1]),2)
> tmp$lower.CI <- round(exp(tmp[,1] - 1.96* tmp[,2]),2)
> tmp$upper.CI <- round(exp(tmp[,1] + 1.96* tmp[,2]),2)
> names(tmp)[4] <- 'P-value'
>
> kable(tmp[,c('OR','lower.CI','upper.CI','P-value')])
OR lower.CI upper.CI P-value
(Intercept) 10.27   3.75   28.17   0.000006
age 1.00     0.98    1.01    0.775598
sexFemale 1.04     0.71    1.53    0.840835
>

```

```

> # Assess the predictive ability of the model
>
> # code using the pROC package
> require(pROC)
> pred <- predict(fit, type='response')
> tmp <- pROC::roc(mockstudy$mdquality.s[!is.na(mockstudy$mdquality.s)]~ pred, plot=TRUE, percent=TRUE)

> tmp$auc
Area under the curve: 50.69%
Extract data using broom package
The broom package makes it easy to extract information from the fit.

> tidy(fit, exp=T, conf.int=T) # coefficients, p-values, conf.intervals
# A tibble: 3 x 7
  term          estimate std.error statistic    p.value conf.low conf.high
<chr>         <dbl>     <dbl>     <dbl>    <dbl>    <dbl>    <dbl>
1 (Intercept)  10.3      0.515      4.53 0.00000601    3.83    28.9
2 age          0.998    0.00826   -0.285 0.776          0.981    1.01
3 sexFemale    1.04     0.195     0.201 0.841          0.712    1.53
>
> glance(fit) # model summary statistics
# A tibble: 1 x 7
  null.deviance df.null logLik   AIC   BIC deviance df.residual
      <dbl>    <int>  <dbl> <dbl> <dbl>   <dbl>    <int>
1      808.    1246 -404.  814.  829.   808.    1244
Create a summary table using modelsum
> summary(modelsum(mdquality.s ~ age + bmi, data=mockstudy, adjust=~sex, family=binomial))
OR  CI.lower.OR CI.upper.OR p.value concordance Nmiss
(Intercept) 10.272  3.831  28.876 < 0.001 0.507  0
Age in Years  0.998  0.981  1.014  0.776
sex Female  1.040  0.712  1.534  0.841
(Intercept) 4.814  1.709  13.221 0.003  0.550  33
Body Mass Index (kg/m^2) 1.023  0.987  1.063  0.220
sex Female  1.053  0.717  1.561  0.794
>
> fitall <- modelsum(mdquality.s ~ age, data=mockstudy, family=binomial,
+                   binomial.stats=c("Nmiss2", "OR", "p.value"))
> summary(fitall)
OR  p.value Nmiss2
(Intercept) 10.493 < 0.001 0
Age in Years  0.998  0.766
Survival
Fit and summarize a Cox regression model
> require(survival)
Loading required package: survival

Attaching package: 'survival'
The following object is masked from 'package:rpart':

  solder
>
> # multivariable model with all 3 terms
> fit <- coxph(Surv(fu.time, fu.stat) ~ age + sex + arm, data=mockstudy)
> summary(fit)
Call:
coxph(formula = Surv(fu.time, fu.stat) ~ age + sex + arm, data = mockstudy)

```

```

n= 1499, number of events= 1356

      coef exp(coef) se(coef)      z Pr(>|z|)
age      0.004600  1.004611  0.002501  1.839  0.0659 .
sexFemale 0.039893  1.040699  0.056039  0.712  0.4765
armF: FOLFOX -0.454650  0.634670  0.064878 -7.008 2.42e-12 ***
armG: IROX   -0.140785  0.868676  0.072760 -1.935  0.0530 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

      exp(coef) exp(-coef) lower .95 upper .95
age      1.0046      0.9954      0.9997      1.0095
sexFemale 1.0407      0.9609      0.9324      1.1615
armF: FOLFOX 0.6347      1.5756      0.5589      0.7207
armG: IROX   0.8687      1.1512      0.7532      1.0018

Concordance= 0.563 (se = 0.009 )
Rsquare= 0.037 (max possible= 1 )
Likelihood ratio test= 56.21 on 4 df,  p=2e-11
Wald test = 56.26 on 4 df,  p=2e-11
Score (logrank) test = 56.96 on 4 df,  p=1e-11
>
> # check proportional hazards assumption
> fit.z <- cox.zph(fit)
> fit.z

      rho chisq      p
age      -0.0311  1.46 0.226
sexFemale -0.0325  1.44 0.230
armF: FOLFOX 0.0343  1.61 0.205
armG: IROX   0.0337  1.54 0.214
GLOBAL      NA  4.59 0.332
> plot(fit.z[1], resid=FALSE) # makes for a cleaner picture in this case
> abline(h=coef(fit)[1], col='red')

>
> # check functional form for age using pspline (penalized spline)
> # results are returned for the linear and non-linear components
> fit2 <- coxph(Surv(fu.time, fu.stat) ~ pspline(age) + sex + arm, data=mockstudy)
> fit2
Call:
coxph(formula = Surv(fu.time, fu.stat) ~ pspline(age) + sex +
      arm, data = mockstudy)

      coef se(coef)      se2      Chisq      DF      p
pspline(age), linear 0.00443 0.00237 0.00237 3.48989 1.00 0.0617
pspline(age), nonlin      13.11270 3.08 0.0047
sexFemale      0.03993 0.05610 0.05607 0.50663 1.00 0.4766
armF: FOLFOX     -0.46240 0.06494 0.06493 50.69608 1.00 1.1e-12
armG: IROX       -0.15243 0.07301 0.07299 4.35876 1.00 0.0368

Iterations: 6 outer, 16 Newton-Raphson
Theta= 0.954
Degrees of freedom for terms= 4.1 1.0 2.0
Likelihood ratio test=70.1 on 7.08 df, p=2e-12
n= 1499, number of events= 1356
>
> # plot smoothed age to visualize why significant

```

```
> termplot(fit2, se=T, terms=1)
> abline(h=0)

>
> # The c-statistic comes out in the summary of the fit
> summary(fit2)$concordance
      C      se(C)
0.5684325 0.5684325
>
> # It can also be calculated using the survConcordance function
> survConcordance(Surv(fu.time, fu.stat) ~ predict(fit2), data=mockstudy)
Call:
survConcordance(formula = Surv(fu.time, fu.stat) ~ predict(fit2),
  data = mockstudy)
```

```
n= 1499
Concordance= 0.5684325 se= 0.008779125
concordant discordant tied.risk tied.time std(c-d)
620221.00 470282.00 5021.00 766.00 19235.49
Extract data using broom package
The broom package makes it easy to extract information from the fit.
```

```
> tidy(fit) # coefficients, p-values
# A tibble: 4 x 7
  term          estimate std.error statistic p.value  conf.low conf.high
<chr>         <dbl>      <dbl>      <dbl>   <dbl>    <dbl>    <dbl>
1 age           0.00460    0.00250     1.84 6.59e- 2 -0.000302  0.00950
2 sexFemale     0.0399     0.0560     0.712 4.77e- 1 -0.0699    0.150
3 armF: FOLFOX -0.455     0.0649    -7.01 2.42e-12 -0.582    -0.327
4 armG: IROX   -0.141     0.0728    -1.93 5.30e- 2 -0.283     0.00182
>
```

```
> glance(fit) # model summary statistics
# A tibble: 1 x 15
  n nevent statistic.log p.value.log statistic.sc p.value.sc
<int> <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
1 1499 1356      56.2      1.81e-11    57.0      1.26e-11
# ... with 9 more variables: statistic.wald <dbl>, p.value.wald <dbl>,
#   r.squared <dbl>, r.squared.max <dbl>, concordance <dbl>,
#   std.error.concordance <dbl>, logLik <dbl>, AIC <dbl>, BIC <dbl>
Create a summary table using modelsum
```

```
> ##Note: You must use quotes when specifying family="survival"
> ##      family=survival will not work
> summary(modelsum(Surv(fu.time, fu.stat) ~ arm,
+   adjust=~age + sex, data=mockstudy, family="survival"))
HR CI.lower.HR CI.upper.HR p.value concordance
Treatment Arm F: FOLFOX 0.635 0.559 0.721 < 0.001 0.563
Treatment Arm G: IROX 0.869 0.753 1.002 0.053
Age in Years 1.005 1.000 1.010 0.066
sex Female 1.041 0.932 1.162 0.477
>
```

```
> ##Note: the pspline term is not working yet
> #summary(modelsum(Surv(fu.time, fu.stat) ~ arm,
> #   adjust=~pspline(age) + sex, data=mockstudy, family='survival'))
```

Poisson

Poisson regression is useful when predicting an outcome variable representing counts. It can also be useful

Example 1: fit and summarize a Poisson regression model

For the first example, use the solder dataset available in the rpart package. The endpoint skips has a defi

```
> require(rpart) ##just to get access to solder dataset
> data(solder)
> hist(solder$skips)

>
> fit <- glm(skips ~ Opening + Solder + Mask , data=solder, family=poisson)
> stats::anova(fit, test='Chi')
Analysis of Deviance Table
```

Model: poisson, link: log

Response: skips

Terms added sequentially (first to last)

	Df	Deviance	Resid. Df	Resid. Dev	Pr(>Chi)
NULL			899	8788.2	
Opening	2	2920.5	897	5867.7	< 2.2e-16 ***
Solder	1	1168.4	896	4699.3	< 2.2e-16 ***
Mask	4	2015.7	892	2683.7	< 2.2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```
> summary(fit)
```

Call:

```
glm(formula = skips ~ Opening + Solder + Mask, family = poisson,
    data = solder)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-6.1251	-1.4720	-0.7826	0.5986	6.6031

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-1.12220	0.07742	-14.50	< 2e-16 ***
OpeningM	0.57161	0.05707	10.02	< 2e-16 ***
OpeningS	1.81475	0.05044	35.98	< 2e-16 ***
SolderThin	0.84682	0.03327	25.45	< 2e-16 ***
MaskA3	0.51315	0.07098	7.23	4.83e-13 ***
MaskA6	1.81103	0.06609	27.40	< 2e-16 ***
MaskB3	1.20225	0.06697	17.95	< 2e-16 ***
MaskB6	1.86648	0.06310	29.58	< 2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 8788.2 on 899 degrees of freedom
 Residual deviance: 2683.7 on 892 degrees of freedom
 AIC: 4802.2

Number of Fisher Scoring iterations: 5

Overdispersion is when the Residual deviance is larger than the degrees of freedom. This can be tested, app

```
> 1-pchisq(fit$deviance, fit$df.residual)
```

```
[1] 0
```

One possible solution is to use the quasipoisson family instead of the poisson family. This adjusts for the

```
> fit2 <- glm(skips ~ Opening + Solder + Mask, data=solder, family=quasipoisson)
> summary(fit2)
```

Call:

```
glm(formula = skips ~ Opening + Solder + Mask, family = quasipoisson,
     data = solder)
```

Deviance Residuals:

	Min	1Q	Median	3Q	Max
	-6.1251	-1.4720	-0.7826	0.5986	6.6031

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-1.12220	0.13483	-8.323	3.19e-16 ***
OpeningM	0.57161	0.09939	5.751	1.22e-08 ***
OpeningS	1.81475	0.08784	20.660	< 2e-16 ***
SolderThin	0.84682	0.05794	14.615	< 2e-16 ***
MaskA3	0.51315	0.12361	4.151	3.62e-05 ***
MaskA6	1.81103	0.11510	15.735	< 2e-16 ***
MaskB3	1.20225	0.11663	10.308	< 2e-16 ***
MaskB6	1.86648	0.10989	16.984	< 2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasipoisson family taken to be 3.033198)

Null deviance: 8788.2 on 899 degrees of freedom
 Residual deviance: 2683.7 on 892 degrees of freedom
 AIC: NA

Number of Fisher Scoring iterations: 5

Extract data using broom package

The broom package makes it easy to extract information from the fit.

```
> tidy(fit) # coefficients, p-values
```

```
# A tibble: 8 x 5
```

	term	estimate	std.error	statistic	p.value
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>
1	(Intercept)	-1.12	0.0774	-14.5	1.29e- 47
2	OpeningM	0.572	0.0571	10.0	1.29e- 23
3	OpeningS	1.81	0.0504	36.0	1.66e-283
4	SolderThin	0.847	0.0333	25.5	6.47e-143
5	MaskA3	0.513	0.0710	7.23	4.83e- 13
6	MaskA6	1.81	0.0661	27.4	2.45e-165
7	MaskB3	1.20	0.0670	18.0	4.55e- 72
8	MaskB6	1.87	0.0631	29.6	2.71e-192

```
>
```

```
> glance(fit) # model summary statistics
```

```
# A tibble: 1 x 7
```

	null.deviance	df.null	logLik	AIC	BIC	deviance	df.residual
	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<int>
1	8788.	899	-2393.	4802.	4841.	2684.	892

Create a summary table using modelsum

```
> summary(modelsum(skips~Opening + Solder + Mask, data=solder, family="quasipoisson"))
```



```

RR   CI.lower.RR CI.upper.RR p.value
(Intercept) 1.533   1.179   1.952   < 0.001
Opening M    2.328   1.733   3.167   < 0.001
Opening S    7.491   5.780   9.888   < 0.001
(Intercept) 2.904   2.423   3.446   < 0.001
Solder Thin  2.808   2.295   3.458   < 0.001
(Intercept) 1.611   1.135   2.204   0.005
Mask A3      1.469   0.995   2.214   0.059
Mask A6      8.331   5.839   12.222  < 0.001
Mask B3      3.328   2.309   4.920   < 0.001
Mask B6      6.466   4.598   9.378   < 0.001
> summary(modelsum(skips~Opening + Solder + Mask, data=solder, family="poisson"))

```

```

RR   CI.lower.RR CI.upper.RR p.value
(Intercept) 1.533   1.397   1.678   < 0.001
Opening M    2.328   2.089   2.599   < 0.001
Opening S    7.491   6.805   8.267   < 0.001
(Intercept) 2.904   2.750   3.065   < 0.001
Solder Thin  2.808   2.637   2.992   < 0.001
(Intercept) 1.611   1.433   1.804   < 0.001
Mask A3      1.469   1.280   1.690   < 0.001
Mask A6      8.331   7.341   9.487   < 0.001
Mask B3      3.328   2.923   3.800   < 0.001
Mask B6      6.466   5.724   7.331   < 0.001

```

Example 2: fit and summarize a Poisson regression model

This second example uses the survival endpoint available in the mockstudy dataset. There is a close relation

```

> # add .01 to the follow-up time (.01*1 day) in order to keep everyone in the analysis
> fit <- glm(fu.stat ~ offset(log(fu.time+.01)) + age + sex + arm, data=mockstudy, family=poisson)
> summary(fit)

```

Call:

```

glm(formula = fu.stat ~ offset(log(fu.time + 0.01)) + age + sex +
    arm, family = poisson, data = mockstudy)

```

Deviance Residuals:

```

      Min       1Q   Median       3Q      Max
-3.1188  -0.4041   0.3242   0.9727   4.3588

```

Coefficients:

```

              Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.875627    0.108984 -53.913  < 2e-16 ***
age          0.003724    0.001705   2.184   0.0290 *
sexFemale    0.027321    0.038575   0.708   0.4788
armF: FOLFOX -0.335141    0.044600  -7.514 5.72e-14 ***
armG: IROX   -0.107776    0.050643  -2.128   0.0333 *
---

```

```

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

(Dispersion parameter for poisson family taken to be 1)

```

Null deviance: 2113.5  on 1498  degrees of freedom
Residual deviance: 2048.0  on 1494  degrees of freedom
AIC: 5888.2

```

Number of Fisher Scoring iterations: 5

```

> 1-pchisq(fit$deviance, fit$df.residual)
[1] 0

```

```

>
> coef(coxph(Surv(fu.time,fu.stat) ~ age + sex + arm, data=mockstudy))
      age      sexFemale armF: FOLFOX   armG: IROX
0.004600011 0.039892735 -0.454650445 -0.140784996
> coef(fit)[-1]
      age      sexFemale armF: FOLFOX   armG: IROX
0.003723763 0.027320917 -0.335141090 -0.107775577
>
> # results from the Poisson model can then be described as risk ratios (similar to the hazard ratio)
> exp(coef(fit)[-1])
      age      sexFemale armF: FOLFOX   armG: IROX
1.0037307 1.0276976 0.7152372 0.8978291
>
> # As before, we can model the dispersion which alters the standard error
> fit2 <- glm(fu.stat ~ offset(log(fu.time+.01)) + age + sex + arm,
+           data=mockstudy, family=quasipoisson)
> summary(fit2)

```

Call:

```

glm(formula = fu.stat ~ offset(log(fu.time + 0.01)) + age + sex +
    arm, family = quasipoisson, data = mockstudy)

```

Deviance Residuals:

	Min	1Q	Median	3Q	Max
	-3.1188	-0.4041	0.3242	0.9727	4.3588

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-5.875627	0.566666	-10.369	<2e-16 ***
age	0.003724	0.008867	0.420	0.675
sexFemale	0.027321	0.200572	0.136	0.892
armF: FOLFOX	-0.335141	0.231899	-1.445	0.149
armG: IROX	-0.107776	0.263318	-0.409	0.682

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasipoisson family taken to be 27.03493)

Null deviance: 2113.5 on 1498 degrees of freedom
 Residual deviance: 2048.0 on 1494 degrees of freedom
 AIC: NA

Number of Fisher Scoring iterations: 5

Extract data using broom package

The broom package makes it easy to extract information from the fit.

```

> tidy(fit) ##coefficients, p-values
# A tibble: 5 x 5
  term      estimate std.error statistic  p.value
<chr>      <dbl>      <dbl>      <dbl>    <dbl>
1 (Intercept) -5.88      0.109     -53.9    0.
2 age          0.00372  0.00171     2.18  2.90e- 2
3 sexFemale    0.0273  0.0386     0.708  4.79e- 1
4 armF: FOLFOX -0.335    0.0446    -7.51  5.72e-14
5 armG: IROX   -0.108    0.0506    -2.13  3.33e- 2
>
> glance(fit) ##model summary statistics

```

```
# A tibble: 1 x 7
  null.deviance df.null logLik   AIC   BIC deviance df.residual
      <dbl>     <int>  <dbl> <dbl> <dbl>   <dbl>      <int>
1      2114.     1498 -2939. 5888. 5915.   2048.       1494
```

Create a summary table using modelsum

Remember that the result from modelsum is different from the fit above. The modelsum summary shows the results of the model fit.

```
> summary(modelsum(fu.stat ~ age, adjust=~offset(log(fu.time+.01))+ sex + arm,
+                 data=mockstudy, family=poisson))
RR   CI.lower.RR CI.upper.RR p.value
(Intercept) 0.003   0.002   0.003   < 0.001
Age in Years  1.004   1.000   1.007   0.029
sexFemale    1.028   0.953   1.108   0.479
armF: FOLFOX  0.715   0.656   0.781   < 0.001
armG: IROX    0.898   0.813   0.991   0.033
```

Additional Examples

Here are multiple examples showing how to use some of the different options.

1. Change summary statistics globally

There are standard settings for each type of model regarding what information is summarized in the table. The default settings for the modelsum function are:

```
> mycontrols <- modelsum.control(gaussian.stats=c("estimate","std.error","adj.r.squared","Nmiss"),
+                               show.adjust=FALSE, show.intercept=FALSE)
> tab2 <- modelsum(bmi ~ age, adjust=~sex, data=mockstudy, control=mycontrols)
> summary(tab2)
estimate      std.error      adj.r.squared
Age in Years   0.012    0.012    0.004
```

You can also change these settings directly in the modelsum call.

```
> tab3 <- modelsum(bmi ~ age, adjust=~sex, data=mockstudy,
+                 gaussian.stats=c("estimate","std.error","adj.r.squared","Nmiss"),
+                 show.intercept=FALSE, show.adjust=FALSE)
> summary(tab3)
estimate      std.error      adj.r.squared
Age in Years   0.012    0.012    0.004
```

2. Add labels to independent variables

In the above example, age is shown with a label (Age in Years), but sex is listed "as is". This is because the variable sex is a factor and does not have a label.

```
> ## Look at one variable's label
> attr(mockstudy$age,'label')
[1] "Age in Years"
>
> ## See all the variables with a label
> unlist(lapply(mockstudy,'attr','label'))
      age      arm
"Age in Years" "Treatment Arm"
      race      bmi
"Race" "Body Mass Index (kg/m^2)"
>
> ## or
> cbind(sapply(mockstudy,attr,'label'))
      [,1]
case      NULL
age      "Age in Years"
arm      "Treatment Arm"
sex      NULL
race     "Race"
```

```
fu.time      NULL
fu.stat      NULL
ps           NULL
hgb          NULL
bmi          "Body Mass Index (kg/m^2)"
alk.phos     NULL
ast          NULL
mdquality.s  NULL
age.ord      NULL
```

If you want to add labels to other variables, there are a couple of options. First, you could add labels to

```
> attr(mockstudy$age,'label') <- 'Age, yrs'
>
> tab1 <- modelsum(bmi ~ age, adjust=~sex, data=mockstudy)
> summary(tab1)
estimate      std.error    p.value adj.r.squared
(Intercept) 26.793  0.766    < 0.001 0.004
Age, yrs     0.012   0.012    0.348
sex Female   -0.718  0.291    0.014
You can also use the built-in data.frame method for labels<:-
```

```
> labels(mockstudy) <- c(age = 'Age, yrs')
>
> tab1 <- modelsum(bmi ~ age, adjust=~sex, data=mockstudy)
> summary(tab1)
estimate      std.error    p.value adj.r.squared
(Intercept) 26.793  0.766    < 0.001 0.004
Age, yrs     0.012   0.012    0.348
sex Female   -0.718  0.291    0.014
Another option is to add labels after you have created the table
```

```
> mylabels <- list(sexFemale = "Female", age="Age, yrs")
> summary(tab1, labelTranslations = mylabels)
estimate      std.error    p.value adj.r.squared
(Intercept) 26.793  0.766    < 0.001 0.004
Age, yrs     0.012   0.012    0.348
Female       -0.718  0.291    0.014
```

Alternatively, you can check the variable labels and manipulate them with a function called labels, which w

```
> labels(tab1)
              bmi                      age
"Body Mass Index (kg/m^2)"          "Age, yrs"
              sexFemale
              "sex Female"
> labels(tab1) <- c(sexFemale="Female", age="Baseline Age (yrs)")
> labels(tab1)
              bmi                      age
"Body Mass Index (kg/m^2)"          "Baseline Age (yrs)"
              sexFemale
              "Female"
> summary(tab1)
estimate      std.error    p.value adj.r.squared
(Intercept) 26.793  0.766    < 0.001 0.004
Baseline Age (yrs) 0.012  0.012    0.348
Female          -0.718  0.291    0.014
3. Don't show intercept values
> summary(modelsum(age~mdquality.s+sex, data=mockstudy), show.intercept=FALSE)
```

```
estimate    std.error    p.value adj.r.squared    Nmiss
mdquality.s -0.326    1.093    0.766    -0.001    252
sex Female  -1.208    0.610    0.048    0.002    0
```

4. Don't show results for adjustment variables

```
> summary(modelsum(mdquality.s ~ age + bmi, data=mockstudy, adjust=~sex, family=binomial),
+          show.adjust=FALSE)
```

```
OR  CI.lower.OR CI.upper.OR p.value concordance Nmiss
(Intercept) 10.272    3.831    28.876    < 0.001 0.507    0
Age, yrs     0.998    0.981    1.014    0.776
(Intercept) 4.814    1.709    13.221    0.003    0.550    33
Body Mass Index (kg/m^2)    1.023    0.987    1.063    0.220
```

5. Summarize multiple variables without typing them out

Often one wants to summarize a number of variables. Instead of typing by hand each individual variable, an

```
> # create a vector specifying the variable names
> myvars <- names(mockstudy)
>
```

```
> # select the 8th through the 12th
> # paste them together, separated by the + sign
> RHS <- paste(myvars[8:12], collapse="+")
> RHS
```

```
[1] "ps+hgb+bmi+alk.phos+ast"
```

```
>
> # create a formula using the as.formula function
> as.formula(paste('mdquality.s ~ ', RHS))
mdquality.s ~ ps + hgb + bmi + alk.phos + ast
```

```
>
> # use the formula in the modelsum function
> summary(modelsum(as.formula(paste('mdquality.s ~ ', RHS)), family=binomial, data=mockstudy))
OR  CI.lower.OR CI.upper.OR p.value concordance Nmiss
(Intercept) 14.628    10.755    20.399    < 0.001 0.620    266
ps  0.461    0.332    0.639    < 0.001
(Intercept) 1.236    0.272    5.560    0.783    0.573    266
hgb 1.176    1.040    1.334    0.011
(Intercept) 4.963    1.818    13.292    0.002    0.549    33
Body Mass Index (kg/m^2)    1.023    0.987    1.062    0.225
(Intercept) 10.622    7.687    14.794    < 0.001 0.552    266
alk.phos    0.999    0.998    1.000    0.159
(Intercept) 10.936    7.912    15.232    < 0.001 0.545    266
ast 0.995    0.988    1.001    0.099
```

These steps can also be done using the formulize function.

```
> ## The formulize function does the paste and as.formula steps
> tmp <- formulize('mdquality.s',myvars[8:10])
> tmp
mdquality.s ~ ps + hgb + bmi
```

```
>
> ## More complex formulas could also be written using formulize
> tmp2 <- formulize('mdquality.s',c('ps','hgb','sqrt(bmi)'))
>
```

```
> ## use the formula in the modelsum function
> summary(modelsum(tmp, data=mockstudy, family=binomial))
OR  CI.lower.OR CI.upper.OR p.value concordance Nmiss
(Intercept) 14.628    10.755    20.399    < 0.001 0.620    266
```

```

ps 0.461 0.332 0.639 < 0.001
(Intercept) 1.236 0.272 5.560 0.783 0.573 266
hgb 1.176 1.040 1.334 0.011
(Intercept) 4.963 1.818 13.292 0.002 0.549 33
Body Mass Index (kg/m^2) 1.023 0.987 1.062 0.225

```

6. Subset the dataset used in the analysis

Here are two ways to get the same result (limit the analysis to subjects age>50 and in the F: FOLFOX treatment group)

The first approach uses the subset function applied to the dataset mockstudy. This example also selects a subset of variables

```

> newdata <- subset(mockstudy, subset=age>50 & arm=='F: FOLFOX', select = c(age,sex, bmi:alk.phos))
> dim(newdata)
[1] 1499  4
> table(newdata$arm)

```

```

      A: IFL F: FOLFOX      G: IROX
      428      691      380

```

```

> dim(newdata)
[1] 1499  4
> names(newdata)
[1] "age"      "sex"      "bmi"      "alk.phos"
> summary(modelsum(alk.phos ~ ., data=newdata))
estimate      std.error      p.value adj.r.squared      Nmiss
(Intercept) 122.577 46.924 0.009 -0.001 0
age 0.619 0.719 0.390
(Intercept) 164.814 7.673 < 0.001 -0.002 0
sex Female -5.497 12.118 0.650
(Intercept) 238.658 33.705 < 0.001 0.010 15
bmi -2.776 1.207 0.022

```

The second approach does the same analysis but uses the subset argument within modelsum to subset the data.

```

> summary(modelsum(log(alk.phos) ~ sex + ps + bmi, subset=age>50 & arm=="F: FOLFOX", data=mockstudy))

```

```

estimate      std.error      p.value adj.r.squared      Nmiss
(Intercept) 4.872 0.039 < 0.001 -0.002 0
sex Female -0.005 0.062 0.931
(Intercept) 4.770 0.040 < 0.001 0.027 108
ps 0.183 0.050 < 0.001
(Intercept) 5.207 0.172 < 0.001 0.007 15
Body Mass Index (kg/m^2) -0.012 0.006 0.044

```

```

> summary(modelsum(alk.phos ~ ps + bmi, adjust=~sex, subset = age>50 & bmi<24, data=mockstudy))

```

```

estimate      std.error      p.value adj.r.squared      Nmiss
(Intercept) 178.812 14.550 < 0.001 0.007 77
ps 20.834 13.440 0.122
sex Female -17.542 16.656 0.293
(Intercept) 373.008 104.272 < 0.001 0.009 24
Body Mass Index (kg/m^2) -8.239 4.727 0.083
sex Female -24.058 16.855 0.155

```

```

> summary(modelsum(alk.phos ~ ps + bmi, adjust=~sex, subset=1:30, data=mockstudy))

```

```

estimate      std.error      p.value adj.r.squared      Nmiss
(Intercept) 169.112 57.013 0.006 0.294 0
ps 254.901 68.100 < 0.001
sex Female 49.566 67.643 0.470
(Intercept) 453.070 200.651 0.033 -0.049 1
Body Mass Index (kg/m^2) -5.993 7.408 0.426
sex Female -22.308 79.776 0.782

```

7. Create combinations of variables on the fly

```

> ## create a variable combining the levels of mdquality.s and sex
> with(mockstudy, table(interaction(mdquality.s,sex)))

```

```

0.Male 1.Male 0.Female 1.Female
77      686      47      437

```

```
> summary(modelsum(age ~ interaction(mdquality.s,sex), data=mockstudy))
```

```

estimate   std.error   p.value adj.r.squared   Nmiss
(Intercept) 59.714 1.314 < 0.001 0.003 252
interaction(mdquality.s, sex) 1.Male 0.730 1.385 0.598
interaction(mdquality.s, sex) 0.Female 0.988 2.134 0.643
interaction(mdquality.s, sex) 1.Female -1.021 1.425 0.474

```

8. Transform variables on the fly

Certain transformations need to be surrounded by I() so that R knows to treat it as a variable transformation

```
> summary(modelsum(arm=="F: FOLFOX" ~ I(age/10) + log(bmi) + mdquality.s,
+ data=mockstudy, family=binomial))
```

```

OR CI.lower.OR CI.upper.OR p.value concordance Nmiss
(Intercept) 0.656 0.382 1.124 0.126 0.514 0
Age, yrs 1.045 0.957 1.142 0.326
(Intercept) 0.633 0.108 3.698 0.611 0.508 33
Body Mass Index (kg/m^2) 1.092 0.638 1.867 0.748
(Intercept) 0.722 0.503 1.029 0.074 0.502 252
mdquality.s 1.045 0.719 1.527 0.819

```

9. Change the ordering of the variables or delete a variable

```
> mytab <- modelsum(bmi ~ sex + alk.phos + age, data=mockstudy)
```

```
> mytab2 <- mytab[c('age','sex','alk.phos')]
```

```
> summary(mytab2)
```

```

estimate   std.error   p.value adj.r.squared   Nmiss
(Intercept) 26.424 0.752 < 0.001 0.000 0
Age, yrs 0.013 0.012 0.290
(Intercept) 27.491 0.181 < 0.001 0.004 0
sex Female -0.731 0.290 0.012
(Intercept) 27.944 0.253 < 0.001 0.011 266
alk.phos -0.005 0.001 < 0.001

```

```
> summary(mytab[c('age','sex')])
```

```

estimate   std.error   p.value adj.r.squared
(Intercept) 26.424 0.752 < 0.001 0.000
Age, yrs 0.013 0.012 0.290
(Intercept) 27.491 0.181 < 0.001 0.004
sex Female -0.731 0.290 0.012

```

```
> summary(mytab[c(3,1)])
```

```

estimate   std.error   p.value adj.r.squared
(Intercept) 26.424 0.752 < 0.001 0.000
Age, yrs 0.013 0.012 0.290
(Intercept) 27.491 0.181 < 0.001 0.004
sex Female -0.731 0.290 0.012

```

10. Merge two modelsum objects together

It is possible to combine two modelsum objects so that they print out together, however you need to pay attention

```
> ## demographics
```

```
> tab1 <- modelsum(bmi ~ sex + age, data=mockstudy)
```

```
> ## lab data
```

```
> tab2 <- modelsum(mdquality.s ~ hgb + alk.phos, data=mockstudy, family=binomial)
```

```
>
```

```
> tab12 <- merge(tab1,tab2)
```

```
> class(tab12)
```

```
[1] "modelsumList"
```

```
>
```

```
> ##ERROR: The merge works, but not the summary
```

```
> #summary(tab12)
```

```
11. Add a title to the table
```

When creating a pdf the tables are automatically numbered and the title appears below the table. In Word and LaTeX the title appears above the table.

```
> t1 <- modelsum(bmi ~ sex + age, data=mockstudy)
```

```
> summary(t1, title='Demographics')
```

```
Demographics
```

```
estimate      std.error    p.value adj.r.squared
```

```
(Intercept) 27.491 0.181 < 0.001 0.004
```

```
sex Female -0.731 0.290 0.012
```

```
(Intercept) 26.424 0.752 < 0.001 0.000
```

```
Age, yrs 0.013 0.012 0.290
```

```
12. Modify how missing values are treated
```

Depending on the report you are writing you have the following options:

Use all values available for each variable

Use only those subjects who have measurements available for all the variables

```
> ## look at how many missing values there are for each variable
```

```
> apply(is.na(mockstudy),2,sum)
```

case	age	arm	sex	race	fu.time
0	0	0	0	7	0
fu.stat	ps	hgb	bmi	alk.phos	ast
0	266	266	33	266	266
mdquality.s	age.ord				
252	0				

```
> ## Show how many subjects have each variable (non-missing)
```

```
> summary(modelsum(bmi ~ ast + age, data=mockstudy,
```

```
+ control=modelsum.control(gaussian.stats=c("N","estimate"))))
```

```
estimate      N
```

```
(Intercept) 27.331 1233
```

```
ast -0.005
```

```
(Intercept) 26.424 1499
```

```
Age, yrs 0.013
```

```
>
```

```
> ## Always list the number of missing values
```

```
> summary(modelsum(bmi ~ ast + age, data=mockstudy,
```

```
+ control=modelsum.control(gaussian.stats=c("Nmiss2","estimate"))))
```

```
estimate      Nmiss2
```

```
(Intercept) 27.331 266
```

```
ast -0.005
```

```
(Intercept) 26.424 0
```

```
Age, yrs 0.013
```

```
>
```

```
> ## Only show the missing values if there are some (default)
```

```
> summary(modelsum(bmi ~ ast + age, data=mockstudy,
```

```
+ control=modelsum.control(gaussian.stats=c("Nmiss","estimate"))))
```

```
estimate      Nmiss
```

```
(Intercept) 27.331 266
```

```
ast -0.005
```

```
(Intercept) 26.424 0
```

```
Age, yrs 0.013
```

```
>
```

```
> ## Don't show N at all
```

```
> summary(modelsum(bmi ~ ast + age, data=mockstudy,
```

```
+ control=modelsum.control(gaussian.stats=c("estimate"))))
```



```

estimate
(Intercept) 27.331
ast -0.005
(Intercept) 26.424
Age, yrs    0.013
13. Modify the number of digits used
Within modelsum.control function there are 3 options for controlling the number of significant digits shown

digits: controls the number of digits after the decimal point for continuous values

digits.ratio: controls the number of digits after the decimal point for continuous values

digits.p: controls the number of digits after the decimal point for continuous values

> summary(modelsum(bmi ~ sex + age + fu.time, data=mockstudy), digits=4, digits.test=2)
Warning: Using 'digits.test = ' is deprecated. Use 'digits.p = ' instead.
estimate      std.error    p.value adj.r.squared
(Intercept) 27.4915 0.1813 < 0.001 0.0036
sex Female  -0.7311 0.2903 0.012
(Intercept) 26.4237 0.7521 < 0.001 0.0001
Age, yrs    0.0130 0.0123 0.290
(Intercept) 26.4937 0.2447 < 0.001 0.0079
fu.time 0.0011 0.0003 < 0.001
14. Use case-weights in the models
Occasionally it is of interest to fit models using case weights. The modelsum function allows you to pass o

> mockstudy$agegp <- cut(mockstudy$age, breaks=c(18,50,60,70,90), right=FALSE)
>
> ## create weights based on agegp and sex distribution
> tab1 <- with(mockstudy, table(agegp, sex))
> tab1
      sex
agegp  Male Female
[18,50) 152    110
[50,60) 258    178
[60,70) 295    173
[70,90) 211    122
> tab2 <- with(mockstudy, table(agegp, sex, arm))
> gpwts <- rep(tab1, length(unique(mockstudy$arm)))/tab2
>
> ## apply weights to subjects
> index <- with(mockstudy, cbind(as.numeric(agegp), as.numeric(sex), as.numeric(as.factor(arm))))
> mockstudy$wts <- gpwts[index]
>
> ## show weights by treatment arm group
> tapply(mockstudy$wts, mockstudy$arm, summary)
$`A: IFL`
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 2.923  3.225   3.548   3.502   3.844   4.045

$`F: FOLFOX`
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 2.033  2.070   2.201   2.169   2.263   2.303

$`G: IROX`
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 3.667  3.734   4.023   3.945   4.031   4.471

```

```
> mockstudy$newvarA <- as.numeric(mockstudy$arm=='A: IFL')
> tab1 <- modelsum(newvarA ~ ast + bmi + hgb, data=mockstudy, subset=(arm != 'G: IROX'),
+                   family=binomial)
> summary(tab1, title='No Case Weights used')
No Case Weights used
OR  CI.lower.OR CI.upper.OR p.value concordance Nmiss
(Intercept) 0.590 0.473 0.735 < 0.001 0.550 210
ast 1.003 0.998 1.008 0.258
(Intercept) 0.578 0.306 1.093 0.091 0.500 29
Body Mass Index (kg/m^2) 1.003 0.980 1.026 0.808
(Intercept) 1.006 0.386 2.631 0.990 0.514 210
hgb 0.965 0.894 1.043 0.372
```

```
>
> suppressWarnings({
+   tab2 <- modelsum(newvarA ~ ast + bmi + hgb, data=mockstudy, subset=(arm != 'G: IROX'),
+                   weights=wts, family=binomial)
+   summary(tab2, title='Case Weights used')
+ })
```

```
Case Weights used
OR  CI.lower.OR CI.upper.OR p.value concordance Nmiss
(Intercept) 0.956 0.837 1.091 0.504 0.550 210
ast 1.003 1.000 1.006 0.068
(Intercept) 0.957 0.658 1.393 0.820 0.500 29
Body Mass Index (kg/m^2) 1.002 0.988 1.016 0.780
(Intercept) 1.829 1.031 3.248 0.039 0.514 210
hgb 0.956 0.913 1.001 0.058
```

15. Use modelsum within an Sweave document

For those users who wish to create tables within an Sweave document, the following code seems to work.

```
\documentclass{article}

\usepackage{longtable}
\usepackage{pdfpages}

\begin{document}

\section{Read in Data}
<<echo=TRUE>>=
require(arsenal)
require(knitr)
require(rmarkdown)
data(mockstudy)

tab1 <- modelsum(bmi~sex+age, data=mockstudy)
@

\section{Convert Summary.modelsum to LaTeX}
<<echo=TRUE, results='hide', message=FALSE>>=
capture.output(summary(tab1), file="Test.md")

## Convert R Markdown Table to LaTeX
render("Test.md", pdf_document(keep_tex=TRUE))
@

\includepdf{Test.pdf}

\end{document}
```

16. Export modelsum results to a .CSV file

When looking at multiple variables it is sometimes useful to export the results to a csv file. The `as.data.frame()` function can be used to export the results to a csv file.

```
> summary(tab2, text=T)
```

	OR	CI.lower.OR	CI.upper.OR	p.value	concordance	Nmiss
(Intercept)	0.956	0.837	1.091	0.504	0.550	210
ast	1.003	1.000	1.006	0.068		
(Intercept)	0.957	0.658	1.393	0.820	0.500	29
Body Mass Index (kg/m^2)	1.002	0.988	1.016	0.780		
(Intercept)	1.829	1.031	3.248	0.039	0.514	210
hgb	0.956	0.913	1.001	0.058		

```
> tmp <- as.data.frame(tab2)
```

```
> tmp
```

model	term	label	term.type	OR
1	1 (Intercept)	(Intercept)	Intercept	0.9559704
2	1 ast	ast	Term	1.0027311
3	2 (Intercept)	(Intercept)	Intercept	0.9573694
4	2 bmi Body Mass Index (kg/m^2)		Term	1.0019251
5	3 (Intercept)	(Intercept)	Intercept	1.8287083
6	3 hgb	hgb	Term	0.9563507

	CI.lower.OR	CI.upper.OR	p.value	concordance	Nmiss
1	0.8373522	1.090904	0.50443340	0.5499494	210
2	0.9998110	1.005696	0.06813456	0.5499494	210
3	0.6579225	1.392859	0.81981779	0.5002561	29
4	0.9884804	1.015561	0.78019163	0.5002561	29
5	1.0311954	3.247941	0.03911088	0.5138162	210
6	0.9132041	1.001419	0.05770821	0.5138162	210

```
> # write.csv(tmp, '/my/path/here/mymodel.csv')
```

17. Write modelsum object to a separate Word or HTML file

```
> ## write to an HTML document
```

```
> write2html(tab2, "~/ibm/trash.html")
```

```
>
```

```
> ## write to a Word document
```

```
> write2word(tab2, "~/ibm/trash.doc", title="My table in Word")
```

18. Use modelsum in R Shiny

The easiest way to output a `modelsum()` object in an R Shiny app is to use the `tableOutput()` UI in combination with `renderTable()`.

```
> # A standalone shiny app
```

```
> library(shiny)
```

```
> library(arsenal)
```

```
> data(mockstudy)
```

```
>
```

```
> shinyApp(
```

```
+   ui = fluidPage(tableOutput("table")),
```

```
+   server = function(input, output) {
```

```
+     output$table <- renderTable({
```

```
+       as.data.frame(summary(modelsum(age ~ sex, data = mockstudy), text = "html"))
```

```
+     }, sanitize.text.function = function(x) x)
```

```
+   }
```

```
+ )
```

This can be especially powerful if you feed the selections from a `selectInput(multiple = TRUE)` into `formuli`.

23. Use modelsum in bookdown

Since the backbone of `modelsum()` is `knitr::kable()`, tables still render well in bookdown. However, `print.summary.modelsum()` can be used to print the results of a `modelsum()` object to a file.

```
> summary(modelsum(age ~ sex, data = mockstudy), title="(\\#tab:mytableby) Caption here")
Available Function Options
Summary statistics
The available summary statistics, by variable type, are:
```

```
ordinal: Ordinal logistic regression models
default: Nmiss, OR, CI.lower.OR, CI.upper.OR, p.value
optional: estimate, CI.OR, CI.estimate, CI.lower.estimate, CI.upper.estimate, N, Nmiss2, endpoint, std.error
binomial,quasibinomial: Logistic regression models
default: OR, CI.lower.OR, CI.upper.OR, p.value, concordance, Nmiss
optional: estimate, CI.OR, CI.estimate, CI.lower.estimate, CI.upper.estimate, N, Nmiss2, endpoint, std.error
gaussian: Linear regression models
default: estimate, std.error, p.value, adj.r.squared, Nmiss
optional: CI.estimate, CI.lower.estimate, CI.upper.estimate, N, Nmiss2, statistic, standard.estimate, endpoint
poisson, quasipoisson: Poisson regression models
default: RR, CI.lower.RR, CI.upper.RR, p.value, Nmiss
optional: CI.RR, CI.estimate, CI.lower.estimate, CI.upper.estimate, CI.RR, Nmiss2, std.error, estimate, statistic
negbin: Negative binomial regression models
default: RR, CI.lower.RR, CI.upper.RR, p.value, Nmiss
optional: CI.RR, CI.estimate, CI.lower.estimate, CI.upper.estimate, CI.RR, Nmiss2, std.error, estimate, statistic
survival: Cox models
default: HR, CI.lower.HR, CI.upper.HR, p.value, concordance, Nmiss
optional: CI.HR, CI.estimate, CI.lower.estimate, CI.upper.estimate, N, Nmiss2, estimate, std.error, endpoint
The full description of these parameters that can be shown for models include:
```

```
N: a count of the number of observations used in the analysis
Nmiss: only show the count of the number of missing values if there are some missing values
Nmiss2: always show a count of the number of missing values for a model
endpoint: dependent variable used in the model
std.err: print the standard error
statistic: test statistic
statistic.F: test statistic (F test)
p.value: print the p-value
r.squared: print the model R-square
adj.r.squared: print the model adjusted R-square
r.squared: print the model R-square
concordance: print the model C statistic (which is the AUC for logistic models)
logLik: print the loglikelihood value
p.value.log: print the p-value for the overall model likelihood test
p.value.wald: print the p-value for the overall model wald test
p.value.sc: print the p-value for overall model score test
AIC: print the Akaike information criterion
BIC: print the Bayesian information criterion
null.deviance: null deviance
deviance: model deviance
df.residual: degrees of freedom for the residual
df.null: degrees of freedom for the null model
dispersion: This is used in Poisson models and is defined as the deviance/df.residual
statistic.sc: overall model score statistic
std.error.concordance: standard error for the C statistic
HR: print the hazard ratio (for survival models), i.e. exp(beta)
CI.lower.HR, CI.upper.HR: print the confidence interval for the HR
OR: print the odd's ratio (for logistic models), i.e. exp(beta)
CI.lower.OR, CI.upper.OR: print the confidence interval for the OR
RR: print the risk ratio (for poisson models), i.e. exp(beta)
CI.lower.RR, CI.upper.RR: print the confidence interval for the RR
```

```
estimate: print beta coefficient
standardized.estimate: print the standardized beta coefficient
CI.lower.estimate, CI.upper.estimate: print the confidence interval for the beta coefficient
edf: print the effective degrees of freedom.
theta: print the estimate of theta.
SE.theta: print the estimate of theta's standard error.
modelsum.control settings
```

A quick way to see what arguments are possible to utilize in a function is to use the `args()` command. Setting

```
> args(modelsum.control)
function (digits = 3L, digits.ratio = 3L, digits.p = 3L, format.p = TRUE,
  show.adjust = TRUE, show.intercept = TRUE, conf.level = 0.95,
  ordinal.stats = c("OR", "CI.lower.OR", "CI.upper.OR", "p.value",
    "Nmiss"), binomial.stats = c("OR", "CI.lower.OR", "CI.upper.OR",
    "p.value", "concordance", "Nmiss"), gaussian.stats = c("estimate",
    "std.error", "p.value", "adj.r.squared", "Nmiss"), poisson.stats = c("RR",
    "CI.lower.RR", "CI.upper.RR", "p.value", "Nmiss"), negbin.stats = c("RR",
    "CI.lower.RR", "CI.upper.RR", "p.value", "Nmiss"), survival.stats = c("HR",
    "CI.lower.HR", "CI.upper.HR", "p.value", "concordance",
    "Nmiss"), stat.labels = list(), ...)
```

NULL

summary.modelsum settings

The `summary.modelsum` function has options that modify how the table appears (such as adding a title or modifying

```
> args(arsenal::summary.modelsum)
function (object, ..., labelTranslations = NULL, text = FALSE,
  title = NULL, term.name = "")
NULL
```

NULL

The paired function

<https://cran.r-project.org/web/packages/arsenal/vignettes/paired.html>

The paired function

Ethan Heinzen, Beth Atkinson, Jason Sinnwell

09 November, 2018

Introduction

Simple Example

NAs

Available Function Options

Testing options

paired.control settings

summary.tableby settings

Introduction

Another one of the most common tables in medical literature includes summary statistics for a set of variables

This vignette is light on purpose; `paired()` piggybacks off of `tableby`, so most documentation there applies

Simple Example

The first step when using the `paired()` function is to load the `arsenal` package. We can't use `mockstudy` here

```
library(arsenal)
dat <- data.frame(
```

```

tp = paste0("Time Point ", c(1, 2, 1, 2, 1, 2, 1, 2, 1, 2)),
id = c(1, 1, 2, 2, 3, 3, 4, 4, 5, 6),
Cat = c("A", "A", "A", "B", "B", "B", "B", "A", NA, "B"),
Fac = factor(c("A", "B", "C", "A", "B", "C", "A", "B", "C", "A")),
Num = c(1, 2, 3, 4, 4, 3, 3, 4, 0, NA),
Ord = ordered(c("I", "II", "II", "III", "III", "III", "I", "III", "II", "I")),
Lgl = c(TRUE, TRUE, FALSE, TRUE, FALSE, TRUE, TRUE, FALSE, FALSE, FALSE),
Dat = as.Date("2018-05-01") + c(1, 1, 2, 2, 3, 4, 5, 6, 3, 4),
stringsAsFactors = FALSE
)

```

To create a simple table stratified by time point, use a formula= statement to specify the variables that y

```

p <- paired(tp ~ Cat + Fac + Num + Ord + Lgl + Dat, data = dat, id = id, signed.rank.exact = FALSE)
summary(p)

```

	Time Point 1 (N=4)	Time Point 2 (N=4)	Difference (N=4)	p value
Cat	1.000			
A	2 (50.0%)	2 (50.0%)	1 (50.0%)	
B	2 (50.0%)	2 (50.0%)	1 (50.0%)	
Fac	0.261			
A	2 (50.0%)	1 (25.0%)	2 (100.0%)	
B	1 (25.0%)	2 (50.0%)	1 (100.0%)	
C	1 (25.0%)	1 (25.0%)	1 (100.0%)	
Num	0.391			
Mean (SD)	2.750 (1.258)	3.250 (0.957)	0.500 (1.000)	
Range	1.000 - 4.000	2.000 - 4.000	-1.000 - 1.000	
Ord	0.174			
I	2 (50.0%)	0 (0.0%)	2 (100.0%)	
II	1 (25.0%)	1 (25.0%)	1 (100.0%)	
III	1 (25.0%)	3 (75.0%)	0 (0.0%)	
Lgl	1.000			
FALSE	2 (50.0%)	1 (25.0%)	2 (100.0%)	
TRUE	2 (50.0%)	3 (75.0%)	1 (50.0%)	
Dat	0.182			
median	2018-05-03	2018-05-04	0.500	
Range	2018-05-02 - 2018-05-06	2018-05-02 - 2018-05-07	0.000 - 1.000	

The third column shows the difference between time point 1 and time point 2. For categorical variables, it

NAs

Note that by default, observations which do not have both timepoints are removed. This is easily changed us

```

p <- paired(tp ~ Cat + Fac + Num + Ord + Lgl + Dat, data = dat, id = id,
signed.rank.exact = FALSE, na.action = na.paired("fill"))

```

```

summary(p)

```

	Time Point 1 (N=6)	Time Point 2 (N=6)	Difference (N=6)	p value
Cat	1.000			
N-Miss	2	1	2	
A	2 (50.0%)	2 (40.0%)	1 (50.0%)	
B	2 (50.0%)	3 (60.0%)	1 (50.0%)	
Fac	0.261			
N-Miss	1	1	2	
A	2 (40.0%)	2 (40.0%)	2 (100.0%)	
B	1 (20.0%)	2 (40.0%)	1 (100.0%)	
C	2 (40.0%)	1 (20.0%)	1 (100.0%)	
Num	0.391			
N-Miss	1	2	2	
Mean (SD)	2.200 (1.643)	3.250 (0.957)	0.500 (1.000)	
Range	0.000 - 4.000	2.000 - 4.000	-1.000 - 1.000	

```

Ord          0.174
  N-Miss    1   1   2
  I         2 (40.0%)   1 (20.0%)   2 (100.0%)
  II        2 (40.0%)   1 (20.0%)   1 (100.0%)
  III       1 (20.0%)   3 (60.0%)   0 (0.0%)
Lgl          1.000
  N-Miss    1   1   2
  FALSE     3 (60.0%)   2 (40.0%)   2 (100.0%)
  TRUE      2 (40.0%)   3 (60.0%)   1 (50.0%)
Dat          0.182
  N-Miss    1   1   2
  median    2018-05-04 2018-05-05 0.500
  Range     2018-05-02 - 2018-05-06 2018-05-02 - 2018-05-07 0.000 - 1.000
For more details, see the help page for na.paired().

```

Available Function Options

Testing options

The tests used to calculate p-values differ by the variable type, but can be specified explicitly in the function arguments.

The following tests are accepted:

`paired.t`: A paired t-test.

`mcnemar`: McNemar's test.

`signed.rank`: the signed-rank test.

`sign.test`: the sign test.

`notest`: Don't perform a test.

`paired.control` settings

A quick way to see what arguments are possible to utilize in a function is to use the `args()` command. Setting

```
args(paired.control)
```

```

## function (test = TRUE, diff = TRUE, test.pname = NULL, numeric.test = "paired.t",
##   cat.test = "mcnemar", ordered.test = "signed.rank", date.test = "paired.t",
##   numeric.stats = c("Nmiss", "meansd", "range"), cat.stats = c("Nmiss",
##     "countpct"), ordered.stats = c("Nmiss", "countpct"),
##   date.stats = c("Nmiss", "median", "range"), stats.labels = list(Nmiss = "N-Miss",
##     Nmiss2 = "N-Miss", meansd = "Mean (SD)", medianq1q3 = "Median (Q1, Q3)",
##     q1q3 = "Q1, Q3", range = "Range", countpct = "Count (Pct)"),
##   digits = 3L, digits.count = 0L, digits.p = 3L, format.p = TRUE,
##   conf.level = 0.95, mcnemar.correct = TRUE, signed.rank.exact = NULL,
##   signed.rank.correct = TRUE, ...)
## NULL

```

`summary.tableby` settings

Since the "paired" object inherits "tableby", the `summary.tableby` function is what's actually used to format

```
args(arsenal::summary.tableby)
```

```

## function (object, ..., labelTranslations = NULL, text = FALSE,
##   title = NULL, pfootnote = FALSE, term.name = "")
## NULL

```

The tableby function

<https://cran.r-project.org/web/packages/arsenal/vignettes/tableby.html>

The tableby function

Beth Atkinson, Ethan Heinzen, Jason Sinnwell, Shannon McDonnell and Greg Dougherty

09 November, 2018

Introduction

Simple Example

Pretty text version of table

Pretty Rmarkdown version of table

Data frame version of table

Summaries using standard R code

Modifying Output

Add labels

Change summary statistics globally

Change summary statistics within the formula

Controlling Options for Categorical Tests (Chisq and Fisher's)

Modifying the look & feel in Word documents

Additional Examples

1. Summarize without a group/by variable

2. Display footnotes indicating which "test" was used

3. Summarize an ordered factor

4. Summarize a survival variable

5. Summarize date variables

6. Summarize multiple variables without typing them out

7. Subset the dataset used in the analysis

8. Create combinations of variables on the fly

9. Transform variables on the fly

10. Subsetting (change the ordering of the variables, delete a variable, sort by p-value, filter by p-value)

11. Merge two tableby objects together

12. Add a title to the table

13. Modify how missing values are displayed

14. Modify the number of digits used

15. Create a user-defined summary statistic

16. Use case-weights for creating summary statistics

17. Create your own p-value and add it to the table

18. For two-level categorical variables or one-line numeric variables, simplify the output.

19. Use tableby within an Sweave document

20. Export tableby object to a .CSV file

21. Write tableby object to a separate Word or HTML file

22. Use tableby in R Shiny

23. Use tableby in bookdown

24. Adjust tableby for multiple p-values

Available Function Options

Summary statistics

Testing options

tableby.control settings

summary.tableby settings

Introduction

One of the most common tables in medical literature includes summary statistics for a set of variables, oft

In developing the tableby() function, the goal was to bring the best features of these macros into an R fun

This report provides step-by-step directions for using the functions associated with tableby(). All functio

Simple Example

The first step when using the `tableby` function is to load the `arsenal` package. All the examples in this report

```
require(arsenal)
require(knitr)
require(survival)
data(mockstudy) ##load data
dim(mockstudy) ##look at how many subjects and variables are in the dataset
## [1] 1499 14
# help(mockstudy) ##learn more about the dataset and variables
str(mockstudy) ##quick look at the data
## 'data.frame': 1499 obs. of 14 variables:
## $ case : int 110754 99706 105271 105001 112263 86205 99508 90158 88989 90515 ...
## $ age : atomic 67 74 50 71 69 56 50 57 51 63 ...
## .. attr(*, "label")= chr "Age in Years"
## $ arm : atomic F: FOLFOX A: IFL A: IFL G: IROX ...
## .. attr(*, "label")= chr "Treatment Arm"
## $ sex : Factor w/ 2 levels "Male","Female": 1 2 2 2 2 1 1 1 2 1 ...
## $ race : atomic Caucasian Caucasian Caucasian Caucasian ...
## .. attr(*, "label")= chr "Race"
## $ fu.time : int 922 270 175 128 233 120 369 421 387 363 ...
## $ fu.stat : int 2 2 2 2 2 2 2 2 2 2 ...
## $ ps : int 0 1 1 1 0 0 0 0 1 1 ...
## $ hgb : num 11.5 10.7 11.1 12.6 13 10.2 13.3 12.1 13.8 12.1 ...
## $ bmi : atomic 25.1 19.5 NA 29.4 26.4 ...
## .. attr(*, "label")= chr "Body Mass Index (kg/m^2)"
## $ alk.phos : int 160 290 700 771 350 569 162 152 231 492 ...
## $ ast : int 35 52 100 68 35 27 16 12 25 18 ...
## $ mdquality.s: int NA 1 1 1 NA 1 1 1 1 1 ...
## $ age.ord : Ord.factor w/ 8 levels "10-19"<"20-29"<...: 6 7 4 7 6 5 4 5 5 6 ...
```

To create a simple table stratified by treatment arm, use a formula statement to specify the variables that

```
tab1 <- tableby(arm ~ sex + age, data=mockstudy)
```

If you want to take a quick look at the table, you can use `summary()` on your `tableby` object and the table will

Pretty text version of table

If you want a nicer version in your console window then add the `text=TRUE` option.

```
summary(tab1, text=TRUE)
##
##
## | | A: IFL (N=428) | F: FOLFOX (N=691) | G: IROX (N=380) | Total (N=1499) | p value |
## |-----|-----|-----|-----|-----|-----|
## | sex | | | | | | 0.190 |
## | Male | 277 (64.7%) | 411 (59.5%) | 228 (60.0%) | 916 (61.1%) | |
## | Female | 151 (35.3%) | 280 (40.5%) | 152 (40.0%) | 583 (38.9%) | |
## | Age in Years | | | | | | 0.614 |
## | Mean (SD) | 59.673 (11.365) | 60.301 (11.632) | 59.763 (11.499) | 59.985 (11.519) | |
## | Range | 27.000 - 88.000 | 19.000 - 88.000 | 26.000 - 85.000 | 19.000 - 88.000 | |
```

Pretty Rmarkdown version of table

In order for the report to look nice within an R markdown (`knitr`) report, you just need to specify `results=`

```
summary(tab1)
A: IFL (N=428) F: FOLFOX (N=691) G: IROX (N=380) Total (N=1499) p value
sex 0.190
  Male 277 (64.7%) 411 (59.5%) 228 (60.0%) 916 (61.1%)
  Female 151 (35.3%) 280 (40.5%) 152 (40.0%) 583 (38.9%)
```

```
Age in Years          0.614
  Mean (SD)    59.673 (11.365) 60.301 (11.632) 59.763 (11.499) 59.985 (11.519)
  Range    27.000 - 88.000 19.000 - 88.000 26.000 - 85.000 19.000 - 88.000
```

Data frame version of table

If you want a data.frame version, simply use as.data.frame.

```
as.data.frame(tab1)
##   variable      term      label variable.type      A: IFL      F: FOLFOX
## 1      sex      sex      sex      categorical
## 2      sex countpct      Male      categorical 277.00000, 64.71963 411.00000, 59.47902
## 3      sex countpct      Female      categorical 151.00000, 35.28037 280.00000, 40.52098
## 4      age      age Age in Years      numeric
## 5      age      meansd      Mean (SD)      numeric 59.67290, 11.36454 60.30101, 11.63225
## 6      age      range      Range      numeric      27, 88      19, 88
##      G: IROX      Total      test      p.value
## 1      Pearson's Chi-squared test 0.1904388
## 2      228, 60 916.0000, 61.1074 Pearson's Chi-squared test 0.1904388
## 3      152, 40 583.0000, 38.8926 Pearson's Chi-squared test 0.1904388
## 4      Linear Model ANOVA 0.6143859
## 5 59.76316, 11.49930 59.98532, 11.51877 Linear Model ANOVA 0.6143859
## 6      26, 85      19, 88 Linear Model ANOVA 0.6143859

Summaries using standard R code
## base R frequency example
tmp <- table(Gender=mockstudy$sex, "Study Arm"=mockstudy$arm)
tmp
##      Study Arm
## Gender  A: IFL F: FOLFOX G: IROX
## Male    277    411    228
## Female  151    280    152
# Note: The continuity correction is applied by default in R (not used in %table)
chisq.test(tmp)
##
## Pearson's Chi-squared test
##
## data: tmp
## X-squared = 3.3168, df = 2, p-value = 0.1904
## base R numeric summary example
tapply(mockstudy$age, mockstudy$arm, summary)
## $`A: IFL`
##   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  27.00  53.00   61.00   59.67  68.00   88.00
##
## $`F: FOLFOX`
##   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   19.0   52.0   61.0   60.3   69.0   88.0
##
## $`G: IROX`
##   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  26.00  52.00   61.00   59.76  68.00   85.00
summary(aov(age ~ arm, data=mockstudy))
##      Df Sum Sq Mean Sq F value Pr(>F)
## arm    2    129    64.7    0.487  0.614
## Residuals 1496 198628    132.8
```

Modifying Output

Add labels

In the above example, age is shown with a label (Age in Years), but sex is listed "as is" with lower case l

```
## Look at one variable's label
attr(mockstudy$age, 'label')
## [1] "Age in Years"
## See all the variables with a label
unlist(lapply(mockstudy, 'attr', 'label'))
##              age              arm              race
##      "Age in Years"      "Treatment Arm"      "Race"
##              bmi
## "Body Mass Index (kg/m^2)"
# Can also use labels(mockstudy)
If you want to add labels to other variables, there are a couple of options. First, you could add labels to

attr(mockstudy$sex, 'label') <- 'Gender'

tab1 <- tableby(arm ~ sex + age, data=mockstudy)
summary(tab1)
A: IFL (N=428)  F: FOLFOX (N=691)  G: IROX (N=380) Total (N=1499)  p value
Gender              0.190
  Male 277 (64.7%) 411 (59.5%) 228 (60.0%) 916 (61.1%)
  Female 151 (35.3%) 280 (40.5%) 152 (40.0%) 583 (38.9%)
Age in Years              0.614
  Mean (SD) 59.673 (11.365) 60.301 (11.632) 59.763 (11.499) 59.985 (11.519)
  Range 27.000 - 88.000 19.000 - 88.000 26.000 - 85.000 19.000 - 88.000
You can also use the built-in data.frame method for labels<-:

labels(mockstudy) <- c(age = 'Age, yrs', sex = "Gender")

tab1 <- tableby(arm ~ sex + age, data=mockstudy)
summary(tab1)
A: IFL (N=428)  F: FOLFOX (N=691)  G: IROX (N=380) Total (N=1499)  p value
Gender              0.190
  Male 277 (64.7%) 411 (59.5%) 228 (60.0%) 916 (61.1%)
  Female 151 (35.3%) 280 (40.5%) 152 (40.0%) 583 (38.9%)
Age, yrs              0.614
  Mean (SD) 59.673 (11.365) 60.301 (11.632) 59.763 (11.499) 59.985 (11.519)
  Range 27.000 - 88.000 19.000 - 88.000 26.000 - 85.000 19.000 - 88.000
Another option is to add labels after you have created the table

mylabels <- list(sex = "SEX", age = "Age, yrs")
summary(tab1, labelTranslations = mylabels)
A: IFL (N=428)  F: FOLFOX (N=691)  G: IROX (N=380) Total (N=1499)  p value
SEX              0.190
  Male 277 (64.7%) 411 (59.5%) 228 (60.0%) 916 (61.1%)
  Female 151 (35.3%) 280 (40.5%) 152 (40.0%) 583 (38.9%)
Age, yrs              0.614
  Mean (SD) 59.673 (11.365) 60.301 (11.632) 59.763 (11.499) 59.985 (11.519)
  Range 27.000 - 88.000 19.000 - 88.000 26.000 - 85.000 19.000 - 88.000
Alternatively, you can check the variable labels and manipulate them with a function called labels, which w

labels(tab1)
##      arm      sex      age
##      "arm"    "Gender" "Age, yrs"
labels(tab1) <- c(arm="Treatment Assignment", age="Baseline Age (yrs)")
labels(tab1)
##      arm      sex      age
## "Treatment Assignment"      "Gender"      "Baseline Age (yrs)"
summary(tab1)
```

A: IFL (N=428) F: FOLFOX (N=691) G: IROX (N=380) Total (N=1499) p value

Gender 0.190

Male 277 (64.7%) 411 (59.5%) 228 (60.0%) 916 (61.1%)

Female 151 (35.3%) 280 (40.5%) 152 (40.0%) 583 (38.9%)

Baseline Age (yrs) 0.614

Mean (SD) 59.673 (11.365) 60.301 (11.632) 59.763 (11.499) 59.985 (11.519)

Range 27.000 - 88.000 19.000 - 88.000 26.000 - 85.000 19.000 - 88.000

Change summary statistics globally

Currently the default behavior is to summarize continuous variables with: Number of missing values, Mean (SD)

```
mycontrols <- tableby.control(test=FALSE, total=FALSE,
                             numeric.test="kwt", cat.test="chisq",
                             numeric.stats=c("N", "median", "q1q3"),
                             cat.stats=c("countpct"),
                             stats.labels=list(N='Count', median='Median', q1q3='Q1,Q3'))
```

```
tab2 <- tableby(arm ~ sex + age, data=mockstudy, control=mycontrols)
```

```
summary(tab2)
```

A: IFL (N=428) F: FOLFOX (N=691) G: IROX (N=380)

Gender

Male 277 (64.7%) 411 (59.5%) 228 (60.0%)

Female 151 (35.3%) 280 (40.5%) 152 (40.0%)

Age, yrs

Count 428 691 380

Median 61.000 61.000 61.000

Q1,Q3 53.000, 68.000 52.000, 69.000 52.000, 68.000

You can also change these settings directly in the tableby call.

```
tab3 <- tableby(arm ~ sex + age, data=mockstudy, test=FALSE, total=FALSE,
               numeric.stats=c("median", "q1q3"), numeric.test="kwt")
```

```
summary(tab3)
```

A: IFL (N=428) F: FOLFOX (N=691) G: IROX (N=380)

Gender

Male 277 (64.7%) 411 (59.5%) 228 (60.0%)

Female 151 (35.3%) 280 (40.5%) 152 (40.0%)

Age, yrs

Median 61.000 61.000 61.000

Q1, Q3 53.000, 68.000 52.000, 69.000 52.000, 68.000

Change summary statistics within the formula

In addition to modifying summary options globally, it is possible to modify the test and summary statistics

The tests function can do a quick check on what tests were performed on each variable in tableby.

```
tab.test <- tableby(arm ~ kwt(age) + anova(bmi) + notest(ast), data=mockstudy)
```

```
tests(tab.test)
```

```
##           Variable  p.value           Method
## age           Age, yrs 0.6390614 Kruskal-Wallis rank sum test
## bmi Body Mass Index (kg/m^2) 0.8916552 Linear Model ANOVA
## ast           ast      NA           No test
```

```
summary(tab.test)
```

A: IFL (N=428) F: FOLFOX (N=691) G: IROX (N=380) Total (N=1499) p value

Age, yrs 0.639

Mean (SD) 59.673 (11.365) 60.301 (11.632) 59.763 (11.499) 59.985 (11.519)

Range 27.000 - 88.000 19.000 - 88.000 26.000 - 85.000 19.000 - 88.000

Body Mass Index (kg/m^2) 0.892

N-Miss 9 20 4 33

Mean (SD) 27.290 (5.552) 27.210 (5.173) 27.106 (5.751) 27.206 (5.432)

Range 14.053 - 53.008 16.649 - 49.130 15.430 - 60.243 14.053 - 60.243

```
ast
```

```
  N-Miss    69   141   56   266
  Mean (SD)   37.292 (28.036) 35.202 (26.659) 35.670 (25.807) 35.933 (26.843)
  Range    10.000 - 205.000    7.000 - 174.000 5.000 - 176.000 5.000 - 205.000
```

Summary statistics for any individual variable can also be modified, but it must be done as secondary argument

```
tab.test <- tableby(arm ~ kwt(ast, "Nmiss2","median") + anova(age, "N","mean") +
  notest(bmi, "Nmiss","median"), data=mockstudy)
```

```
summary(tab.test)
```

```
A: IFL (N=428) F: FOLFOX (N=691) G: IROX (N=380) Total (N=1499) p value
```

```
ast
  N-Miss    69   141   56   266
  Median   29.000 25.500 27.000 27.000
```

```
Age, yrs
  N      428 691 380 1499
  mean 59.7  60.3 59.8  60
```

```
Body Mass Index (kg/m^2)
```

```
  N-Miss    9   20   4   33
  Median   26.234 26.525 25.978 26.325
```

Controlling Options for Categorical Tests (Chisq and Fisher's)

The formal tests for categorical variables against the levels of the by variable, chisq and fe, have options

```
set.seed(100)
```

```
tab.catsim <- tableby(arm ~ sex + race, cat.test="fe", simulate.p.value=TRUE, B=500, data=mockstudy)
```

```
tests(tab.catsim)
```

```
Variable p.value
sex Gender 0.2195609 race Race 0.3093812 Method sex Fisher's Exact Test for Count Data with simulated p-values
```

The chis-square test on 2x2 tables applies Yates' continuity correction by default, so we provide an option

```
cat.correct <- tableby(arm ~ sex + race, cat.test="chisq", subset = !grepl("^F", arm), data=mockstudy)
```

```
tests(cat.correct)
```

```
Variable p.value Method
sex Gender 0.1666280 Pearson's Chi-squared test race Race 0.8108543 Pearson's Chi-squared test
```

```
cat.nocorrect <- tableby(arm ~ sex + race, cat.test="chisq", subset = !grepl("^F", arm),
  chisq.correct=FALSE, data=mockstudy)
```

```
tests(cat.nocorrect)
```

```
Variable p.value Method
sex Gender 0.1666280 Pearson's Chi-squared test race Race 0.8108543 Pearson's Chi-squared test
```

Modifying the look & feel in Word documents

You can easily create Word versions of tableby output via an Rmarkdown report and the default options will

The functionality listed in this next paragraph is coming soon but needs an upgraded version of RStudio If

```
output: word_document
```

```
reference_docx: /projects/bsi/gentools/R/lib320/arsenal/doc/WordStylesReference01.docx
```

For more information on changing the look/feel of your Word document, see the Rmarkdown documentation website

Additional Examples

Here are multiple examples showing how to use some of the different options.

1. Summarize without a group/by variable

```
tab.noby <- tableby(~ bmi + sex + age, data=mockstudy)
```

```
summary(tab.noby)
```

```
Overall (N=1499)
```

Body Mass Index (kg/m²)

N-Miss 33

Mean (SD) 27.206 (5.432)

Range 14.053 - 60.243

Gender

Male 916 (61.1%)

Female 583 (38.9%)

Age, yrs

Mean (SD) 59.985 (11.519)

Range 19.000 - 88.000

2. Display footnotes indicating which "test" was used

summary(tab.test) #, pfootnote=TRUE)

A: IFL (N=428) F: FOLFOX (N=691) G: IROX (N=380) Total (N=1499) p value

ast 0.039

N-Miss 69 141 56 266

Median 29.000 25.500 27.000 27.000

Age, yrs 0.614

N 428 691 380 1499

mean 59.7 60.3 59.8 60

Body Mass Index (kg/m²)

N-Miss 9 20 4 33

Median 26.234 26.525 25.978 26.325

3. Summarize an ordered factor

When comparing groups of ordered data there are a couple of options. The default uses a general independence

```
mockstudy$age.ordnew <- ordered(c("a",NA,as.character(mockstudy$age.ord[-(1:2)])))
```

```
table(mockstudy$age.ord, mockstudy$sex)
```

```
##
```

```
##           Male Female
```

```
## 10-19      1      0
```

```
## 20-29      8     11
```

```
## 30-39     37     30
```

```
## 40-49    127     83
```

```
## 50-59    257    179
```

```
## 60-69    298    170
```

```
## 70-79    168    101
```

```
## 80-89     20      9
```

```
table(mockstudy$age.ordnew, mockstudy$sex)
```

```
##
```

```
##           Male Female
```

```
## 10-19      1      0
```

```
## 20-29      8     11
```

```
## 30-39     37     30
```

```
## 40-49    127     83
```

```
## 50-59    257    179
```

```
## 60-69    297    170
```

```
## 70-79    168    100
```

```
## 80-89     20      9
```

```
## a          1      0
```

```
class(mockstudy$age.ord)
```

```
## [1] "ordered" "factor"
```

```
summary(tableby(sex ~ age.ordnew, data = mockstudy)) #, pfootnote = TRUE)
```

```
Male (N=916)      Female (N=583)  Total (N=1499)  p value
```

```
age.ordnew              0.040
```

```
N-Miss      0      1      1
```

```
10-19      1 (0.1%)      0 (0.0%)      1 (0.1%)
```

```
20-29      8 (0.9%)     11 (1.9%)     19 (1.3%)
```

```

30-39    37 (4.0%)   30 (5.2%)   67 (4.5%)
40-49   127 (13.9%)  83 (14.3%)  210 (14.0%)
50-59   257 (28.1%) 179 (30.8%) 436 (29.1%)
60-69   297 (32.4%) 170 (29.2%) 467 (31.2%)
70-79   168 (18.3%) 100 (17.2%) 268 (17.9%)
80-89    20 (2.2%)   9 (1.5%)   29 (1.9%)
a      1 (0.1%)    0 (0.0%)    1 (0.1%)
summary(tableby(sex ~ kwt(age.ord), data = mockstudy)) #) #, pfootnote = TRUE)
Male (N=916)      Female (N=583) Total (N=1499) p value
age.ord          0.067
10-19    1 (0.1%)    0 (0.0%)    1 (0.1%)
20-29    8 (0.9%)   11 (1.9%)   19 (1.3%)
30-39    37 (4.0%)   30 (5.1%)   67 (4.5%)
40-49   127 (13.9%)  83 (14.2%)  210 (14.0%)
50-59   257 (28.1%) 179 (30.7%) 436 (29.1%)
60-69   298 (32.5%) 170 (29.2%) 468 (31.2%)
70-79   168 (18.3%) 101 (17.3%) 269 (17.9%)
80-89    20 (2.2%)   9 (1.5%)   29 (1.9%)

```

4. Summarize a survival variable

First look at the information that is presented by the `survfit()` function, then see how the same results can be obtained using `summary(survfit())`.

```

survfit(Surv(fu.time, fu.stat)~sex, data=mockstudy)
## Call: survfit(formula = Surv(fu.time, fu.stat) ~ sex, data = mockstudy)
##
##              n events median 0.95LCL 0.95UCL
## sex=Male    916      829      550      515      590
## sex=Female  583      527      543      511      575
survdiff(Surv(fu.time, fu.stat)~sex, data=mockstudy)
## Call:
## survdiff(formula = Surv(fu.time, fu.stat) ~ sex, data = mockstudy)
##
##              N Observed Expected (O-E)^2/E (O-E)^2/V
## sex=Male    916      829      830  0.000370  0.000956
## sex=Female  583      527      526  0.000583  0.000956
##
##  Chisq= 0  on 1 degrees of freedom, p= 1
summary(tableby(sex ~ Surv(fu.time, fu.stat), data=mockstudy))
Male (N=916)      Female (N=583) Total (N=1499) p value
Surv(fu.time, fu.stat)          0.975
Events      829 527 1356
Median Survival  550.000 543.000 546.000

```

It is also possible to obtain summaries of the % survival at certain time points (say the probability of survival at 1, 2, 3, 4, and 5 years).

```

summary(survfit(Surv(fu.time/365.25, fu.stat)~sex, data=mockstudy), times=1:5)
## Call: survfit(formula = Surv(fu.time/365.25, fu.stat) ~ sex, data = mockstudy)
##
##              sex=Male
## time n.risk n.event survival std.err lower 95% CI upper 95% CI
##    1    626    286    0.6870  0.0153    0.6576    0.7177
##    2    309    311    0.3437  0.0158    0.3142    0.3761
##    3    152    151    0.1748  0.0127    0.1516    0.2015
##    4     57     61    0.0941  0.0104    0.0759    0.1168
##    5     24     16    0.0628  0.0095    0.0467    0.0844
##
##              sex=Female
## time n.risk n.event survival std.err lower 95% CI upper 95% CI
##    1    380    202    0.6531  0.0197    0.6155    0.693

```

```
##      2      190      189      0.3277      0.0195      0.2917      0.368
##      3       95       90      0.1701      0.0157      0.1420      0.204
##      4       51       32      0.1093      0.0133      0.0861      0.139
##      5       18       12      0.0745      0.0126      0.0534      0.104
```

```
summary(tableby(sex ~ Surv(fu.time/365.25, fu.stat), data=mockstudy, times=1:5, surv.stats=c("NeventsSurv",
```

```
Male (N=916)      Female (N=583)      Total (N=1499)      p value
```

```
Surv(fu.time/365.25, fu.stat)      0.975
```

```
time = 1 286 (68.7) 202 (65.3) 488 (67.4)
```

```
time = 2 597 (34.4) 391 (32.8) 988 (33.7)
```

```
time = 3 748 (17.5) 481 (17.0) 1229 (17.3)
```

```
time = 4 809 (9.4) 513 (10.9) 1322 (10.1)
```

```
time = 5 825 (6.3) 525 (7.4) 1350 (6.8)
```

```
time = 1 626 380 1006
```

```
time = 2 309 190 499
```

```
time = 3 152 95 247
```

```
time = 4 57 51 108
```

```
time = 5 24 18 42
```

5. Summarize date variables

Date variables by default are summarized with the number of missing values, the median, and the range. For

```
set.seed(100)
```

```
N <- nrow(mockstudy)
```

```
mockstudy$dtentry <- mdy.Date(month=sample(1:12,N,replace=T), day=sample(1:29,N,replace=T),
                             year=sample(2005:2009,N,replace=T))
```

```
summary(tableby(sex ~ dtentry, data=mockstudy))
```

```
Male (N=916)      Female (N=583)      Total (N=1499)      p value
```

```
dtentry      0.554
```

```
N-Miss      3      2      5
```

```
Median      2007-06-16 2007-06-15 2007-06-15
```

```
Range      2005-01-03 - 2009-12-27 2005-01-01 - 2009-12-28 2005-01-01 - 2009-12-28
```

6. Summarize multiple variables without typing them out

Often one wants to summarize a number of variables. Instead of typing by hand each individual variable, an

```
## create a vector specifying the variable names
```

```
myvars <- names(mockstudy)
```

```
## select the 8th through the last variables
```

```
## paste them together, separated by the + sign
```

```
RHS <- paste(myvars[8:10], collapse="+")
```

```
RHS
```

```
[1] "ps+hgb+bmi"
```

```
## create a formula using the as.formula function
```

```
as.formula(paste('arm ~ ', RHS))
```

```
arm ~ ps + hgb + bmi
```

```
## use the formula in the tableby function
```

```
summary(tableby(as.formula(paste('arm ~ ', RHS)), data=mockstudy))
```

```
A: IFL (N=428)      F: FOLFOX (N=691)      G: IROX (N=380)      Total (N=1499)      p value
```

```
ps      0.903
```

```
N-Miss      69      141      56      266
```

```
Mean (SD)      0.529 (0.597)      0.547 (0.595)      0.537 (0.606)      0.539 (0.598)
```

```
Range      0.000 - 2.000      0.000 - 2.000      0.000 - 2.000      0.000 - 2.000
```

```
hgb      0.639
```

```
N-Miss      69      141      56      266
```

```
Mean (SD)      12.276 (1.686)      12.381 (1.763)      12.373 (1.680)      12.348 (1.719)
```

```
Range      9.060 - 17.300      9.000 - 18.200      9.000 - 17.000      9.000 - 18.200
```



```

Body Mass Index (kg/m^2)                                0.892
  N-Miss    9    20    4    33
  Mean (SD)   27.290 (5.552) 27.210 (5.173) 27.106 (5.751) 27.206 (5.432)
  Range    14.053 - 53.008 16.649 - 49.130 15.430 - 60.243 14.053 - 60.243

```

These steps can also be done using the `formulize` function.

```
## The formulize function does the paste and as.formula steps
```

```
tmp <- formulize('arm',myvars[8:10])
```

```
tmp
```

```
arm ~ ps + hgb + bmi
```

```
## More complex formulas could also be written using formulize
```

```
tmp2 <- formulize('arm',c('ps','hgb^2','bmi'))
```

```
## use the formula in the tableby function
```

```
summary(tableby(tmp, data=mockstudy))
```

```
A: IFL (N=428) F: FOLFOX (N=691) G: IROX (N=380) Total (N=1499) p value
```

```
ps                                0.903
```

```
  N-Miss    69   141   56   266
```

```
  Mean (SD)   0.529 (0.597) 0.547 (0.595) 0.537 (0.606) 0.539 (0.598)
```

```
  Range    0.000 - 2.000 0.000 - 2.000 0.000 - 2.000 0.000 - 2.000
```

```
hgb                                0.639
```

```
  N-Miss    69   141   56   266
```

```
  Mean (SD)  12.276 (1.686) 12.381 (1.763) 12.373 (1.680) 12.348 (1.719)
```

```
  Range    9.060 - 17.300 9.000 - 18.200 9.000 - 17.000 9.000 - 18.200
```

```
Body Mass Index (kg/m^2)                                0.892
```

```
  N-Miss    9    20    4    33
```

```
  Mean (SD)   27.290 (5.552) 27.210 (5.173) 27.106 (5.751) 27.206 (5.432)
```

```
  Range    14.053 - 53.008 16.649 - 49.130 15.430 - 60.243 14.053 - 60.243
```

7. Subset the dataset used in the analysis

Here are two ways to get the same result (limit the analysis to subjects `age>5` and in the F: FOLFOX treatment group)

The first approach uses the `subset` function applied to the dataset `mockstudy`. This example also selects a subset of variables (sex, ps, hgb, bmi)

```
newdata <- subset(mockstudy, subset=age>50 & arm=='F: FOLFOX', select = c(sex,ps:bmi))
```

```
dim(mockstudy)
```

```
## [1] 1499   16
```

```
table(mockstudy$arm)
```

```
##
```

```
##      A: IFL F: FOLFOX   G: IROX
```

```
##      428       691       380
```

```
dim(newdata)
```

```
## [1] 557    4
```

```
names(newdata)
```

```
## [1] "sex" "ps"  "hgb" "bmi"
```

```
summary(tableby(sex ~ ., data=newdata))
```

```
Male (N=333)   Female (N=224) Total (N=557)   p value
```

```
ps                                0.652
```

```
  N-Miss    64   44   108
```

```
  Mean (SD)   0.554 (0.600) 0.528 (0.602) 0.543 (0.600)
```

```
  Range    0.000 - 2.000 0.000 - 2.000 0.000 - 2.000
```

```
hgb                                < 0.001
```

```
  N-Miss    64   44   108
```

```
  Mean (SD)  12.720 (1.925) 12.063 (1.395) 12.457 (1.760)
```

```
  Range    9.000 - 18.200 9.100 - 15.900 9.000 - 18.200
```

```
bmi                                0.650
```

```
  N-Miss    9    6    15
```

```
  Mean (SD)  27.539 (4.780) 27.337 (5.508) 27.458 (5.081)
```

```

Range      17.927 - 47.458 16.649 - 49.130 16.649 - 49.130
The second approach does the same analysis but uses the subset argument within tableby to subset the data.
summary(tableby(sex ~ ps + hgb + bmi, subset=age>50 & arm=="F: FOLFOX", data=mockstudy))
Male (N=333)   Female (N=224)   Total (N=557)   p value
ps
  N-Miss    64   44   108
  Mean (SD)   0.554 (0.600)   0.528 (0.602)   0.543 (0.600)
  Range      0.000 - 2.000   0.000 - 2.000   0.000 - 2.000
hgb
  N-Miss    64   44   108
  Mean (SD)  12.720 (1.925)  12.063 (1.395)  12.457 (1.760)
  Range      9.000 - 18.200  9.100 - 15.900  9.000 - 18.200
Body Mass Index (kg/m^2)
  N-Miss     9    6    15
  Mean (SD)  27.539 (4.780)  27.337 (5.508)  27.458 (5.081)
  Range      17.927 - 47.458 16.649 - 49.130 16.649 - 49.130
8. Create combinations of variables on the fly
## create a variable combining the levels of mdquality.s and sex
with(mockstudy, table(interaction(mdquality.s,sex)))
##
##   0.Male   1.Male 0.Female 1.Female
##      77     686      47     437
summary(tableby(arm ~ interaction(mdquality.s,sex), data=mockstudy))
A: IFL (N=428) F: FOLFOX (N=691) G: IROX (N=380) Total (N=1499) p value
interaction(mdquality.s, sex)
  N-Miss    55   156   41   252
  0.Male    29 (7.8%)   31 (5.8%)   17 (5.0%)   77 (6.2%)
  1.Male   214 (57.4%)  285 (53.3%)  187 (55.2%)  686 (55.0%)
  0.Female  12 (3.2%)   21 (3.9%)   14 (4.1%)   47 (3.8%)
  1.Female 118 (31.6%)  198 (37.0%)  121 (35.7%)  437 (35.0%)
## create a new grouping variable with combined levels of arm and sex
summary(tableby(interaction(mdquality.s, sex) ~ age + bmi, data=mockstudy, subset=arm=="F: FOLFOX"))
0.Male (N=31)   1.Male (N=285)   0.Female (N=21) 1.Female (N=198)   Total (N=535)   p value
Age, yrs
  N-Miss    0    6    1    5    12
  Mean (SD)  63.065 (11.702) 60.653 (11.833) 60.810 (10.103) 58.924 (11.366) 60.159 (11.612)
  Range     41.000 - 82.000 19.000 - 88.000 42.000 - 81.000 29.000 - 83.000 19.000 - 88.000
Body Mass Index (kg/m^2)
  N-Miss     0    6    1    5    12
  Mean (SD)  26.633 (5.094) 27.387 (4.704) 27.359 (4.899) 27.294 (5.671) 27.307 (5.100)
  Range     20.177 - 41.766 17.927 - 47.458 19.801 - 39.369 16.799 - 44.841 16.799 - 47.458
9. Transform variables on the fly
Certain transformations need to be surrounded by I() so that R knows to treat it as a variable transformation
trans <- tableby(arm ~ I(age/10) + log(bmi) + factor(mdquality.s, levels=0:1, labels=c('N','Y')),
  data=mockstudy)
summary(trans)
A: IFL (N=428) F: FOLFOX (N=691) G: IROX (N=380) Total (N=1499) p value
Age, yrs
  N-Miss    55   156   41   252
  Mean (SD)   5.967 (1.136)   6.030 (1.163)   5.976 (1.150)   5.999 (1.152)
  Range      2.700 - 8.800   1.900 - 8.800   2.600 - 8.500   1.900 - 8.800
Body Mass Index (kg/m^2)
  N-Miss     9    20    4    33
  Mean (SD)   3.287 (0.197)   3.286 (0.183)   3.279 (0.200)   3.285 (0.192)
  Range      2.643 - 3.970   2.812 - 3.894   2.736 - 4.098   2.643 - 4.098
factor(mdquality.s, levels = 0:1, labels = c("N", "Y"))
  N-Miss    55   156   41   252
  N         41 (11.0%)  52 (9.7%)   31 (9.1%)   124 (9.9%)

```

Y	332 (89.0%)	483 (90.3%)	308 (90.9%)	1123 (90.1%)
---	-------------	-------------	-------------	--------------

The labels for these variables isn't exactly what we'd like so we can change modify those after the fact. I

```
labels(trans)
##                                arm
##                                "arm"
##                                I(age/10)
##                                "Age, yrs"
##                                log(bmi)
##                                "Body Mass Index (kg/m^2)"
##                                factor(mdquality.s, levels = 0:1, labels = c("N", "Y"))
## "factor(mdquality.s, levels = 0:1, labels = c(\"N\", \"Y\"))"
labels(trans)[2:4] <- c('Age per 10 yrs', 'log(BMI)', 'MD Quality')
labels(trans)
##                                arm
##                                "arm"
##                                I(age/10)
##                                "Age per 10 yrs"
##                                log(bmi)
##                                "log(BMI)"
## factor(mdquality.s, levels = 0:1, labels = c("N", "Y"))
##                                "MD Quality"
summary(trans)
A: IFL (N=428)   F: FOLFOX (N=691)   G: IROX (N=380) Total (N=1499) p value
Age per 10 yrs                                0.614
Mean (SD)      5.967 (1.136)    6.030 (1.163)    5.976 (1.150)    5.999 (1.152)
Range          2.700 - 8.800     1.900 - 8.800     2.600 - 8.500     1.900 - 8.800
log(BMI)                               0.811
N-Miss         9    20    4    33
Mean (SD)       3.287 (0.197)    3.286 (0.183)    3.279 (0.200)    3.285 (0.192)
Range           2.643 - 3.970     2.812 - 3.894     2.736 - 4.098     2.643 - 4.098
MD Quality                               0.694
N-Miss        55   156  41   252
N              41 (11.0%)  52 (9.7%)   31 (9.1%)   124 (9.9%)
Y             332 (89.0%) 483 (90.3%) 308 (90.9%) 1123 (90.1%)
```

Note that if we had not changed `mdquality.s` to a factor, it would have been summarized as though it were a

```
class(mockstudy$mdquality.s)
[1] "integer"
```

```
summary(tableby(arm-mdquality.s, data=mockstudy))
A: IFL (N=428)  F: FOLFOX (N=691)  G: IROX (N=380) Total (N=1499)  p value
mdquality.s      0.695
  N-Miss    55  156  41  252
  Mean (SD)    0.890 (0.313)  0.903 (0.297)  0.909 (0.289)  0.901 (0.299)
  Range      0.000 - 1.000  0.000 - 1.000  0.000 - 1.000  0.000 - 1.000
```

Another option would be to specify the test and summary statistics. In fact, if I had a set of variables co

```
summary(tableby(arm ~ chisq(mdquality.s, "Nmiss","countpct"), data=mockstudy))
A: IFL (N=428)  F: FOLFOX (N=691)  G: IROX (N=380) Total (N=1499)  p value
mdquality.s      0.694
  N-Miss    55  156  41   252
0    41 (11.0%)  52 (9.7%)   31 (9.1%)   124 (9.9%)
1   332 (89.0%) 483 (90.3%) 308 (90.9%) 1123 (90.1%)
```

```
10. Subsetting (change the ordering of the variables, delete a variable, sort by p-value, filter by p-value)
mytab <- tableby(arm ~ sex + alk.phos + age, data=mockstudy)
mytab2 <- mytab[c('age','sex','alk.phos')]
```

```
summary(mytabs2)
A: IFL (N=428) F: FOLFOX (N=691) G: IROX (N=380) Total (N=1499) p value
Age, yrs 0.614
  Mean (SD) 59.673 (11.365) 60.301 (11.632) 59.763 (11.499) 59.985 (11.519)
  Range 27.000 - 88.000 19.000 - 88.000 26.000 - 85.000 19.000 - 88.000
Gender 0.190
  Male 277 (64.7%) 411 (59.5%) 228 (60.0%) 916 (61.1%)
  Female 151 (35.3%) 280 (40.5%) 152 (40.0%) 583 (38.9%)
alk.phos 0.226
  N-Miss 69 141 56 266
  Mean (SD) 175.577 (128.608) 161.984 (121.978) 173.506 (138.564) 168.969 (128.492)
  Range 11.000 - 858.000 10.000 - 1014.000 7.000 - 982.000 7.000 - 1014.000
summary(mytabs[c('age','sex')], digits = 2)
A: IFL (N=428) F: FOLFOX (N=691) G: IROX (N=380) Total (N=1499) p value
Age, yrs 0.614
  Mean (SD) 59.67 (11.36) 60.30 (11.63) 59.76 (11.50) 59.99 (11.52)
  Range 27.00 - 88.00 19.00 - 88.00 26.00 - 85.00 19.00 - 88.00
Gender 0.190
  Male 277 (64.7%) 411 (59.5%) 228 (60.0%) 916 (61.1%)
  Female 151 (35.3%) 280 (40.5%) 152 (40.0%) 583 (38.9%)
summary(mytabs[c(3,1)], digits = 3)
A: IFL (N=428) F: FOLFOX (N=691) G: IROX (N=380) Total (N=1499) p value
Age, yrs 0.614
  Mean (SD) 59.673 (11.365) 60.301 (11.632) 59.763 (11.499) 59.985 (11.519)
  Range 27.000 - 88.000 19.000 - 88.000 26.000 - 85.000 19.000 - 88.000
Gender 0.190
  Male 277 (64.7%) 411 (59.5%) 228 (60.0%) 916 (61.1%)
  Female 151 (35.3%) 280 (40.5%) 152 (40.0%) 583 (38.9%)
summary(sort(mytabs, decreasing = TRUE))
A: IFL (N=428) F: FOLFOX (N=691) G: IROX (N=380) Total (N=1499) p value
Age, yrs 0.614
  Mean (SD) 59.673 (11.365) 60.301 (11.632) 59.763 (11.499) 59.985 (11.519)
  Range 27.000 - 88.000 19.000 - 88.000 26.000 - 85.000 19.000 - 88.000
alk.phos 0.226
  N-Miss 69 141 56 266
  Mean (SD) 175.577 (128.608) 161.984 (121.978) 173.506 (138.564) 168.969 (128.492)
  Range 11.000 - 858.000 10.000 - 1014.000 7.000 - 982.000 7.000 - 1014.000
Gender 0.190
  Male 277 (64.7%) 411 (59.5%) 228 (60.0%) 916 (61.1%)
  Female 151 (35.3%) 280 (40.5%) 152 (40.0%) 583 (38.9%)
summary(mytabs[mytabs < 0.5])
A: IFL (N=428) F: FOLFOX (N=691) G: IROX (N=380) Total (N=1499) p value
Gender 0.190
  Male 277 (64.7%) 411 (59.5%) 228 (60.0%) 916 (61.1%)
  Female 151 (35.3%) 280 (40.5%) 152 (40.0%) 583 (38.9%)
alk.phos 0.226
  N-Miss 69 141 56 266
  Mean (SD) 175.577 (128.608) 161.984 (121.978) 173.506 (138.564) 168.969 (128.492)
  Range 11.000 - 858.000 10.000 - 1014.000 7.000 - 982.000 7.000 - 1014.000
head(mytabs, 1) # can also use tail()
Tableby Object
```

```
Function Call: tableby(formula = arm ~ sex + alk.phos + age, data = mockstudy)
```

```
y variable: [1] "arm" x variables: [1] "sex"
```

```
11. Merge two tableby objects together
```

It is possible to combine two tableby objects so that they print out together.

```
## demographics
tab1 <- tableby(arm ~ sex + age, data=mockstudy,
               control=tableby.control(numeric.stats=c("Nmiss","meansd"), total=FALSE))

## lab data
tab2 <- tableby(arm ~ hgb + alk.phos, data=mockstudy,
               control=tableby.control(numeric.stats=c("Nmiss","median","q1q3"),
                                       numeric.test="kwt", total=FALSE))

names(tab1$x)
[1] "sex" "age"

names(tab2$x)
[1] "hgb" "alk.phos"

tab12 <- merge(tab1,tab2)
class(tab12)
[1] "tableby"

names(tab12$x)
[1] "sex" "age" "hgb" "alk.phos"
```

```
summary(tab12) #, pfootnote=TRUE)
A: IFL (N=428)  F: FOLFOX (N=691)  G: IROX (N=380)  p value
Gender                0.190
  Male 277 (64.7%) 411 (59.5%) 228 (60.0%)
  Female 151 (35.3%) 280 (40.5%) 152 (40.0%)
Age, yrs                0.614
  Mean (SD) 59.673 (11.365) 60.301 (11.632) 59.763 (11.499)
hgb                0.570
  N-Miss 69 141 56
  Median 12.100 12.200 12.400
  Q1, Q3 11.000, 13.450 11.100, 13.600 11.175, 13.625
alk.phos                0.104
  N-Miss 69 141 56
  Median 133.000 116.000 122.000
  Q1, Q3 89.000, 217.000 85.000, 194.750 87.750, 210.250
```

12. Add a title to the table

When creating a pdf the tables are automatically numbered and the title appears below the table. In Word and

```
t1 <- tableby(arm ~ sex + age, data=mockstudy)
summary(t1, title='Demographics')
Demographics
A: IFL (N=428)  F: FOLFOX (N=691)  G: IROX (N=380)  Total (N=1499)  p value
Gender                0.190
  Male 277 (64.7%) 411 (59.5%) 228 (60.0%) 916 (61.1%)
  Female 151 (35.3%) 280 (40.5%) 152 (40.0%) 583 (38.9%)
Age, yrs                0.614
  Mean (SD) 59.673 (11.365) 60.301 (11.632) 59.763 (11.499) 59.985 (11.519)
  Range 27.000 - 88.000 19.000 - 88.000 26.000 - 85.000 19.000 - 88.000
```

13. Modify how missing values are displayed

Depending on the report you are writing you have the following options:

Show how many subjects have each variable

Show how many subjects are missing each variable

Show how many subjects are missing each variable only if there are any missing values

Don't indicate missing values at all

```
## look at how many missing values there are for each variable
```

```
apply(is.na(mockstudy),2,sum)
```

```
##          case          age          arm          sex          race          fu.time          fu.stat          ps
##           0           0           0           0           7           0           0           266
##          hgb          bmi          alk.phos          ast mdquality.s          age.ord          age.ordnew          dtentry
##          266          33          266          266          252          0           1           5
```

```
## Show how many subjects have each variable (non-missing)
```

```
summary(tableby(sex ~ ast + age, data=mockstudy,
                 control=tableby.control(numeric.stats=c("N","median"), total=FALSE)))
```

```
Male (N=916)   Female (N=583)   p value
```

```
ast           0.921
  N           754 479
  Median      27.000 27.000
Age, yrs      0.048
  N           916 583
  Median      61.000 60.000
```

```
## Always list the number of missing values
```

```
summary(tableby(sex ~ ast + age, data=mockstudy,
                 control=tableby.control(numeric.stats=c("Nmiss2","median"), total=FALSE)))
```

```
Male (N=916)   Female (N=583)   p value
```

```
ast           0.921
  N-Miss      162 104
  Median      27.000 27.000
Age, yrs      0.048
  N-Miss      0    0
  Median      61.000 60.000
```

```
## Only show the missing values if there are some (default)
```

```
summary(tableby(sex ~ ast + age, data=mockstudy,
                 control=tableby.control(numeric.stats=c("Nmiss","mean"),total=FALSE)))
```

```
Male (N=916)   Female (N=583)   p value
```

```
ast           0.921
  N-Miss      162 104
  mean 35.9    36
Age, yrs      0.048
  mean 60.5    59.2
```

```
## Don't show N at all
```

```
summary(tableby(sex ~ ast + age, data=mockstudy,
                 control=tableby.control(numeric.stats=c("mean"),total=FALSE)))
```

```
Male (N=916)   Female (N=583)   p value
```

```
ast           0.921
  mean 35.9    36
Age, yrs      0.048
  mean 60.5    59.2
```

One might also consider the use of `includeNA()` to include NAs in the counts and percents for categorical variables

```
mockstudy$ps.cat <- factor(mockstudy$ps)
```

```
attr(mockstudy$ps.cat, "label") <- "ps"
```

```
summary(tableby(sex ~ includeNA(ps.cat), data = mockstudy, cat.stats = "countpct"))
```

```
Male (N=916)   Female (N=583)   Total (N=1499)   p value
```

```
ps           0.354
  0          391 (42.7%) 244 (41.9%) 635 (42.4%)
  1          329 (35.9%) 202 (34.6%) 531 (35.4%)
  2           34 (3.7%)   33 (5.7%)   67 (4.5%)
```

(Missing) 162 (17.7%) 104 (17.8%) 266 (17.7%)

14. Modify the number of digits used

Within `tableby`.control function there are 4 options for controlling the number of significant digits shown.

`digits`: controls the number of digits after the decimal place for continuous values

`digits.count`: controls the number of digits after the decimal point for counts

`digits.pct`: controls the number of digits after the decimal point for percents

`digits.p`: controls the number of digits after the decimal point for p-values

```
summary(tableby(arm ~ sex + age + fu.time, data=mockstudy), digits=4, digits.p=2, digits.pct=1)
```

A: IFL (N=428) F: FOLFOX (N=691) G: IROX (N=380) Total (N=1499) p value

Gender 0.19

Male 277 (64.7%) 411 (59.5%) 228 (60.0%) 916 (61.1%)

Female 151 (35.3%) 280 (40.5%) 152 (40.0%) 583 (38.9%)

Age, yrs 0.61

Mean (SD) 59.6729 (11.3645) 60.3010 (11.6323) 59.7632 (11.4993) 59.9853 (11.5188)

Range 27.0000 - 88.0000 19.0000 - 88.0000 26.0000 - 85.0000 19.0000 - 88.0000

fu.time < 0.01

Mean (SD) 553.5841 (419.6065) 731.2460 (487.7443) 607.2421 (435.5092) 649.0841 (462.5109)

Range 9.0000 - 2170.0000 0.0000 - 2472.0000 17.0000 - 2118.0000 0.0000 - 2472.0000

With the exception of `digits.p`, all of these can be specified on a per-variable basis using the `in-formula`

```
summary(tableby(arm ~ chisq(sex, digits.pct=1) + anova(age, digits=4) +
               anova(fu.time, digits = 1), data=mockstudy))
```

A: IFL (N=428) F: FOLFOX (N=691) G: IROX (N=380) Total (N=1499) p value

Gender 0.190

Male 277 (64.7%) 411 (59.5%) 228 (60.0%) 916 (61.1%)

Female 151 (35.3%) 280 (40.5%) 152 (40.0%) 583 (38.9%)

Age, yrs 0.614

Mean (SD) 59.6729 (11.3645) 60.3010 (11.6323) 59.7632 (11.4993) 59.9853 (11.5188)

Range 27.0000 - 88.0000 19.0000 - 88.0000 26.0000 - 85.0000 19.0000 - 88.0000

fu.time < 0.001

Mean (SD) 553.6 (419.6) 731.2 (487.7) 607.2 (435.5) 649.1 (462.5)

Range 9.0 - 2170.0 0.0 - 2472.0 17.0 - 2118.0 0.0 - 2472.0

15. Create a user-defined summary statistic

For purposes of this example, the code below creates a trimmed mean function (`trims 10%`) and use that to su

```
myfunc <- function(x, weights=rep(1,length(x)), ...){
  mean(x, trim=.1, ...)
}
```

```
summary(tableby(sex ~ hgb, data=mockstudy,
               control=tableby.control(numeric.stats=c("Nmiss","myfunc"), numeric.test="kwt",
               stats.labels=list(Nmiss='Missing values', myfunc="Trimmed Mean, 10%"))))
```

Male (N=916) Female (N=583) Total (N=1499) p value

hgb < 0.001

Missing values 162 104 266

Trimmed Mean, 10% 12.6 11.9 NA

16. Use case-weights for creating summary statistics

When comparing groups, they are often unbalanced when it comes to nuisances such as age and sex. The `tableb`

```
##create fake group that is not balanced by age/sex
```

```
set.seed(200)
```

```
mockstudy$fake_arm <- ifelse(mockstudy$age>60 & mockstudy$sex=='Female',sample(c('A','B'),replace=T, prob=c
```

```

sample(c('A','B'),replace=T, prob=c(.8,.4)))

mockstudy$agegp <- cut(mockstudy$age, breaks=c(18,50,60,70,90), right=FALSE)

## create weights based on agegp and sex distribution
tab1 <- with(mockstudy, table(agegp, sex))
tab2 <- with(mockstudy, table(agegp, sex, fake_arm))
tab2
## , , fake_arm = A
##
##           sex
## agegp      Male Female
## [18,50)      73      62
## [50,60)     128      94
## [60,70)     139       7
## [70,90)     102       0
##
## , , fake_arm = B
##
##           sex
## agegp      Male Female
## [18,50)      79      48
## [50,60)     130      84
## [60,70)     156     166
## [70,90)     109     122
gpwts <- rep(tab1, length(unique(mockstudy$fake_arm)))/tab2
gpwts[gpwts>50] <- 30

## apply weights to subjects
index <- with(mockstudy, cbind(as.numeric(agegp), as.numeric(sex), as.numeric(as.factor(fake_arm))))
mockstudy$wts <- gpwts[index]

## show weights by treatment arm group
tapply(mockstudy$wts, mockstudy$fake_arm, summary)
## $A
##   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  1.774  1.894   2.069   2.276  2.082  24.714
##
## $B
##   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  1.000  1.042   1.924   1.677  1.985   2.292
orig <- tableby(fake_arm ~ age + sex + Surv(fu.time/365, fu.stat), data=mockstudy, test=FALSE)
summary(orig, title='No Case Weights used')
No Case Weights used
A (N=605)   B (N=894)   Total (N=1499)
Age, yrs
  Mean (SD)   57.413 (11.618) 61.726 (11.125) 59.985 (11.519)
  Range      22.000 - 85.000 19.000 - 88.000 19.000 - 88.000
Gender
  Male 442 (73.1%) 474 (53.0%) 916 (61.1%)
  Female 163 (26.9%) 420 (47.0%) 583 (38.9%)
Surv(fu.time/365, fu.stat)
  Events   554 802 1356
  Median Survival 1.504   1.493   1.496
tab1 <- tableby(fake_arm ~ age + sex + Surv(fu.time/365, fu.stat), data=mockstudy, weights=wts)
summary(tab1, title='Case Weights used')
Case Weights used

```



```

A (N=605)   B (N=894)   Total (N=1499)
Age, yrs
  Mean (SD)   58.009 (10.925) 60.151 (11.428) 59.126 (11.235)
  Range      22.000 - 85.000 19.000 - 88.000 19.000 - 88.000
Gender
  Male 916 (66.5%) 916 (61.1%) 1832 (63.7%)
  Female 461 (33.5%) 583 (38.9%) 1044 (36.3%)
Surv(fu.time/365, fu.stat)
  Events 1252 1348 2599
  Median Survival 1.534 1.496 1.532

```

17. Create your own p-value and add it to the table

When using weighted summary statistics, it is often desirable to then show a p-value from a model that corrects for confounding.

To add the p-value you simply need to create a data frame and use the function `modpval.tableby`. The first 2

```

mypval <- data.frame(variable=c('age','sex','Surv(fu.time/365, fu.stat)'),
                     adj.pvalue=c(.953,.811,.01),
                     method=c('Age/Sex adjusted model results'))
tab2 <- modpval.tableby(tab1, mypval, use.pname=TRUE)
summary(tab2, title='Case Weights used, p-values added') #, pfootnote=TRUE)
Case Weights used, p-values added
A (N=605)   B (N=894)   Total (N=1499)   adj.pvalue
Age, yrs                0.953
  Mean (SD)   58.009 (10.925) 60.151 (11.428) 59.126 (11.235)
  Range      22.000 - 85.000 19.000 - 88.000 19.000 - 88.000
Gender                0.811
  Male 916 (66.5%) 916 (61.1%) 1832 (63.7%)
  Female 461 (33.5%) 583 (38.9%) 1044 (36.3%)
Surv(fu.time/365, fu.stat) 0.010
  Events 1252 1348 2599
  Median Survival 1.534 1.496 1.532

```

18. For two-level categorical variables or one-line numeric variables, simplify the output.

If the `cat.simplify` option is set to `TRUE`, then only the second level of two-level categorical variables is shown.

```

table2 <- tableby(arm~sex + factor(mdquality.s), data=mockstudy, cat.simplify=TRUE)
summary(table2, labelTranslations=c(sex="Female", "factor(mdquality.s)"="MD Quality"))
A: IFL (N=428) F: FOLFOX (N=691) G: IROX (N=380) Total (N=1499) p value
Female 151 (35.3%) 280 (40.5%) 152 (40.0%) 583 (38.9%) 0.190
MD Quality 0.694
  N-Miss 55 156 41 252
  0 41 (11.0%) 52 (9.7%) 31 (9.1%) 124 (9.9%)
  1 332 (89.0%) 483 (90.3%) 308 (90.9%) 1123 (90.1%)

```

Similarly, if `numeric.simplify` is set to `TRUE`, then any numerics which only have one row of summary statistics are simplified.

```

summary(tableby(arm ~ age + ast, data = mockstudy,
               numeric.simplify=TRUE, numeric.stats=c("Nmiss", "meansd")))
A: IFL (N=428) F: FOLFOX (N=691) G: IROX (N=380) Total (N=1499) p value
Age, yrs 59.673 (11.365) 60.301 (11.632) 59.763 (11.499) 59.985 (11.519) 0.614
ast 0.507
  N-Miss 69 141 56 266
  Mean (SD) 37.292 (28.036) 35.202 (26.659) 35.670 (25.807) 35.933 (26.843)

```

The in-formula functions to change which tests are run can also be used to specify these options for each variable.

```

summary(tableby(arm ~ anova(age, "meansd", numeric.simplify=TRUE) +
               chisq(sex, cat.simplify=TRUE), data = mockstudy))

```

```
##
##
```

```
## |           | A: IFL (N=428) | F: FOLFOX (N=691) | G: IROX (N=380) | Total (N=1499) | p value|
## |-----|:-----|:-----|:-----|:-----|:-----|
## |**Age, yrs** | 59.673 (11.365) | 60.301 (11.632) | 59.763 (11.499) | 59.985 (11.519) | 0.614|
## |**Gender**   | 151 (35.3%)   | 280 (40.5%)    | 152 (40.0%)    | 583 (38.9%)    | 0.190|
```

19. Use tableby within an Sweave document

For those users who wish to create tables within an Sweave document, the following code seems to work.

```
\documentclass{article}

\usepackage{longtable}
\usepackage{pdfpages}

\begin{document}

\section{Read in Data}
<<echo=TRUE>>=
require(arsenal)
require(knitr)
require(rmarkdown)
data(mockstudy)

tab1 <- tableby(arm~sex+age, data=mockstudy)
@

\section{Convert Summary.Tableby to LaTeX}
<<echo=TRUE, results='hide', message=FALSE>>=
capture.output(summary(tab1), file="Test.md")

## Convert R Markdown Table to LaTeX
render("Test.md", pdf_document(keep_tex=TRUE))
@

\includepdf{Test.pdf}
```

\end{document}

20. Export tableby object to a .CSV file

When looking at multiple variables it is sometimes useful to export the results to a csv file. The as.data.

```
tab1 <- tableby(arm~sex+age, data=mockstudy)
as.data.frame(tab1)
##   variable      term      label variable.type      A: IFL      F: FOLFOX
## 1      sex      sex      Gender      categorical      277.00000, 64.71963 411.00000, 59.47902
## 2      sex countpct      Male      categorical      151.00000, 35.28037 280.00000, 40.52098
## 3      sex countpct      Female      categorical      59.67290, 11.36454 60.30101, 11.63225
## 4      age      age      Age, yrs      numeric      27, 88      19, 88
## 5      age meansd Mean (SD)      numeric      27, 88      19, 88
## 6      age      range      Range      numeric      27, 88      19, 88
##   G: IROX      Total      test      p.value
## 1      Pearson's Chi-squared test 0.1904388
## 2      228, 60 916.0000, 61.1074 Pearson's Chi-squared test 0.1904388
## 3      152, 40 583.0000, 38.8926 Pearson's Chi-squared test 0.1904388
## 4      Linear Model ANOVA 0.6143859
## 5 59.76316, 11.49930 59.98532, 11.51877 Linear Model ANOVA 0.6143859
## 6      26, 85      19, 88      Linear Model ANOVA 0.6143859
# write.csv(tmp, '/my/path/here/mymodel.csv')
21. Write tableby object to a separate Word or HTML file
## write to an HTML document
```

```
tab1 <- tableby(arm ~ sex + age, data=mockstudy)
write2html(tab1, "~/trash.html")
```

```
## write to a Word document
write2word(tab1, "~/trash.doc", title="My table in Word")
```

22. Use `tableby` in R Shiny

The easiest way to output a `tableby()` object in an R Shiny app is to use the `tableOutput()` UI in combination with the `renderTableby()` server function.

```
# A standalone shiny app
```

```
library(shiny)
library(arsenal)
data(mockstudy)
```

```
shinyApp(  
  ui = fluidPage(tableOutput("table")),  
  server = function(input, output) {  
    output$table <- renderTable({  
      as.data.frame(summary(tableby(sex ~ age, data = mockstudy), text = "html"))  
    }, sanitize.text.function = function(x) x)  
  }  
)
```

This can be especially powerful if you feed the selections from a `selectInput(multiple = TRUE)` into `formuli`

23. Use `tableby` in `bookdown`

Since the backbone of `tableby()` is `knitr::kable()`, tables still render well in bookdown. However, `print.summary`

```
summary(tableby(sex ~ age, data = mockstudy), title="(\\#tab:mytableby) Caption here")
```

24. Adjust table by for multiple p-values

The `padjust()` function is a new S3 generic piggybacking off of `p.adjust()`. It works on both `tableby` and `sum`

```
tab <- summary(tableby(sex ~ age + fu.time + bmi + mdquality.s, data = mockstudy))
```

tab

##

##

##		Male (N=916)	Female (N=583)	Total (N=1499)	p value
##	***Age, yrs***				0.048
##	Mean (SD)	60.455 (11.369)	59.247 (11.722)	59.985 (11.519)	
##	Range	19.000 - 88.000	22.000 - 88.000	19.000 - 88.000	
##	***fu.time***				0.978
##	Mean (SD)	649.345 (454.332)	648.674 (475.472)	649.084 (462.511)	
##	Range	0.000 - 2472.000	9.000 - 2441.000	0.000 - 2472.000	
##	***Body Mass Index (kg/m ²)***				0.012
##	N-Miss	22	11	33	
##	Mean (SD)	27.491 (5.030)	26.760 (5.984)	27.206 (5.432)	
##	Range	14.053 - 60.243	15.430 - 53.008	14.053 - 60.243	
##	***mdquality.s***				0.827
##	N-Miss	153	99	252	
##	Mean (SD)	0.899 (0.301)	0.903 (0.296)	0.901 (0.299)	
##	Range	0.000 - 1.000	0.000 - 1.000	0.000 - 1.000	

padjust(tab, method = "bonferroni")

##

##

##		Male (N=916)	Female (N=583)	Total (N=1499)	p value
##	:-----:	:-----:	:-----:	:-----:	:-----:
##	**Age, yrs**				0.191
##	Mean (SD)	60.455 (11.369)	59.247 (11.722)	59.985 (11.519)	

notest: Don't perform a test.

tableby.control settings

A quick way to see what arguments are possible to utilize in a function is to use the args() command. Setti

```
args(tableby.control)
## function (test = TRUE, total = TRUE, test.pname = NULL, cat.simplify = FALSE,
##     numeric.simplify = FALSE, numeric.test = "anova", cat.test = "chisq",
##     ordered.test = "trend", surv.test = "logrank", date.test = "kwt",
##     numeric.stats = c("Nmiss", "meansd", "range"), cat.stats = c("Nmiss",
##     "countpct"), ordered.stats = c("Nmiss", "countpct"),
##     surv.stats = c("Nevents", "medSurv"), date.stats = c("Nmiss",
##     "median", "range"), stats.labels = list(Nmiss = "N-Miss",
##     Nmiss2 = "N-Miss", meansd = "Mean (SD)", medianrange = "Median (Range)",
##     median = "Median", medianq1q3 = "Median (Q1, Q3)", q1q3 = "Q1, Q3",
##     iqr = "IQR", range = "Range", countpct = "Count (Pct)",
##     Nevents = "Events", medSurv = "Median Survival", medTime = "Median Follow-Up"),
##     digits = 3L, digits.count = 0L, digits.pct = 1L, digits.p = 3L,
##     format.p = TRUE, conf.level = 0.95, chisq.correct = FALSE,
##     simulate.p.value = FALSE, B = 2000, ...)
```

```
## NULL
```

summary.tableby settings

The summary.tableby function has options that modify how the table appears (such as adding a title or modifi

```
args(arsenal::summary.tableby)
## function (object, ..., labelTranslations = NULL, text = FALSE,
##     title = NULL, pfootnote = FALSE, term.name = "")
## NULL
```

The write2 function

<https://cran.r-project.org/web/packages/arsenal/vignettes/write2.html>

The write2 function

Ethan Heinzen

09 November, 2018

Introduction

A note on piping

Examples Using arsenal Objects

tableby

modelsum

freqlist

compare

Examples Using Other Objects

knitr::kable()

xtable::xtable()

pander::pander_return()

Output Multiple Tables to One Document

Output Other Objects Monospaced (as if in a terminal)

Add a YAML Header to the Output

FAQs

How do I suppress the note about my document getting rendered?

How do I look at the temporary .md file?

How do I prevent my document from being rendered?

How do I output headers, raw HTML/LaTeX, paragraphs, etc.?

How do I tweak the default format from `write2word()`, `write2html()`, or `write2pdf()`?

How do I output to a file format other than word, HTML, and PDF?

How do I avoid prefixes on my table captions in PDF?

How do I output multiple tables with different titles?

Introduction

The `write2*()` functions were designed as an alternative to SAS's ODS procedure for users who want to save R

There are three shortcut functions for the most common output types: HTML, PDF, and Word. Each of these thr

The two most important things to recognize with `write2()` are the following:

Which function is being used to output the object. Sometimes the `write2` functions use `summary()`, while othe

How the ... arguments are passed. To change the options for the summary-like or print-like function, you ca

A note on piping

arsenal is piping-compatible!

The `write2*()` functions are probably the most useful place to take advantage of the `magrittr` package's pipi

This vignette will sprinkle the foward pipe (`%>%`) throughout as a hint at the power and flexibility of arse

Examples Using arsenal Objects

```
library(arsenal)
```

```
library(magrittr)
```

```
data(mockstudy)
```

```
tmpdir <- tempdir()
```

```
tableby
```

For `tableby` objects, the output function in `write2()` is `summary()`. For `summary.tableby` objects, the output

```
mylabels <- list(sex = "SEX", age = "Age, yrs")
```

```
tab1 <- tableby(arm ~ sex + age, data=mockstudy)
```

```
write2html(
```

```
  tab1, paste0(tmpdir, "/test.tableby.html"), quiet = TRUE,
```

```
  title = "My test table",          # passed to summary.tableby
```

```
  labelTranslations = mylabels,    # passed to summary.tableby
```

```
  total = FALSE                    # passed to summary.tableby
```

```
)
```

```
modelsum
```

For `modelsum` objects, the output function in `write2()` is `summary()`. For `summary.modelsum` objects, the output

```
tab2 <- modelsum(alk.phos ~ arm + ps + hgb, adjust= ~ age + sex, family = "gaussian", data = mockstudy)
```

```
write2pdf(
```

```
  tab2, paste0(tmpdir, "/test.modelsum.pdf"), quiet = TRUE,
```

```
  title = "My test table", # passed to summary.modelsum
```

```
  show.intercept = FALSE,  # passed to summary.modelsum
```

```
  digits = 5               # passed to summary.modelsum
```

```
)
```

```
freqlist
```

For `freqlist` objects, the output function in `write2()` is `summary()`. For `summary.freqlist` objects, the output

```
mockstudy[, c("arm", "sex", "mdquality.s")] %>%
```

```

table(useNA = "ifany") %>%
freqlist(groupBy = c("arm", "sex")) %>%
write2word(
  paste0(tmpdir, "/test.freqlist.doc"), quiet = TRUE,
  single = FALSE,          # passed to summary.freqlist
  title = "My cool title" # passed to summary.freqlist
)

```

compare

For compare.data.frame objects, the output function in write2() is summary(). For summary.compare.data.frame

Examples Using Other Objects

knitr::kable()

For objects resulting from a call to kable(), the output function in write2() is print(). There aren't any

```

mockstudy %>%
  head() %>%
  knitr::kable() %>%
  write2html(paste0(tmpdir, "/test.kable.html"), quiet = TRUE)

```

xtable::xtable()

For xtable objects, the output function in write2() is print(). For available arguments, see the help pages

```

mockstudy %>%
  head() %>%
  xtable::xtable(caption = "My xtable") %>%
  write2pdf(
    paste0(tmpdir, "/test.xtable.pdf"), quiet = TRUE,
    comment = FALSE, # passed to print.xtable to turn off the default message about xtable version
    include.rownames = FALSE, # passed to print.xtable
    caption.placement = "top" # passed to print.xtable
  )

```

To make an HTML document, use the print.xtable() option type = "html".

```

mockstudy %>%
  head() %>%
  xtable::xtable(caption = "My xtable") %>%
  write2html(
    paste0(tmpdir, "/test.xtable.html"), quiet = TRUE,
    type = "html",          # passed to print.xtable
    comment = FALSE, # passed to print.xtable to turn off the default message about xtable version
    include.rownames = FALSE, # passed to print.xtable
    caption.placement = "top" # passed to print.xtable
  )

```

User beware! xtable() is not compatible with write2word().

pander::pander_return()

Pander is a little bit more tricky. Since pander::pander() doesn't return an object, the user should instead

```
write2word(pander::pander_return(head(mockstudy)), file = paste0(tmpdir, "/test.pander.doc"), quiet = TRUE)
```

Output Multiple Tables to One Document

To output multiple tables into a document, simply make a list of them and call the same function as before.

```

mylist <- list(
  tableby(sex ~ age, data = mockstudy),
  freqlist(table(mockstudy[, c("sex", "arm")])),
  knitr::kable(head(mockstudy))
)

```

```
write2pdf(mylist, paste0(tmpdir, "/test.mylist.pdf"), quiet = TRUE)
```

One neat side-effect of this function is that you can output text and headers, etc. The possibilities are e

```
mylist2 <- list(
  "# Header 1",
  "This is a small paragraph introducing tableby.",
  tableby(sex ~ age, data = mockstudy),
  "<hr>",
  "# Header 2",
  "<font color='red'>I can change color of my text!</font>"
)
write2html(mylist2, paste0(tmpdir, "/test.mylist2.html"), quiet = TRUE)
In fact, you can even recurse on the lists!
```

```
write2pdf(list(mylist2, mylist), paste0(tmpdir, "/test.mylists.pdf"), quiet = TRUE)
```

Output Other Objects Monospaced (as if in a terminal)

It may be useful at times to write output that would normally be copied from the terminal. The default meth

```
lm(age ~ sex, data = mockstudy) %>%
  summary() %>%
  write2pdf(paste0(tmpdir, "/test.lm.pdf"), quiet = TRUE)
```

The verbatim() function is another option to explicitly alert write2() to do this. This becomes particularl

For example, suppose you wanted to just print a tableby object (as if it were to print in the terminal):

```
tab4 <- tableby(arm ~ sex + age, data=mockstudy)
write2html(verbatim(tab4), paste0(tmpdir, "/test.print.tableby.html"), quiet = TRUE)
Or suppose you wanted to print a character vector (as if it were to print in the terminal):
```

```
chr <- paste0("MyVector", 1:10)
write2pdf(verbatim(chr), paste0(tmpdir, "/test.character.pdf"), quiet = TRUE)
Add a YAML Header to the Output
You can add a YAML header to write2() output using the yaml() function.
```

```
mylist3 <- list(
  yaml(title = "Test YAML Title", author = "My cool author name"),
  "# Header 1",
  "This is a small paragraph introducing tableby.",
  tableby(sex ~ age, data = mockstudy)
)
```

```
write2html(mylist3, paste0(tmpdir, "/test.yaml.html"), quiet = TRUE)
```

In fact, all detected YAML pieces will be moved as the first output, so that the above code chunk gives the

```
mylist4 <- list(
  "# Header 1",
  "This is a small paragraph introducing tableby.",
  yaml(title = "Test YAML Title"),
  tableby(sex ~ age, data = mockstudy),
  yaml(author = "My cool author name")
)
write2html(mylist3, paste0(tmpdir, "/test.yaml2.html"), quiet = TRUE)
```

FAQs

How do I suppress the note about my document getting rendered?

This is easily accomplished by using the argument quiet = TRUE (passed to the rmarkdown::render() function)

```
write2html(
  knitr::kable(head(mockstudy)), paste0(tmpdir, "/test.kable.quiet.html"),
```



```

  quiet = TRUE # passed to rmarkdown::render
)
How do I look at the temporary .md file?
This is easily accomplished by using the option keep.md = TRUE.

write2html(
  knitr::kable(head(mockstudy)), paste0(tmpdir, "/test.kable.keep.md.html"),
  quiet = TRUE, # passed to rmarkdown::render
  keep.md = TRUE
)
How do I prevent my document from being rendered?
This is easily accomplished by using the option render. = FALSE. Note that this will then default to keep.md = TRUE.

write2html(
  knitr::kable(head(mockstudy)), paste0(tmpdir, "/test.kable.dont.render.html"),
  render. = FALSE
)
How do I output headers, raw HTML/LaTeX, paragraphs, etc.?
One can simply abuse the list S3 method for write2(!)

mylist2 <- list(
  "# Header 1",
  "This is a small paragraph introducing tableby.",
  tableby(sex ~ age, data = mockstudy),
  "<hr>",
  "# Header 2",
  "<font color='red'>I can change color of my text!</font>"
)
write2html(mylist2, paste0(tmpdir, "/test.mylist2.html"), quiet = TRUE)
How do I tweak the default format from write2word(), write2html(), or write2pdf()?
You can pass arguments to the format functions used behind the scenes.

write2html(
  knitr::kable(head(mockstudy)), paste0(tmpdir, "/test.kable.theme.html"),
  quiet = TRUE, # passed to rmarkdown::render
  theme = "yeti" # passed to rmarkdown::html_document
)
See the help pages for rmarkdown::word_document(), rmarkdown::html_document(), and rmarkdown::pdf_document().

How do I output to a file format other than word, HTML, and PDF?
This can be done using the generic write2() function. The last argument in the function can be another format.

write2(
  knitr::kable(head(mockstudy[, 1:4])), paste0(tmpdir, "/test.kable.rtf"),
  quiet = TRUE, # passed to rmarkdown::render
  output_format = rmarkdown::rtf_document
)
How do I avoid prefixes on my table captions in PDF?
You can do this pretty easily with the yaml() function:

mylist5 <- list(
  yaml("header-includes" = list("\\usepackage[labelformat=empty]{caption}")),
  "# Header 1",
  "This is a small paragraph introducing tableby.",
  tableby(sex ~ age, data = mockstudy)
)
write2pdf(mylist5, paste0(tmpdir, "/test.noprefixes.pdf"), title = "My tableby")

```

How do I output multiple tables with different titles?

There are now `write2()` methods for the summary objects of arsenal functions. This allows you to specify a t

```
mylist6 <- list(
  summary(tableby(sex ~ age, data = mockstudy), title = "A Title for tableby"),
  summary(modelsum(age ~ sex, data = mockstudy), title = "A Title for modelsum"),
  summary(freqlist(~ sex, data = mockstudy), title = "A Title for freqlist")
)
write2pdf(mylist6, paste0(tmpdir, "/test.multiple.titles.pdf"))
```

```
author: "Kristian Larsen"
```

```
output:
```

```
  flexdashboard::flex_dashboard:
    orientation: rows
    vertical_layout: scroll
```

```
from: https://datascienceplus.com/automated-dashboard-visualizations-with-deviation-in-r/?fbclid=IwAR2JcAMQ
```

20.2 Row

20.2.1 Chart A: Diverging Barcharts

20.2.2 Chart B: Diverging Lollipop Chart

20.3 Row

20.3.1 Cart C: Diverging Dot Plot

```
print(paste0("Git Update Started at: ", Sys.time()))
CommitMessage <- paste("updated on: ", Sys.time(), sep = "")
wd <- "~/serdarbalci"
setorigin <- "git remote set-url origin git@github.com:sbalci/MyJournalWatch.git \n"
gitCommand <- paste("cd ", wd, "\n git add . \n git commit --message '", CommitMessage, "' \n", setorigin,
system(command = paste(gitCommand, "\n") , intern = TRUE, wait = TRUE)
Sys.sleep(5)
print(paste0("Git Update Ended at: ", Sys.time()))
```

20.4 Describe results of analysis

Copy/paste t-tests Directly to Manuscripts: https://neuropsychology.github.io/psycho.R//2018/06/19/analyze_tttest.html

<https://github.com/neuropsychology/psycho.R>

Chapter 21

citation

My next citation is here¹.

“r dimensionBadge”

“r altmetricBadge”

¹“r cit_25783680”

Chapter 22

BBC Visual and Data Journalism cookbook for R graphics

<https://bbc.github.io/rcookbook/>

A brief introduction to bibliometrix

<https://cran.r-project.org/web/packages/bibliometrix/vignettes/bibliometrix-vignette.html>

Bibliographic Network Visualization for Academic Literature Reviews

<http://www.mburnamfink.com/blog/bibliographic-network-visualization-for-academic-literature-reviews>

<https://embed.kumu.io/0b991b02bb20975fde904f4bf7433333#jpsp-top-50?s=%23doi-101037-0022-35147451252>

More Than Words? Computer-Aided Text Analysis in Organizational Behavior and Psychology Research

<https://www.annualreviews.org/doi/10.1146/annurev-orgpsych-032117-104622>

<https://www.kumu.io/nicholasjkelley/jpsp-top-50>

Chapter 23

knitcitations

<https://github.com/cboettig/knitcitations>

```
citation "r citep("10.1890/11-0011.1")" in text
```

```
citation "r citet("10.1098/rspb.2013.1372")" in text
```

```
write.bibtex(file="references.bib")
```


Chapter 24

rcrossref

<https://github.com/ropensci/rcrossref>

Chapter 25

rorcid tutorial

https://ropensci.org/tutorials/rorcid_tutorial/

Chapter 26

rentrez tutorial

https://ropensci.org/tutorials/rentrez_tutorial/

Chapter 27

WebSciCorpus

<https://www.clarehooper.net/WebSciCorpus/>

Chapter 28

WEB OF SCIENCE (WOS) CORPUS | PARSING SCRIPT

<https://docs.cortext.net/question/web-of-science-wos-corpus-parsing-script-2/>

Chapter 29

T-LAB PLUS 2019

https://tlab.it/en/allegati/help_en_online/mmappe2.htm

Chapter 30

Tools for bibliometric analyses

<https://ju.se/library/research--teaching-support/bibliometrics/tools-for-bibliometric-analyses.html>

Chapter 31

evidencepartners

<https://www.evidencepartners.com/>

Chapter 32

R script for creating a cross-citation network

https://www.researchgate.net/publication/327790285_R_script_for_creating_a_cross-citation_network

Repository: <https://github.com/arsiders/citation-network>

```
# RCitation - Quick Citation Network
# Fall 2018
# A.R. Siders (siders@alumni.stanford.edu)

# Creates a network of the citations among a set of academic papers.
# Rationale: If full title of Article 2 is present in text of Article 1, Article 1 cites Article 2.
# NOTE: Will only work in fields where full, unabbreviated titles are used in reference/bibliography citations.
# NOTE: Will have high error rate if titles are very short or comprised of common words (e.g., paper "Vulnerability").
# NOTE: Error rate may be reduced by using only reference sections of the articles of interest, rather than full articles.

# ==> FIVE STEPS TO CITATION NETWORK

# STEP 1. FORMAT INPUT
# a. Papers: Folder of papers in txt format (UTF-8) organized *in SAME ORDER* as Titles
# b. Titles: Column of paper titles in csv spreadsheet (Column #1) *in SAME ORDER* as documents in Papers folder
# Recommend naming all texts in Papers folder using author last name listed alphabetically. Organize Titles similarly.

# STEP 2. PREP
# set working directory
setwd("C:\\[name of working space]") # make sure \\ not / in name
setwd("C:/Users/User/OneDrive/Adaptive Capacity Text Mining/Citation Network Test/CitationNetwork Test Data")
# load packages
install.packages(c("tm","plyr"))
library(tm)
library(plyr)

# STEP 3. LOAD INPUTS
# a. Papers
papers<-Corpus(DirSource("[name of folder where papers located]"))
papers<-Corpus(DirSource("Papers"))
# b. Titles
titletable<-read.csv("[name of titles file].csv") #make sure column has a header
```

```

titletable<-read.csv("TestTitles.csv")
titles<-as.vector(titletable[,1])
# load functions at bottom of this script (below Step 5)

length(papers)
length(titles)

# STEP 4. RUN FUNCTION

CitationNetwork<-CreateCitationNetwork(papers,titles)
# add date
currentDate <- Sys.Date()
csvFileName <- paste("CitationEdges",currentDate,".csv",sep="")
# save results
write.csv(CitationNetwork, file=csvFileName)

# STEP 5. VISUALIZE NETWORK

# Install Gephi or other network visualization software and load CitationEdges.csv
# Load list of titles or other spreadsheet as nodes to visualize network
# Gephi available at https://gephi.org/

# ==> FUNCTIONS TO LOAD

CreateCitationNetwork<-function(papers,titles){
  # prep papers corpus
  papers<-tm_map(papers, content_transformer(tolower))
  papers<-tm_map(papers, removePunctuation)
  papers<-tm_map(papers, removeNumbers)
  papers<-tm_map(papers, stripWhitespace)
  # prep titles
  titles<-removePunctuation(titles)
  titles<-stripWhitespace(titles)
  titles<-tolower(titles)
  # create citation true/false matrix
  Cites.TF<-CiteMatrix(titles, papers)
  # format matrix into edges file
  CitationEdges<-EdgesFormat(Cites.TF, titles)
  return(CitationEdges)
}

# format true/false matrix into edges file
EdgesFormat<-function(Cites.TF, titles){
  #create an empty object to put information in
  edges<-data.frame(matrix(NA), nrow=NA, ncol=NA)
  colnames(edges)<- c("Source","Target","Weight")
  for (i in 1:length(Cites.TF)){
    #for each document, run through all titles accross columns
    for (j in 1:ncol(Cites.TF)){
      # for each title, see if document [row] cited that title [column]
      if (Cites.TF[i,j]==TRUE){ #if document is cited
        temp<-data.frame(matrix(NA), nrow=NA, ncol=NA)
        colnames(temp)<- c("Source","Target","Weight")
        # first column <- document doing the citing
        temp[1,1]<-titles[i]

```

```

        # second column <- document being cited
        temp[1,2]<-titles[j]
        # third column the yes/no [weight]
        temp[1,3]<-1
        temp[1,4]<-"Directed"
        edges<-rbind(edges,temp)
    }
}
}
return(edges[-1,]) #-1 removes initial row of null values
}

# Citation true/false matrix
CiteMatrix<-function(search.vector, Ref.corpus){
  # Creates a csv matrix with True/False for citation patterns
  citations<-data.frame(matrix(NA, nrow = length(Ref.corpus), ncol=length(search.vector)))
  #Columns are the document being cited
  colnames(citations)<-search.vector
  #Rows are the document doing the citing
  rownames(citations)<-search.vector
  for (i in 1:length(search.vector)){
    searchi<-search.vector[i]
    papercite<-grepl(searchi, Ref.corpus$content, fixed=TRUE)
    citations[,i]<-papercite
  }
  return(citations)
}

```

-
- The application of methods of social network analysis in bibliometrics and webometrics. Measures and tools

https://www.researchgate.net/publication/327817518_The_application_of_methods_of_social_network_analysis_in_bibliometrics_and_webometrics_Measures_and_tools

Chapter 33

ScientoMiner ICR

<https://zenodo.org/record/1432557#.XItjfxO2k1J>

Chapter 34

onodo

<https://onodo.org/dashboard>

<https://onodo.org/tutorials>

Chapter 35

BibExcel

<https://homepage.univie.ac.at/juan.gorraiz/bibexcel/>

Chapter 36

Scientometric Portal

<https://sites.google.com/site/hjamali/scientometric-portal>

Chapter 37

leydesdorff

<https://www.leydesdorff.net/software.htm>

Chapter 38

Publish or Perish

<https://harzing.com/resources/publish-or-perish>

Chapter 39

Pajek: analysis and visualization of large networks

<http://mrvar.fdv.uni-lj.si/pajek/>

- <https://www.bioconductor.org/>

```
## try http:// if https:// URLs are not supported
source("https://bioconductor.org/biocLite.R")
biocLite()
```

- The Bioconductor 2018 Workshop Compilation <https://bioconductor.github.io/BiocWorkshops/index.html>

<https://github.com/Bioconductor/BiocWorkshops>

https://raw.githubusercontent.com/Bioconductor/BiocWorkshops/master/100_Morgan_RBiocForAll/ALL-phenoData.csv

<https://support.bioconductor.org/>

<https://bioconductor.org/help/course-materials/>

https://genome.ucsc.edu/cgi-bin/hgTables?hgsid=578954849_wF1QP81SIHdfr8b0kmZUOcsZcHYr&clade=mammal&org=Human&db=hg38&hgta_group=regulation&hgta_track=knownGene&hgta_table=0&hgta_regionType=genome&position=chr9%3A133252000-133280861&hgta_outputType=primaryTable&hgta_outFileName=

<https://bioconductor.github.io/BiocWorkshops/r-and-bioconductor-for-everyone-an-introduction.html>

- **Introduction to Bioconductor**

<https://www.datacamp.com/community/tutorials/intro-bioconductor>

Important packages:

- DNASTringSet
 - Biostrings
 - GenomicRanges
-

<https://bioconductor.org/packages>

<https://support.bioconductor.org/>

<http://bioconductor.org/help/course-materials/>

-
- DESeq results to pathways in 60 Seconds with the fgsea package

<https://stephenturner.github.io/deseq-to-fgsea/>

Chapter 40

Bioconductor

<https://www.youtube.com/user/bioconductor>

40.1 Courses & Conferences

<https://www.bioconductor.org/help/course-materials/>

Chapter 41

Neuroconductor Tutorials

<https://neuroconductor.org/tutorials>

Chapter 42

Neuroconductor Courses

<https://neuroconductor.org/courses>

An R interface for computational modeling of tumor progression

<https://bioconductor.org/packages/release/bioc/html/CancerInSilico.html>

<https://bioconductor.org/packages/release/bioc/vignettes/CancerInSilico/inst/doc/CancerInSilico.html>

Chapter 43

Running a Cell Simulation

43.1 Run Simple Simulation

43.2 Plot CellModel Object

43.3 Query Cell Information

Chapter 44

Drugs

Chapter 45

Cell Types

45.1 Adding a Single Cell Type

45.2 Adding Multiple Cell Types

45.3 Getting Cell Type

Chapter 46

Pathways

46.1 Calibrate Gene Expression Range

46.2 Generate Pathway Activity

46.3 Visualize Pathway Activity

46.4 Accounting for Model Effects

46.5 Normalize Pathway Activity

Chapter 47

Simulating Bulk Gene Expression Data

47.1 Simulating Microarray Data

47.2 Visualize Bulk Gene Expression Data

Chapter 48

Simulating Single Cell Gene Expression Data

48.1 Cell Type Pathways

48.2 Simulating Single Cell RNA-seq

48.3 Visualize Single Cell Data

Chapter 49

BCRA

<https://cran.r-project.org/web/packages/BCRA/index.html>

Chapter 50

cgdsr

cgdsr: R-Based API for Accessing the MSKCC Cancer Genomics Data Server (CGDS)

<https://cran.r-project.org/web/packages/cgdsr/index.html>

Chapter 51

TCGAbiolinksGUI

<https://bioconductor.org/packages/release/bioc/html/TCGAbiolinksGUI.html>

Chapter 52

RTCGA

Chapter 53

CancerSubtypes

Chapter 54

CancerMutationAnalysis

Chapter 55

cancerclass

Chapter 56

canceR

Chapter 57

bioCancer

Chapter 58

TCGAretriever

TCGAretriever: Retrieve Genomic and Clinical Data from TCGA

<https://cran.r-project.org/web/packages/TCGAretriever/index.html>

Chapter 59

TCGA2STAT

<https://cran.r-project.org/web/packages/TCGA2STAT/vignettes/TCGA2STAT.html>

Chapter 60

TCIApathfinder

TCIApathfinder: Client for the Cancer Imaging Archive REST API

<https://cran.r-project.org/web/packages/TCIApathfinder/index.html>

Chapter 61

MILC

MILC: Microsimulation Lung Cancer (MILC) model

<https://cran.r-project.org/web/packages/MILC/index.html>

Chapter 62

InfiniumPurify

InfiniumPurify: Estimate and Account for Tumor Purity in Cancer Methylation Data Analysis

<https://cran.r-project.org/web/packages/InfiniumPurify/index.html>

Chapter 63

rclone

<https://rclone.org/drive/>

Chapter 64

rmdrive

<https://github.com/ekothe/rmdrive>

```
rstudioapi::selectDirectory()
```

```
xaringan::inf_mr()
```

Load required packages

Load required packages

- Load required packages

Gerekli paketleri yükle

```
library(tidyverse)
```

Chapter 65

tips

Chapter 66

environment memory

<http://r-statistics.co/R-Tutorial.html>

As you create new variables, by default they get store in what is called a global environment.

```
a <- 10 b <- 20 ls() # list objects in global env rm(a) # delete the object 'a' rm(list = ls()) # caution: delete all objects in .GlobalEnv gc() # free system memory
```

However if you choose, you can create a new environment and store them there.

```
rm(list=ls()) # remove all objects in work space env1 <- new.env() # create a new environment assign("a", 3, envir = env1) # store a=3 inside env1 ls() # returns objects in .GlobalEnv ls(env1) # returns objects in env1 get('a', envir=env1) # retrieve value from env1
```

```
sort(vec1) # ascending sort sort(vec1, decreasing = TRUE) # Descending sort Sorting can also be achieved using the order() function which returns the indices of elements in ascending order.
```

```
vec1[order(vec1)] # ascending sort vec1[rev(order(vec1))] # descending sort
```

```
seq(1, 10, by = 2) # diff between adj elements is 2 seq(1, 10, length=25) # length of the vector is 25 rep(1, 5) # repeat 1, five times. rep(1:3, 5) # repeat 1:3, 5 times rep(1:3, each=5) # repeat 1 to 3, each 5 times.
```

```
subset(airquality, Day == 1, select = -Temp) # select Day=1 and exclude 'Temp' airquality[which(airquality$Day==1), -c(4)] # same as above
```

```
set.seed(100) trainIndex <- sample(c(1:nrow(airquality)), size=nrow(airquality)*0.7, replace=F) # get test sample indices airquality[trainIndex, ] # training data airquality[-trainIndex, ] # test data
```

```
if(checkConditionIfTrue) { ....statements.. ....statements.. } else { # place the 'else' in same line as '}' ....statements.. ....statements.. }
```

```
for(counterVar in c(1:n)){ .... statements.. }
```

66.0.1 Compare Means

Chapter 67

infer

Randomization Examples using nycflights13 flights data

https://cran.r-project.org/web/packages/infer/vignettes/flights_examples.html

Hypothesis tests One numerical variable (mean)

One numerical variable (median)

One categorical (one proportion)

```
null_distn <- fli_small %>%
  specify(response = day_hour, success = "morning") %>%
  hypothesize(null = "point", p = .5) %>%
  generate(reps = 1000, type = "simulate") %>%
  calculate(stat = "prop")
ggplot(null_distn, aes(x = stat)) +
  geom_bar() +
  geom_vline(xintercept = p_hat, color = "red")
```

```
null_distn %>%
  summarize(p_value = mean(stat <= p_hat) * 2)
p_value
0.132
```

Logical variables will be coerced to factors:

```
null_distn <- fli_small %>%
  mutate(day_hour_logical = (day_hour == "morning")) %>%
  specify(response = day_hour_logical, success = "TRUE") %>%
  hypothesize(null = "point", p = .5) %>%
  generate(reps = 1000, type = "simulate") %>%
  calculate(stat = "prop")
Two categorical (2 level) variables
d_hat <- fli_small %>%
  group_by(season) %>%
  summarize(prop = mean(day_hour == "morning")) %>%
  summarize(diff(prop)) %>%
  pull()
null_distn <- fli_small %>%
  specify(day_hour ~ season, success = "morning") %>%
  hypothesize(null = "independence") %>%
  generate(reps = 1000, type = "permute") %>%
  calculate(stat = "diff in props", order = c("winter", "summer"))
ggplot(null_distn, aes(x = stat)) +
```

```

geom_density() +
geom_vline(xintercept = d_hat, color = "red")

null_distn %>%
  summarize(p_value = mean(stat <= d_hat) * 2) %>%
  pull()
## [1] 0.758
One categorical (>2 level) - GoF
Chisq_hat <- fli_small %>%
  specify(response = origin) %>%
  hypothesize(null = "point",
    p = c("EWR" = .33, "JFK" = .33, "LGA" = .34)) %>%
  calculate(stat = "Chisq")
null_distn <- fli_small %>%
  specify(response = origin) %>%
  hypothesize(null = "point",
    p = c("EWR" = .33, "JFK" = .33, "LGA" = .34)) %>%
  generate(reps = 1000, type = "simulate") %>%
  calculate(stat = "Chisq")
ggplot(null_distn, aes(x = stat)) +
  geom_density() +
  geom_vline(xintercept = pull(Chisq_hat), color = "red")

null_distn %>%
  summarize(p_value = mean(stat >= pull(Chisq_hat))) %>%
  pull()
## [1] 0.002
Two categorical (>2 level) variables
Chisq_hat <- fli_small %>%
  chisq_stat(formula = day_hour ~ origin)
null_distn <- fli_small %>%
  specify(day_hour ~ origin, success = "morning") %>%
  hypothesize(null = "independence") %>%
  generate(reps = 1000, type = "permute") %>%
  calculate(stat = "Chisq")
ggplot(null_distn, aes(x = stat)) +
  geom_density() +
  geom_vline(xintercept = pull(Chisq_hat), color = "red")

null_distn %>%
  summarize(p_value = mean(stat >= pull(Chisq_hat))) %>%
  pull()
## [1] 0.017
One numerical variable, one categorical (2 levels) (diff in means)
d_hat <- fli_small %>%
  group_by(season) %>%
  summarize(mean_stat = mean(dep_delay)) %>%
  # Since summer - winter
  summarize(-diff(mean_stat)) %>%
  pull()
null_distn <- fli_small %>%
  specify(dep_delay ~ season) %>% # alt: response = dep_delay,
  # explanatory = season
  hypothesize(null = "independence") %>%
  generate(reps = 1000, type = "permute") %>%
  calculate(stat = "diff in means", order = c("summer", "winter"))
ggplot(null_distn, aes(x = stat)) +

```

```

geom_density() +
geom_vline(xintercept = d_hat, color = "red")

null_distn %>%
  summarize(p_value = mean(stat <= d_hat) * 2) %>%
  pull()
## [1] 1.574
One numerical variable, one categorical (2 levels) (diff in medians)
d_hat <- fli_small %>%
  group_by(season) %>%
  summarize(median_stat = median(dep_delay)) %>%
  # Since summer - winter
  summarize(-diff(median_stat)) %>%
  pull()
null_distn <- fli_small %>%
  specify(dep_delay ~ season) %>% # alt: response = dep_delay,
  # explanatory = season
  hypothesize(null = "independence") %>%
  generate(reps = 1000, type = "permute") %>%
  calculate(stat = "diff in medians", order = c("summer", "winter"))
ggplot(null_distn, aes(x = stat)) +
  geom_bar() +
  geom_vline(xintercept = d_hat, color = "red")

null_distn %>%
  summarize(p_value = mean(stat >= d_hat) * 2) %>%
  pull()
## [1] 0.068
One numerical, one categorical (>2 levels) - ANOVA
F_hat <- anova(
  aov(formula = arr_delay ~ origin, data = fli_small)
)$`F value`[1]
null_distn <- fli_small %>%
  specify(arr_delay ~ origin) %>% # alt: response = arr_delay,
  # explanatory = origin
  hypothesize(null = "independence") %>%
  generate(reps = 1000, type = "permute") %>%
  calculate(stat = "F")
ggplot(null_distn, aes(x = stat)) +
  geom_density() +
  geom_vline(xintercept = F_hat, color = "red")

null_distn %>%
  summarize(p_value = mean(stat >= F_hat)) %>%
  pull()
## [1] 0.351
Two numerical vars - SLR
slope_hat <- lm(arr_delay ~ dep_delay, data = fli_small) %>%
  broom::tidy() %>%
  filter(term == "dep_delay") %>%
  pull(estimate)
null_distn <- fli_small %>%
  specify(arr_delay ~ dep_delay) %>%
  hypothesize(null = "independence") %>%
  generate(reps = 1000, type = "permute") %>%
  calculate(stat = "slope")
ggplot(null_distn, aes(x = stat)) +

```

```

geom_density() +
geom_vline(xintercept = slope_hat, color = "red")

null_distn %>%
  summarize(p_value = mean(stat >= slope_hat) * 2) %>%
  pull()
## [1] 0
Confidence intervals
One numerical (one mean)
x_bar <- fli_small %>%
  summarize(mean(arr_delay)) %>%
  pull()
boot <- fli_small %>%
  specify(response = arr_delay) %>%
  generate(reps = 1000, type = "bootstrap") %>%
  calculate(stat = "mean") %>%
  pull()
c(lower = x_bar - 2 * sd(boot),
  upper = x_bar + 2 * sd(boot))
##      lower      upper
## 1.122209 8.021791
One categorical (one proportion)
p_hat <- fli_small %>%
  summarize(mean(day_hour == "morning")) %>%
  pull()
boot <- fli_small %>%
  specify(response = day_hour, success = "morning") %>%
  generate(reps = 1000, type = "bootstrap") %>%
  calculate(stat = "prop") %>%
  pull()
c(lower = p_hat - 2 * sd(boot),
  upper = p_hat + 2 * sd(boot))
##      lower      upper
## 0.4194756 0.5125244
One numerical variable, one categorical (2 levels) (diff in means)
d_hat <- fli_small %>%
  group_by(season) %>%
  summarize(mean_stat = mean(arr_delay)) %>%
  # Since summer - winter
  summarize(-diff(mean_stat)) %>%
  pull()
boot <- fli_small %>%
  specify(arr_delay ~ season) %>%
  generate(reps = 1000, type = "bootstrap") %>%
  calculate(stat = "diff in means", order = c("summer", "winter")) %>%
  pull()
c(lower = d_hat - 2 * sd(boot),
  upper = d_hat + 2 * sd(boot))
##      lower      upper
## -7.704370 6.213971
Two categorical variables (diff in proportions)
d_hat <- fli_small %>%
  group_by(season) %>%
  summarize(prop = mean(day_hour == "morning")) %>%
  # Since summer - winter
  summarize(-diff(prop)) %>%
  pull()

```

```

boot <- fli_small %>%
  specify(day_hour ~ season, success = "morning") %>%
  generate(reps = 1000, type = "bootstrap") %>%
  calculate(stat = "diff in props", order = c("summer", "winter")) %>%
  pull()
c(lower = d_hat - 2 * sd(boot),
  upper = d_hat + 2 * sd(boot))
##      lower      upper
## -0.07149487  0.11258550
Two numerical vars - SLR
slope_hat <- lm(arr_delay ~ dep_delay, data = fli_small) %>%
  broom::tidy() %>%
  filter(term == "dep_delay") %>%
  pull(estimate)
boot <- fli_small %>%
  specify(arr_delay ~ dep_delay) %>%
  generate(reps = 1000, type = "bootstrap") %>%
  calculate(stat = "slope") %>%
  pull()
c(lower = slope_hat - 2 * sd(boot),
  upper = slope_hat + 2 * sd(boot))
##      lower      upper
##  0.9657595  1.0681384

```

Examples using mtcars data

https://cran.r-project.org/web/packages/infer/vignettes/mtcars_examples.html

Examples using mtcars data

Chester Ismay and Andrew Bray

2018-01-05

Note: The type argument in generate() is automatically filled based on the entries for specify() and hypoth

Data preparation

```
library(infer)
```

```
library(dplyr)
```

```
mtcars <- mtcars %>%
```

```
  mutate(cyl = factor(cyl),
         vs = factor(vs),
         am = factor(am),
         gear = factor(gear),
         carb = factor(carb))
```

```
# For reproducibility
```

```
set.seed(2018)
```

```
One numerical variable (mean)
```

```
mtcars %>%
```

```
  specify(response = mpg) %>% # formula alt: mpg ~ NULL
  hypothesize(null = "point", mu = 25) %>%
  generate(reps = 100, type = "bootstrap") %>%

```

```

  calculate(stat = "mean")
## # A tibble: 100 x 2
##   replicate stat
##   <int> <dbl>
## 1         1 26.6
## 2         2 25.1
## 3         3 25.2
## 4         4 24.7
## 5         5 24.6
## 6         6 25.8
## 7         7 24.7
## 8         8 25.6
## 9         9 25.0
## 10        10 25.1
## # ... with 90 more rows
One numerical variable (median)

```

```

mtcars %>%
  specify(response = mpg) %>% # formula alt: mpg ~ NULL
  hypothesize(null = "point", med = 26) %>%
  generate(reps = 100, type = "bootstrap") %>%
  calculate(stat = "median")
## # A tibble: 100 x 2
##   replicate stat
##   <int> <dbl>
## 1         1 28.2
## 2         2 27.2
## 3         3 26.2
## 4         4 26
## 5         5 26.5
## 6         6 24.5
## 7         7 26
## 8         8 28.2
## 9         9 28.2
## 10        10 23.2
## # ... with 90 more rows
One categorical (2 level) variable

```

```

mtcars %>%
  specify(response = am, success = "1") %>% # formula alt: am ~ NULL
  hypothesize(null = "point", p = .25) %>%
  generate(reps = 100, type = "simulate") %>%
  calculate(stat = "prop")
## # A tibble: 100 x 2
##   replicate stat
##   <fct> <dbl>
## 1 1      0.375
## 2 2      0.0625
## 3 3      0.125
## 4 4      0.25
## 5 5      0.188
## 6 6      0.406
## 7 7      0.219
## 8 8      0.375
## 9 9      0.344
## 10 10     0.188
## # ... with 90 more rows

```

Two categorical (2 level) variables

```
mtcars %>%
  specify(am ~ vs, success = "1") %>% # alt: response = am, explanatory = vs
  hypothesize(null = "independence") %>%
  generate(reps = 100, type = "permute") %>%
  calculate(stat = "diff in props", order = c("0", "1"))
## # A tibble: 100 x 2
##   replicate    stat
##   <int>    <dbl>
## 1         1 -0.421
## 2         2 -0.167
## 3         3 -0.421
## 4         4 -0.0397
## 5         5  0.0873
## 6         6 -0.0397
## 7         7 -0.0397
## 8         8 -0.0397
## 9         9  0.0873
## 10        10 -0.167
## # ... with 90 more rows
One categorical (>2 level) - GoF
```

```
mtcars %>%
  specify(cyl ~ NULL) %>% # alt: response = cyl
  hypothesize(null = "point", p = c("4" = .5, "6" = .25, "8" = .25)) %>%
  generate(reps = 100, type = "simulate") %>%
  calculate(stat = "Chisq")
## # A tibble: 100 x 2
##   replicate    stat
##   <fct>    <dbl>
## 1 1         6.75
## 2 2         1.69
## 3 3         3.19
## 4 4         1.69
## 5 5         6
## 6 6         2.69
## 7 7         4.75
## 8 8         0.75
## 9 9         0.688
## 10 10        3.69
## # ... with 90 more rows
Two categorical (>2 level) variables
```

```
mtcars %>%
  specify(cyl ~ am) %>% # alt: response = cyl, explanatory = am
  hypothesize(null = "independence") %>%
  generate(reps = 100, type = "permute") %>%
  calculate(stat = "Chisq")
## # A tibble: 100 x 2
##   replicate    stat
##   <int>    <dbl>
## 1         1  1.34
## 2         2  1.63
## 3         3  1.63
## 4         4  2.63
## 5         5  3.90
```

```
## 6      6 1.74
## 7      7 0.126
## 8      8 1.74
## 9      9 1.34
## 10     10 1.34
## # ... with 90 more rows
One numerical variable one categorical (2 levels) (diff in means)
```

```
mtcars %>%
  specify(mpg ~ am) %>% # alt: response = mpg, explanatory = am
  hypothesize(null = "independence") %>%
  generate(reps = 100, type = "permute") %>%
  calculate(stat = "diff in means", order = c("0", "1"))
## # A tibble: 100 x 2
##   replicate  stat
##   <int> <dbl>
## 1         1 -1.10
## 2         2  0.217
## 3         3 -1.08
## 4         4 -3.80
## 5         5  3.08
## 6         6  0.489
## 7         7  2.34
## 8         8  4.10
## 9         9 -1.86
## 10        10 -0.210
## # ... with 90 more rows
One numerical variable one categorical (2 levels) (diff in medians)
```

```
mtcars %>%
  specify(mpg ~ am) %>% # alt: response = mpg, explanatory = am
  hypothesize(null = "independence") %>%
  generate(reps = 100, type = "permute") %>%
  calculate(stat = "diff in medians", order = c("0", "1"))
## # A tibble: 100 x 2
##   replicate  stat
##   <int> <dbl>
## 1         1  0.5
## 2         2 -1.10
## 3         3  5.20
## 4         4  1.8
## 5         5  0.5
## 6         6  3.3
## 7         7 -1.60
## 8         8 -2.3
## 9         9  2.90
## 10        10 -0.5
## # ... with 90 more rows
One numerical one categorical (>2 levels) - ANOVA
```

```
mtcars %>%
  specify(mpg ~ cyl) %>% # alt: response = mpg, explanatory = cyl
  hypothesize(null = "independence") %>%
  generate(reps = 100, type = "permute") %>%
  calculate(stat = "F")
## # A tibble: 100 x 2
##   replicate  stat
```



```
##          <int> <dbl>
##  1          1 1.43
##  2          2 1.65
##  3          3 0.318
##  4          4 0.393
##  5          5 1.05
##  6          6 0.826
##  7          7 1.32
##  8          8 0.833
##  9          9 0.144
## 10         10 0.365
## # ... with 90 more rows
Two numerical vars - SLR

mtcars %>%
  specify(mpg ~ hp) %>% # alt: response = mpg, explanatory = cyl
  hypothesize(null = "independence") %>%
  generate(reps = 100, type = "permute") %>%
  calculate(stat = "slope")
## # A tibble: 100 x 2
##   replicate    stat
##   <int>      <dbl>
##  1         1 -0.0151
##  2         2  0.00224
##  3         3 -0.0120
##  4         4  0.00292
##  5         5  0.0203
##  6         6 -0.00730
##  7         7 -0.0246
##  8         8  0.00555
##  9         9  0.0109
## 10        10  0.0176
## # ... with 90 more rows
One numerical variable (standard deviation)

Not currently implemented

mtcars %>%
  specify(response = mpg) %>% # formula alt: mpg ~ NULL
  hypothesize(null = "point", sigma = 5) %>%
  generate(reps = 100, type = "bootstrap") %>%
  calculate(stat = "sd")
Confidence intervals
One numerical (one mean)

mtcars %>%
  specify(response = mpg) %>%
  generate(reps = 100, type = "bootstrap") %>%
  calculate(stat = "mean")
## # A tibble: 100 x 2
##   replicate  stat
##   <int>    <dbl>
##  1         1  19.6
##  2         2  21.8
##  3         3  18.7
##  4         4  19.2
##  5         5  21.6
```

```

## 6      6 19.9
## 7      7 20.7
## 8      8 19.3
## 9      9 21.2
## 10     10 21.3
## # ... with 90 more rows
One numerical (one median)

mtcars %>%
  specify(response = mpg) %>%
  generate(reps = 100, type = "bootstrap") %>%
  calculate(stat = "median")
## # A tibble: 100 x 2
##   replicate stat
##   <int> <dbl>
## 1      1 19.2
## 2      2 20.1
## 3      3 21
## 4      4 17.8
## 5      5 20.1
## 6      6 19.2
## 7      7 18.4
## 8      8 19.2
## 9      9 19.2
## 10     10 18.0
## # ... with 90 more rows
One numerical (standard deviation)

mtcars %>%
  specify(response = mpg) %>%
  generate(reps = 100, type = "bootstrap") %>%
  calculate(stat = "sd")
## # A tibble: 100 x 2
##   replicate stat
##   <int> <dbl>
## 1      1 5.28
## 2      2 6.74
## 3      3 5.29
## 4      4 5.41
## 5      5 5.56
## 6      6 5.65
## 7      7 6.17
## 8      8 6.40
## 9      9 6.31
## 10     10 6.11
## # ... with 90 more rows
One categorical (one proportion)

mtcars %>%
  specify(response = am, success = "1") %>%
  generate(reps = 100, type = "bootstrap") %>%
  calculate(stat = "prop")
## # A tibble: 100 x 2
##   replicate stat
##   <int> <dbl>
## 1      1 0.375
## 2      2 0.406

```

```
## 3      3 0.406
## 4      4 0.312
## 5      5 0.312
## 6      6 0.469
## 7      7 0.438
## 8      8 0.281
## 9      9 0.438
## 10     10 0.5
## # ... with 90 more rows
One numerical variable one categorical (2 levels) (diff in means)
```

```
mtcars %>%
  specify(mpg ~ am) %>%
  generate(reps = 100, type = "bootstrap") %>%
  calculate(stat = "diff in means", order = c("0", "1"))
## # A tibble: 100 x 2
##   replicate    stat
##   <int>    <dbl>
## 1         1 -9.38
## 2         2 -5.11
## 3         3 -4.88
## 4         4 -5.39
## 5         5 -9.19
## 6         6 -7.20
## 7         7 -5.34
## 8         8 -3.20
## 9         9 -5.95
## 10        10 -11.0
## # ... with 90 more rows
Two categorical variables (diff in proportions)
```

```
mtcars %>%
  specify(am ~ vs, success = "1") %>%
  generate(reps = 100, type = "bootstrap") %>%
  calculate(stat = "diff in props", order = c("0", "1"))
## # A tibble: 100 x 2
##   replicate    stat
##   <int>    <dbl>
## 1         1 -0.352
## 2         2 -0.15
## 3         3 -0.294
## 4         4 -0.254
## 5         5 -0.438
## 6         6 -0.126
## 7         7 -0.188
## 8         8  0.167
## 9         9 -0.143
## 10        10 -0.5
## # ... with 90 more rows
Two numerical vars - SLR
```

```
mtcars %>%
  specify(mpg ~ hp) %>%
  generate(reps = 100, type = "bootstrap") %>%
  calculate(stat = "slope")
## # A tibble: 100 x 2
##   replicate    stat
```

```
##           <int>   <dbl>
##  1             1 -0.0850
##  2             2 -0.0512
##  3             3 -0.0736
##  4             4 -0.0569
##  5             5 -0.0930
##  6             6 -0.0659
##  7             7 -0.0710
##  8             8 -0.0767
##  9             9 -0.0556
## 10            10 -0.0627
## # ... with 90 more rows
Two numerical vars - correlation
```

```
mtcars %>%
  specify(mpg ~ hp) %>%
  generate(reps = 100, type = "bootstrap") %>%
  calculate(stat = "correlation")
## # A tibble: 100 x 2
##   replicate    stat
##   <int>    <dbl>
##  1         1 -0.821
##  2         2 -0.812
##  3         3 -0.802
##  4         4 -0.723
##  5         5 -0.885
##  6         6 -0.777
##  7         7 -0.752
##  8         8 -0.758
##  9         9 -0.826
## 10        10 -0.779
## # ... with 90 more rows
```

Two sample t test example using nycflights13 flights data

https://cran.r-project.org/web/packages/infer/vignettes/two_sample_t.html

Two sample t test example using nycflights13 flights data

Chester Ismay

2018-11-15

Note: The type argument in generate() is automatically filled based on the entries for specify() and hypoth

Data preparation

```
library(nycflights13)
library(dplyr)
library(stringr)
library(infer)
set.seed(2017)
fli_small <- flights %>%
  sample_n(size = 500) %>%
  mutate(half_year = case_when(
    between(month, 1, 6) ~ "h1",
    between(month, 7, 12) ~ "h2"
```

```

)) %>%
mutate(day_hour = case_when(
  between(hour, 1, 12) ~ "morning",
  between(hour, 13, 24) ~ "not morning"
)) %>%
select(arr_delay, dep_delay, half_year,
  day_hour, origin, carrier)
Two numeric - arr_delay, dep_delay
Two categories
half_year ("h1", "h2"),
day_hour ("morning", "not morning")
Three categories - origin ("EWR", "JFK", "LGA")
Sixteen categories - carrier
One numerical variable, one categorical (2 levels)
Calculate observed statistic
The recommended approach is to use specify() %>% calculate():

obs_t <- fli_small %>%
  specify(arr_delay ~ half_year) %>%
  calculate(stat = "t", order = c("h1", "h2"))
## Warning: Removed 15 rows containing missing values.
The observed t statistic is
stat
0.8685
.

Or using t_test in infer

obs_t <- fli_small %>%
  t_test(formula = arr_delay ~ half_year, alternative = "two_sided",
    order = c("h1", "h2")) %>%
  dplyr::pull(statistic)
The observed t statistic is 0.8685.

Or using another shortcut function in infer:

obs_t <- fli_small %>%
  t_stat(formula = arr_delay ~ half_year, order = c("h1", "h2"))
The observed t statistic is
statistic
0.8685
.

Randomization approach to t-statistic
t_null_perm <- fli_small %>%
  # alt: response = arr_delay, explanatory = half_year
  specify(arr_delay ~ half_year) %>%
  hypothesize(null = "independence") %>%
  generate(reps = 1000, type = "permute") %>%
  calculate(stat = "t", order = c("h1", "h2"))
## Warning: Removed 15 rows containing missing values.
visualize(t_null_perm) +
  shade_p_value(obs_stat = obs_t, direction = "two_sided")

Calculate the randomization-based p-value
t_null_perm %>%
  get_p_value(obs_stat = obs_t, direction = "two_sided")

```

```

p_value
0.408
Theoretical distribution
t_null_theor <- fli_small %>%
  # alt: response = arr_delay, explanatory = half_year
  specify(arr_delay ~ half_year) %>%
  hypothesize(null = "independence") %>%
  # generate() ## Not used for theoretical
  calculate(stat = "t", order = c("h1", "h2"))
## Warning: Removed 15 rows containing missing values.
visualize(t_null_theor, method = "theoretical") +
  shade_p_value(obs_stat = obs_t, direction = "two_sided")
## Warning: Check to make sure the conditions have been met for the
## theoretical method. {infer} currently does not check these for you.

Overlay appropriate t distribution on top of permuted t-statistics
visualize(t_null_perm, method = "both") +
  shade_p_value(obs_stat = obs_t, direction = "two_sided")
## Warning: Check to make sure the conditions have been met for the
## theoretical method. {infer} currently does not check these for you.

Compute theoretical p-value
fli_small %>%
  t_test(formula = arr_delay ~ half_year,
          alternative = "two_sided",
          order = c("h1", "h2")) %>%
  dplyr::pull(p_value)
## [1] 0.3855

```

67.0.1 Compare Proportions

chisq.test {stats} R Documentation
 Pearson's Chi-squared Test for Count Data
 Description
 chisq.test performs chi-squared contingency table tests and goodness-of-fit tests.

Usage

```
chisq.test(x, y = NULL, correct = TRUE,
           p = rep(1/length(x), length(x)), rescale.p = FALSE,
           simulate.p.value = FALSE, B = 2000)
```

Arguments

x

a numeric vector or matrix. x and y can also both be factors.

y

a numeric vector; ignored if x is a matrix. If x is a factor, y should be a factor of the same length.

correct

a logical indicating whether to apply continuity correction when computing the test statistic for 2 by 2 tables.

p

a vector of probabilities of the same length of x. An error is given if any entry of p is negative.

rescale.p

a logical scalar; if TRUE then p is rescaled (if necessary) to sum to 1. If rescale.p is FALSE, and p does

simulate.p.value

a logical indicating whether to compute p-values by Monte Carlo simulation.

B

an integer specifying the number of replicates used in the Monte Carlo test.

Details

If x is a matrix with one row or column, or if x is a vector and y is not given, then a goodness-of-fit test

If x is a matrix with at least two rows and columns, it is taken as a two-dimensional contingency table: the

If simulate.p.value is FALSE, the p-value is computed from the asymptotic chi-squared distribution of the test

In the contingency table case simulation is done by random sampling from the set of all contingency tables

In the goodness-of-fit case simulation is done by random sampling from the discrete distribution specified

Value

A list with class "htest" containing the following components:

statistic

the value the chi-squared test statistic.

parameter

the degrees of freedom of the approximate chi-squared distribution of the test statistic, NA if the p-value

p.value

the p-value for the test.

method

a character string indicating the type of test performed, and whether Monte Carlo simulation or continuity

data.name

a character string giving the name(s) of the data.

observed

the observed counts.

expected

the expected counts under the null hypothesis.

residuals

the Pearson residuals, $(\text{observed} - \text{expected}) / \sqrt{\text{expected}}$.

stdres

standardized residuals, $(\text{observed} - \text{expected}) / \sqrt{V}$, where V is the residual cell variance (Agresti, 2002).

Source

The code for Monte Carlo simulation is a C translation of the Fortran algorithm of Patefield (1981).

References

Hope, A. C. A. (1968). A simplified Monte Carlo significance test procedure. *Journal of the Royal Statistical Society B* 30, 598. <http://www.jstor.org/stable/2984263>.

Patefield, W. M. (1981). Algorithm AS 159: An efficient method of generating $r \times c$ tables with given row and column totals.

97. doi: 10.2307/2346669.

Agresti, A. (2007). An Introduction to Categorical Data Analysis, 2nd ed. New York: John Wiley & Sons. Page

See Also

For goodness-of-fit testing, notably of continuous distributions, `ks.test`.

Examples

```
## From Agresti(2007) p.39
M <- as.table(rbind(c(762, 327, 468), c(484, 239, 477)))
dimnames(M) <- list(gender = c("F", "M"),
                    party = c("Democrat", "Independent", "Republican"))
(Xsq <- chisq.test(M)) # Prints test summary
Xsq$observed      # observed counts (same as M)
Xsq$expected      # expected counts under the null
Xsq$residuals     # Pearson residuals
Xsq$stdres        # standardized residuals

## Effect of simulating p-values
x <- matrix(c(12, 5, 7, 7), ncol = 2)
chisq.test(x)$p.value      # 0.4233
chisq.test(x, simulate.p.value = TRUE, B = 10000)$p.value
                           # around 0.29!

## Testing for population probabilities
## Case A. Tabulated data
x <- c(A = 20, B = 15, C = 25)
chisq.test(x)
chisq.test(as.table(x))      # the same
x <- c(89,37,30,28,2)
p <- c(40,20,20,15,5)
try(
  chisq.test(x, p = p)      # gives an error
)
chisq.test(x, p = p, rescale.p = TRUE)
                           # works
p <- c(0.40,0.20,0.20,0.19,0.01)
                           # Expected count in category 5
                           # is 1.86 < 5 ==> chi square approx.
chisq.test(x, p = p)      # maybe doubtful, but is ok!
chisq.test(x, p = p, simulate.p.value = TRUE)

## Case B. Raw data
x <- trunc(5 * runif(100))
chisq.test(table(x))      # NOT 'chisq.test(x)!'
[Package stats version 3.5.1 Index]
```

Chapter 68

infer

Chi-squared test example using nycflights13 flights data

https://cran.r-project.org/web/packages/infer/vignettes/chisq_test.html

Chapter 69

comparisons between correlations

<http://comparingcorrelations.org/>

Chapter 70

Exploring correlations in R with corrr

<https://drsimonj.svbtle.com/exploring-correlations-in-r-with-corrr>

output:

```
html_notebook:
  fig_caption: yes
  highlight: tango
  number_sections: yes
  theme: paper
  toc: yes
  toc_depth: 5
  toc_float: yes
html_document:
  code_folding: hide
  df_print: kable
  keep_md: yes
  number_sections: yes
  theme: cerulean
  toc: yes
  toc_float: yes
  highlight: kate
```


Chapter 71

Data List

- Learning Clinical Epidemiology with R

<http://datacompass.lshtm.ac.uk/599/>

- ISLR
- acs

Download, Manipulate, and Present American Community Survey and Decennial Data from the US Census

<https://cran.r-project.org/web/packages/acs/index.html>

- eurostat

Tools for Eurostat Open Data

<https://cran.r-project.org/web/packages/eurostat/index.html>

- Rilostat

<https://github.com/ilostat/Rilostat>

- OECD

https://cran.r-project.org/web/packages/OECD/vignettes/oecd_vignette_main.pdf

- gapminder

Factfulness: Building Gapminder Income Mountains

<http://staff.math.su.se/hoehle/blog/2018/07/02/factfulness.html>

- nycflights13
- fivethirtyeight
- projects

<https://www.analyticsvidhya.com/blog/2014/11/data-science-projects-learn/>

- Miscellaneous Datasets

<http://users.stat.ufl.edu/~winner/datasets.html>

- datasets

<https://www.rdocumentation.org/packages/datasets/versions/3.5.1>

<https://livebook.datascienceheroes.com/>

Chapter 72

Rdatatable

<https://github.com/Rdatatable>

72.1 Introduction to data.table

<https://cloud.r-project.org/web/packages/data.table/vignettes/datatable-intro.html>

```
DT = data.table(  
  ID = c("b","b","b","a","a","c"),  
  a = 1:6,  
  b = 7:12,  
  c = 13:18  
)  
DT
```

```
class(DT$ID)
```

```
getOption("datatable.print.nrows")
```

```
ans <- flights[origin == "JFK" & month == 6L]  
head(ans)
```

```
ans <- flights[1:2]  
ans
```

```
ans <- flights[origin == "JFK" & month == 6L][1:2]  
head(ans)
```

```
ans <- flights[order(origin, -dest)]  
head(ans)
```

```
ans <- flights[, arr_delay]  
head(ans)
```

```

ans <- flights[, arr_delay, dest]
head(ans)

ans <- flights[, list(arr_delay)]
head(ans)

ans <- flights[, .(arr_delay)]
head(ans)

ans <- flights[, .(arr_delay, dep_delay)]
head(ans)

ans <- flights[, .(delay_arr = arr_delay, delay_dep = dep_delay)]
head(ans)

ans <- flights[, sum( (arr_delay + dep_delay) < 0 )]
ans

ans <- flights[origin == "JFK" & month == 6L,
               .(m_arr = mean(arr_delay), m_dep = mean(dep_delay))]
ans

ans <- flights[origin == "JFK" & month == 6L, length(dest)]
ans

ans <- flights[origin == "JFK" & month == 6L, .N]
ans

ans <- flights[, c("arr_delay", "dep_delay")]
head(ans)

select_cols = c("arr_delay", "dep_delay")
flights[, ..select_cols]

flights[, select_cols, with = FALSE]

ans <- flights[, !c("arr_delay", "dep_delay")]

ans <- flights[, -c("arr_delay", "dep_delay")]

ans <- flights[, year:day]

ans <- flights[, day:year]

ans <- flights[, -(year:day)]
ans <- flights[, !(year:day)]

ans <- flights[, .(.N), by = .(origin)]
ans

```

```
ans <- flights[, .(N), by = "origin"]
ans

ans <- flights[, .N, by = origin]
ans

ans <- flights[carrier == "AA", .N, by = origin]
ans

ans <- flights[carrier == "AA", .N, by = .(origin, dest)]
head(ans)

ans <- flights[carrier == "AA", .N, by = c("origin", "dest")]
ans

ans <- flights[carrier == "AA",
               .(mean(arr_delay), mean(dep_delay)),
               by = .(origin, dest, month)]
ans

ans <- flights[carrier == "AA",
               .(mean(arr_delay), mean(dep_delay)),
               keyby = .(origin, dest, month)]
ans

ans <- flights[carrier == "AA", .N, by = .(origin, dest)]
ans

ans <- flights[carrier == "AA", .N, by = .(origin, dest)][order(origin, -dest)]
head(ans, 10)

ans <- flights[, .N, .(dep_delay>0, arr_delay>0)]
ans

flights[, .N, .(dep_delayed = dep_delay>0, arr_delayed = arr_delay>0)]
```


Chapter 73

cheat sheet

<https://www.datacamp.com/community/tutorials/data-table-cheat-sheet>

https://s3.amazonaws.com/assets.datacamp.com/blog_assets/datatable_Cheat_Sheet_R.pdf

<http://r-datatable.com>

<https://github.com/Rdatatable/data.table/wiki>

73.1 Subsetting Rows Using `i`

73.2 Manipulating Columns in `j`

sonuç vektör olarak alınacaksa sadece sütun ismi yazılıyor

sonuç data.frame olarak alınacaksa sütun ismi önünde `.` yazılıyor

tek sütun üzerinden özet alma

birden fazla sütun üzerinden özet alma

73.3 Doing `j` by Group

73.4 Adding/Updating Columns By Reference in `j` Using `:=`

73.5 Indexing And Keys

- Installations for Data Science. Anaconda, RStudio, Spark, TensorFlow, AWS (Amazon Web Services).

<https://medium.com/@GalarnykMichael>

https://github.com/mGalarnyk/Installations_Mac_Ubuntu_Windows

- Google Cloud for Data Science: Beginner's Guide <https://www.datacamp.com/community/tutorials/google-cloud-data-science>
- Deep Learning With Jupyter Notebooks In The Cloud <https://www.datacamp.com/community/tutorials/deep-learning-jupyter-aws>

<https://www.datacamp.com/community/tutorials/homebrew-install-use>

system() function works when I use R from terminal but not from RStudio #2193

<https://github.com/rstudio/rstudio/issues/2193>

```
myTerm <- rstudioapi::terminalCreate(show = FALSE)
rstudioapi::terminalSend(myTerm, "esearch -db pubmed -query '(diabetes AND pregnancy) AND (\"2017/01/01\" [P
Sys.sleep(1)
repeat{
  Sys.sleep(0.1)
  if(rstudioapi::terminalBusy(myTerm) == FALSE){
    print("Code Executed")
    break
  }
}
```

Chapter 74

Decision Trees

```
prune.carseats = prune.misclass(tree.carseats, best = 12)
plot(prune.carseats)
text(prune.carseats, pretty=0)
```

It's a bit shallower than previous trees, and you can actually read the labels. Let's evaluate it on the test set.

```
tree.pred = predict(prune.carseats, carseats[-train,], type="class")
with(carseats[-train,], table(tree.pred, High))
(74 + 39) / 150
```

Seems like the correct classifications dropped a little bit. It has done about the same as your original tree.

Often case, trees don't give very good prediction errors, so let's go ahead take a look at random forests and boosting.

Random Forests

For this part, you will use the Boston housing data to explore random forests and boosting. The dataset is

```
library(MASS)
data(package="MASS")
boston<-Boston
dim(boston)
names(boston)
```

Let's also load the randomForest package.

```
require(randomForest)
```

To prepare data for random forest, let's set the seed and create a sample training set of 300 observations.

```
set.seed(101)
train = sample(1:nrow(boston), 300)
```

In this dataset, there are 506 suburbs of Boston. For each suburb, you have variables such as crime per capita.

Let's fit a random forest and see how well it performs. As being said, you use the response medv, the median value of a house.

```
rf.boston = randomForest(medv~., data = boston, subset = train)
rf.boston
```

Printing out the random forest gives its summary: the # of trees (500 were grown), the mean squared residual error.

The only tuning parameter in a random Forests is the argument called mtry, which is the number of variables to sample at each node.

You're going to fit a series of random forests. There are 13 variables, so let's have mtry range from 1 to 13.

In order to record the errors, you set up 2 variables oob.err and test.err.

In a loop of `mtry` from 1 to 13, you first fit the `randomForest` with that value of `mtry` on the train dataset. Then you extract the mean-squared-error on the object (the out-of-bag error).

Then you predict on the test dataset (`boston[-train]`) using `fit` (the fit of `randomForest`).

Lastly, you compute the test error: mean-squared error, which is equals to `mean((medv - pred) ^ 2)`.

```
oob.err = double(13)
test.err = double(13)
for(mtry in 1:13){
  fit = randomForest(medv~., data = boston, subset=train, mtry=mtry, ntree = 350)
  oob.err[mtry] = fit$mse[350]
  pred = predict(fit, boston[-train,])
  test.err[mtry] = with(boston[-train,], mean( (medv-pred)^2 ))
}
```

Basically you just grew 4550 trees (13 times 350). Now let's make a plot using the `matplot` command. The test

```
matplot(1:mtry, cbind(test.err, oob.err), pch = 23, col = c("red", "blue"), type = "b", ylab="Mean Squared
legend("topright", legend = c("OOB", "Test"), pch = 23, col = c("red", "blue"))
```

Ideally, these 2 curves should line up, but it seems like the test error is a bit lower. However, there's a

Notice that the red curve is smoothly above the blue curve? These error estimates are very correlated, because

So with very few tiers, you have fitted a very powerful prediction model using random forests. How so? The

Boosting

Compared to random forests, boosting grows smaller and stubbier trees and goes at the bias. You will use the

```
require(gbm)
```

GBM asks for the distribution, which is Gaussian, because you'll be doing squared error loss. You're going

```
boost.boston = gbm(medv~., data = boston[train,], distribution = "gaussian", n.trees = 10000, shrinkage = 0.1)
summary(boost.boston)
```

The `summary` function gives a variable importance plot. It seems like there are 2 variables that have high r

```
plot(boost.boston,i="lstat")
plot(boost.boston,i="rm")
```

The 1st plot shows that the higher the proportion of lower status people in the suburb, the lower the value

It's time to predict a boosted model on the test dataset. Let's look at the test performance as a function

First, you make a grid of number of trees in steps of 100 from 100 to 10,000.

Then, you run the `predict` function on the boosted model. It takes `n.trees` as an argument, and produces a matrix

The dimensions of the matrix are 206 test observations and 100 different predict vectors at the 100 different

```
n.trees = seq(from = 100, to = 10000, by = 100)
predmat = predict(boost.boston, newdata = boston[-train,], n.trees = n.trees)
dim(predmat)
```

It's time to compute the test error for each of the predict vectors:

`predmat` is a matrix, `medv` is a vector, thus `(predmat - medv)` is a matrix of differences. You can use the `ap`

Then you make a plot using similar parameters to that one used for Random Forest. It would show a boosting


```
boost.err = with(boston[-train,], apply( (predmat - medv)^2, 2, mean) )  
plot(n.trees, boost.err, pch = 23, ylab = "Mean Squared Error", xlab = "# Trees", main = "Boosting Test Error")  
abline(h = min(test.err), col = "red")
```

The boosting error pretty much drops down as the number of trees increases. This is an evidence showing that

Conclusion

So that's the end of this R tutorial on building decision tree models: classification trees, random forests

If you would like to learn more, be sure to take a look at our Machine Learning Toolbox course for R.

Chapter 75

decision tree

<https://analytics4all.org/2016/11/23/r-decision-trees-regression/>

Chapter 76

DECISION TREE CLASSIFIER IMPLEMENTATION IN R

<https://dataaspirant.com/2017/01/30/how-decision-tree-algorithm-works/>

<https://dataaspirant.com/2017/02/03/decision-tree-classifier-implementation-in-r/>

Chapter 77

caret

Classification And REgression Training

77.1 Descriptive Statistics

77.2 skimr

https://cran.r-project.org/web/packages/skimr/vignettes/Using__skimr.html

- Exploratory Data Analysis in R (introduction)

<https://blog.datascienceheroes.com/exploratory-data-analysis-in-r-intro/>

-
- What's so hard about histograms?

<http://tinlizzie.org/~aran/histograms/>

Chapter 78

DataExplorer

Chapter 79

Webinar: Tidyverse Exploratory Analysis (Emily Robinson)

<https://hookedondata.org/the-lesser-known-stars-of-the-tidyverse/>

<https://www.rstudio.com/resources/videos/the-lesser-known-stars-of-the-tidyverse/>

https://github.com/robinsones/robinsones__blog/blob/master/content/post/multipleChoiceResponses.csv

https://github.com/robinsones/robinsones__blog/blob/master/content/post/2018-11-16-the-lesser-known-stars-of-the-tidyverse.Rmd

Chapter 80

**I “only” use R for descriptive stats —
and that’s OK**

<https://rforeval.com/descriptive-stats-r/>

Chapter 81

histograms

<http://tinlizzie.org/histograms/>

SEER China vs others

<https://www.rdocumentation.org/packages/bayesTFR/versions/6.1-2/topics/country.names>

<https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/state.html>

Chapter 82

Who works on SEER

If you want to see the code used in the analysis please click the code button on the right upper corner or throughout the page.

Select from the tabs below.

82.1 Aim

Aim:

82.2 Data retrieval from PubMed using EDirect

Articles are downloaded as `xml`.

At the time of the research the number of articles with ‘SEER Program’[Mesh] formula is “`r dim(SEER_countries)[1]`”.

While helping the preparation of #PBPath Journal Watch (<https://t.co/WiBsJixzlc>) I thought that many SEER ? studies are from China. So using edirect ? and #RStats I draw the attached graph. What do you think? Do Chinese do research on SEER that much? pic.twitter.com/3Op5r9ofbK

— Serdar Balci (?) October 6, 2018

- eurostat

<http://ec.europa.eu/eurostat>

<http://ec.europa.eu/eurostat/data/database>

- eurostat R package

<http://ropengov.github.io/eurostat/>

- Retrieval and Analysis of Eurostat Open Data with the eurostat Package

<https://journal.r-project.org/archive/2017/RJ-2017-019/index.html>

- CheatSheet

https://github.com/rOpenGov/eurostat/blob/master/vignettes/cheatsheet/eurostat_cheatsheet.pdf

<https://github.com/rstudio/cheatsheets/raw/master/eurostat.pdf>

- Searching, downloading and manipulating Eurostat data with R

<http://ropengov.github.io/r/2015/05/01/eurostat-package-examples/>

- Mapping Eurostat information

<https://www.mytinyshinys.com/2017/07/11/eurostat/>

- eurostat-package published

<https://rpubs.com/muuankarski/27120>

- Tutorial (vignette) for the eurostat R package

http://ropengov.github.io/eurostat/articles/eurostat_tutorial.html

Chapter 83

revtools

revtools: Tools to Support Evidence Synthesis

<https://cran.r-project.org/package=revtools>

<https://revtools.net/>

https://revtools.net/user_manual/1_introduction.html

```
data1 <- read_bibliography("my_data.ris")
data2 <- read_bibliography("my_data.bib")

# If the files are in the working directory:
file_names <- list.files()

# Or if they are in a subdirectory:
file_names <- paste0(
  "./raw_data/",
  list.files(path = "./raw_data/")
)

# Then import to a list
data_list <- lapply(
  file_names,
  function(x){read_bibliography(x)}
)

data <- read_bibliography("my_data.ris")

matches <- find_duplicates(
  data = data,
  match_variable = "title",
  group_variable = NULL,
  match_function = "fuzzdist",
  method = "fuzz_partial_ratio",
  threshold = 0
)

data_unique <- extract_unique_references(data, matches)
```


Chapter 84

screen__duplicates

https://revtools.net/user_manual/4_removing_duplicates.html

```
# 1. standalone; load in data in the app
screen_titles()

# 2. the same, but save back to workspace on exit
result <- screen_titles() # ditto,

data <- read_bibliography("my_data.ris") # load in data

# 3. launch the app using data from the workspace
screen_titles(data)

# 4. specify an object to return data to
result <- screen_titles(data)
```


Chapter 85

RefManageR

RefManageR: Straightforward ‘BibTeX’ and ‘BibLaTeX’ Bibliography Management

<https://cran.r-project.org/web/packages/RefManageR/index.html>

Chapter 86

bibtex

bibtex: Bibtex Parser

<https://cran.r-project.org/web/packages/bibtex/index.html>

Chapter 87

DataExplorer

<https://cran.r-project.org/web/packages/DataExplorer/vignettes/dataexplorer-intro.html>

<https://boxuancui.github.io/DataExplorer/>

87.1 File organization best practices

This page summarises how to organize files and analysis before everything gets jumbled up: Setting up a reproducible data analysis workflow in R

Basically they suggest: - using a project and project folder in RStudio for each analysis - using **packrat** as much as possible

`setwd()` and `getwd()` is not necessary when you use projects.

-
- Why should I use the here package when I'm already using projects?

<https://malco.io/2018/11/05/why-should-i-use-the-here-package/>

```
output: rmarkdown::html_vignette
vignette: >
  %\VignetteIndexEntry{All tables examples}
  %\VignetteEngine{knitr::rmarkdown}
  %\VignetteEncoding{UTF-8}
```

87.2 1 Cross tables

Two-way tables are used extensively in healthcare research, e.g. a 2x2 table comparing two factors with two levels each, or table 1 from a typical clinical study or trial

The main functions all take a **dependent** variable - the outcome (maximum of 5 levels) - and **explanatory** variables - predictors or exposures (any number categorical or continuous variables).

87.2.1 1.01 Default

Note, chi-squared warnings will be generated when the expected count in any cell is less than 5. Fisher's exact test can be used as below, or go straight to a univariable logistic regression, e.g. `colon_s %>% finalfit(dependent, explanatory)`

87.2.2 1.02 Add or edit variable labels**87.2.3 1.03 P-value for hypothesis test**

Chi-squared for categorical, Kruskal-Wallis/Mann-Whitney for continuous

87.2.4 1.04 With Fisher's exact test**87.2.5 1.05 Median (interquartile range) instead of mean (standard deviation)**

... for continuous variables.

87.2.6 1.06 Missing values for the explanatory variables

Always do this when describing your data.

87.2.7 1.07 Column proportions (rather than row)**87.2.8 1.08 Total column****87.2.9 1.09 Order a variable by total**

This is intended for when there is only one explanatory variable.

87.2.10 1.10 Label with dependent name

The dependent name cannot be passed directly to the table intentionally. This is to avoid errors when code is copied and the name is not updated. Change the dependent label using the following. The prefix ("Dependent:") and any suffix can be altered.

87.2.11 1.11 Dependent variable with any number of factor levels supported**87.2.12 1.12 Explanatory variable defaults to factor when 5 distinct values****87.2.13 1.13 Keep as continuous variable when 5 distinct values****87.2.14 1.14 Stratified crosstables**

I've been meaning to include support for table stratification for a while. I have delayed for a good reason. Perhaps the most straightforward way to implement stratification is with `dplyr::group_by()`. However, the non-standard evaluation required for multiple strata may confuse as it is not implemented elsewhere in the package (doesn't work with `group_by_()`). This translates to whether variable names are passed in quotes or not. Finally, `dplyr::do()` is planned for deprecation, but there is no good alternative at the moment. Anyway, here is a solution, which while not that pretty, is very effective.

87.3 2 Model tables with `finalfit()`**87.3.1 2.01 Default**

Logistic regression first.

87.3.2 2.02 Hide reference levels

Most appropriate when all explanatory variables are continuous or well-known binary variables, such as sex.

87.3.3 2.03 Model metrics**87.3.4 2.04 Model metrics can be applied to all supported base models****87.3.5 2.05 Reduced model****87.3.6 2.06 Include all models****87.3.7 2.06 Interactions**

Interactions can be specified in the normal way. Formatting the output is trickier. At the moment, we have left the default model output. This can be adjusted as necessary.

87.3.8 2.07 Interactions: create interaction variable with two factors**87.3.9 2.08 Dependent name**

The dependent name cannot be specified directly intentionally. This is to prevent errors when copying code. Re-label using `ff_label()`. The dependent prefix and suffix can also be altered.

87.3.10 2.09 Estimate name**87.3.11 2.10 Digits / decimal places**

Number of digits to round to regression results. (1) estimate, (2) confidence interval limits, (3) p-value. Default is `c(2,2,3)`. Trailing zeros are preserved. Number of decimal places for counts and mean (sd) / median (IQR) not currently supported. Defaults are sensible :)

87.3.12 2.11 Confidence interval type

One of `c("profile", "default")` for GLM models (`confint.glm()`). Note, a little awkwardly, the ‘default’ setting is `profile`, rather than `default`. Profile levels are probably a little more accurate. Only go to default if taking a significant length of time for profile, i.e. data is greater than hundreds of thousands of lines.

For glmer/lmer models (`confint.merMod()`), `c("profile", "Wald", "boot")`. Not implemented for `lm()`, `coxph()` or `coxphlist`, which use default.

87.3.13 2.12 Confidence interval level

Probably never change this :) Note, the p-value is intentionally not included for confidence levels other than 95% to avoid confusion.

87.3.14 2.13 Confidence interval separation

Some like to avoid the hyphen so as not to confuse with minus sign. Obviously not an issue in logistic regression.

87.3.15 2.14 Mixed effects random-intercept model

At its simplest, a random-intercept model can be specified using a single quoted variable. In this example, it is the equivalent of quoting `{r # andom_effect = "(1 | hospital)"`.

87.3.16 2.15 Mixed effects random-slope model

In the example below, allow the effect of age on outcome to vary by hospital. Note, this specification must have parentheses included.

87.3.17 2.16 Mixed effects random-slope model directly from lme4

Clearly, as models get more complex, parameters such as random effect group variances may require to be extracted directly from model outputs.

87.3.18 2.17 Exclude all missing data in final model from univariable analyses

This can be useful if you want the numbers in the final table to match the final multivariable model. However, be careful to include a full explanation of this in the methods and the reason for excluding the missing data.

87.3.19 2.18 Linear regression

87.3.20 2.19 Mixed effects random-intercept linear regression

87.3.21 2.20 Mixed effects random-slope linear regression

87.3.22 2.21 Cox proportional hazards model (survival / time to event)

87.3.23 2.22 Cox proportional hazards model: change dependent label

As above, the dependent label cannot be specified directly in the model to avoid errors. However, in survival modelling the survival object specification can be long or awkward. Therefore, here is the work around.

87.4 3 Model tables manually using ff_merge()

87.4.1 3.1 Basic table

Note `summary_factorlist()` needs argument, `fit_id = TRUE`.

87.4.2 3.2 Complex table (all in single pipe)**87.4.3 3.3 Other GLM models****87.4.3.1 Poisson****87.4.3.2 Gamma****87.4.4 3.4 Weighted regression****87.4.5 3.5 Using base R functions**

Note `ff_formula()` convenience function to make multivariable formula (`y ~ x1 + x2 + x3` etc.) from a `dependent` and `explanatory` vector of names.

87.4.6 3.6 Edit table rows

This can be done as any dataframe would be edited.

87.4.7 3.7 Base model + individual explanatory variables

This was an email enquiry about how to build on a base model. The example request was in a survival context.

87.5 4 Support for complex survey structures via `library(survey)`**87.5.1 4.1 Linear regression**

Examples taken from `survey::svyglm()` help page.

87.5.2 4.2 Binomial example

Note model family needs specified and exponentiation set to `TRUE` if desired.

```
devtools::install_github("ewenharrison/finalfit")
```

87.6 Table 1 - Demographics

```
kable(table1, row.names=FALSE, align=c("l", "l", "r", "r", "r", "r"))
```

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

87.7 Table 2 - Association between tumour factors and 5 year mortality

```
kable(table2, row.names=FALSE, align=c("l", "l", "r", "r", "r", "r"))
```

country	year	type	count
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

87.8 Figure 1 - Association between tumour factors and 5 year mortality

```
devtools::install_github("ewenharrison/finalfit")
```

87.9 Table 1 - Demographics

```
kable(table1, row.names=FALSE, align=c("l", "l", "r", "r", "r", "r"))
```

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

87.10 Table 2 - Association between tumour factors and 5 year mortality

```
kable(table2, row.names=FALSE, align=c("l", "l", "r", "r", "r", "r"))
```


country	year	type	count
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

87.11 Figure 1 - Association between tumour factors and 5 year mortality

87.12 Table 1 - Demographics

```
kable(table1, row.names=FALSE, align=c("l", "l", "r", "r", "r", "r"),
      booktabs=TRUE)
```

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

87.13 Table 2 - Association between tumour factors and 5 year mortality

87.14 Figure 1 - Association between tumour factors and 5 year mortality

```
devtools::install_github("ewenharrison/finalfit")
```

87.15 Table 1 - Demographics

```
kable(table1, row.names=FALSE, align=c("l", "l", "r", "r", "r", "r"))
```

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

87.16 Table 2 - Association between tumour factors and 5 year mortality

```
kable(table2, row.names=FALSE, align=c("l", "l", "r", "r", "r", "r"))
```

country	year	type	count
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

87.17 Figure 1 - Association between tumour factors and 5 year mortality

87.18 Flipping Coin

<https://www.littlemissdata.com/blog/prettytables>

Chapter 88

5 Alternatives to the Default R Outputs for GLMs and Linear Models

https://www.displayr.com/5-alternatives-to-the-default-r-outputs-for-glms-and-linear-models/?utm_medium=Feed&utm_source=Syndication

88.1 Classic Output

88.2 stargazer

88.3 formattable

88.4 flipRegression

88.4.1 Building Online Interactive Simulators for Predictive Models in R

<https://www.displayr.com/building-online-interactive-simulators-for-predictive-models-in-r/>

Chapter 89

Data Science Live Book

<https://livebook.datascienceheroes.com/>
<https://toolbox.google.com/datasetsearch>
<http://archive.ics.uci.edu/ml/index.php>
<http://asdfree.com/>
<https://rstudio-education.github.io/hopr/>

- **What I Wish I Knew When I Started R**

https://www.williamrchase.com/slides/intro_r_anthropology_2018
<https://sbalci.gitbooks.io/pathology-notes/content/pathology-residents/computational-pathology.html>
<http://web.stanford.edu/class/bios221/book/>
https://kbroman.org/minimal_make/
<https://www.gnu.org/software/make/>
https://kbroman.org/minimal_make/
<https://www.datacamp.com/community/tutorials/shell-commands-data-scientist>
<https://moderndive.com/3-viz.html>
<https://www.causeweb.org/cause/ecots/ecots18/breakouts/7>
<https://plotly-book.cpsievert.me/>
<http://r-bio.github.io/01-intro-R/>
<https://www.rdatagen.net/post/by-vs-within/?platform=hootsuite>
<http://www.biomart.org/download.html>
<https://ropensci.org/blog/2018/07/24/educollab-challenges/>
<https://www.datacamp.com/community/tutorials/data-science-pitfalls>
<https://serialmentor.com/dataviz/preface.html>

- https://news.codecademy.com/errors-in-code-think-differently/?utm_source=customer.io&utm_medium=email&utm_campaign=fortnightly_8-1-18&utm_content=ErrorFortnightly
- Data Science Live Book

<https://livebook.datascienceheroes.com/>

- School of Psychology at the University of New South Wales <http://www.compcogscisydney.org/teaching/>
 - Of Minds and Machines <http://www.compcogscisydney.org/mm/>
 - psyr: Using R in Psychological Science <http://www.compcogscisydney.org/psyr/>
 - Perception and Cognition <http://www.compcogscisydney.org/psyc2071/>
 - Learning Statistics with R <http://www.compcogscisydney.org/learning-statistics-with-r/>
 - Computational Cognitive Science <http://www.compcogscisydney.org/ccs/>

- Advanced R

<https://adv-r.hadley.nz/>

- One Page R

<https://togaware.com/onepager/>

- htmlwidgets for R

<http://www.htmlwidgets.org/>

<http://gallery.htmlwidgets.org/>

- Learning R for Clinical Epidemiologists

<http://rpubs.com/michaelmarks/R-Clin-Epi>

- r-tutor

<http://www.r-tutor.com/>

- Statistics Meets Big Data

<http://www.statsoft.org/>

- ModernDive

<https://moderndive.com/>

- Laerd Statistics

<https://statistics.laerd.com/>

- statpages

<http://statpages.info/index.html>

- The R class R programming for biologists

<http://r-bio.github.io/>

- Sosyal Bilimler Araştırmaları İçin R

<https://bookdown.org/connect/#/apps/1531/access>

- R for Psychological Science An introductory resource

<http://compcogscisydney.org/psyr/>

- Jamovi tutorial

<https://datalab.cc/tools/jamovi>

https://www.youtube.com/playlist?list=PLkk92zzyru5OAtc_ItUubaSSq6S_TGfRn

89.1 master course links

Chapter 90

Do More with R

<https://www.infoworld.com/video/series/8563/do-more-with-r>

Chapter 91

Getting Data into R / Veriyi R'a yükleme

91.1 Import Data

91.1.1 Import using RStudio

91.1.2 Import CSV File

91.1.2.1 How to import multiple .csv files at once?

<https://stackoverflow.com/questions/11433432/how-to-import-multiple-csv-files-at-once>

```
temp = list.files(pattern="*.csv")
myfiles = lapply(temp, read.delim)

temp = list.files(pattern="*.csv")
for (i in 1:length(temp)) assign(temp[i], read.csv(temp[i]))

temp = list.files(pattern="*.csv")
list2env(
  lapply(setNames(temp, make.names(gsub("*.csv$", "", temp))),
    read.csv), envir = .GlobalEnv)

# Get the files names
files = list.files(pattern="*.csv")
# First apply read.csv, then rbind
myfiles = do.call(rbind, lapply(files, function(x) read.csv(x, stringsAsFactors = FALSE)))

library(data.table)
DT = do.call(rbind, lapply(files, fread))
# The same using `r # bindlist`
DT = rbindlist(lapply(files, fread))

library(readr)
library(dplyr)
tbl = lapply(files, read_csv) %>% bind_rows()
```

```
data <- read.csv(
  switch(animal,
    "dog" = "dogdata.csv",
    "cat" = "catdata.csv",
    "rabbit" = "rabbitdata.csv")
)
```

91.1.3 Import TXT File

91.1.4 Import Excel File

```
my_data <- read_excel(file.choose())

files <- list.files(pattern = ".xlsx")

data_xlsx_df <- map_df(set_names(files), function(file) {
  file %>%
    excel_sheets() %>%
    set_names() %>%
    map_df(
      ~ read_xlsx(path = file, sheet = .x, range = "H3"),
      .id = "sheet")
}, .id = "file")
```

91.1.4.1 Import Sheets

91.1.5 Import SPSS File

91.1.6 Keep SPSS labels

read.spss komutu ile değer etiketlerini almasını ve bunu liste olarak değil de data.frame olarak kaydetmesini istiyoruz
aktardığımız data.frame'in özellikleri (attr) içinde değişkenlerin etiketleri var, bunları dışarı çıkartıyoruz
elde ettiğimiz data.frame'deki satır isimleri değişkenlerin isimleri oluyor, karşılarında da değişken etiketleri var satır isimlerini de dışarı çıkartıyoruz

Değişken etiketi olanları etiketleri ile diğerlerini olduğu gibi saklıyoruz

son olarak da data.frame'deki sütun isimlerini değiştiriyoruz

Chapter 92

Export Data

92.0.1 Export to SPSS, while keeping labels

R'da `factor` olan label verdiğiniz değişkenleri SPSS ya da diğer istatistik programlarına aktardığınızda bu tanımlamaları korumak işimize yarar. Bunun için `foreign` paketi ile bir `txt` dosyası ve bir `sps` dosyası oluşturuyoruz. SPSS'te `sps` dosyasını açıp kodu çalıştırarak tekrar atanan değerler geri yükleniyor.

<https://twitter.com/WeAreRLadies/status/1034817323922804737>

```
f <- list.files( "my_folder", pattern = "*.csv", full.names = TRUE)
d <- purrr::map_df(f, readr::read_csv, .id = "id")
```

```
m <- lm(mpg ~ qsec + wt, data = mtcars)
broom::tidy(m)
```

Import a Directory of CSV Files at Once Using {purrr} and {readr}

<https://www.gerkelab.com/blog/2018/09/import-directory-csv-purrr-readr/>

```
data_dir %>%
  dir_ls(regex = "\\..csv$") %>%
  map_dfr(read_csv, .id = "source") %>%
  mutate(Month_Year = myd(Month_Year, truncated = 1))
```

<https://suatatan.wordpress.com/2017/10/07/bulk-replacing-turkish-characters-in-r/>

Turkish character sometimes became the menace for the data scientist. To avoid the risks you may want to change it with safe characters. To do that you can use this code:

```
#turkce karakter donusumu
to.plain <- function(s) {

# 1 character substitutions
old1 <- "çğşüöÇĞŞİÜÜ"
new1 <- "cgsiuocgsiou"
s1 <- chartr(old1, new1, s)

# 2 character substitutions
old2 <- c("æ", "ß", "æ", "ø")
new2 <- c("oe", "ss", "ae", "oe")
s2 <- s1
for(i in seq_along(old2)) s2 <- gsub(old2[i], new2[i], s2, fixed = TRUE)

s2
}
df$source=as.vector(sapply(df$source,to.plain))

to.plain(make.names(tolower(names(df))))
```

- Remove all special characters from a string in R?

<https://stackoverflow.com/questions/10294284/remove-all-special-characters-from-a-string-in-r>

```
x <- "a1~!@#%&*(){ }_+:\<>?,./;'[]-="
stringr::str_replace_all(x, "[[:punct:]]", " ")
stringr::str_replace_all(x, "[^[:alnum:]]", " ")

astr <- "Ábcdêãçøàúü"
iconv(astr, from = 'UTF-8', to = 'ASCII//TRANSLIT')

Data <- gsub("[^0-9A-Za-z//' ]","", Data ,ignore.case = TRUE)

Data <- gsub("'", "", Data ,ignore.case = TRUE)
```

Chapter 93

pdftables

https://cran.r-project.org/web/packages/pdftables/vignettes/convert_pdf_tables.html

Chapter 94

tabulizer

Extract Tables from PDFs

<https://github.com/ropensci/tabulizer>

Chapter 95

rio

Import, Export, and Convert Data Files

<https://thomasleeper.com/rio/index.html>

<https://cran.r-project.org/web/packages/rio/vignettes/rio.html>

Chapter 96

read with purrr

R tip: Iterate with purrr's `map_df` function

<https://www.infoworld.com/video/89075/r-tip-iterate-with-purrrs-map-df-function>

Chapter 97

The janitor package

<https://garhtarr.github.io/meatR/janitor.html>

97.1 convert excel number into date

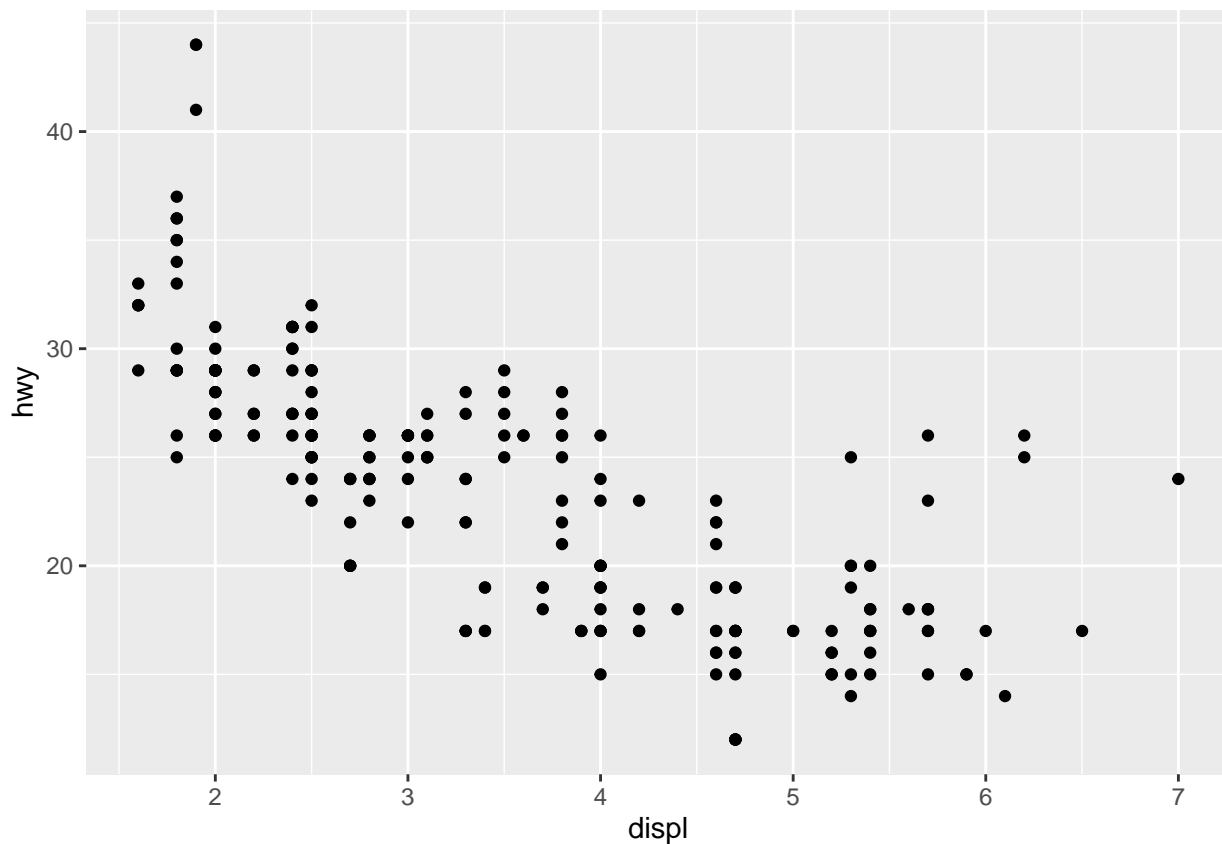
```
output:
  pdf_document: default
  html_document: default
header-includes:
- \usepackage{pdflscape}
- \usepackage{xcolor}
- \newcommand{\blandscape}{\begin{landscape}}
- \newcommand{\elandscape}{\end{landscape}}
```


Chapter 98

ggplot2 —

mpg

```
library("tidyverse")
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy))
```



```
ggplot(mpg, aes(model, manufacturer)) + geom_point()
ggplot(mpg, aes(displ, cty, colour = year)) + geom_point()
ggplot(mpg, aes(displ, hwy)) + geom_point(aes(shape = year))
ggplot(mpg, aes(displ, hwy)) + geom_point() + geom_smooth(span = 0.2)
ggplot(mpg, aes(hwy)) + geom_histogram() + geom_freqpoly()
```

```
ggplot(mpg, aes(cty, hwy)) + geom_point() + geom_smooth()
```

```
ggplot(mpg, aes(class, hwy)) + geom_boxplot() ggplot(mpg, aes(reorder(class, hwy), hwy)) + geom_boxplot()
```

Chapter 99

gganimate —

```
library(gganimate)
p <- ggplot(iris, aes(x = Petal.Width, y = Petal.Length)) + geom_point()
plot(p)
anim <- p + transition_states(Species, transition_length = 2, state_length = 1)
anim
p + enter_appear()
{r # mytext
```

To display the text, type `{r # text_formatted}` outside of the chunk

Chapter 100

ggpubr

<https://rpkgs.datanovia.com/ggpubr>

```
if(!require(devtools)) install.packages("devtools")
devtools::install_github("kassambara/ggpubr")
Distribution
library(ggpubr)
```

```
set.seed(1234)
wdata = data.frame(
  sex = factor(rep(c("F", "M"), each=200)),
  weight = c(rnorm(200, 55), rnorm(200, 58)))
head(wdata, 4)
```

```
ggdensity(wdata, x = "weight",
  add = "mean", rug = TRUE,
  color = "sex", fill = "sex",
  palette = c("#00AFBB", "#E7B800"))
```

```
gghistogram(wdata, x = "weight",
  add = "mean", rug = TRUE,
  color = "sex", fill = "sex",
  palette = c("#00AFBB", "#E7B800"))
```

```
data("ToothGrowth")
df <- ToothGrowth
head(df, 4)
```

```
p <- ggboxplot(df, x = "dose", y = "len",
  color = "dose", palette = c("#00AFBB", "#E7B800", "#FC4E07"),
```

```

      add = "jitter", shape = "dose")

p

# Add p-values comparing groups
# Specify the comparisons you want
my_comparisons <- list( c("0.5", "1"), c("1", "2"), c("0.5", "2") )
p + stat_compare_means(comparisons = my_comparisons)+ # Add pairwise comparisons p-value
  stat_compare_means(label.y = 50)                  # Add global p-value

ggviolin(df, x = "dose", y = "len", fill = "dose",
  palette = c("#00AFBB", "#E7B800", "#FC4E07"),
  add = "boxplot", add.params = list(fill = "white"))+
  stat_compare_means(comparisons = my_comparisons, label = "p.signif")+ # Add significance levels
  stat_compare_means(label.y = 50)                                     # Add global the p-value

data("mtcars")
dfm <- mtcars

dfm$cyl <- as.factor(dfm$cyl)

dfm$name <- rownames(dfm)

head(dfm[, c("name", "wt", "mpg", "cyl")])

ggbarplot(dfm, x = "name", y = "mpg",
  fill = "cyl",                # change fill color by cyl
  color = "white",             # Set bar border colors to white
  palette = "jco",             # jco journal color palett. see ?ggpar
  sort.val = "desc",           # Sort the value in dscending order
  sort.by.groups = FALSE,      # Don't sort inside each group
  x.text.angle = 90            # Rotate vertically x axis texts
)

ggbarplot(dfm, x = "name", y = "mpg",
  fill = "cyl",                # change fill color by cyl
  color = "white",             # Set bar border colors to white
  palette = "jco",             # jco journal color palett. see ?ggpar
  sort.val = "asc",            # Sort the value in dscending order
  sort.by.groups = TRUE,       # Sort inside each group
  x.text.angle = 90            # Rotate vertically x axis texts
)

dfm$mpg_z <- (dfm$mpg - mean(dfm$mpg))/sd(dfm$mpg)

```

```

dfm$mpg_grp <- factor(ifelse(dfm$mpg_z < 0, "low", "high"),
                      levels = c("low", "high"))

head(dfm[, c("name", "wt", "mpg", "mpg_z", "mpg_grp", "cyl")])

ggbarplot(dfm, x = "name", y = "mpg_z",
          fill = "mpg_grp",          # change fill color by mpg_level
          color = "white",          # Set bar border colors to white
          palette = "jco",          # jco journal color palett. see ?ggpar
          sort.val = "asc",          # Sort the value in ascending order
          sort.by.groups = FALSE,    # Don't sort inside each group
          x.text.angle = 90,          # Rotate vertically x axis texts
          ylab = "MPG z-score",
          xlab = FALSE,
          legend.title = "MPG Group"
          )

ggbarplot(dfm, x = "name", y = "mpg_z",
          fill = "mpg_grp",          # change fill color by mpg_level
          color = "white",          # Set bar border colors to white
          palette = "jco",          # jco journal color palett. see ?ggpar
          sort.val = "desc",         # Sort the value in descending order
          sort.by.groups = FALSE,    # Don't sort inside each group
          x.text.angle = 90,          # Rotate vertically x axis texts
          ylab = "MPG z-score",
          legend.title = "MPG Group",
          rotate = TRUE,
          ggtheme = theme_minimal()
          )

ggdotchart(dfm, x = "name", y = "mpg",
           color = "cyl",            # Color by groups
           palette = c("#00AFBB", "#E7B800", "#FC4E07"), # Custom color palette
           sorting = "ascending",    # Sort value in descending order
           add = "segments",         # Add segments from y = 0 to dots
           ggtheme = theme_pubr()    # ggplot2 theme
           )

ggdotchart(dfm, x = "name", y = "mpg",
           color = "cyl",            # Color by groups
           palette = c("#00AFBB", "#E7B800", "#FC4E07"), # Custom color palette
           sorting = "descending",    # Sort value in descending order
           add = "segments",         # Add segments from y = 0 to dots
           rotate = TRUE,            # Rotate vertically
           group = "cyl",            # Order by groups
           dot.size = 6,             # Large dot size
           label = round(dfm$mpg),   # Add mpg values as dot labels
           font.label = list(color = "white", size = 9,
                             vjust = 0.5), # Adjust label parameters
           )

```

```

ggtheme = theme_pubr()                                # ggplot2 theme
)

ggdotchart(dfm, x = "name", y = "mpg_z",
  color = "cyl",                                       # Color by groups
  palette = c("#00AFBB", "#E7B800", "#FC4E07"),      # Custom color palette
  sorting = "descending",                             # Sort value in descending order
  add = "segments",                                   # Add segments from y = 0 to dots
  add.params = list(color = "lightgray", size = 2),   # Change segment color and size
  group = "cyl",                                       # Order by groups
  dot.size = 6,                                       # Large dot size
  label = round(dfm$mpg_z,1),                         # Add mpg values as dot labels
  font.label = list(color = "white", size = 9,
                    vjust = 0.5),                    # Adjust label parameters
  ggtheme = theme_pubr()                             # ggplot2 theme
)+
geom_hline(yintercept = 0, linetype = 2, color = "lightgray")

```

```

ggdotchart(dfm, x = "name", y = "mpg",
  color = "cyl",                                       # Color by groups
  palette = c("#00AFBB", "#E7B800", "#FC4E07"),      # Custom color palette
  sorting = "descending",                             # Sort value in descending order
  rotate = TRUE,                                       # Rotate vertically
  dot.size = 2,                                       # Large dot size
  y.text.col = TRUE,                                  # Color y text by groups
  ggtheme = theme_pubr()                             # ggplot2 theme
)+
theme_cleveland()                                     # Add dashed grids

```

```

print(paste0("Git Update Started at: ", Sys.time()))
CommitMessage <- paste("updated on: ", Sys.time(), sep = "")
wd <- "~/serdarbalci"
setorigin <- "git remote set-url origin git@github.com:sbalci/MyJournalWatch.git \n"
gitCommand <- paste("cd ", wd, "\n git add . \n git commit --message '", CommitMessage, "' \n", setorigin,
system(command = paste(gitCommand, "\n") , intern = TRUE, wait = TRUE)
Sys.sleep(5)
print(paste0("Git Update Ended at: ", Sys.time()))

```

Chapter 101

Happy Git and GitHub for the useR

<https://happygitwithr.com>

- An introduction to Git and how to use it with RStudio

<http://r-bio.github.io/intro-git-rstudio/>

https://andrewbtran.github.io/NICAR/2018/workflow/docs/03-integrating_github.html

<https://abderdeenstudygroup.github.io/studyGroup/lessons/SG-T1-GitHubVersionControl/VersionControl/>

<http://r-bio.github.io/intro-git-rstudio/>

<https://stackoverflow.com/questions/41688164/using-rstudio-to-make-pull-requests-in-git>

<https://bookdown.org/rdpeng/RProgDA/version-control-and-github.html>

<https://www.r-bloggers.com/rstudio-and-github/>

<http://happygitwithr.com/fork.html>

https://kbroman.org/github_tutorial/

https://kbroman.org/simple_site/

- Helping you make your first pull request!

<https://github.com/thisisnic/first-contributions>

101.0.1 scholar

Analyse citation data from Google Scholar: <https://github.com/jkeirstead/scholar/>

101.0.2 coauthornetwork

Exploring Google Scholar coauthorship: <https://cimentadaj.github.io/blog/2018-06-19-exploring-google-scholar-coauthorship/exploring-google-scholar-coauthorship/>

101.1 scholar.shiny

A shiny application that interacts with Google Scholar

<https://github.com/agbarnett/scholar.shiny>

Chapter 102

flatly

Texas Housing Prices: flatly theme

<https://elastic-lovelace-155848.netlify.com/gallery/themes/flatly.html>

Chapter 103

easyalluvial

<https://github.com/erblast/easyalluvial>

https://www.datisticsblog.com/2018/10/intro_easyalluvial/#features

<https://cran.r-project.org/web/packages/easyalluvial/index.html>

Chapter 104

RColorBrewer

How to expand color palette with ggplot and RColorBrewer

<https://www.r-bloggers.com/how-to-expand-color-palette-with-ggplot-and-rcolorbrewer/>

Chapter 105

highcharter

<http://jkunst.com/highcharter/>

<https://github.com/jbkunst/highcharter>

<http://www.htmlwidgets.org/index.html>

<https://cran.r-project.org/web/packages/highcharter/index.html>

<https://www.datacamp.com/community/tutorials/data-visualization-highcharter-r>

`hchart` works like `ggplot2`'s `qplot`.

`hc_add_series` works like `ggplot2`'s `geom_S`.

`hcaes` works like `ggplot2`'s `aes`.

Highmaps - Map Collection

<https://code.highcharts.com/mapdata/>

`download_map_data`: Download the geojson data from the highcharts collection.

`get_data_from_map`: Get the properties for each region in the map, as the keys from the map data.

Chapter 106

taucharts

<https://www.infoworld.com/video/87337/r-tip-how-to-create-easy-interactive-scatter-plots-with-taucharts>



Chapter 107

gganimate

<https://www.infoworld.com/video/89987/r-tip-animations-in-r>

Chapter 108

ggplot2

<http://r-statistics.co/ggplot2-Tutorial-With-R.html>

- <https://ggplot2.tidyverse.org/reference/>

continue from here <http://r-statistics.co/ggplot2-Tutorial-With-R.html>

Chapter 109

gganimate

<https://cran.r-project.org/web/packages/gganimate/vignettes/gganimate.html>

Chapter 110

ggforce

Chapter 111

g2r

```
remotes::install_github("JohnCoene/g2r")
```

```
http://h2o-release.s3.amazonaws.com/h2o/rel-wright/10/docs-website/h2o-r/docs/articles/getting\_started.html
```

```
https://datascienceplus.com/hierarchical-clustering-in-r/
```

```
author: '[Serdar Balcı, MD, Pathologist] (https://sbalci.github.io/)'
```

```
date: "{r # format(Sys.Date())}"
```

```
output:
```

```
  revealjs::revealjs_presentation:
```

```
    incremental: yes
```

```
    theme: sky
```

```
    highlight: pygments
```

```
    center: no
```

```
    smart: yes
```

```
    transition: fade
```

```
    self_contained: yes
```

```
    ig_width: 7
```

```
    fig_height: 6
```

```
    fig_caption: yes
```

```
    reveal_options:
```

```
      slideNumber: yes
```

```
      previewLinks: yes
```

```
prettydoc::html_pretty:
```

```
  theme: leonids
```

```
  highlight: github
```

```
rmrshower::shower_presentation: null
```

```
beamer_presentation:
```

```
  incremental: yes
```

```
  highlight: tango
```

```
html_notebook:
```

```
  fig_caption: yes
```

```
  highlight: kate
```

```
  number_sections: yes
```

```
  theme: flatly
```

```
  toc: yes
```

```
  toc_depth: 5
```

```
  toc_float: yes
```

```
slidy_presentation: null
```

```
pdf_document:
```

```
  toc: yes
```

```
    toc_depth: '5'
html_document:
  fig_caption: yes
  keep_md: yes
  toc: yes
  toc_depth: 5
  toc_float: yes
xaringan::moon_reader:
  lib_dir: libs
  nature:
    beforeInit:
      - macros.js
      - https://platform.twitter.com/widgets.js
    highlightStyle: github
    highlightLines: yes
    countIncrementalSlides: no
  self_contained: yes
ioslides_presentation:
  incremental: yes
  highlight: github
institute: '[serdarbalci.com](https://www.serdarbalci.com)'
editor_options:
  chunk_output_type: inline
```

Chapter 112

How to Prepare Data for Histopathology Research?

Outline

- Why is Data Preparation Important?
 - Do I need a specific Software?
 - What are the Golden Rules?
 - What do I do with Data after analysis?
 - I got all the tables from the biostatistician, is it enough?
 - What is a Good (Clean/Ideal/Tidy) Data?
 - What is a Bad (Dirty/Common/Untidy) Data?
 - Do I need to know statistics before collecting Data?
 - Do I need to have a hypothesis before collecting Data?
 - Do I need a research question before collecting Data?
-

Chapter 113

How to Prepare Data for Histopathology Research?

We Should Collect the Data Related to What We will Report

- Recommendations for reporting histopathology studies: a proposal
- {r # PMID_25846513\$title

{r # citation_25846513

{r # PubMed_25846513

{r # addthis_inline_25846513

{r # PMID_25846513\$abstract

{r # doi_25846513

{r # dimensionBadge_25846513

{r # altmetricBadge_25846513

Chapter 114

Tables and Graphs to be Formed

- Table One: Clinical Features Related to this disease and Histopathological Features (like a CAP synoptic)
 - Cross Tables
 - IHC Tables
 - Survival Tables and Graphs
-

Chapter 115

Age

Chapter 116

Gender

- Male
- Female
- Non-binary (based on research)

For missing values:

{gender}

<https://lincolnmullen.com/software/gender/>

<https://github.com/ropensci/gender>

Chapter 117

Surgery Type
