Using `uwIntroStats`

Authors: Brian D. Williamson and Scott S. Emerson, M.D., Ph.D.
University of Washington Department of Biostatistics

Contents

1 Introduction

2 Preparing uwIntroStats

Before we can dive in and run any analyses, we first need to install the package. This is done via

```
> install.packages("uwIntroStats")
```

Regardless of the graphical user interface (GUI) that you are using, R will prompt you to select a CRAN mirror. It is essentially asking you where you want to download the package files from. Select the mirror closest to you - for us at the University of Washington it is `WA(1)` or the Fred Hutchinson Cancer Research Center (FHCRC) - and the package will download and say that it has installed. Now each time we open a new R session (whether that is at the command line, a new RGui window, or a new RStudio window) we need to load the package for use.

The `uwIntroStats` package relies on five other packages. These other packages provide key functions that the `uwIntroStats` package uses or adds functionality to. We must install these packages like we did above if we have not installed them previously, and then load `uwIntroStats`. While these packages do not need to be loaded every time (in fact, some are only used for specific functions) it is good practice to load them for the R session where you need to use `uwIntroStats`.

```
> library(Exact)
> library(geepack)
> library(plyr)
> library(sandwich)
> library(survival)
> library(uwIntroStats)
```

Last, we load the data, `mri` that we will be using throughout this document. Information about the dataset can be found at http://www.emersonstatistics.com/datasets/mri.pdf. Since the data is part of the package, we can load it via

```
> data(mri)
```

The `uwIntroStats` package should be used for descriptive statistics, basic plotting (like scatterplots and boxplots), and regression analyses. The following sections will go through examples of these tasks, in addition to pointing out how our package differs from base R and other existing packages. We will assume familiarity with basic data manipulation and statistical tasks (for a refresher, see http://www.emersonstatistics.com/GeneralMaterials/R/IntroToR.pdf).

3 Descriptive Statistics

Descriptive statistics are an important part of any analysis. Often, they are just as important for the statistician or data analysist as they are for the scientific collaborators on a project. For example, descriptive statistics can help identify errors in the data, like observations that are particularly unusual (usually far out of the expected or observed range). They can also help to identify patterns of missing data, examine the study population, and identify aspects of the data that might lead to statistical issues. Descriptive statistics can also unearth relationships that had not previously been considered for analysis, thereby generating new studies. More information on this, and an outline of an approach to analysing a dataset, was written by Scott Emerson and can be found at http://www.emersonstatistics.com/GeneralMaterials/analysis.pdf.

3.1 The basics: `descrip()`

The `uwIntroStats` package was designed with descriptive statistics in mind. The basic function for descriptive statistics is `descrip()`. This function takes in an arbitrary number of variables, and by default calculates the number of observations, the number of missing values, the mean, standard deviation, mininum value, maximum value, and the 25th, 50th, and 75th percentiles of each variable. For instance, if we wanted a quick glance at the `mri` data, we could type

```
> descrip(mri)
```

	N	Msng	Mean	Std Dev	Min	25%	Mdn	75%	Max
ptid:	735	0	368.0	212.3	1.000	184.5	368.0	551.5	735.0
mridate:	735	0	76423	31896	10192	66642	80992	91392	1.232e+05
age:	735	0	74.57	5.451	65.00	71.00	74.00	78.00	99.00
male:	735	0	0.4980	0.5003	0.0000	0.0000	0.0000	1.000	1.000
race:	735	0	1.318	0.6659	1.000	1.000	1.000	1.000	4.000
weight:	735	0	159.9	30.74	74.00	138.5	158.0	179.0	264.0
height:	735	0	165.8	9.710	139.0	158.0	165.9	173.2	190.5
packyrs:	735	1	19.60	27.11	0.0000	0.0000	6.500	33.75	240.0
yrsquit:	735	0	9.661	14.10	0.0000	0.0000	0.0000	18.50	56.00
alcoh:	735	0	2.109	4.852	0.0000	0.0000	0.01920	1.144	35.00
physact:	735	0	1.922	2.052	0.0000	0.5538	1.312	2.513	13.81
chf:	735	0	0.05578	0.2297	0.0000	0.0000	0.0000	0.0000	1.000
chd:	735	0	0.3347	0.6862	0.0000	0.0000	0.0000	0.0000	2.000
stroke:	735	0	0.2367	0.6207	0.0000	0.0000	0.0000	0.0000	2.000
diabetes:	735	0	0.1075	0.3099	0.0000	0.0000	0.0000	0.0000	1.000
genhlth:	735	0	2.588	0.9382	1.000	2.000	3.000	3.000	5.000
ldl:	735	10	125.8	33.60	11.00	102.0	125.0	147.0	247.0
alb:	735	2	3.994	0.2690	3.200	3.800	4.000	4.200	5.000
crt:	735	2	1.064	0.3030	0.5000	0.9000	1.000	1.200	4.000
plt:	735	7	246.0	65.80	92.00	201.8	239.0	285.0	539.0
sbp:	735	0	131.1	19.66	78.00	118.0	130.0	142.0	210.0
aai:	735	9	1.103	0.1828	0.3171	1.027	1.112	1.207	1.728
fev:	735	10	2.207	0.6875	0.4083	1.745	2.158	2.649	4.471
dsst:	735	12	41.06	12.71	0.0000	32.00	40.00	50.00	82.00
atrophy:	735	0	35.98	12.92	5.000	27.00	35.00	44.00	84.00
whgrd:	735	1	2.007	1.410	0.0000	1.000	2.000	3.000	9.000
numinf:	735	0	0.6109	0.9895	0.0000	0.0000	0.0000	1.000	5.000
volinf:	735	1	3.223	17.36	0.0000	0.0000	0.0000	0.09420	197.0
obstime:	735	0	1804	392.3	68.00	1837	1879	2044	2159
death:	735	0	0.1810	0.3852	0.0000	0.0000	0.0000	0.0000	1.000

This call gives us a quick look at the distribution of each variable in the sample, and can be easily exported to a word processing software. This is important, because displaying raw R output in a publication is not usually ideal, and taking the time to format the output is a pain.

Notice that there are a lot of variables measured in these data, and that some interesting phenomena are measured by multiple variables. For instance, smoking is measured by `packyrs` and `yrsquit` - if someone has zero pack-years (that is, they have never smoked) then they will also have zero years quit. Similarly, a current smoker of any amount of packs per year will have zero years quit. We can also look at the variables measured through blood samples, like LDL cholesterol (`ldl`) and get an idea of the range in our sample population. Last, in the variables like `male` which consist of only two values (sex was measured as male or female in this dataset), the mean tells us the proportion of our sample who was male (or for other binary variables, it will tell us the proportion which is coded as 1). The `descrip` function works similarly to `summary` in base R, but gives more information and returns it in a format that, as we described above, is easier to export from R.

Another powerful feature of `descrip` is its ability to present stratified summaries, or summaries on a subset of the data. For instance, let's calculate descriptive statistics on the `age` variable within males and females:

```
> descrip(mri$age, strata = mri$male)
                    N     Msng  Mean      Std Dev    Min       25%       Mdn       75%       Max
mri$age:  All       735      0   74.57     5.451     65.00     71.00     74.00     78.00     99.00
mri$age:    Str  0  369      0   74.41     5.258     65.00     71.00     73.00     78.00     91.00
mri$age:    Str  1  366      0   74.73     5.642     66.00     71.00     74.00     78.00     99.00
```

Here the row for `Str 0` corresponds to females (since they are coded as 0 for `male`), and males are `Str 1`. If we further wanted to restrict to only those people who were over 75 years old, we could use the `subset` argument:

```
> descrip(mri$age, strata = mri$male, subset = mri$age > 75)
                    N     Msng  Mean      Std Dev    Min       25%       Mdn       75%       Max
mri$age:  All       260      0   80.59     4.111     76.00     77.75     79.00     83.00     99.00
mri$age:    Str  0  127      0   80.44     3.572     76.00     78.00     79.00     83.00     91.00
mri$age:    Str  1  133      0   80.73     4.576     76.00     77.00     79.00     83.00     99.00
```

However, the descrip function can only give us a fixed number of summaries (mean, count, etc). Oftentimes we need only a subset of these, or perhaps different summary measures. For this, we turn to more powerful functions.

3.2 Flexibility: `tableStat()` and `tabulate()`

The main draw of `tableStat()` and `tabulate()` is their ability to supply only a subset of the summary measures from `descrip()`. These functions focus on display and flexibility. The user supplies the format for the summary statistics to be displayed in, which makes exporting results from a call to `tablestat()` or `tabulate()` even easier than it was for `descrip()`. Of the two, `tableStat()` is the base, and `tabulate()` adds some additional formatting and options. For example, let's say that we want the same summary statistics as we would get from `descrip()`, but we want to control the printout. For

this, we use the `stat` argument. The options are

Name	Summary Measure	Name	Summary Measure
`"count"`	Number of observations on a variable	`"sd"`	Standard Deviation
`"missing"`	Number of missing observations	`"variance"`	Variance
`"mean"`	Arithmetic mean	`"min"`	Minimum value
`"geometric mean"`	Geometric mean	`"max"`	Maximum value
`"median"`	50th percentile	`"quantiles"`	Display quantiles
`"probabilities"`	Display percentiles	`"mn(sd)"`	Display Mean (SD) format
`"range"`	Maximum value - minimum value	`"iqr"`	Interquartile range (75th - 25th percentiles)
`"all"`	Display all summary measures	`"row%"`	Display row percentages
`"col%"`	Display column percentages	`"tot%"`	Display percentage of total

The following gets us close to results we could display in a table for a puplication. We want to have

$$N: \ n; \ Mean \ (SD): \ mn \ (sd); \ Range: \ max - min,$$

which we get via:

```
> tableStat(mri$age, stat = "N: @count@; Mean (SD): @mean@ (@sd@); Range: @max@ - @min@")
Tabled descriptive statistics by strata
Call:
      tableStat(variable = mri$age, stat = "N: @count@; Mean (SD): @mean@ (@sd@); Range: @max@ - @min@")
            - NaN denotes strata with no observations
            - NA arises from missing or censored data

Format:  N: Cnt; Mean (SD): Mean (SD); Range: Max - Min

                                                    .ALL
N: 735.0; Mean (SD): 74.57 (5.451); Range: 99.00 - 65.00
```

Note that all of the values enclosed in `@` symbols are from 3.2 and are run by `tableStat()`, while the other values in the string are used for formatting. This function can also take stratification variables; if we want to see the above stratified by sex, we use

```
> tableStat(mri$age, strata = mri$male, stat = "N: @count@; Mean (SD): @mean@ (@sd@); Range: @min@ - @max@")
Tabled descriptive statistics by strata
Call:
      tableStat(variable = mri$age, strata = mri$male, stat = "N: @count@; Mean (SD): @mean@ (@sd@); Range: @min@ - @max@")
            - NaN denotes strata with no observations
            - NA arises from missing or censored data
```

```
Format:  N: Cnt; Mean (SD): Mean (SD); Range: Min - Max

                                                 strata.0                                                                  strata.1
N: 369.0; Mean (SD): 74.41 (5.258); Range: 65.00 - 91.00 N: 366.0; Mean (SD): 74.73 (5.642); Range: 66.00 - 99.00 N: 735.0; Mean (SD): 74
```

4 Analyses

First, a disclaimer. All of the following analyses are for teaching. This is not the way that the authors recommend performing an analysis. We do emphasize heavily the need to prespecify all analyses in any setting, in order to have correct reproduceability and error rates. The two functions we will use in these analyses are `regress` and `lincom`. To learn more about either of these functions, type

```
> ?regress
> ?lincom
```

From these help files, we learn that the minimum we need to enter into `regress` is a functional, a formula, and a dataset. A functional takes an object and returns a value; for instance, the mean is a functional because it takes a distribution and returns the mean. The allowed functionals for `regress` are:

Functional	Type of Regression
`"mean"`	Linear Regression
`"geometric mean"`	Linear Regression on logarithmically transformed Y
`"odds"`	Logistic Regression
`"rate"`	Poisson Regression
`"hazard"`	Proportional Hazards Regression

4.1 Analysis 1

A natural question is, "Do males have lower cerebral atrophy score than females"? To answer this question, we run a linear regression of atrophy on the variable male:

```
> regress("mean", atrophy~age, data=mri)
Call:
regress(fnctl = "mean", formula = atrophy ~ age, data = mri)

Residuals:
    Min      1Q  Median      3Q     Max
-36.870  -8.589  -0.870   7.666  51.203

Coefficients:
                Estimate  Naive SE  Robust SE    95%L      95%H         F stat    df Pr(>F)
```

```
[1] Intercept     -16.06     6.256     6.701       -29.22    -2.907          5.75 1    0.0168
[2] age            0.6980   0.08368   0.09002       0.5213    0.8747        60.12 1  < 0.00005

Residual standard error: 12.36 on 733 degrees of freedom
Multiple R-squared:  0.08669,        Adjusted R-squared:  0.08545
F-statistic: 60.12 on 1 and 733 DF,  p-value: 2.988e-14
```