Tutorial on anoint package

FDA PACES Workshop, June 25, 2013

Stephanie Kovalchik
Research Fellow, National Cancer Institute

> “`anoint`, for ANalysis Of INTeractions, is a package written in the `R` language, which provides a suite of tools for investigating heterogeneity of treatment effect in a clinical trial.”

Description of `anoint` Package

- Open-source software
- Written in the R language
- Provides methods for assessing heterogeneity of treatment effect, including:
  - Proportional interactions modeling
  - Unrestricted multiple interaction regression
  - Conventional subgroup analyses
  - Forest plots

First...A Very Brief Introduction to R

- R is a statistical programming environment
- It is maintained by the R Development Core Team
- It can be run on Windows, Linux, and Mac OS platiforms
- It can be downloaded from http://cran.r-project.org/

You Can Use R as a Calculator

```
(1:3)^2
```

```
## [1] 1 4 9
```

```
(1:3) * (4:6)
```

```
## [1]  4 10 18
```

You Can Use R to Store & Manipulate Data

```
object <- data.frame(x = 1:6, y = rep(1, 6))

object
```

```
##   x y
## 1 1 1
## 2 2 1
## 3 3 1
## 4 4 1
## 5 5 1
## 6 6 1
```

```


object[1, ]
```

```
##   x y
## 1 1 1
```

You Can Use R to Import Data

```
thedata <- read.csv("filename")
```

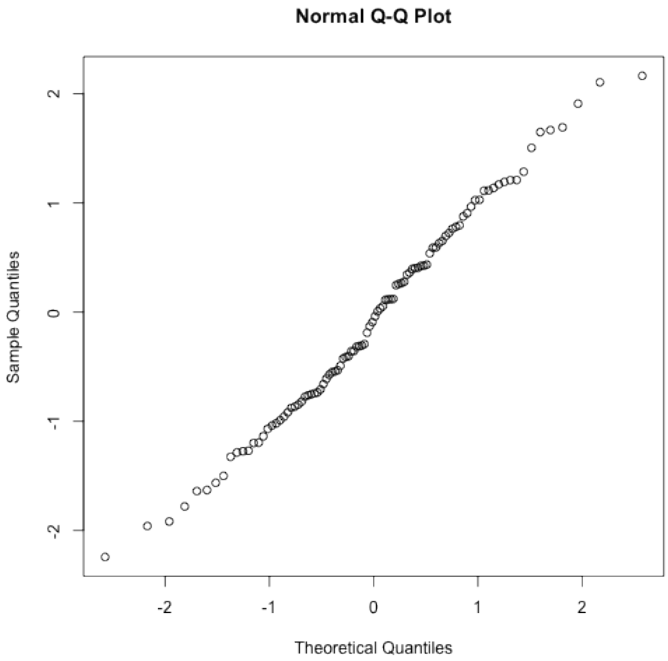
You Can Use R to Perform Statistical Analyses of All Kinds

```
quantile(object$x)
```

```
##   0%  25%  50%  75% 100%
## 1.00 2.25 3.50 4.75 6.00
```

You Can Use R to Perform Statistical Analyses of All Kinds

```
qqnorm(rnorm(100))
```



You Can Extend R by Writing Functions

```
is.missing <- function(x) NA %in% x
```

```
is.missing(object$x)
```

```
## [1] FALSE
```

```
sapply(object, is.missing)
```

```
##     x     y
## FALSE FALSE
```

You Can Also Extend R by Installing Contributed Packages

- The rapid development of R is owing to the contributions of a growing community of programmers (mostly academics), who are willing to share their work with others.
- After passing some quality checks, these packages are posted to the Comprehensive R Archive Network (CRAN), which can be accessed directly from the R environment.

```
install.packages("package-name")
```

Getting Started

1. Install `R` (http://cran.r-project.org/)
2. Install `anoint` package from CRAN (Version 1.3)

- Run `R`
- If you have an active web connection, use the following code:

```
install.packages("anoint")
```

- If needed, install any dependent packages

Getting Started with `anoint`

All sessions with the `anoint` begin by loading the package.

```
library(anoint)
```

- `anoint`'s dependent packages will need to be installed

Getting Help

An index of all of the functions of `anoint` can be obtained by using the `help` function.

Click the link for any function to bring up documentation describing its use.

```
help(package = "anoint")
```

Getting Help

You can also use `?' as a shortcut to a function's documentation.

```
`?`(anoint)
```

Overview of `anoint` package

Function	Description
`anoint`	Creates an analysis of interactions object
`obo`	Extract one-by-one subgroup analyses from `anoint` object
`uim`	Extract unrestricted multiple interaction regression from `anoint` object
`pim.subsets`	Perform all subsets proportional interactions modeling
`pim.fit`	Fit a specific proportional interactions model
`forest.subsets`	Forest plot of all subsets procedure

Examples with bootstrap SOLVD-T

We will demonstrate the major tools of the `anoint` package with a bootstrap SOLVD-T dataset.

If this is located in the current directory, it can be loaded into the `R` session with the function `load`.

```
setwd("~/master/project/anoint/tutorial")  # On My System

load("solvd.RData")
```

Description of SOLVD-T Dataset

```
str(solvd)  # Show Structure
```

```
## 'data.frame':    2569 obs. of  10 variables:
##  $ enal      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ deathdays : num  723 835 1547 1051 1449 ...
##  $ death     : num  0 0 0 0 0 1 1 0 0 0 ...
##  $ vaso      : int  0 0 1 0 1 0 0 1 0 1 ...
##  $ ischemia  : int  1 1 1 1 1 0 1 1 0 1 ...
##  $ sodium    : int  144 140 142 140 135 143 139 136 142 136 ...
##  $ lvef      : int  35 28 24 26 24 15 17 35 22 16 ...
##  $ nyha      : int  2 2 3 3 3 2 2 2 3 2 ...
##  $ sodium.cat: Ord.factor w/ 3 levels "<140"<"140-141"<..: 3 2 3 2 1 3 1 1 3 1 ...
##  $ lvef.cat  : Ord.factor w/ 3 levels "<23"<"23-29"<..: 3 2 2 2 2 1 1 3 1 1 ...
```

Description of SOLVD-T Variables

- `enal`: Indicator of enalapril treatment group (1=enalapril, 0=placebo)
- `deathdays`: Days to death or last date of follow-up if still alive
- `death`: Indicator of death
- `vaso`: Indicator of prior vasodilator use (1=use, 0=no use)
- `ischemia`: Indicator of ischemic cause of congestive heart failure
- `sodium`: Sodium level (mmol/liter)
- `lvef`: Left ventricular ejection fraction (as percent)
- `nyha`: NYHA prognostic class (1 - 4, 1=best and 4=worst)

Syntax of `anoint` Object

```
anoint(formula, data, family, ...)
```

Argument	Description
`formula`	Model formula
`data`	Data frame containing formula variables
`family`	coxph or any of `glm` families, i.e. gaussian, binomial, etc.

`anoint` Formula Specification

The `formula` used to create an `anoint` object requires

- a response (appropriate for the model)
- covariates
- a treatment variable

Example: `anoint` Formula Specification

Suppose:

- response = `y`
- covariates = `a, b, c`
- treatment variable = `trt`

```
formula = y ~ (a + b + c) * trt
```

Example: anoint

```
fit <- anoint(
    formula = Surv(deathdays, death) ~ (vaso + ischemia + lvef + sodium + nyha) * enal,
    data = solvd,
    family = "coxph")
```

Example: `anoint`

```
class(fit)
```

```
## [1] "anoint"
## attr(,"package")
## [1] "anoint"
```

Methods for `anoint` Object

Method	Description
`print`	Shows formula
`summary`	Shows formula, covariates
`obo`	Returns results of one-by-one subgroup analyses
`uim`	Returns unrestricted interaction model

Example: `print` Method for `anoint`

```
fit
```

```
## Surv(deathdays, death) ~ (vaso + ischemia + lvef + sodium + nyha) *
##     enal
```

The `obo` Method for `anoint`

Returns a list with the following named components:

- `fit`: list of model objects; length equal to the number of covariates
- `LRT`: vector of likelihood ratio test (LRT) statistics for interaction
- `pvalue`: vector of unadjusted p-values for LRTs

Example: obo Method

```
subgroups <- obo(fit)

names(subgroups)
```

```
## [1] "fit"    "LRT"    "pvalue"
```

Example: obo Method

```
subgroups$LRT
```

```
##     vaso ischemia     lvef   sodium     nyha
##   2.4068   1.4669   4.1930   0.2081   3.7393
```

```
subgroups$pvalue
```

```
##     vaso ischemia     lvef   sodium     nyha
##  0.12081  0.22584  0.04059  0.64828  0.05315
```

Example: obo method

```
subgroups$fit[[1]]  ## Cox model for vasodilator
```

```
## Call:
## coxph(formula = f, data = anoint@data)
##
##
##                coef exp(coef) se(coef)       z     p
## vaso       -0.00504     0.995   0.0902 -0.0559 0.960
## enal       -0.17119     0.843   0.0968 -1.7678 0.077
## vaso:enal  0.20119     1.223   0.1297  1.5516 0.120
##
## Likelihood ratio test=5.45  on 3 df, p=0.142  n= 2569, number of events= 966
```

Example: obo method

```
betas <- sapply(subgroups$fit, function(x) x$coef[1])  ## Covariate effects in placebo group
betas
```

```
##         Surv(deathdays, death) ~ vaso *  enal.vaso
##                                          -0.005044
## Surv(deathdays, death) ~ ischemia *  enal.ischemia
##                                          -0.057539
##         Surv(deathdays, death) ~ lvef *  enal.lvef
##                                          -0.047876
##     Surv(deathdays, death) ~ sodium *  enal.sodium
##                                          -0.040647
##         Surv(deathdays, death) ~ nyha *  enal.nyha
##                                           0.549418
```

Example: obo Method

```
interactions <- sapply(subgroups$fit, function(x) x$coef[3])  # Interaction
interactions
```

```
##         Surv(deathdays, death) ~ vaso *  enal.vaso:enal
##                                                0.201191
## Surv(deathdays, death) ~ ischemia *  enal.ischemia:enal
##                                                0.174479
##         Surv(deathdays, death) ~ lvef *  enal.lvef:enal
##                                                0.019561
##     Surv(deathdays, death) ~ sodium *  enal.sodium:enal
##                                                0.009408
##         Surv(deathdays, death) ~ nyha *  enal.nyha:enal
##                                               -0.189664
```

Example: obo Method

```
interactions/betas + 1  # implied proportional effects
```

```
##         Surv(deathdays, death) ~ vaso *  enal.vaso:enal
##                                                -38.8847
## Surv(deathdays, death) ~ ischemia *  enal.ischemia:enal
##                                                -2.0324
##         Surv(deathdays, death) ~ lvef *  enal.lvef:enal
##                                                 0.5914
##     Surv(deathdays, death) ~ sodium *  enal.sodium:enal
##                                                 0.7685
##         Surv(deathdays, death) ~ nyha *  enal.nyha:enal
##                                                 0.6548
```

The `uim` Method for `anoint`

Returns a list with the same named components as `obo`:

- `fit`: fitted model object
- `LRT`: value of global likelihood ratio test (LRT) for any interaction
- `pvalue`: value of corresponding LRT

Example: `uim` Method

```
unrestricted <- uim(fit)  # Cox multiple interaction model

names(unrestricted)
```

```
## [1] "fit"    "LRT"    "pvalue"
```

Example: `uim` Method

```
unrestricted$LRT
```

```
## [1] 7.693
```

```
unrestricted$pvalue
```

```
## [1] 0.174
```

Example: `uim` Method

```
unrestricted$fit
```

```
## Call:
## coxph(formula = object@formula@formula, data = object@data)
##
##
##                    coef exp(coef) se(coef)         z       p
## vaso          -3.15e-02    0.9690  0.09271 -0.339303 7.3e-01
## ischemia       1.16e-05    1.0000  0.10174  0.000114 1.0e+00
## lvef          -3.82e-02    0.9626  0.00677 -5.636172 1.7e-08
## sodium        -4.07e-02    0.9601  0.01434 -2.837351 4.5e-03
## nyha           4.64e-01    1.5897  0.06782  6.835390 8.2e-12
## enal          -2.45e+00    0.0863  2.89633 -0.845774 4.0e-01
## vaso:enal      1.73e-01    1.1889  0.13286  1.302541 1.9e-01
## ischemia:enal  1.44e-01    1.1547  0.14823  0.970554 3.3e-01
## lvef:enal      9.85e-03    1.0099  0.00974  1.010600 3.1e-01
## sodium:enal    1.65e-02    1.0167  0.02062  0.801963 4.2e-01
## nyha:enal     -1.44e-01    0.8655  0.09909 -1.457243 1.5e-01
##
## Likelihood ratio test=154  on 11 df, p=0  n= 2569, number of events= 966
```

Example: `uim` Method

```
betas <- unrestricted$fit$coef[1:5]  # covariate placebo effects
betas
```

```
##       vaso   ischemia       lvef     sodium       nyha
## -3.146e-02  1.157e-05 -3.815e-02 -4.069e-02  4.636e-01
```

Example: `uim` Method

```
interactions <- unrestricted$fit$coef[7:11]  # interaction effects
interactions
```

```
##     vaso:enal ischemia:enal     lvef:enal   sodium:enal     nyha:enal
##      0.173049      0.143865      0.009846      0.016537     -0.144392
```

Example: `uim` Method

```
interactions/betas + 1  # implied proportional effect
```

```
##     vaso:enal ischemia:enal     lvef:enal   sodium:enal     nyha:enal
##       -4.5014    12438.0717        0.7419        0.5936        0.6885
```

Proportional Interactions All Subsets

- We see from the subgroup and unrestricted interaction effects that there is some suggestion of proportionality among the SOLVD-T candidate effect modifiers.
- To investigate this formally, we can use the function `pim.subsets`