Package ‘BERG’

October 2, 2018

Type Package

Title A set of standardized functions for ACS BERG analyses

Version 1.0.2

Author Brian Carter

Maintainer Brian Carter <brian.carter@cancer.org>

Description A set of standardized functions and program documentation.

License What license is it under?

Encoding UTF-8

LazyData true

Imports metafor, dplyr, survival, rms, xlsx, officer

R topics documented:

`ageStdCrossTab` *Age-standardized frequencies*

Description

This function calculates age-standardized cross-tab frequencies for two variables

Usage

```
ageStdCrossTab(dat,mar=2,agedist,age,var,strata)
```

Arguments

`dat`	data frame used for the analysis
`mar`	1= row percents, 2=column percents (default)
`agedist`	Age-specific standardized weights. See documentaion on ?nutweightlist for details
`age`	Character vector for your age at baseline variable. This could be "AGE_INT" or "AGE92M", etc.
`var`	Character vector for the primary frequency variable (typically formatted as the row variables)
`strata`	Character vector for the strata variable for which you need frequencies (typically formatted as your column variable)

Value

Data frame including the only the age-adjusted row/column percents

Author(s)

Maret Maliniak

See Also

`nutweightlist`

Examples

```
df <- example.data
mytable <- ageStdCrossTab(dat=df,
                          mar=2,
                          agedist="YEARS00",
                          age="age92m",
                          var="bmicat92",
                          strata="sex")
```

ageStdFreqs *Age-standardized frequencies*

Description

This function calculates age-standardized one-variable frequencies. A separate function is available to produce age-standardized 2x2 tables of two variables, which is probably more useful for typical analyses.

Usage

```
ageStdFreqs(dat,agedist,age,var)
```

Arguments

`dat`	data frame used for the analysis
`agedist`	Age-specific standardized weights. See documentaion on ?nutweightlist for details
`age`	Character vector for your age at baseline variable. This could be "AGE_INT" or "AGE92M", etc.
`var`	Character vector for the variable for which you need frequencies

Value

Data frame including the raw frequencies and proportions, and a age-standardized proportion.

Author(s)

Maret Maliniak

See Also

`nutweightlist` `ageStdCrossTab`

Examples

```
df <- example.data

mytable <- ageStdFreqs(dat=df,
                       agedist="YEARS00",
                       age="age92m",
                       var="bmicat92")

# Run multiple frequencies using lapply()

manytables <- lapply(c("bmicat92","smoke92"),function(x){
     ageStdFreqs(dat=df,
                 agedist="YEARS00",
                 age="age92m",
                 var=x)
                 })
# create a single table from the results
```

```
final.frequencies <- Reduce(function(x,y) rbind(x,y), manytables)
```

---

bergMeta *Simple meta-analysis function using the metafor package*

Description

The metafor package does all the hard work for metaanalyses. This function simply takes a standardized input of relative risks and confidence intervals and calculates pooled estimates using a variety of methods. Also computes study heterogeneity statistics.

Usage

```
bergMeta(dat,method="DL")
```

Arguments

`dat`	A data frame. Must have THREE variables: RR = relative risk estimates, LL=lower bound confidence limit, and UL=upper bound confidence limit. The data frame can have anything else in it, as long as those three variables are present.
`method`	Random effects estimates method. The metafor package provides several options: "DL" - DerSimonian-Laird - default "HS" - Hunter-Schmidt "SJ" - Sidik-Jonkman "ML" - Maximum-likelihood "REML" - restricted maximum-likelihood

Details

bergMeta() calculates fixed and random effect spooled estimates using the rma() function in the metafor package. It is written specifically for pooling relative risks. The user should use this function if they are interested in quick results. The metafor package contains a wide variety of methods for meta-analysis. The bergMeta() function simply wraps the most standard methods and formats the results. Users should refer to the metafor functions if they wish to do more advanced meta-analyses; however, this should work for most purposes.

Value

A data frame with fixed and random effects estimates and an I2 statistic.

Author(s)

Brian Carter

See Also

`rma`, ~~~

Examples

```
myRR <- data.frame(name=c("Brian","James","Maret"),
                   year=c("2000","2001","2002"),
                   RR=c(2.0,2.3,2.5),
                   LL=c(1.8,2.0,2.2),
                   UL=c(2.2,2.5,2.7))
foo <- bergMeta(myRR,"DL")
```

`contents` *Creates and prints PROC CONTENTS style output*

Description

The contents() function creates two tables similar to those that are generated by SAS PROC CONTENTS: 1) Summary table with number of observations, rows, and file size; 2) A list of all variables (alphabetically or by variable number) with the option to write them to an excel sheet or an RTF file.

Usage

```
contents(data, varnum = T, write = NULL, outfile = NULL, filename = NULL)
```

Arguments

`data`	dataframe used in the analysis
`varnum`	boolean input controlling order or variable list - set to false for alphabetical variable list.
`write`	OPTIONAL - character string either "word" or "excel" - controls whether to write to file and which file type.
`outfile`	File path for output file folder. Required if write argument is specified.
`filename`	File name for output file - DO NOT include file type extension. Required if write argument is specified.

Details

Generates two tables akin to output from SAS Proc Contents - 1 with summary information (i.e. name of dataframe, number of observations, and number of variables) and a second with a list of variables. Includes option to order the variable table alphabetically or by variable number (default, equivalent to the index value). Includes additional option to output to a word doc or excel file allowing the user to specify a file path. In the current iteration, the file path and file name must be specified separately in the outfile and filename arguments.

Dependent packages: dplyr, xlsx, officer

Value

The function returns a 2 level list with the following tables:

`metadata`	4 line table with dataframe name, number of observations, number of variables and object size (in bytes)
`var_table`	Table showing all variables in dataframe along with factor levels, and labels (if pulled in from SAS)

Author(s)

James Hodge

Examples

```
#-------------- Examples using base data set mtcars

# Default usage will sort variables by index number
test1 <- contents(data = mtcars)
test1$metadata
test1$var_table

# Changing the varnum option will order variables alphabetically
test2 <- contents(data = mtcars, varnum = F)
test2$metadata
test2$var_table
```

`cox_models` *Cox proportional hazards models and documentation*

Description

This function runs age- and multivariable-adjusted cox proportional hazards models. The final results are compiled into a single table and all documentation is organized and prepared for program review. Age-standardized rates will also be calculated for the exposure. Additionl functions are available for interaction and stratified models.

Usage

```
cox_models(dat,start,stop,outcome,expo,birthday,covariates=NULL,agedist,agegrps=NULL)
```

Arguments

`dat`	Data frame containing the data for the analysis
`start`	Character vector naming your start of followup time, can be numeric or date format. Typically this is one of our DTINT variables. For time dependent or late-entry models, this will be the name of your start time variable.
`stop`	Character vector naming your end of followup time, can be numeric or date format. Typically this is one of our DATEFT or DATEDD variables. For time dependent or late-entry models, this will be the name of your stop time variable.
`outcome`	Censor variable name (numeric), formatted 0=control, 1=case
`expo`	Exposure variable name. Exposure can be either continuous (numeric) or categorical (factor)
`birthday`	Variable name for exact birth date. Typically this is our "BDAYDATE" variable from the master file.
`covariates`	Vector of covariates used in the analysis. Can be a mixture of numeric/character/factor variables. No requirements on how they are coded. Note: missing values in the covariates will drop people from the multivariable models.
`agedist`	Age distribution standard used for the rate analysis (see nutweightlist documentation for details)

`agegrps`	Vector of start ages for the analysis. Default is 5-year age groups starting at age 40-44, 45-59,... 85+. To collapse ages, provide a vector of the ages you want each group to start with. For example, if you wanted to use 10-year age groups you would provide: c(40,50,60,70,80) to create age groups 40-49, 50-59, 60-69, 70-79, 80+. For 2 age groups (<65 and 65+) you would provide the vector c(40,65).

Details

The cox_models() function automates the coxph() function found in the survival package and will run standard, time-dependent, or late-entry models depending on how the input data frame is constructed. All models are stratified on single year of age at interview. The function will run two sets of models with the following formulae:

Age-adjusted models: coxph(Surv(START,STOP,OUTCOME)~EXPOSURE + strata(BASELINE_AGE))

Multivariable-adjusted models: coxph(Surv(START,STOP,OUTCOME)~EXPOSURE + covariates + strata(BASELINE_AGE))

The final output list will contain a data frame organizing all of the results that is suitable for delivery to your PI. It will also contain all the model output for the age-adjusted and multivariable-adjusted models required for checking your work while doing the analysis and program review.

Value

The function outputs a 4-level list including all the output from the analysis.

`final`	A data frame containing the organized output of the function. Includes exposure name, categories, case numbers, person years, standardized rates, age-adjusted estimates and p-values, and multivariable adjusted estimates and p-values
`rates`	Output from the rate analysis. Multilevel list containing the age-specific person-years/events/rates for each level of the exposure as well as a rates$Std.Rates that includes summarized results. These summarized results are used in the $final data.frame
`age`	age-adjusted model output documentation
`multi`	multivariable-adjusted model output documentation

Author(s)

Brian Carter

See Also

coxph, incrate

Examples

```
df <- example_data

foo <- cox_models(dat=df,
                  start="dtint92",
                  stop="dateft",
                  outcome="dead",
                  expo="smoke92",
                  birthday="bdaydate",
                  covariates=NULL,
                  agedist="YEARS00",
                  agegrps=c(40,65))

print(foo$final)
```

`documentation` *Documents number of observations and variables for merging datasets*

Description

The documentation() function replicates some of the output found in the SAS log. When loading or merging datasets, calling documentation() on any number of data frames will return the number of observations and number of variables. This can be compared to the final merged dataset to spot-check that no variables or observations were dropped.

Usage

```
documentation(...)
```

Arguments

`...`	data frames requiring documentation. The user can insert as many dataframes as necessary to check, separated by a coma.

Author(s)

Brian Carter and James Hodge

Examples

```
mydata1 <- data.frame(x=1:5,y=x^2)
mydata2 <- data.frame(a=letters[1:10],b=rnorm(10,5))
document(mydata1,mydata2)
```

`dxdateClean` *Cleans up diagnosis dates - 1997-2013*

Description

This standard code that cleans up diagnosis dates based on date of interview for each of the followup surveys. If diagnosis dates are <180 after interview date, the date is recoded to DTINT-1. If the diagnosis dates are >180 days after interview date, the date is recoded to 1900-01-01 and will be excluded from analyses or flagged for further review.

Usage

```
dxdateClean(data,d)
```

Arguments

`data`	Data frame used in the analysis
`d`	Date of diagnosis variable

Value

Cleaned diagnosis date is returned, named identically to the one given to the function.

Author(s)

Brian Carter

`example_data` *This is an example dataset useful for demonstrating the functions in the BERG package.*

Description

The example_data data frame uses data from a recent analysis of BMI and myeloid leukemias. The IDs and dates are fake, so please don't merge it with any existing data or believe and results that come out of it. The data are only useful for learning the various BERG functions

Usage

```
df <- example_data
```

Format

A data frame with 152,046 observations on the following 15 variables.

`racenew` Factor w/ 3 levels "White","Black",..: 1 1 1 1 1 1 1 1 3 2 ...

`age92m` num 60 65 61 68 68 65 58 66 66 62 ...

`sex` Factor w/ 2 levels "Men","Women": 1 1 1 1 1 1 1 1 1 1 ...

`bmicat92` Factor w/ 3 levels "18.5-24.9","25-29.9",..: 1 1 1 2 1 2 2 1 1 2 ...

`bmi92` num 23.7 24.8 20.3 25.4 23.2 ...

`smoke92` Factor w/ 5 levels "Never","Current",..: 3 3 1 3 3 1 3 3 1 3 ...

`dead` num 0 0 0 1 1 1 1 1 0 0 ...

`myeloid` num 0 0 0 0 0 0 0 0 0 0 ...

`bdaydate` Date, format: "1931-01-01" "1927-01-01" "1931-01-01" "1924-01-01" ...

`dtint92` Date, format: "1992-01-01" "1992-01-01" "1992-01-01" "1992-01-01" ...

`dateft` Date, format: "2010-03-26" "2013-03-28" "2007-01-08" "2007-10-31" ...

`AML` num 0 0 0 0 0 0 0 0 0 0 ...

`CML` num 0 0 0 0 0 0 0 0 0 0 ...

`educ` Factor w/ 4 levels "High school or less",..: 3 3 3 3 3 3 3 3 3 1 ...

`id` Character vector similar to our ID variable `incrate` *Age standardized rate analysis*

Description

This function calculates age-standardized incidence and mortality rates for CPSII Mortality and Nutrition Cohorts analyses.

Usage

```
incrate(dat=df, agedist, agegrps=NULL, dtint, birthday,
                    failtime, outcome, expo)
```

Arguments

`dat`	data.frame used in the analysis (default is "df" for use in modeling functions)
`agedist`	Age distribution standard (see nutweightlist documentation for details)
`agegrps`	Vector of start ages for the analysis. Default is 5-year age groups starting at age 40-44, 45-59,... 85+. To collapse ages, provide a vector of the ages you want each group to start with. For example, if you wanted to use 10-year age groups you would provide: c(40,50,60,70,80) to create age groups 40-49, 50-59, 60-69, 70-79, 80+. For 2 age groups (<65 and 65+) you would provide the vector c(40,65).
`dtint`	Variable name for the start of followup. Example: "dtint92", "dtint97", etc.
`birthday`	Variable name exact birthdate, typically "BDAYDATE" in our master file.
`failtime`	Variable name for fail time in days
`outcome`	Variable name for your outcome, formatted 0=control, 1=case
`expo`	Factor variable for your exposure.

Details

These rates are calculated using the pyrs() function in the survival package. Age-specific time and events are tabulated as each CPSII participant moves through calendar time. For example, if a participant enrolls in the study at age 49, they will contribute person time to the age 45-49 age group until their 50th birthday, and then start contributing person time to the 50-54 age group. Events are assessed at this attained age.

The pyrs() function simply calculated age- and strata-specific person years and events. The rest of the function calculates the age-adjustment and final rates.

Value

The function returns a list of age-specific person-years and rates for each exposure level of your variable. Also includes:

`Std.Rates`	Final age-standardized rates calculated for each exposure level

Author(s)

Brian Carter

See Also

weightlist

Examples

```
df <- example.data
df$failtime <- as.numeric(df$dateft)-as.numeric(df$dtint92)
rates <- incrate(dat=df,
                 agedist="YEARS00",
                 agegrps=c(40,65),
                 dtint="dtint92",
                 birthday="bdaydate",
                 failtime="df$failtime",
                 outcome="dead",
                 expo="smoke92")

print(rates$Std.Rate)
```

---

interaction_cox *Runs single referent group cox models using interaction variables.*

Description

The interaction_cox() function runs single referent group interaction cox models. The main interaction variable is coded using the interaction(var1, var2) function.

Usage

```
interaction_cox(dat,start,stop,outcome,expoVar,strataVar,age,covariates=NULL)
```

Arguments

`dat`	data frame used in the analysis
`start`	Character vector naming your start of followup time, can be numeric or date format. Typically this is one of our DTINT variables. For time dependent or late-entry models, this will be the name of your start time variable.
`stop`	Character vector naming your end of followup time, can be numeric or date format. Typically this is one of our DATEFT or DATEDD variables. For time dependent or late-entry models, this will be the name of your stop time variable.
`outcome`	Character vector for the censor variable used in the analysis, must be coded as 0=control, 1=case
`expoVar`	Character vector for our main exposure variable, may be categorical or numeric
`strataVar`	Character vector for our stratification variable, must be categorical
`age`	Character vector for our age variable used for stratifying on single year of age
`covariates`	Character vector of covariates. Default is NULL (age-adjusted analysis only)

Details

The interaction_cox() function will run simple interaction models, format the results into a table, and return all model output in a list format that is suitable for program review. The function will run standard, time-dependent, or late-entry models, depending on how the data are structured prior to running the function

Interaction models are coded using the following formula:

y <- formula(Surv(start,stop,outcome)~ interaction(expoVar,strataVar) + covariates + strata(age))

To calculate a p-interaction, a reduced model is also calculated:

y <- formula(Surv(start,stop,outcome) strataVar + expoVar + covariates + strata(age))

The p-interaction is then calculated:

anova(interaction.model, reduced.model)

Value

The function outputs a 3-level list including all the output from the analysis.

`final`	A data frame containing the organized output of the function. Includes exposure name, categories, case numbers, stratified estimates and p-values, and a p-value for interaction.
`int.model`	All model output for the interaction model
`base.model`	All model output from the base model

Author(s)

Brian Carter

References

G:/Intramural Research/Epidemiology Research/Analysts EPI/Memos, presentations, resources and code/Interaction/Memo, Interaction Methods.doc

See Also

coxph, cox_models, stratified_cox

Examples

```
df <- example_data

foo <- interaction_cox(dat=df,
                       start="dtint92",
                       stop="dateft",
                       outcome="dead",
                       expoVar="smoke92",
                       strataVar="bmicat92",
                       age="age92m",
                       covariates=NULL)
```

jointCox — *Function to run joint-cox proportional hazards models for subtype analyses*

Description

Joint-cox models examine associations with risk factors across multiple disease subtypes. Works with single exposure or time-dependent variables. See details:

Usage

```
jointCox(dat, primarySubtype, otherSubtypes, start, stop, expo, age, idvar, covariates = NULL)
```

Arguments

`dat`	Data frame used in the analysis.
`primarySubtype`	Character vector indicating your primary subtype. This is the outcome for which you want risk estimates. Variable should be coded as 0=noncase, 1=case.
`otherSubtypes`	Character vector indicating the other subtypes. These variables should also be coded as 0=noncase, 1=case. There is no limit on how many subtypes can be included in the model.
`start`	Character vector of the variable name used for the start of followup. Typically this will be one of our date of interview variables, "dtint92","dtint97", etc.
`stop`	Character vector of the end of followup variable. Typically this would be "dateft", "datedd", etc.
`expo`	Character vector of main exposure variable. Must be a factor or numeric variable.
`age`	Character vector for baseline age. Models stratify on single year of baseline age.
`idvar`	Character vector for the ID variable, typically "ID". coxph() uses the cluster(ID) function to compute robust sandwhich variance estimators.
`covariates`	Character vector of covariates required for the analysis.

Details

The function assumes common followup time. The joint-cox model requires that data be restructured as long-form with indicator variables delineating each subtype. This function restructures the data, fits the models, and returns the formatted results and model output. P-tumor heterogeneity is calculated as a liklihood ratio test comparing the interaction joint-cox model to a simple additive base model without interactions with subtype.

The function will only calculate risk estimates for the primarySubtype variable. You will need to rerun the function for each subtype in the analysis.

It is important that you look at the p-heterogeneity calculated for each subtype. If you are modeling the same set of subtypes (and covariates) in each model, the p-heterogeneity should be identical for each set. If they are different, something has gone wrong.

Value

A list of three objects

`final`	Final formatted dataset of estimates
`interaction`	Model output for the joint-cox models
`base`	Model output for the base comparison models (used for p-heterogeneity calculation)

Author(s)

Brian Carter

References

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3565022/

See Also

`prepJoint`

Examples

```
df <- example_data


# Run the function with AML being our primary outcome
aml.results <- jointCox(dat=df,
              primarySubtype="AML",  # Acute myeloid
              otherSubtypes="CML",
              start="dtint92",
              stop="dateft",
              expo="smoke92",
              age="age92m",
              idvar="id",
              covariates = c("racenew","educ","sex"))

# Run with CML being the primary outcome
cml.results <- jointCox(dat=df,
              primarySubtype="CML", # chronic myeloid
              otherSubtypes="AML",
              start="dtint92",
              stop="dateft",
              expo="smoke92",
              age="age92m",
              idvar="id",
              covariates = c("racenew","educ","sex"))

# Bind the results into the final table
final.results <- rbind(aml.results$final,cml.results$final)
print(final.results)
```

prepJoint — *Preps a cohort for a joint-cox regression model*

Description

Joint cox proportional hazards modeling in R requires that data be reformatted as long-form. In short, one dataset is created for each subtype the user wants to model, indicator variables are defined, and then each subtype dataset is rbind() back together into one long dataset. The result is a dataset with duplicates for each original observation. The prepJoint() function simply automates this process.

Usage

```
prepJoint(dat,primarySubtype,otherSubtypes)
```

Arguments

`dat`	Data frame used in the analysis.
`primarySubtype`	Character vector indicating your primary subtype. This is the outcome for which you want risk estimates. Variable should be coded as 0=noncase, 1=case.
`otherSubtypes`	Character vector indicating the other subtypes. These variables should also be coded as 0=noncase, 1=case. There is no limit on how many subtypes can be included in the model.

Value

Returns a dataset structured for a joint-cox regression analysis. The dataset will have duplicated observations for each subtype requested. Two new variables will be added: "OUTCOME" which will match the subtype dummy codes, and "EVENT2" which indicates which subtype is the primary outcome used in the analysis.

Note

This function expands the size of your analytic dataset. It is unlikely that the user will run into memory issues restructuring the data using this function. However, the dataset returned by prepJoint() will be used in the final modeling procedure which can be quite memory intensive. Please consider subsetting your analytic dataset to only the variables you need prior to running the prepJoint() function. Otherwise, you may run into memory overflow issues when fitting the joint cox models.

Author(s)

Brian Carter

References

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3565022/

See Also

`jointCox`

Examples

```
df <- example_data
joint <- prepJoint(df,primarySubtype="AML",otherSubtypes="CML")
```

`prophazCheck` *Check proportional hazards assumption and log-log survival plots.*

Description

prophazCheck quickly checks the proportional hazards assumption and returns a data frame of p-values and log-log survival plots.

Usage

```
prophazCheck(dat,start,stop,outcome,age,expo,outcome.title=NULL,expo.title=NULL)
```

Arguments

`dat`	Data frame used in the analysis
`start`	Character vector of the variable name for your start of followup. Typically this would be "dtint92", "dtint97", etc. Variable can be formatted as a date or numeric variable. It is converted to a numeric variable within the function.
`stop`	Character vector of the variable name for your end of followup. Typically this would be "dateft", "datedd". Variable can be formatted as a date or numeric variable, but is converted to numeric within the function to calculate continuous followup time.
`outcome`	Character vector indicating the variable name for your censor variable. Variable should be coded as a numeric variable, with 0=noncase, and 1=control
`age`	Character vector indicating your baseline age variable. Cox models used the strata() function to stratify on single year of baseline age. Typically this would correspond to variables "AGE_INT", "AGE92M", etc.
`expo`	Character vector indicating your main exposure variable for which you want to assess proportional hazards.
`outcome.title`	Provide a character vector describing your censor variable. This will be used in the plots title. The function defaults to the "outcome" argument, but you can call it anything you want.
`expo.title`	Provide a character vector describing your exposure variable. This will be used in the plots title. The function defaults to the "expo" argument, but you can call it anything you want.

Details

Proportional hazards are checked using the cox_zph() function in the survival package by using the Schoenfeld residuals against transformed time. Log-log survival plots are also calculated and returned.

Value

A list containing 2 objects

pval	Data frame with three variables: exposure variable, outcome variable, and p-value for proportional hazards assumption
plots	log-log survival plots

Author(s)

Brian Carter

See Also

cox.zph, ~~~

Examples

```
foo <- prophazCheck(dat=example_data,
             start="dtint92",
             stop="dateft",
             outcome="myeloid",
             age="age92m",
             expo="bmicat92",
             outcome.title="All myeloid leukemias",
             expo.title="baseline BMI")
foo$plots  # draws plots

# You can save plots using the png() function

png(filename="My LogLog Plots.png", width=5,height=5,units="in",res=400)
foo$plots
dev.off()
```

---

splineFun *Restricted cubic splines in R*

Description

The splineFun() function plots restricted cubic splines adjusted for covariates.

Usage

```
splineFun(dat,expo,covariates=NULL,reference=NULL,knots,
                        start,stop,outcome,agestrat,
                        expo.label=NULL,
                        outcome.text=NULL)
```

Arguments

`dat`	Data frame used for the analysis
`expo`	Character vector of your main exposure variable. Must be a continuous-numeric variable.
`covariates`	Character vector of your list of covariates. Covariates can be a mixture of numeric, character, or factor variables. Default is NULL, splines will only be age adjusted.
`reference`	The spline plot is centered on a reference value. If left NULL, this reference value is computed as the median of your expo variable.
`knots`	Number of knots (must be >=3) calculated using default quantiles of expo. For 3-5 knots, the outer quantiles used are 0.05 and 0.95. For knots > 5, the outer quantiles are 0.025 and 0.975. The remaining knots are equally spaced between these outer quantiles.
`failtime`	Character vector of your failtime variable, must be coded as numeric.
`outcome`	Outcome variable. Must be numeric and coded as 0=control, 1=case
`agestrat`	Character vector of continuous age. BERG analyses stratify Cox models on single year of baseline age (i.e. AGE_INT, AGE92M, etc). These splines are calculated using the same stratification procedure.
`expo.label`	Character label for your x-axis. If left NULL it will default to the expo variable name. Otherwise you can label your x-axis anything you want, i.e. "Baseline BMI", "Cigarettes per day", etc.
`outcome.text`	Character vector describing your outcome that will go into the figure title. If left NULL, this value will default to the name of our outcome variable name. Otherwise you can label your outcome any way you want, i.e. "Incident breast cancer", "Fatal lung cancer in men", etc.

Value

A ggplot graph is returned after running the function. The user can output this figure using any of the default graphic output functions included in base R. The final figure is editable using normal ggplot2 commands.

Warning

In datadist(df) : [variable] is constant. The RMS package creates a a datadist object that includes summaries of all variables in a dataset and is required for adjusting the spline models for covariates. If any of the variables in your dataset are nonvarying, you will get this warning. It is not necessarily a fatal error. But if the variable is an important part of your model, you might want to consider whether it is coded correctly.

Notes

The final plots are limited on the x and y axis. On the Y-axis (Hazard ratios) range from 0-4.0. The X-axis limits are based on the distribution of your data, with cutoffs at the 0.01 and 0.99 quantiles. This may result in a warning message that some data have been dropped from the figure. These values are dropped from the X-axis due to formatting issues in ggplot(). If this is a problem, it can be changed in the function, but it is recommended that you allow the truncating of the extreme ends of our exposure variable.

Author(s)

Maret Maliniak and Brian Carter

See Also

`rcs`

Examples

```
df <- example_data
figure <- splineFun(dat=df,
                    expo="bmi92",
                    covariates=NULL,
                    reference=25,
                    knots=5,
                    start="dtint92",
                    stop="dateft",
                    outcome="myeloid",
                    agestrat="age92m",
                    expo.label="Baseline BMI",
                    outcome.text="Incident myeloid leukemia")

plot(figure)
```

`stratified_cox` *Runs multiple referent group cox models using interaction variables.*

Description

The stratified_cox() function runs multiple referent group interaction cox models. The results will show the effect of our main exposure variable within each level of our stratification variable.

Usage

```
stratified_cox(dat,start,stop,outcome,expoVar,strataVar,age,covariates=NULL)
```

Arguments

`dat`	data frame used in the analysis
`start`	Character vector naming your start of followup time, can be numeric or date format. Typically this is one of our DTINT variables. For time dependent or late-entry models, this will be the name of your start time variable.
`stop`	Character vector naming your end of followup time, can be numeric or date format. Typically this is one of our DATEFT or DATEDD variables. For time dependent or late-entry models, this will be the name of your stop time variable.
`outcome`	Character vector for the outcome variable used in the analysis, must be coded as 0=control, 1=case
`expoVar`	Character vector for our main exposure variable, may be categorical or numeric
`strataVar`	Character vector for our stratification variable, must be categorical
`age`	Character vector for our age variable used for stratifying on single year of age
`covariates`	Character vector of covariates. Default is NULL (age-adjusted analysis only)

Details

The stratified_cox() function will run your stratified interaction models, format the results into a table, and return all model output in a list format that is suitable for program review. Function will work for standard, time-dependent, or late-entry models, depending on how the data are formatted prior to running the function.

Interaction models are coded using the following formula:

y <- formula(Surv(start,stop,outcome)~ strataVar + strataVar:expoVar + covariates + strata(age))

To calculate a p-interaction, a reduced model is also calculated:

y <- formula(Surv(start,stop,outcome) strataVar + expoVar + covariates + strata(age))

The p-interaction is then calculated:

anova(interaction.model, reduced.model)

Value

The function outputs a 3-level list including all the output from the analysis.

`final`	A data frame containing the organized output of the function. Includes exposure name, categories, case numbers, stratified estimates and p-values, and a p-value for interaction.
`int.model`	All model output for the interaction model
`base.model`	All model output from the base model

Author(s)

Brian Carter

References

G:/Intramural Research/Epidemiology Research/Analysts EPI/Memos, presentations, resources and code/Interaction/Memo, Interaction Methods.doc

See Also

coxph, cox_models, interaction_cox

Examples

```
df <- example_data

foo <- stratified_cox(dat=df,
                      start="dtint92",
                      stop="dateft",
                      outcome="dead",
                      expoVar="smoke92",
                      strataVar="bmicat92",
                      age="age92m",
                      covariates=NULL)
```

`tbl1` *Creates and formats a typical Table 1 for categorical and continuous variables*

Description

The tbl1() function will create a typical descriptive table for categorical and continuous variables. Frequencies are stratified by an exposure variable. Percentages can be calculated as column or row percents.

Usage

```
tbl1(dat, variable, stratvar, percents=2, freq.type=0)
```

Arguments

`dat`	A data.frame in which to interpret the variables used in the function.
`variable`	Main frequency variable or vector of variables. In an N*N table, this would correspond to the rows
`stratvar`	Strata variable. In an N*N table, this would correspond to the columns
`percents`	1=row percents; 2=column percents (default); 0=overall percents
`freq.type`	0=Both N and (percent); 1= N-only; 2=Percentages-only

Details

The tbl1() function will take any set of variables and calculate frequencies (N and percentages) across strata of another variable and format the output typical for BERG Table 1 publications. Output for continuous variables will be mean (SD), categorical variables will be N (percent). Total output is formatted so that continuous variables are at the top of the table, followed by results for categorical variables. Final output is a data frame.

Author(s)

Brian Carter

Examples

```
df <- example_data
table1 <- tbl1(dat=df,
         variable=c("bmi92","age92m","smoke92","educ"),
         stratvar="myeloid")
```

`weightlist` *Age-specific weights used for rate analyses and age-adjusted frequencies*

Description

This dataset contains age-specific weights that are required for running time-dependent rate analyses in the Nutrition Cohort. Also required for running age-adjusted frequency tables.

Usage

```
weightlist[[x]])
```

Format

A list with 71 data.frames.

`YEARS70`	US 1970 population
`YEARS80`	US 1980 population
`YEARS90`	US 1990 population
`YEARS00`	US 2000 population
`YEARSW70`	World 1970 population
`YEARS1N2`	Unknown
`YEARSC`	Unknown
`YEARS1Q2`	Unknown
`YEARS12M`	Mortality cohort 12 year FU (men)
`YEARS12F`	Mortality cohort 12 year FU (women)
`YEARS12B`	Mortality cohort 12 year FU
`YEARS14B`	Mortality cohort 14 year FU
`YEARS14F`	Mortality cohort 14 year FU (women)
`YEARS14M`	Mortality cohort 14 year FU (men)
`YEARS16B`	Mortality cohort 16 year FU
`YEARS16F`	Mortality cohort 16 year FU (women)
`YEARS16M`	Mortality cohort 16 year FU (men)
`YEARS18B`	Mortality cohort 18 year FU
`YEARS18F`	Mortality cohort 18 year FU (women)
`YEARS18M`	Mortality cohort 18 year FU (men)
`YEARS20B`	Mortality cohort 20 year FU
`YEARS20F`	Mortality cohort 20 year FU (women)
`YEARS20M`	Mortality cohort 20 year FU (men)
`YEARS22B`	Mortality cohort 22 year FU
`YEARS22F`	Mortality cohort 22 year FU (women)
`YEARS22M`	Mortality cohort 22 year FU (men)

`YEARS24B`	Mortality cohort 24 year FU
`YEARS24F`	Mortality cohort 24 year FU (women)
`YEARS24M`	Mortality cohort 24 year FU (men)
`YEARS26B`	Mortality cohort 26 year FU
`YEARS26F`	Mortality cohort 26 year FU (women)
`YEARS26M`	Mortality cohort 26 year FU (men)
`YEARS28B`	Mortality cohort 28 year FU
`YEARS28F`	Mortality cohort 28 year FU (women)
`YEARS28M`	Mortality cohort 28 year FU (men)
`YEARS30B`	Mortality cohort 30 year FU
`YEARS30F`	Mortality cohort 30 year FU (women)
`YEARS30M`	Mortality cohort 30 year FU (men)
`YEARSNIB`	Nutrition cohort 22 year FU
`YEARSNIF`	Nutrition cohort 22 year FU (women)
`YEARSNIM`	Nutrition cohort 22 year FU (men)
`YEARSNHB`	Nutrition cohort 20 year FU
`YEARSNHF`	Nutrition cohort 20 year FU (women)
`YEARSNHM`	Nutrition cohort 20 year FU (men)
`YEARSNGB`	Nutrition cohort 18 year FU
`YEARSNGF`	Nutrition cohort 18 year FU (women)
`YEARSNGM`	Nutrition cohort 18 year FU (men)
`YEARSNFB`	Nutrition cohort 16 year FU
`YEARSNFF`	Nutrition cohort 16 year FU (women)
`YEARSNFM`	Nutrition cohort 16 year FU (men)
`YEARSNEB`	Nutrition cohort 14 year FU
`YEARSNEF`	Nutrition cohort 14 year FU (women)
`YEARSNEM`	Nutrition cohort 14 year FU (men)
`YEARSNDB`	Nutrition cohort 12 year FU
`YEARSNDF`	Nutrition cohort 12 year FU (women)
`YEARSNDM`	Nutrition cohort 12 year FU (men)
`YEARSNCB`	Nutrition cohort 10 year FU
`YEARSNCF`	Nutrition cohort 10 year FU (women)
`YEARSNCM`	Nutrition cohort 10 year FU (men)
`YEARSNBB`	Nutrition cohort 8 year FU
`YEARSNBF`	Nutrition cohort 8 year FU (women)
`YEARSNBM`	Nutrition cohort 8 year FU (men)
`YEARSNBB`	Nutrition cohort 6 year FU
`YEARSNBF`	Nutrition cohort 6 year FU (women)
`YEARSNBM`	Nutrition cohort 6 year FU (men)
`YEARS30MLIM`	Mortality (age limited) cohort 30 year FU (men)
`YEARS30FLIM`	Mortality (age limited) cohort 30 year FU (men)
`YEARS30BLIM`	Mortality (age limited) cohort 30 year FU (men)
`YEARS32MLIM`	Mortality (age limited) cohort 32 year FU (men)
`YEARS32FLIM`	Mortality (age limited) cohort 32 year FU (men)
`YEARS32BLIM`	Mortality (age limited) cohort 32 year FU (men)

Details

Each dataset in weightlist corresponds to a person year or age distribution. In rate analyses, the user must choose one of these age distributions to calculate the adjustment. See documentation for rate and age-adjustments for details how to use these datasets in each function.

Examples

```
weightlist[["YEARS00"]]
```

Index