|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Getting Data In/Out | | | | | |
| read data | read.table(file.choose(), header=TRUE) | |  | | file.choose() will open a finder window so you can choose the file you need. You can alternatively type in the path to the file. |
| read .csv | read.csv(file.choose(), header=TRUE) | |  | |  |
| read spss | read.spss(file.choose(), header=TRUE) | |  | | Requires the foreign package. |
| read sas | sas.get(file.choose(), header=TRUE) | |  | | Requires the foreign package. |
| save data to a file | write.table(dataframe, “Filename.txt”,sep=“\t”, row.names=FALSE) | | write.table(myData,"~/R/Devin/myData.txt",sep=t,row.names=FALSE) | |  |
| **Data Container/Object Operations (creating, assigning, reading etc.)** | | | | | |
| create a vector | c(x,y,x) | | c(“bill”,”john”,”mary”) or c(1,2,3) | | creates a list with the elements: “bill”,”john”,”mary” |
| save a vector to a name | listName = c(x,y,z) | | myList=c(1,2,3) | | creates a list with the elements: 1, 2, 3 |
| create a dataframe | frameName=data.frame(variableName1=vectorName1,variableName2=vectorName2...) | | myDataFrame=data.frame(gender=genderList,age=ageList) | | creates a dataframe (spreadsheet) called “myDataFrame” such that the first column will be made up of the items in the previously created list called “genderList” and the previously created list called “ageList”. The first column header will be “gender” and the second column header will be “age”. |
| create a factor from existing vector | factorName=factor(variableName,levels=c(x,y,z...),labels=c(“label1”,”label2”...)) | | gender=factor(gender,levels=c(1:2),labels=c(“Male”,”Female”)) | | turns an existing vector (list) called “gender” into a factor called “gender” with 2 levels which are coded as “1” and “2”. The label for “1” will be “Male” and the label for “2” will be “Female” |
| create a factor from scratch | factorName=gl(numberOfLevels,numberOfCasesInEachLevel,totalCases,labels=c(“label1”,”label2”...)) | | gender=gl(2,5,labels=c(“Male”,”Female”)) | | creates a factor called “gender” that will have 2 levels with 5 cases in each level. The first level will be called “Male” and the second “Female”. |
| rename a column title/header | names(dataFrame)=c(“label1”,”label2”...) | | names(myDataFrame)=c(“Person”,”Sex”,”Score”) | | changes or creates the first column label (headers) for the dataframe called “myDataFrame” so that it is “Person”. The second would be “Sex” and so forth. |
| convert column to a factor | dataFrameName$columnName=as.factor(dataFrameName$columnName) | | mtcars$am=as.factor(mtcars$am) | | converts the column named “am” in the dataframe named “mtcars” into a factor |
| label for a factor | levels(dataSet$varName)=c(“One”,”Two”,”Three”…) | |  | |  |
| change column names | colnames(dataSet)=c(“colOneName”,”colTwoName”...) | |  | |  |
| recode a value | variableToChange=recode(variableToChange,”x=y;a=b;c=d...”) | | dataSet$sex=recode(dataSet$sex,”1=0;2=1”) | | changes all the 1s in the “sex” column of the dataframe “dataSet” to 0s and the 2s become 1 |
| merge two dataFrames | merge(receivingDataFrame, donatingDataFrame, by.x=“receivingDataFrameKeyNameInQuotes”, by.y=“donatingDataFrameKeyNameInQuotes”) | | heid2=merge(heid2,items,by.x=“Word”,by.y=“Word”) | | Merges the dataFrame called "items" into the dataFrame called "heid2" and matches the values for the "Word" column in both dataFrames |
| Accessing Data in Data Containers | | | | | |
| refer to variable in a dataframe by name | dataFrameName$variableName | | cars$speed | | lists the values in the column called “speed” from the dataframe called “cars” |
| describe dataframe structure | str(dataFrameName) | | str(cars) | |  |
| list variables in a dataframe | names(dataFrameName) | | names(cars) | |  |
| sort dataframe | dataFrame[order(dataFrame$sortColumn1, dataFrame$sortColumn2), ] | |  | | |  |
| list headers | head(dataFrame, n = NumRowsToPrint) | | head(verbs, n = 10) | |  |
| accessing information in data frames: specify row and column number | dataSet[RowNum, ColNum] | | verbs[1,5]  verbs[ , 5]  verbs[1, ] | |  |
| access info in data frame: specify col name | dataSet$ColName | | | | | verbs$LengthOftheme |  |
| list only rows with unique values (no duplicates) | unique() | | | | | unique(dataSet) |  |
| sort or order a dataframe by column | dataFrame[order(dataFrame$sortColumn1, dataFrame$sortColumn2), ] | |  | | order function occupies the slot in the subscript of the data frame that specifies the conditions on the rows |
| change info in dataframe | verbs.rs[“638”,]$RealizationOfRec = “NP” | | | |  |  |
| Extracting and Subsetting Information from Dataframes | | | | | |
| extract rows that meet a criterion (subset) | subset(dataSet, colName == “criterionString”) | | subset(verbs, AnimacyOfTheme == “animate”) | |  |
| extract row or column as vector | vectorName = dataSet[1, ]  vectorName = dataSet[ , 5] | |  | |  |
| extract several rows | verbs[c(1,2,3),] | |  | |  |
| extract several continuous rows | verbs[1:3, ] | |  | |  |
| extract rows that meet a criterion (subset) | verbs[verbs$AnimacyOfTheme == “animate”, ] | |  | |  |
| extract rows that meet more than one criteria | verbs(verbs$AnimacyOfTheme == “animate” & verbs$LengthOfTheme > 2, ] | | | | AND = &; OR = | |  |
| extract row and col names | rownames(); colnames() | | head(rownames(verbs))  colnames(verbs) | |  |
| specify dataset | with() | | with(verbs, tapply(LengthOfTheme,AnimacyOfRec, mean) | |  |
| subset with tapply | with(dataset,tapply(numericVector,list(Factor1,Factor2...),function)) | | with(verbs,tapply(LengthOfTheme,list(AnimacyOfRec,AnimacyOfTheme),mean)) | |  |
| crosstabulation | xtabs(dependentVariable ~ independVar1 + independenVar2, data=dataSet) | | xtabs(~ RealizationOfRec+AnimacyOfRec,data=verbs) | | Uses the languageR library. Creates a table with column 1 containing the designation of RelizationOfRec (NP or PP) and the number of animate and inanimate NPs and PPs:  RealizationOfRec animate inanimate  NP 521 34  PP 301 47  There is no variable to the left of the ~ since there is no dependent variable -- just a frequency count of items |
| contingency table | xtabs(~ independVar1 + independenVar2, data=dataSet) | | xtabs(~ RealizationOfRec+AnimacyOfRec,data=verbs) | |  |
| table of proportions | prop.table(xtabTableName) | | prop.table(verbs.xtabs) | | can use prop.table(verbs.xtabs,1) to specify to use row total proportions or 2 to use column total proportions (columns sum to 1) |
| calculation on dataframe | tapply(numericVector, factors, function) | |  | |  |
| get averages from dataframe | aggregate(dependentVariable, column), function) | | aggregate(heid$RT, list(heid$Word),mean) | |  |
| Statistical Functions | | | | | |
| normality: shapiro-wild | shapiro.test(vectorName) | |  | |  |
| normality: Kolmogorov-Smirnov | ks.test(vectorName, distribution, parameter1, parameter2) | | ks.test(ver$Frequency, “pnorm”,mean(ver$Frequency),sd(ver$Frequency)) | |  |
| compare chance of two prob distributions | chisq.test(vectorName) | | chisq.test(intro) | |  |
| probability of N | pnorm(value, mean, sd) | |  | | For a left tailed distribution:  2\*(1-pnorm(abs(value),mean,sd)) |
| probability of t | pt(tValue,df) | |  | |  |
| probabilty of F | pf(fValue, df) | |  | |  |
| probability of ChiSq | pchisq(ChiSqValue, df) | |  | |  |
| one-sample t test | t.test(vectorName, mu=comparisonMean) | | t.test(scores,mu=3) | |  |
| pairwise t-test | pairwise.t.test(dependentVariable, independentVariable, p.adjust.method=“methodName”) | | pairwise.t.test(score, group, p.adjust.method=“bonferroni”) | | runs pairwise t-tests following an ANOVA |
| basic independent anova | | objectName=aov(dependentVariable ~ independentVariable, data=dataSet); summary(objectName) | | viagra.aov=aov(libido~dose, data=viagraData); summary(viagra.aov) | summary can be run separately |
| repeated measures ANOVA with Mauchley's sphericity | model = ezANOVA(data=dataFrame, dv=.(outcomeVariable), wid=.(subjectVariableName), within=.(repeatedMeasuresVariables),between=.(betweenGroupsVariables), detailed=FALSE, type=2) | | | | requires the “ez” package. "wid" is the subject variable id name. |
| repeated measures ANOVA | newModel=lme(outcome~predictor,random=randomEffects,data=dataFrame,method=“ML”) | |  | | should also create a baseline model and use anova(baselineModel, modelTwo) to compare |
| repeated measures ANOVA | aovModel=aov(DV ~ IV + Error(subject/IV),data=dataFrame)  summary(aovModel) | |  | |  |
| post-hoc tests | pairwise.t.test(outcome,predictor,paired=FALSE,p.adjust.method=“method”) | | pairwise.t.test(dataFrame$score, dataFrame$group, paired=FALSE, p.adjust.method=“bonferroni”) | | methods: bonferroni, BH, |
| baseline LME model | baseline=lme(outcome~1, random=~1|Participant/RepeatedVariable, data=dataFrame, method=“ML") | |  | |  |