Introdução ao R Commander

Umberto Guarnier Mignozzetti*

25 de setembro de 2009

Nesta apostila vamos introduzir as ferramentas principais do R Commander, um pacote complementar para o software estatístico R. Pelo caráter introdutório deste trabalho, vamos organizar a apresentação seguindo a ordem do que seriam os principais passos para a resolução de problemas práticos de pesquisa, assim, apresentaremos os principais comandos e sua utilização. Para facilitar, usaremos bancos disponíveis no próprio R e no pacote `Rcmdr`.

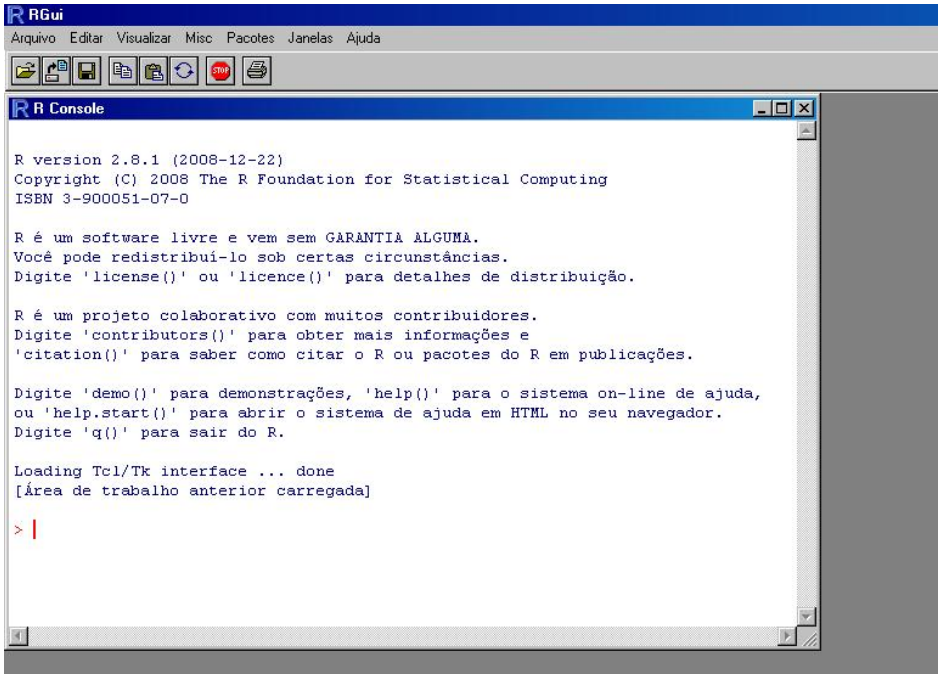
1 Baixando e Instalando o R Commander

Para baixar o R Commander, obviamente você deverá já haver instalado o R em sua maquina. Se este ainda não for seu caso, instale o R[1].

Com o R baixado abra o RGui e você terá a seguinte tela (ou algo parecido):

*Centro de Estudo das Negociações Internacionais (CAENI) - Departamento de Ciência Política - Universidade de São Paulo. Site `www.caeni.com.br`. Este texto é uma versão preliminar e tem mais erros que se pode imaginar. Você pode reporta-los para `matematicari@caeni.com.br`. Os gráficos não ficaram bons mas estão, em geral, compreensíveis. Esta apostila foi pensada para ser usada na ultima aula do curso de R oferecido regularmente pelo CAENI.

[1]Site `www.r-project.org`.

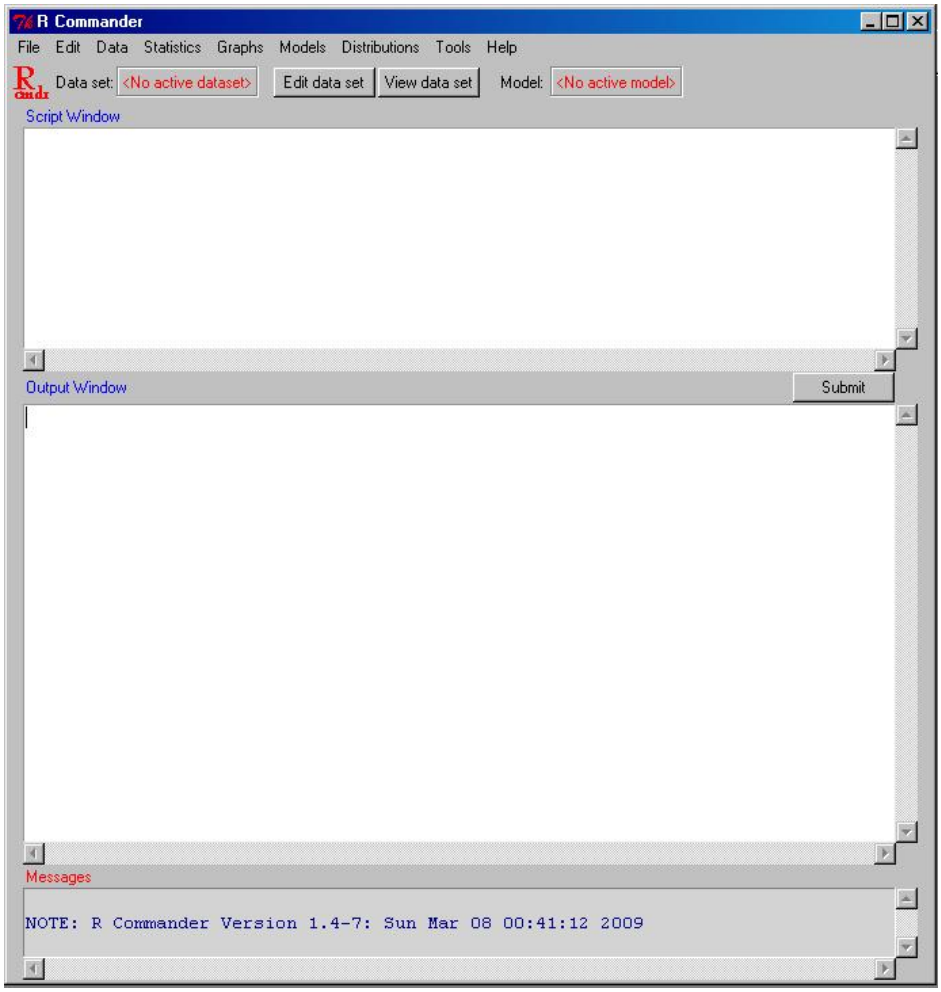


Vá então no menu `Pacotes` e depois em `Instalar Pacotes`. Aparecerá um menu com várias siglas que facilmente podem ser identificadas como países. Estessão os espelhos onde os pacotes do R e o próprio software estão disponíveis. Selecione o espelho do CRAN mais próximo de você e esta tela se fechará e abrirá outra em que você terá disponível todos os pacotes para sua escolha. Nesta tela selecione o pacote `Rcmdr`. Ele irá instalar o pacote e eventualmente alguns outros. Pode acontecer de o programa perguntar se você deseja instalar outros pacotes e você deverá confirmar. Os outros pacotes são os complementos do R Commander.

Depois de instalado basta você digitar:

```
> require(Rcmdr)
```

E o R Commander aparecerá em seu monitor:

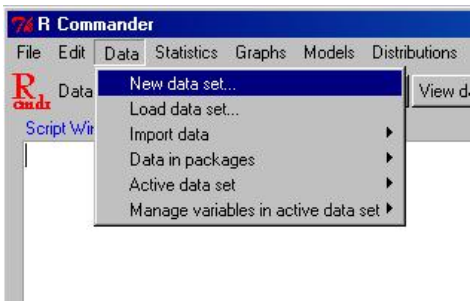


Este basicamente é o R Commander em sua versão default. Existem vários complementos que adiciona ainda mais comandos ao pacote e você poderá explora-los depois. Nesta ocasião, nosso objetivo será apresentar somente os comandos mais elementares.

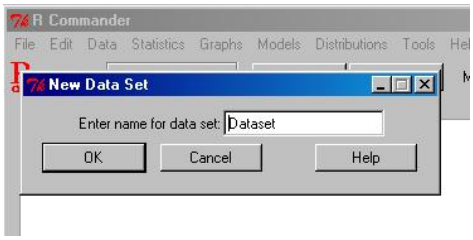
2 Criando ou Carregando um Bancos de Dados

Vários são os formatos que podem ser lidos do R Commander. Além disso, nosso pacote e o próprio R vem com bancos de dados que podem (e serão) trabalhados em nosso texto. O Menu que contém os comandos para esta

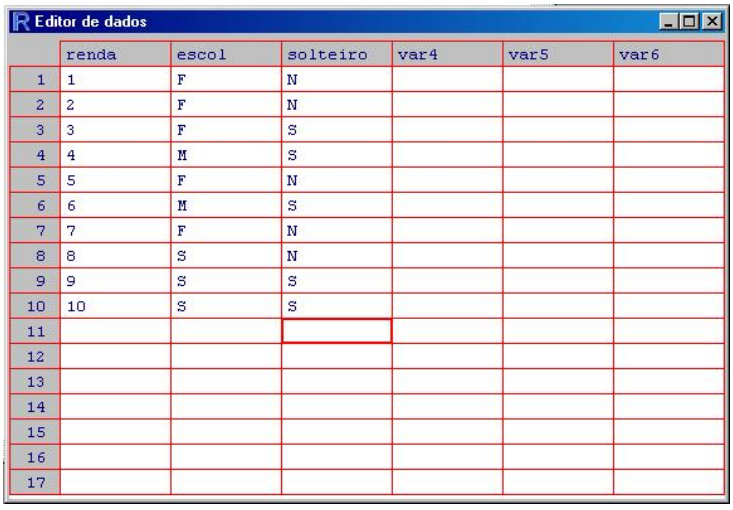
seção é o `Data`. Suponhamos que vamos criar um banco de dados. Para isso basta clicarmos em `Data` e depois em `New data set...`:



Daí o programa irá te pedir para dar o nome ao novo banco de dados:

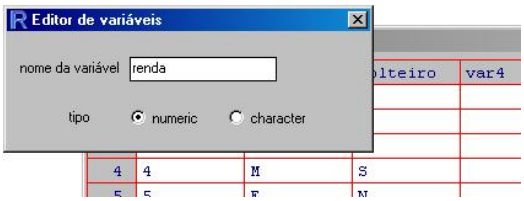


Voce pode por o nome que quiser. Vamos manter `Dataset` por questões de simplicidade... Dando OK teremos,

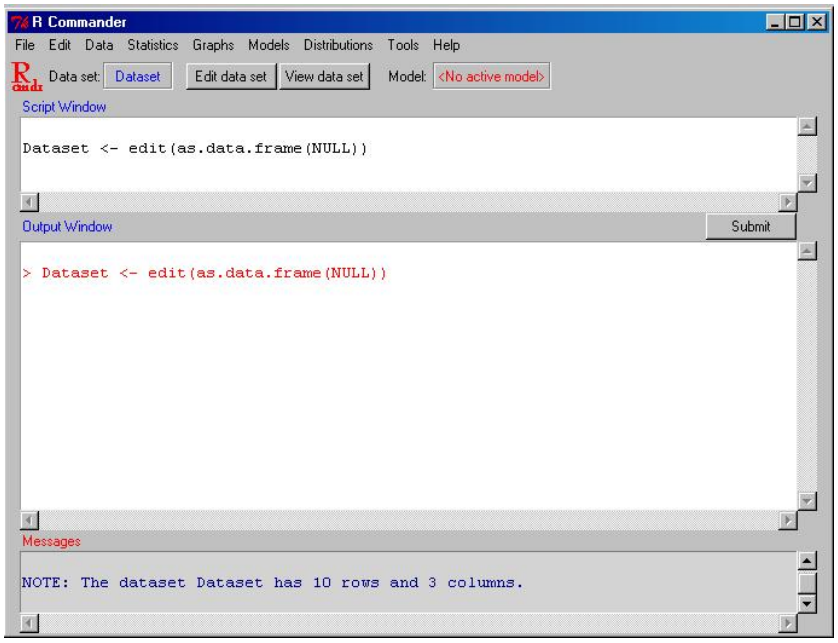


	renda	escol	solteiro	var4	var5	var6
1	1	F	N			
2	2	F	N			
3	3	F	S			
4	4	M	S			
5	5	F	N			
6	6	M	S			
7	7	F	N			
8	8	S	N			
9	9	S	S			
10	10	S	S			
11						
12						
13						
14						
15						
16						
17						

Assim, você pode facilmente editar os dados que você quer inserir. Insira como exercício os dados que aqui estão. Além disso, pode mudar o nome e o tipo de variável (e isso é muito importante...):



Para encerrar, aperte o botão para fechar (pode apertar, seus dados não serão perdidos!). Assim, note que o conjunto de dados já foi corretamente inserido e está ja no R Commander:



Cabe agora explicarmos um pouco mais sobre as duas janelas principais do R Commander. A primeira é a janela do `Script`. Nela são salvos as sintaxes dos comandos que você deu. Isso eventualmente serve para você enviar para outra pessoa uma linha de comandos ou mesmo abrir você mesmo em alguma

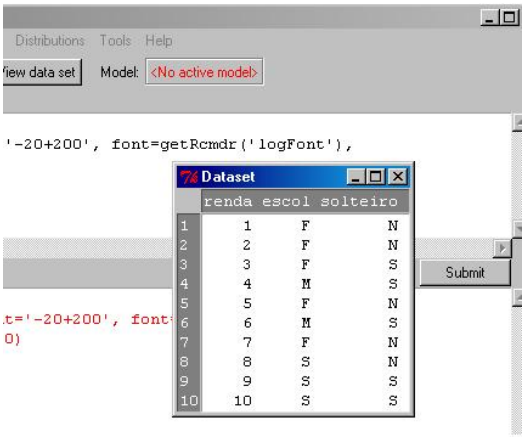
outra ocasião[2]. Caso você queira dar novamente o comando, basta colocar o cursor na linha que foi dado o comando e, pressionar $CTRL + R$.

Na janela de saída (`Output window`), temos os resultados dos comandos dados. Logo mais em baixo, no rodapé, algumas informações que o sistema julga relevante partilhar conosco, no caso deste comando, o numero de linhas (casos) e colunas (variáveis) de nossa matriz de dados.

Note que no canto superior esquerdo o sistema nos indica que está trabalhando com o banco `Dataset`. Se tivermos mais que um unico banco podemos clicar em cima do nome do banco e ele nos abre uma janela que nos permite selecionarmos qual banco queremos usar.

Podemos facilmente consertar algum erro de digitação em nosso banco usando o botão `Edit data set` que está ao lado do nome do banco. Ao clicar nele, novamente a janela anterior se abre e daí podemos editar os valores.

Ao lado deste botão temos o botão `View data set`. Ele é bem parecido com o `Edit` em seu layout mas você não é capaz de alterar o banco por ele. Só serve para ver os dados.

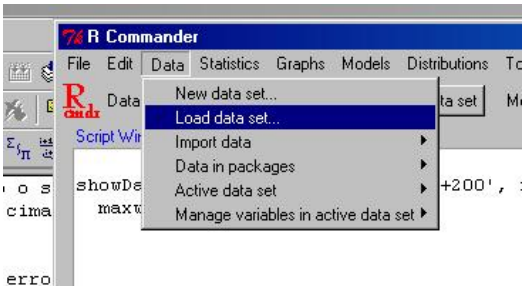


Mais ao lado temos uma caixa onde está escrito `Model`. Veremos o que é

[2]Ou seja, isto lhe auxilia a replicar o que foi feito anteriormente.

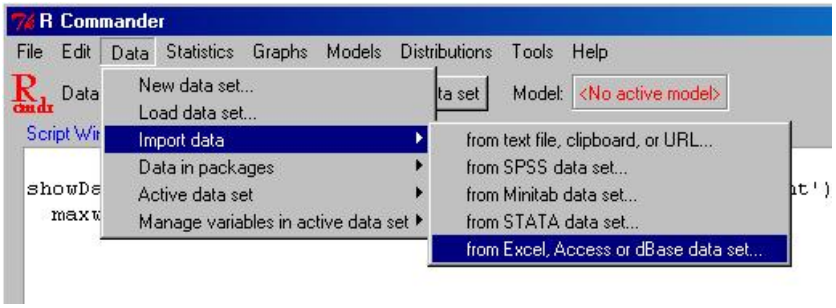
mais a frente. A idéia é que no R, nós salvamos os modelos preditivos que formulamos para nossos dados.

O nosso próximo problema é carregar um banco de dados já existente. Se o banco tiver sido salvo pelo R Commander, basta que você clique em `Data` e depois `Load data set...`.



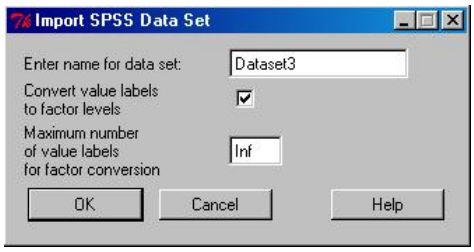
E então selecionar o arquivo que você quer carregar. O R Commander colocará como banco ativo ele e seu nome será o mesmo que estava quando foi salvo.

A idéia é que, além dos bancos do R, tem alguns outros de grande importância. Por exemplo, você pode querer carregar um banco feito em Excel. Isso é bem simples: basta você ir em `Data` e depois em `Import Data` e então selecionar no menu o comando que bate com o seu banco de dados. Note que o R Commander tem suporte para uma série de bancos.



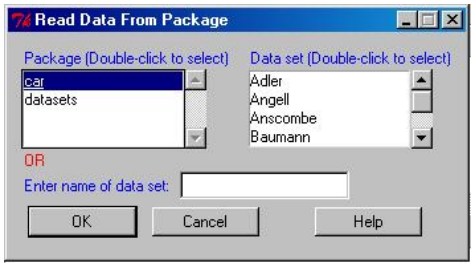
Em nossa área, o software *SPSS* é muito usado. Para carregar um banco feito neste sistema basta usarmos o mesmo procedimento. A única diferença

é que, além do nome do banco, o sistema irá nos perguntar sobre algumas opções da transferência:



No inicio temos o nome do banco. A segunda opção diz respeito a se você deseja manter os fatores com seus respectivos labels ou não. Exemplo: se uma variável assume valores 'Sim' ou 'Não'. Se você ligar o botão, ela vai aparecer no R Commander como 'Sim' ou 'Não'. Se você desligar este botão, ela irá aparecer como numérica[3]. O ultimo diz respeito aos numeros de labels atribuidos aos fatores[4].

Além dos bancos que temos, um recurso bem interessante do R e do R Commander são os bancos que vem junto com o programa. Para acessa-los basta clicar em `Data` e depois em `Data in packages`. Para ver os variados tipos de bancos basta clicar em `Lista data sets in packages`. Para carregar algum basta usar o outro menu. Irá aparecer o seguinte menu:



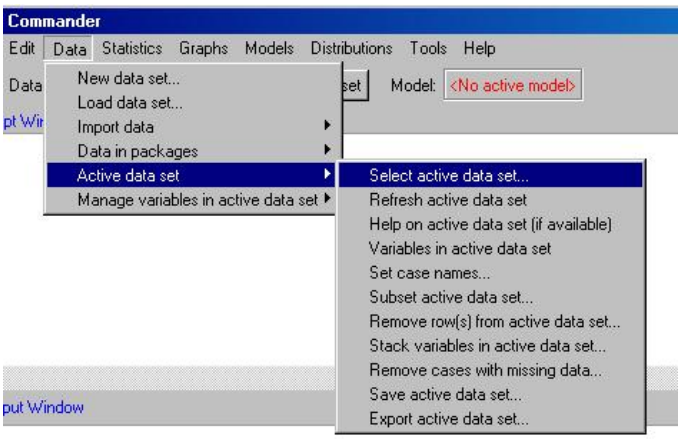
Aí basta você selecionar o banco que você deseja carregar. Como exercício, selecione o banco `Chile` do pacote `car`.

[3]Logicamente que não por isso deixará de ser nominal ou ordinal...

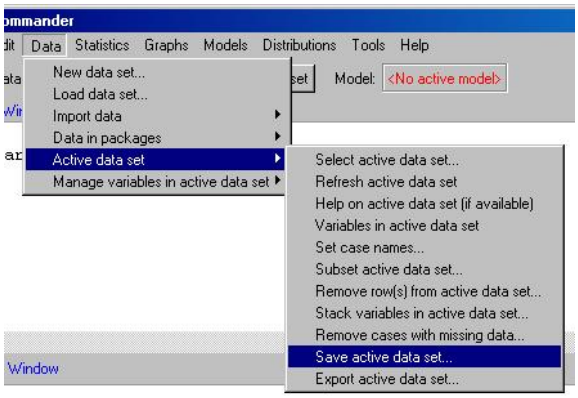
[4]Eu sempre mantenho `Inf` neste valor...

Digamos que agora nosso problema será selecionar outro banco que já carregamos anteriormente. Basta clicarmos em:

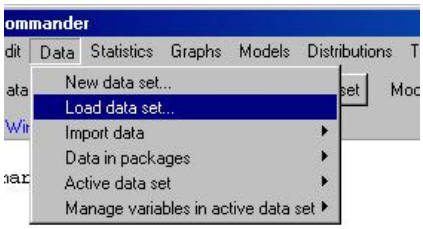
Ou então, pelo menu, basta irmos em:



E daí selecionarmos no menu que irá aparecer, dentre todos os bancos de dados salvos no console do R, o que desejamos. Eventualmente, nosso banco de dados pode ter sido feito em R Commander. Primeiramente, para salvarmos um banco feito em R Commander, basta clicarmos em `Data` em seguida em `Active Data Set` e depois em `Save Active Data Set`. O sistema irá salvar em um formato próprio para depois ser aberto no R Commander:

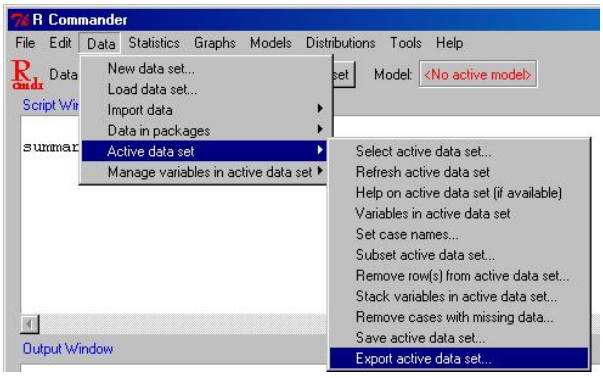


Para carregarmos bancos salvos por meio deste procedimento, basta fazermos:

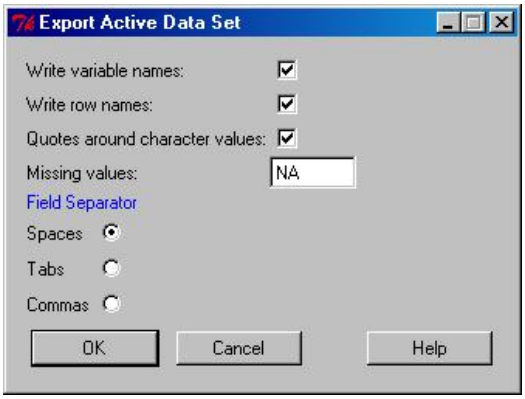


E então, devemos selecionar nosso banco do menu de arquivos.

Eventualmente, podemos querer exportar nosso banco de dados para outros formatos. O R Commander faz isso de modo bem diferente que os outros pacotes de estatística. Ele nos apresenta um menu com algumas opções do modo como deverá ser composto nosso arquivo. Para exportarmos:



As opções são:



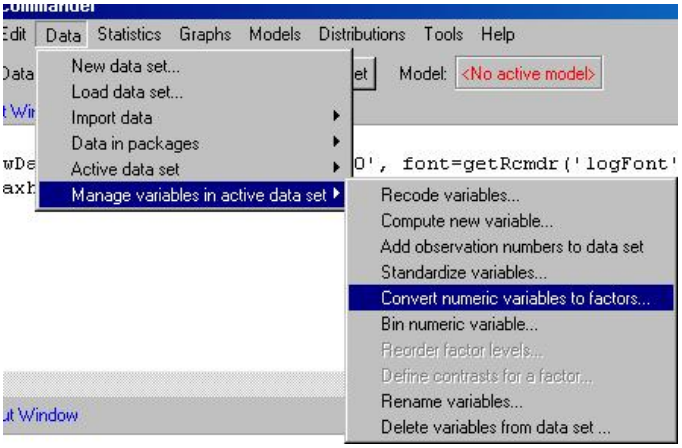
Na primeira caixa, ele pergunta se queremos salvar os nomes das variáveis. Se não fizermos isso, vão só os dados brutos. Na segunda, se queremos salvar o ponto amostral. Eventualmente, em algum banco, ao invés de aparecer 1, 2,... poderia aparecer os nomes de países por exemplo e seria portanto importante preservar o nome da coluna. Na terceira caixa, ele pergunta se queremos aspas separando as cadeias de caracteres. Isso é muito bom, principalmente para exportarmos para programas como o Excel. Na caixa com o valos NA, ele pede qual será o valor do missing. Alguns pesquisadores usam valores negativos para exportarem missings. Abaixo, o separador de campo define qual dos espaços separa os dados. O Excel por exemplo, costuma ler corretamente quando os dados estão separados por tabulações. O formato CSV (comma separated values) pede que separemos portanto com vírgulas. Este formato é lido por quase todos os sistemas. A dica é que você organize a exportação de acordo com o padrão do programa importador dos dados.

3 Manipulando os Dados

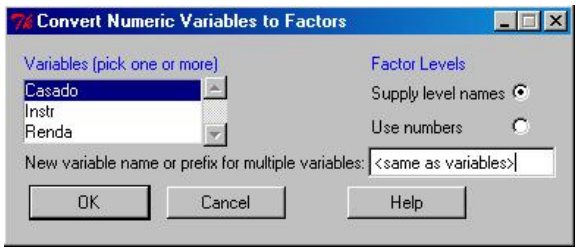
Já vimos como editar os dados no R Commander. O que ainda não vimos é como fazer, em algumas situações específicas, isso de maneira eficiente.

Vejamos como fazer isso em um banco tirado do livro dos profs. Bussab & Morretin. O banco é bem simples[5], tem três variáveis, a primeira é responde se a pessoa é Casada(= 1) ou não(= 0), a segunda, o Grau de Instrução da pessoa, Fundamental (= 0), Médio (= 1) e Superior (= 2) e, a terceira, a Renda da pessoa em multiplos de Salários Mínimos. Digamos que, nosso primeiro problema seja dar 'forma' a estes níveis, ou seja, apresenta-los como algo mais interessante que zeros e uns.

Vamos fazer isso então para a primeira variável, a variável 'Casado'. Basta irmos então no menu:



E a partir daí, aparecerá um box com as opções para nossa manipulação:



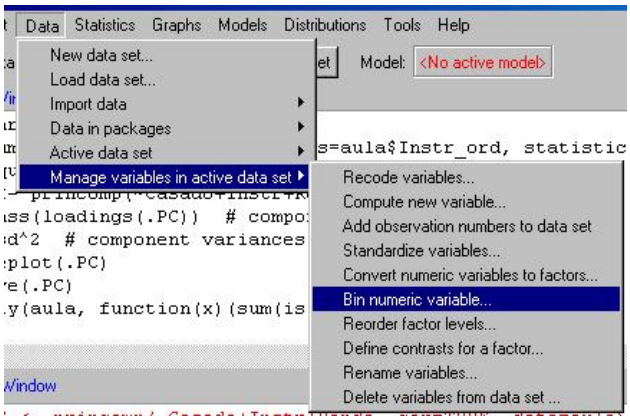
Você então deverá selecionar as variáveis que deseja alterar. Iremos alterar 'Casado' e salvaremos como 'Casado_'. No pequeno botão de seleção ao lado

[5]Segue no Anexo do texto. Sugerimos que você reedite-o no seu R Commander e acompanhe os passos desta seção

(à direita), o sistema nos questiona se queremos manter os numeros como nomes dos fatores ou se iremos altera-los também. Em nosso caso, iremos alterar, zero para Solteiro e um para Casado. Abaixo, o sistema pergunta se queremos reescrever a variável (default) ou se desejamos salvar as alterações com outro nome. Vamos, como já dissemos, salvar como 'Casado_'. A partir daí, o sistema exibe uma janela para edição dos nomes dos fatores:

Vamos então fazer o mesmo para a variável Grau de Instrução(Instr)[6].

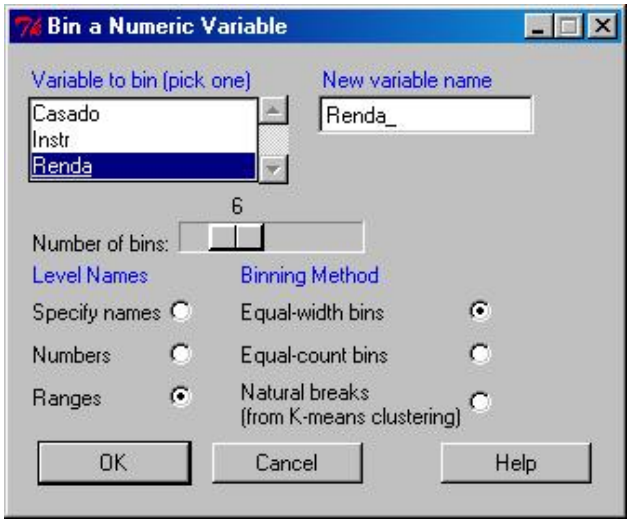
Um modo bem simples de recodificar a variável Renda é usando o comando `Bin numeric variables...`:



A partir daí o programa abre um menu com algumas opções para modelarmos a quebra dos dados:

A primeira opção é naturalmente qual variável entrará em nossa 'quebra'. Ao lado, temos a definição do nome da variável que vamos salvar a quebra

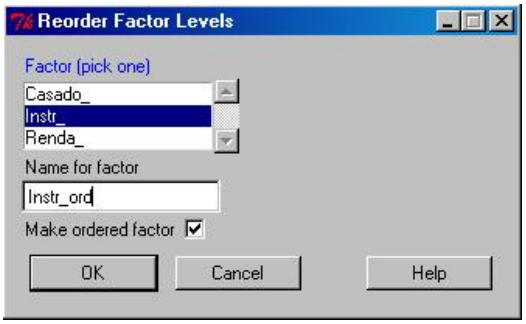
[6]Conforme pedido na nota 5, isso fica como exercício...



realizada. A opção abaixo diz respeito ao numero de quebras. Vamos fazer 6 para nossos dados (o default costuma ser 3). Logo abaixo temos os nomes dos níveis. Poderiamos eventualmente defini-los de acordo com alguma suposição prévia mas, neste caso, vamos nomear a partir dos valores máximos e minimos das quebras efetuadas pelo sistema. Por ultimo, o método usado para 'quebrar' o dado. O primeiro e o segundo são mais intuitivos. Dizem respeito, respectivamente, a igualdade nas variações das faixas e a igualdade no numero de casos em cada faixa (note que isso por si só ja dá uma diferença tremenda, dependendo do caso). O ultimo é a quebra de acordo com as K-médias. A teoria envolvida foge do escopo deste texto e assim sugerimos consultar o texto [k] das referências. Em nosso caso vamos quebrar em seis faixas de amplitudes iguais.

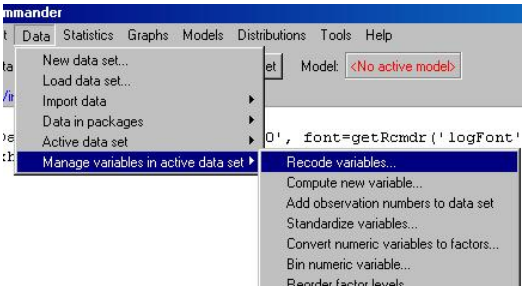
Outro procedimento interessante é informarmos para o sistema quando temos fatores que são ordinais entre si, como no caso do Grau de Instrução. O comando no menu é o `Data`, em seguida `Manage Variables in Data Set` e por fim, `Reorder factor levels`. Irá aparecer o seguinte menu:

Primeiramente a variável a ser selecionada para aplicar a função, em



segundo lugar, o nome da variável nova e, na caixa, se os fatores novos serão ordinais. Caso você ligue esta opção, aparecerá outro menu para que você confirme a ordem correta dos dados. Coloque numeros inteiros, começando em um, para pontuar esta ordenação.

Digamos então que queremos recodificar alguma variável a nosso gosto, sem muitas restrições. Isso em geral é um pouco mais difícil. O comando para fazermos isso:



E então basta selecionarmos a variável que queremos recodificar. O comando `Recode` é bem livre e, com isso, bem complicado... Algumas dicas podem ajudar você a recodificar com eficiência sua variável:

- Uma dica talvez meio desnecessária é para você não salvar a variável nova sobre a variável antiga. Eu não consigo te dar nenhuma justificativa lógica mas, isso já me atrapalhou muito... Assim, veja o resultado e, se satisfatório, apague então a antiga.

- Se a variável for discreta, renomeie cada um dos valores que a variável assume. Por exemplo, se chamarmos o valor 0 de `Não` e o valor 1 de `Sim`, faça:

```
0 = "Não"
1 = "Sim"
```

- Se a variável for contínua e você quer transforma-la em discreta, use limites. Por exemplo: se estamos lidando com a variável `Renda` do banco, caso queiramos que a variável assuma o valor 0 se a renda estiver entre o mínimo e 8 s.m. e, 1 se estiver entre 8 e 12 s.m. e 2 para valores maiores do que 12, basta fazermos o que está no box abaixo.

- Se você não quer diferenciar algum grupo use o comando `else`. Por exemplo, se quisermos no banco anterior, criar uma *dummie* em que temos 1 caso a pessoa tenha ensino fundamental e 0 caso contrário basta acrescentarmos `else=0` à linha de comando.

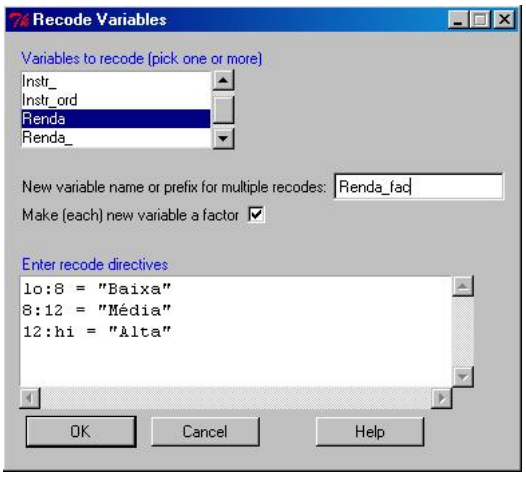
- Os comandos são separados por Enter, portanto, pule de linha depois de terminar um comando.

- Os comandos `lo` e `hi` significam, respectivamente, *low* e *high*. Use-os para limites inferior e superior respectivamente.

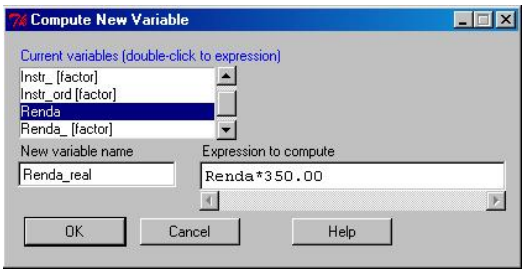
- Use o comando ':' para indicar sequências. Por exemplo, se quisermos que o sistema leia os valores de 1 até 10 dizemos simplesmente para ele 1 : 10.

- O comando recodificará sempre os primeiros valores primeiro, ou seja, no exemplo do box abaixo, se algum valor for igual a 8, ele chamará de

'Baixa' e se algum valor for igual a 12 ele chamará de 'Média'. Assim, ele executa sempre a primeira instrução antes das outras.

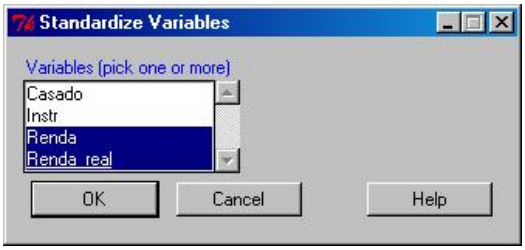


Um comando interessante é o `Compute new variable...` (logo abaixo do comando `Recode`). Muita coisa legal podemos fazer com ele. Podemos aplicar funções às variáveis, bem como combina-las entre si, somando-as, por exemplo. Um exemplo, para ilustrar, é a renda bruta, supondo que o valor do salário mínimo à época tenha sido de R$ 350,00.

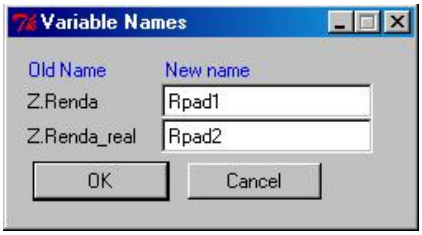


O resultado final das recodificações você pode conferir clicando em `View data set`. O ideal é que, após cada recodificação, você confira o resultado final. Um comando importante é o `Standardize variables`. Ele nos dá o resultado da recodificação em variável padronizada. Se supormos que o fenomeno em questão segue uma distribuição normal. esse comando é muito

importante. Eventualmente ainda podemos usa-lo para regressão com variáveis padrinizadas. Vamos fazer isso com *Renda* e *Renda_real*:



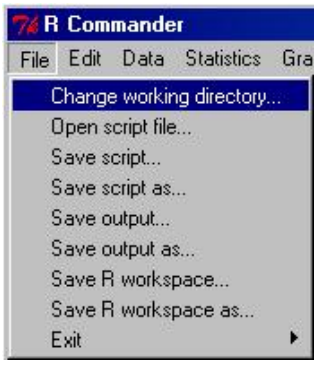
Digamos que por algum motivo qualquer queiramos renomear os dados. Para isso usamos o comando `Rename variables`. O seu uso é bem simples: basta selecionar as variáveis que queremos renomear e depois, no box que aparecerá em seguida, renomea-las:



Ou seja, muito simples. Como sabemos que a variável `Renda` é combinação linear da variável `Renda_real`, sabemos que, quando padronizamos, obtivemos a mesma variável. As duas são iguais portanto... Para descarta-las basta usarmos, ainda no menu `Manage variables in data set` a opção `Delete variables in data set...`. Daí então selecionamos a variável e pronto, ela será descartada. Note que podemos selecionar mais de uma (basta segurarmos o CTRL).

4 O Menu *File*

Este menu será abordado de modo bem breve. Quem está acostumado no padrão windows sabe que em geral o que querem dizer suas funcionalidades.



O primeiro comando, `Change the work directory` é bem importante. Suponhamos que, por motivos de organização, tudo que tiver a ver com bancos de dados você queira guardar em uma pasta distinta, feita só para este propósito. Para tanto, você deve mandar o R Commander salvar ou ler tudo desta pasta especificamente e isso você faz a partir deste comando.

Os três comandos seguintes dizem respeito ao Script. No começo do texto falamos um pouco dele. No Script o R Commander salva os comandos que ele deu (pense na tradução, roteiro...). O comando `Open script file...` serve para você abrir scripts já feitos por você em outras ocasiões. O comando `Save script...` salva o que tiver sido feito no Script que você estiver trabalhando no momento e o `Save script as...` salva com outro nome. Note que na janela do Script ficam os comandos dados. Você pode comenta-los como faz com qualquer script do R, basta que você adicione o parâmetro '#'. Ele fará com que o R pare de interpretar a linha e considere-a somente um comentário.

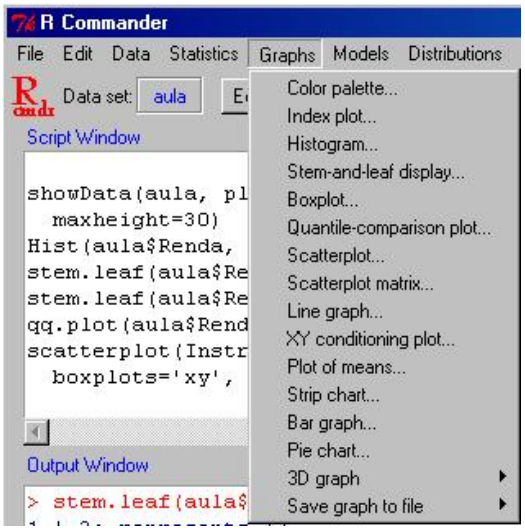
Os quatro comandos seguintes servem para salvar os resultados. Os dois primeiros salvam, em formato *.txt* o que aparece na janela de *Output*. Os dois seguintes, serve para salvarmos o que fora feito ou inserido no R Commander

na memória. Quando abrirmos o R Commander tudo que tivermos feito estará presente para que carreguemos.

O ultimo você descobre...

5 Estatísticas Básicas usando R Commander

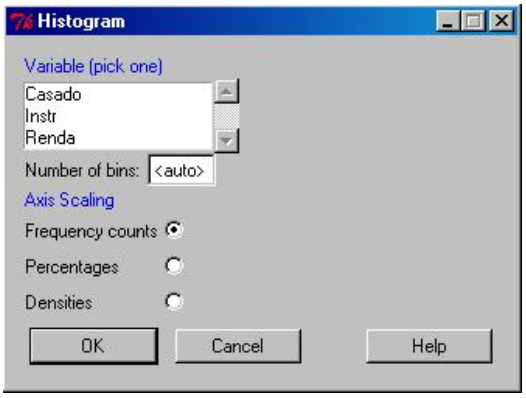
Ja passamos pelos principais comandos de edição e manipulação de dados do R Commander. Nosso objetivo agora deverá ser de apresentar os principais comandos estatísticos do pacote. Nesta seção estudaremos dois menus basicamente. O primeiro é o menu `Statistics` e o outro o menu `Graphs`. Vamos começar com a `Graphs` que é mais legal...



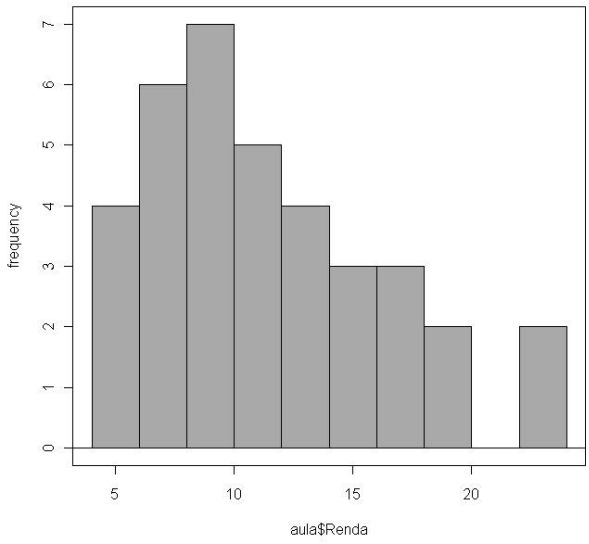
Ou seja, na versão *default* do R Commander, vários são os gráficos que podem ser feitos[7]. Dos gráficos, vamos primeiramente ilustrar um de grande interesse, o **histograma** (comando `Histogram...`):

Vamos fazer um histograma então para a variável *Renda*. O resultado, o R

[7] E veremos que vários são os omplementos que podemos acrescentar para ampliar o 'poder de fogo' de nosso R Commander



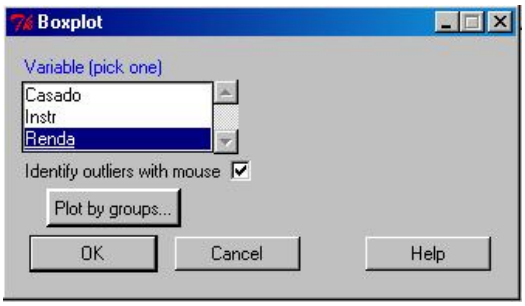
Commander dá na própria janela gráfica do R. Na prática é algo semelhante a isso:



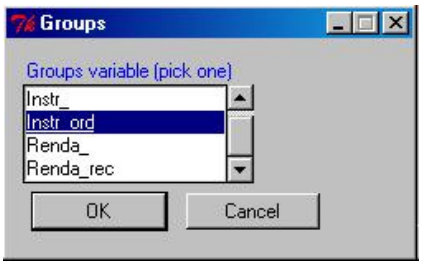
Ou seja, exatamente o histograma que queriamos. Os parâmetros que podem ser passados para o sistema incluem o numero de quebras (no caso usamos o default) e o tipo da frequência que irá aparecer em cada casela. Eu usei a primeira opção, você, por exemplo, poderia ter pedido o resultado em densidade. Fica a seu critério...

Um outro tipo de gráfico legal para fazermos com nosso conjunto de

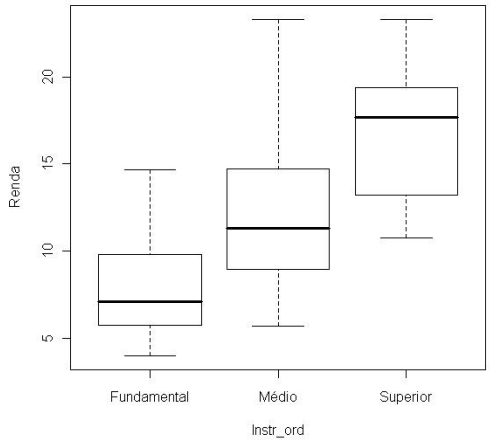
dados é o box-plot. O comando naturalmente é o `Boxplot`. Aparece então o seguinte menu:



Ou seja, selecionamos a variável *Renda* e marcamos que queremos identificar os *outliers*. Podemos ainda usar uma categórica qualquer para segmentarmos nosso box-plot de acordo, por exemplo, com faixas de escolaridade. Fazemos isso com o comando `Plot by groups...`:

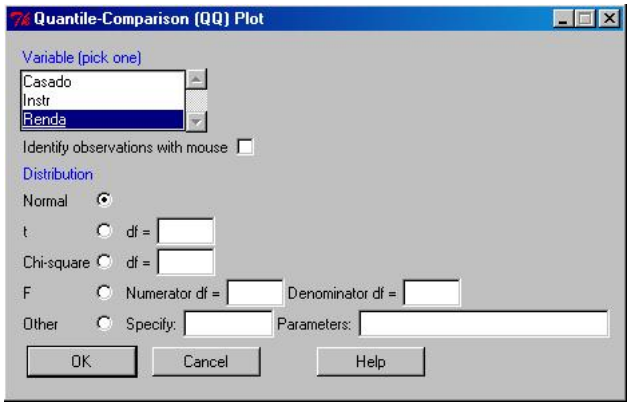


O nosso box-plot então fica:



Ou seja, conforme esperado...

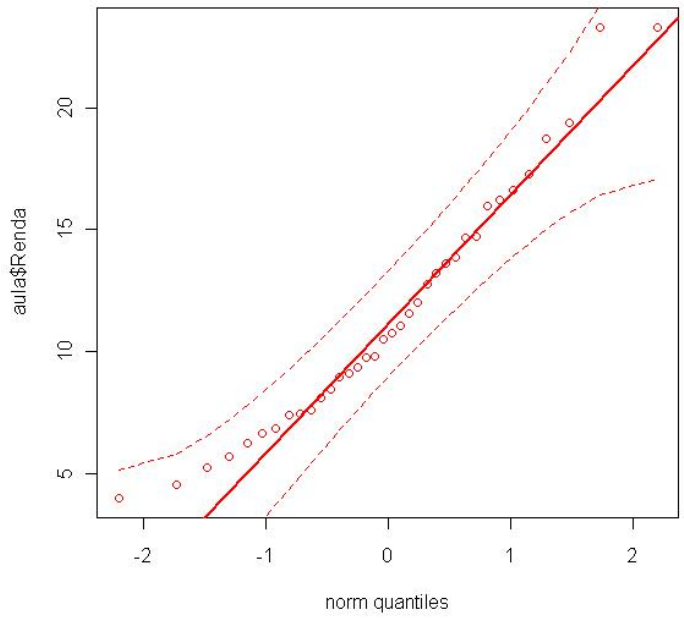
Outro gráfico interessante é o QQ-Plot, comando `Quantile Comparison Plot...`. Para fazermos este tipo de gráfico para nosso conjunto de dados basta selecionarmos a variável e uma distribuição de probabilidade[8] que acreditamos que ela segue:



E o gráfico de QQ-Plot fica:

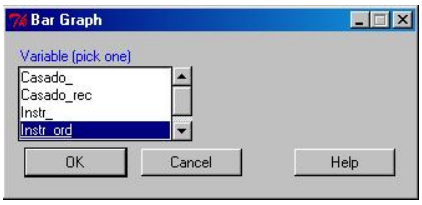
Poderiamos ter inclusive, ligado o identificador de pontos. Não faremos isso aqui, entretanto, em muitos casos práticos vale a pena saber quem seria o

[8]Note que temos varias distribuições no R. Para maiores informações veja o Card de Tom Short, facilmente encontrado no site www.r-project.org

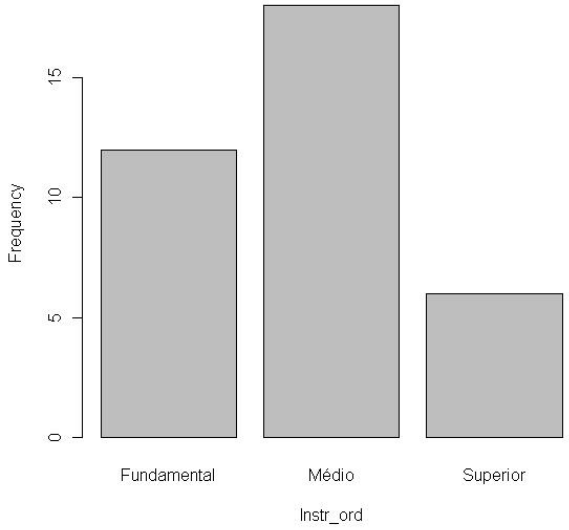


dado que está se comportando mal... Note que a linha tracejada é o intervalo de confiança.

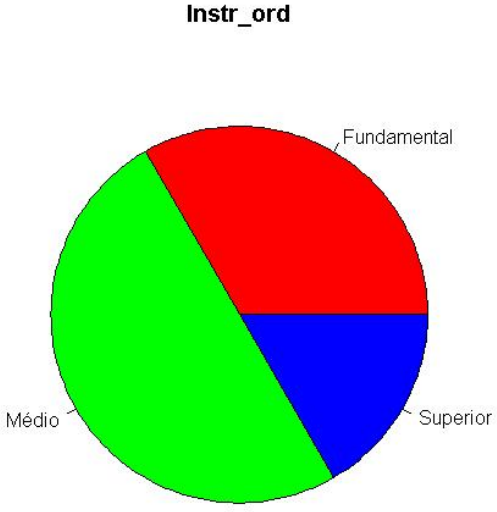
Um gráfico interessante é o gráfico de barras. O comando para este gráfico é o `Bar graph...` Para a variável *Instr_ord* temos de fazer:



E o resultado:

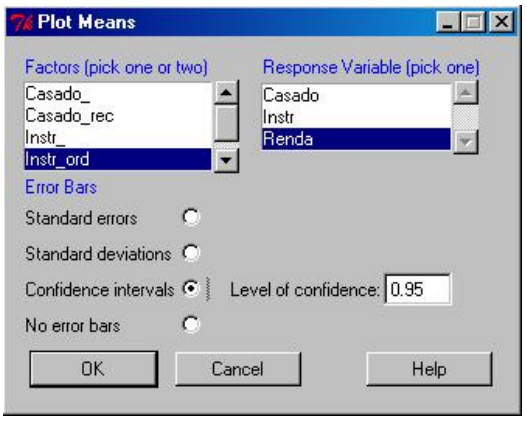


Um outro gráfico que tem mais ou menos a mesma funcionalidade do gráfico de barras é o `Pie chart...` Usamos ele de modo semelhante ao gráfico de barras. Um `pie` dos dados anteriores:

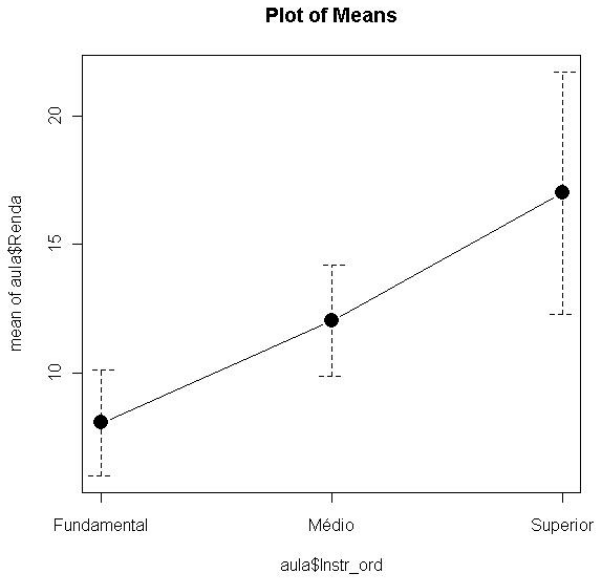


Note que também podemos fazer um gráfico para o intervalo de confiança para médias. Digamos que desejamos estudar a variação da média de renda

quando variamos a escolaridade. O comando para isso é o `Plot of means...` Seus parâmetros:



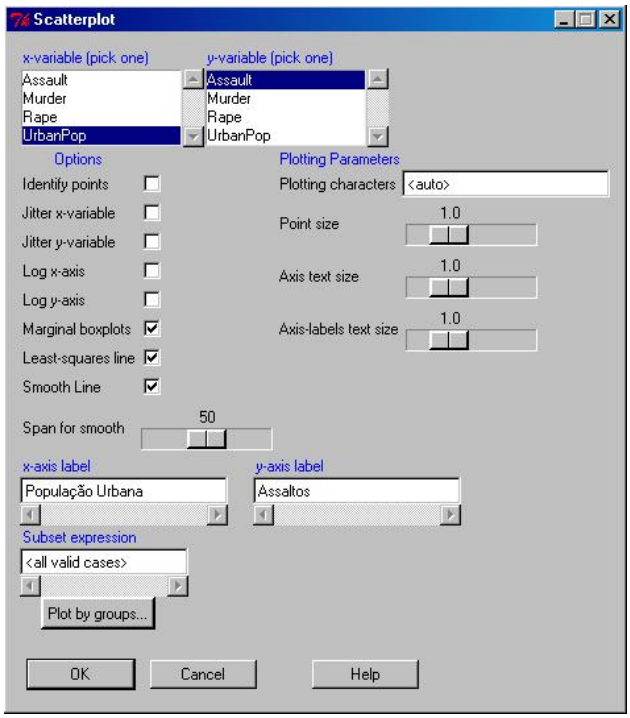
E o resultado:



Em nossos parâmetros escolhemos plotar o intervalo de confiança a 95%. Você poderia ter plotado qualquer um dos outros parâmetros.

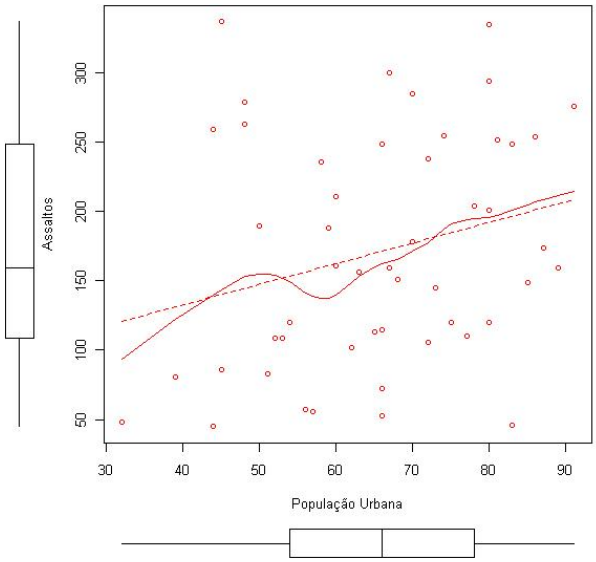
Para duas variáveis temos um gráfico interessante, o chamado `Scatterplot...` Vamos carregar o banco 'USArrests'. Ele está disponível

no pacote 'datasets'. Carregue o banco e chame o comando que acabamos de citar. Vai aparecer algo do tipo:

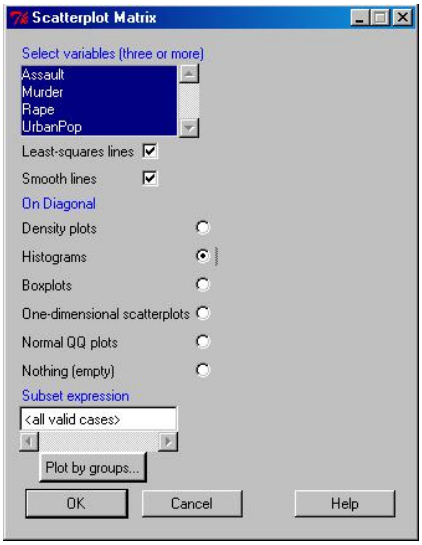


Este comando tem vários parâmetros. Os dois primeiros dizem respeito à seleção das variáveis que figurarão nos eixos x e y respectivamente. Abaixo podemos ativar o identificador dos pontos, este parâmetro `jitter`, que ainda não sei para que serve, podemos aplicar a transformação logaritmica aos dados de x ou y[9], podemos colocar ou não box-plots marginais ao lado dos eixos, o ajuste de minimos quadrados e a curva suave. Ainda podemos alterar o tamanho de cada bolota. Podemos também alterar os titulos dos eixos x e y e segmentar por casos. No caso particular acima, o resultado é o seguinte:

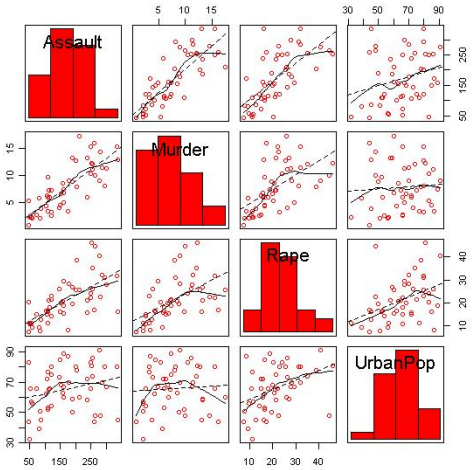
[9]Reduz a escala e ajuda em modelos log-lineares, log-lin, lin-log, entre outros...



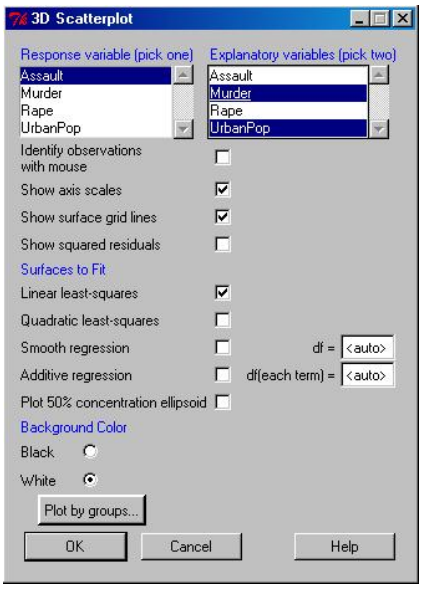
Quando temos mais de duas variáveis, o gráfico equivalente para esta situação é o `Scatterplot matrix...`



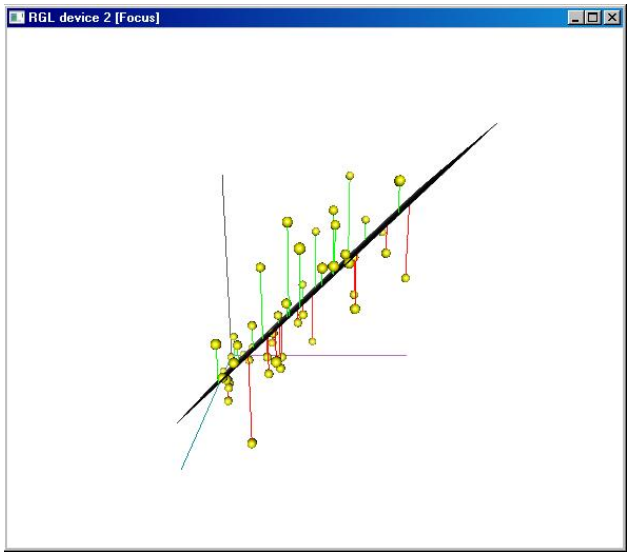
Os parâmetros são pouco diferentes e, a principal pergunta neste caso é o que você irá colocar na diagonal de seu gráfico. Muitos recursos temos disponíveis (conforme você pode ver). No caso acima:



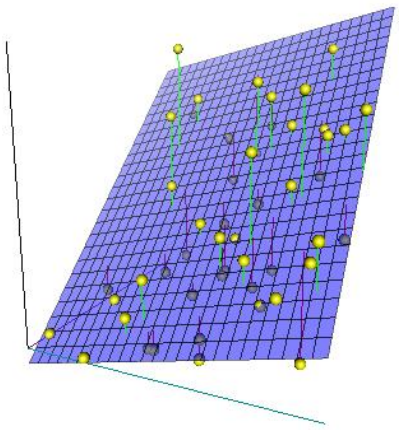
Um gráfico bem interessante é o `3D Scatterplot...` Vejamos o seus parâmetros:



O gráfico fruto destes parâmetros é:



E o interessante é que podemos mudar o ângulo de visão do gráfico:



O ultimo menu[10] diz respeito a possibilidade de salvar os gráficos produzidos como arquivos. Este aqui fica para você explorar. Ele não tem nada de muito discrepante do menu `File`.

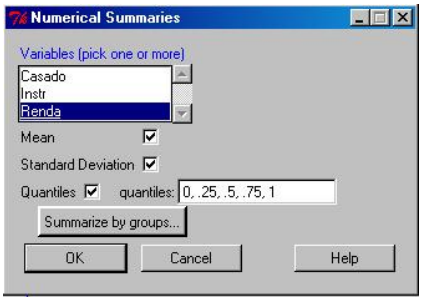
Vamos passar agora para alguns sumários dos dados feitos por via do menu `Statistics`. Vamos explorar no restante desta seção os sub-menus `Summaries` e `Contingency tables`. Para tanto, voltemos ao banco 'aula'.

[10]Naturalmente alguns menus ficaram faltando mas você poderá explora-los sozinho.

Para calcularmos algumas estatísticas descritivas usamos no sub-menu `Summaries` o comando `Active data set`. Este comando não diferencia quais variáveis entram no sumário. Ele faz o sumário de todas as variáveis do banco carregado. O resultado sai no box `Output`:

```
> summary(aula)
     Casado           Instr            Renda            Casado_
 Min.   :0.0000   Min.   :0.0000   Min.   : 4.000   Solteiro:16
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 7.553   Casado  :20
 Median :1.0000   Median :1.0000   Median :10.645
 Mean   :0.5556   Mean   :0.8333   Mean   :11.527
 3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:14.695
 Max.   :1.0000   Max.   :2.0000   Max.   :23.300
         Instr_           Renda_         Instr_ord  Casado_rec Renda_rec
 Fundamental:12   (3.98,7.2] : 7   Fundamental:12   1:16       0:28
 Médio      :18   (7.2,10.4] :10   Médio      :18   2:20       1: 8
 Superior   : 6   (10.4,13.6]: 8   Superior   : 6
                  (13.6,16.9]: 6
                  (16.9,20.1]: 3
                  (20.1,23.3]: 2
```

Outro modo de sumarizar os dados é tirando as principais estatísticas descritivas para algumas variáveis somente. O comando é o `Numerical summaries`. O menu que ele abre:



Note que, podemos, além das opções que pedi no exemplo, dividir em ainda mais quantis os dados ou ainda, separa-los em sub-grupos. Vejamos o resultado do que foi pedido:

```
> numSummary(aula[,"Renda"], statistics=c("mean", "sd", "quantiles"),
+   quantiles=c(0,.25,.5,.75,1))
```

```
     mean       sd 0%    25%    50%    75% 100%  n
 11.52667 4.995144  4 7.5525 10.645 14.695 23.3 36
```

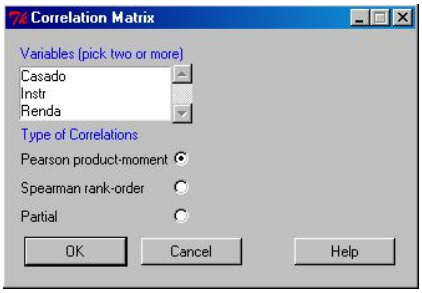
Uma função interessante é a `Count missing observations`. Ela nos dá o numero de dados perdidos em cada variável. Nosso banco, por construção, não tem nenhuma variável missing. Daí:

```
> sapply(aula, function(x)(sum(is.na(x)))) # NA counts
    Casado      Instr      Renda    Casado_     Instr_     Renda_  Instr_ord
         0          0          0          0          0          0          0
Casado_rec  Renda_rec
         0          0
```

Para tirarmos as frequências simples de nossas variaveis usamos o comando `Frequency distributions`. Para as frequências da variável *Casado_* :

```
> .Table <- table(aula$Casado_)
> .Table  # counts for Casado_
Solteiro   Casado
      16       20

> 100*.Table/sum(.Table)  # percentages for Casado_
Solteiro   Casado
44.44444 55.55556

> remove(.Table)
```

Podemos também calcular, para nossos dados, uma matriz de correlação. O comando é o `Correlation Matrix`. Note que temos várias formas de fazer o calculo e, a rigor, não poderiamos aplicar da forma que estamos fazendo porque temos algumas variáveis categóricas no meio de nossa matriz. O menu do teste é o seguinte:

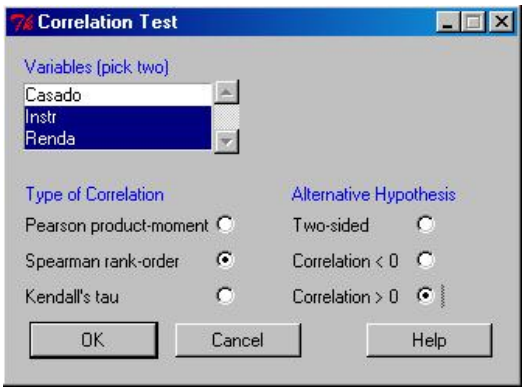


A matriz de correlação de Pearson para as variáveis:

```
> cor(aula[,c("Casado","Instr","Renda")], use="complete.obs")
          Casado     Instr     Renda
Casado 1.0000000 0.1084652 0.2526122
Instr  0.1084652 1.0000000 0.6125011
Renda  0.2526122 0.6125011 1.0000000
```

Note no menu a opção `Type of Correlations`. Nela você pode alterar o método usado para calcular a matriz de correlação. Cada método tem seus pressupostos sobre os dados e, você deve sempre estar atento a isso.

Eventualmente, podemos fazer testes para ver se as correlações encontradas são (ou não) significativas. Usamos então a opção `Correlation test`. Nela encontramos as seguintes opções:



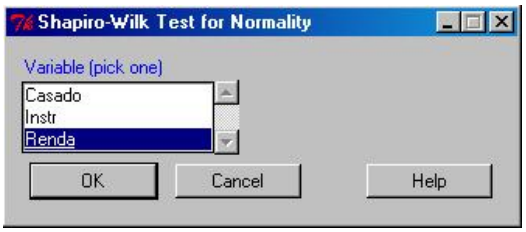
Vamos para os dados em questão fazer então um teste para verificar se a correlação entre *Instr* e *Renda* é positiva (aumento em uma gera um au-

mento na outra) ou não (hipótese nula...). Note que, a suposição nos leva a considerar a hipótese alternativa tal como aparece no texto. Selecionamos então *Spearman* porque funciona para os dados que temos:

```
> cor.test(aula$Instr, aula$Renda, alternative="greater", method="spearman")

Spearman's rank correlation rho

data:  aula$Instr and aula$Renda
S = 2894.646, p-value = 2.104e-05
alternative hypothesis: true rho is greater than 0
sample estimates:
      rho
0.6274586
```

Ou seja, à julgar pelo p-valor, a correlação é então significativa. O coeficiente aparece no final do teste. Vale aqui uma nota sobre a notação ciêntifica do R. O p-valor é $2.104e-05$ isso quer dizer que é igual a $2.104x10^{-5}$ ou seja, em valor exato, 0.00002104. Ou seja, o primeiro valor aparece onde seria a primeira casa decimal. Se você estiver calculando algo e o e for menor que -3, considere que é bem próximo de zero (mas tome cuidado, notação cientifica às vezes nos engana).

O último comando deste sub-menu é o `Shapiro-Wilk normality test`. Ou seja, o teste de normalidade de Shapiro-Wilk. A hipótese nula deste teste é que o dado que queremos testar segue uma distribuição Normal. Assim:

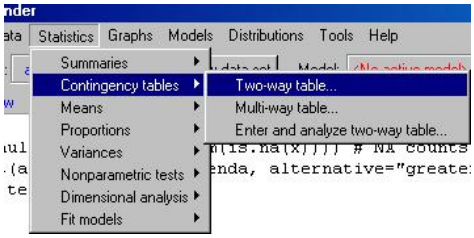


E portanto, vamos testar a hipótese de normalidade para a variável *Renda*:

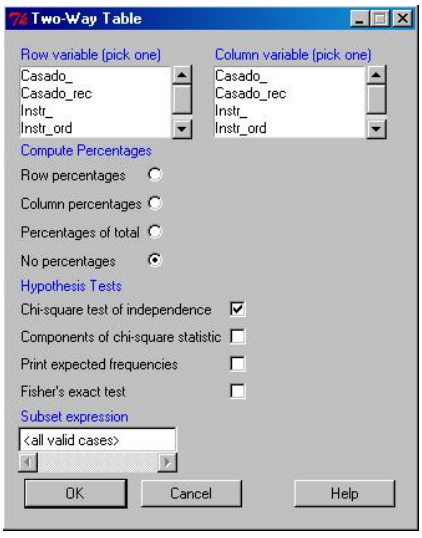
```
> shapiro.test(aula$Renda)

Shapiro-Wilk normality test
data:  aula$Renda
W = 0.9514, p-value = 0.1159
```

Ou seja, estes dados não nos dão evidencia de que não estamos lidando com uma população Normal[11].

Nosso próximo sub-menu, `Contingency tables`:



Para montarmos uma tabela $2x2$ basta clicarmos em `Two-way table`. Aparecerá então o seguinte menu:



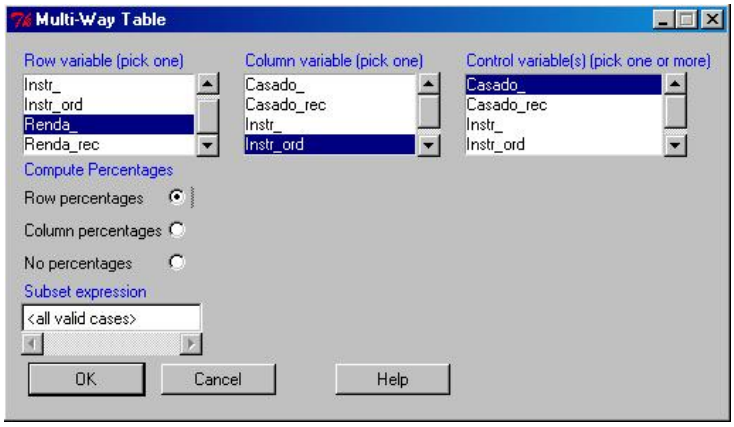
[11] O teste de Shapiro-Wilk é assintótico, o que quer dizer que ele funciona melhor com um numero 'grande' de dados. Portanto, tome cuidado...

Várias são as opções que podemos selecionar para este conjunto de dados. Vamos fazer uma tabela de contingência da renda, tal como recodificamos contra o grau de instrução. Nossa suposição naturalmente é que há uma relação entre estas variáveis. Vamos selecionar para as linhas da tabela *Renda_* e para as colunas *Instr_ord*. Vamos pedir então para que o sistema calcule os percentuais-linha, faça um teste de Qui-quadrado supondo independência e então um Teste Exato de Fisher (pois certamente haverá caselas em que o esperado seja menor que 5 casos). O resultado:

```
> .Table <- xtabs(~Renda_+Instr_ord, data=aula)
> .Table
             Instr_ord
Renda_        Fundamental Médio Superior
  (3.98,7.2]            6     1        0
  (7.2,10.4]            3     7        0
  (10.4,13.6]           2     4        2
  (13.6,16.9]           1     4        1
  (16.9,20.1]           0     1        2
  (20.1,23.3]           0     1        1

> rowPercents(.Table) # Row Percentages
             Instr_ord
Renda_        Fundamental Médio Superior Total Count
  (3.98,7.2]         85.7  14.3      0.0 100.0     7
  (7.2,10.4]         30.0  70.0      0.0 100.0    10
  (10.4,13.6]        25.0  50.0     25.0 100.0     8
  (13.6,16.9]        16.7  66.7     16.7 100.1     6
  (16.9,20.1]         0.0  33.3     66.7 100.0     3
  (20.1,23.3]         0.0  50.0     50.0 100.0     2

> .Test <- chisq.test(.Table, correct=FALSE)
> .Test

Pearson's Chi-squared test
data:  .Table
X-squared = 20.2143, df = 10, p-value = 0.02729

> remove(.Test)
```

```
> fisher.test(.Table)

Fisher's Exact Test for Count Data
data:  .Table
p-value = 0.02585
alternative hypothesis: two.sided

> remove(.Table)
```

Ou seja, há evidencia para considerarmos que as variáveis em questão não são independentes. Para tabelas mais complicadas, usamos o comando que vem na sequência deste, `Multi-way table`. Aparece então o seguinte sub-menu:



Devemos então selecionar a variável para as linhas, as colunas e a variável controle. Pediremos então novamente os percentuais-linha:

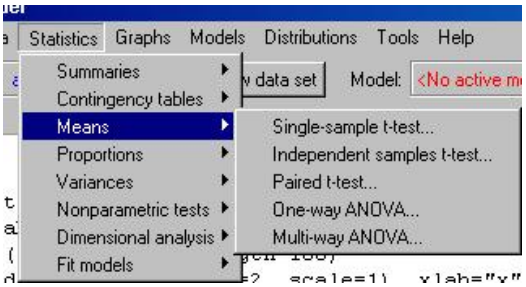
```
> .Table <- xtabs(~Renda_+Instr_ord+Casado_, data=aula)
> .Table
, , Casado_ = Solteiro

             Instr_ord
Renda_        Fundamental Médio Superior
  (3.98,7.2]            3     1        0
  (7.2,10.4]            2     3        0
  (10.4,13.6]           1     2        1
  (13.6,16.9]           1     0        1
```

```
  (16.9,20.1]           0     0        1
  (20.1,23.3]           0     0        0

, , Casado_ = Casado

             Instr_ord
Renda_        Fundamental Médio Superior
  (3.98,7.2]            3     0        0
  (7.2,10.4]            1     4        0
  (10.4,13.6]           1     2        1
  (13.6,16.9]           0     4        0
  (16.9,20.1]           0     1        1
  (20.1,23.3]           0     1        1

> rowPercents(.Table) # Row Percentages
, , Casado_ = Solteiro

             Instr_ord
Renda_        Fundamental Médio Superior Total Count
  (3.98,7.2]           75    25        0   100     4
  (7.2,10.4]           40    60        0   100     5
  (10.4,13.6]          25    50       25   100     4
  (13.6,16.9]          50     0       50   100     2
  (16.9,20.1]           0     0      100   100     1
  (20.1,23.3]         NaN   NaN      NaN   NaN     0

, , Casado_ = Casado

             Instr_ord
Renda_        Fundamental Médio Superior Total Count
  (3.98,7.2]          100     0        0   100     3
  (7.2,10.4]           20    80        0   100     5
  (10.4,13.6]          25    50       25   100     4
  (13.6,16.9]           0   100        0   100     4
  (16.9,20.1]           0    50       50   100     2
  (20.1,23.3]           0    50       50   100     2

> remove(.Table)
```

Ou seja, uma tabela chatíssima de se interpretar mas, em algum sentido,

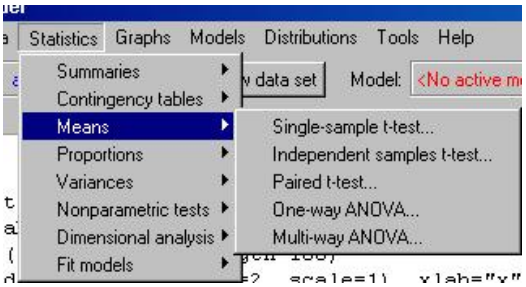
pode ser necessária em sua pesquisa. Note o 'numero' `NaN` na tabela. Ele quer dizer `Not a Number`. Provavelmente porque fizemos uma divisão por zero naquela casela.

O ultimo comando deste sub-menu é o `Enter two-way table`. Ele é util para uma situação em que, por exemplo, temos uma tabela em um artigo que queremos testar a consistência dos testes realizados. Inserimos então a mesma no menu abaixo:

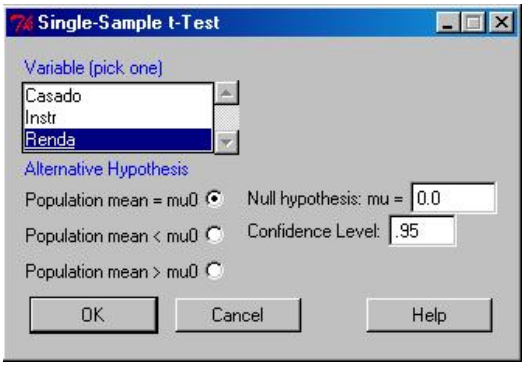


E daí pedimos os testes que quisermos. Note que você primeiramente deverá ajustar o numero de linhas e de colunas de sua matriz. Em seguida, você deve inserir os valores na tabela subjacente e daí você deve escolher os testes que deseja realizar nos dados.

O próximo menu a tratarmos será o `Means`. Em geral, seus comandos nos auxiliam com testes para médias dos dados. Os testes aqui supõem que estamos lidando com dados paramétricos. Vejamos o que temos de bom nele:



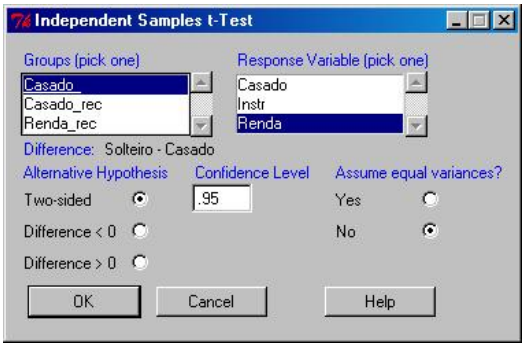
Comecemos então com o primeiro comando, `Single-sample t-test`. Vamos então fazer um teste t para nossos dados:



E o resultado do teste:

```
> t.test(aula$Renda, alternative='two.sided', mu=0.0, conf.level=.95)

One Sample t-test

data:  aula$Renda
t = 13.8454, df = 35, p-value = 9.162e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
  9.836553 13.216780
sample estimates:
mean of x
 11.52667
```

Ou seja, a Renda média é significativamente diferente de zero. Vejamos agora se a renda difere com relação ao estado civil. No R Commander usamos o menu `Independent Samples t-Test`. Escolhemos então as opções tal como aparece no box:



Primeiro informamos os grupos em que se divide a variável. Em seguida, a variável que testaremos a diferença das médias. Em baixo podemos especificar as hipóses e o suposto sobre a igualdade (ou não) da variância nos grupos. Os resultados...

```
> t.test(Renda~Casado_, alternative='two.sided', conf.level=.95,
+   var.equal=FALSE, data=aula)

Welch Two Sample t-test

data:  Renda by Casado_
t = -1.5668, df = 33.975, p-value = 0.1264
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -5.7517582  0.7440082
sample estimates:
mean in group Solteiro   mean in group Casado
              10.13562                12.63950
```

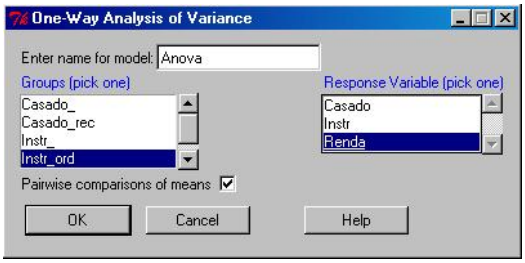
E portanto, não rejeotamos a hipóse de iguadade das médias com relação ao estado civil. Note que temos ainda um intervalo de confiança para a diferença (a $\gamma = 0.95$) e as médias nos grupos.

Para amostras pareadas, usamos o `Paired t-Test`. Usamos este teste quando observamos nossas amostras, por exemplo, antes de um tratamento e depois do tratamento[12]. Os valores são computados em variáveis diferentes

[12]Por exemplo, em um experimento sobre a opinião sobre o governo, podemos pedir

e então, testamos se as médias na primeira e na segunda diferem significativamente[13].

Nosso próximo menu é uma `One-Way ANOVA`. Vejamos as opções que temos para este tipo de modelo:



Ou seja, primeiramente devemos dar um nome para o nosso modelo. A `ANOVA` é, como você deve (ou deveria) saber, um tipo de regressão e o R Commander salva os modelos de Regressão que você computa. Devemos então informar os fatores e a variável explicada. A hipótese nula deste teste é de que a diferença das médias dos grupos é igual a zero e a hipótese alternativa é de que a diferença é diferente de zero em pelo menos um grupo. O resultado para nosso modelo:

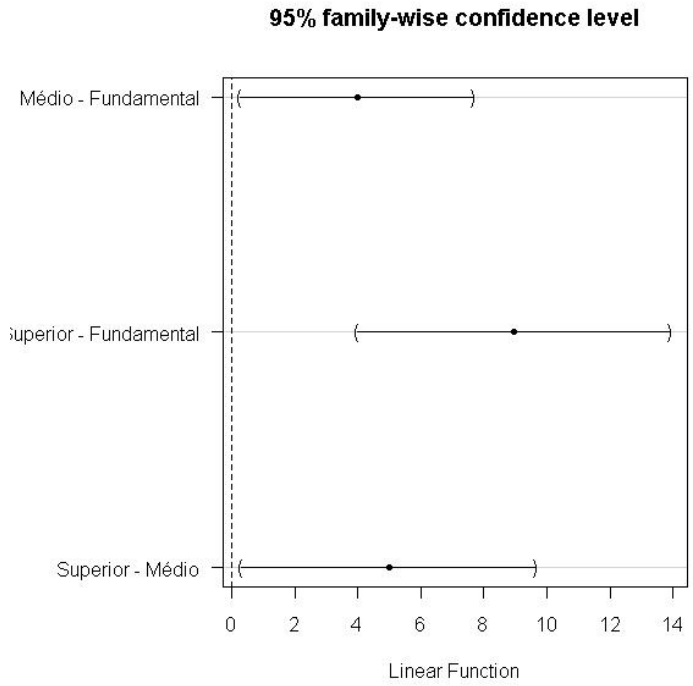
```
> Anova <- aov(Renda ~ Instr_ord, data=aula)
> summary(Anova)
             Df Sum Sq Mean Sq F value    Pr(>F)
Instr_ord     2 329.83  164.91  10.014 0.0003992 ***
Residuals    33 543.47   16.47
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> numSummary(aula$Renda , groups=aula$Instr_ord, statistics=c("mean", "sd"))
```

para um mesmo grupo de indivíduos dar uma nota para o desempenho do governo antes e depois de passarmos uma propaganda política. Testatemos assim se a propaganda afetou a média das notas.

[13]Não vamos fazer este teste porque nossos dados não preenchem o requisito necessário para tanto.

```
                 mean       sd  n
Fundamental  8.048333 3.267645 12
Médio       12.018333 4.367024 18
Superior    17.008333 4.512531  6

> .Pairs <- glht(Anova, linfct = mcp(Instr_ord = "Tukey"))
> confint(.Pairs)

 Simultaneous Confidence Intervals
Multiple Comparisons of Means: Tukey Contrasts

Fit: aov(formula = Renda ~ Instr_ord, data = aula)
Estimated Quantile = 2.4455
95% family-wise confidence level

Linear Hypotheses:
                             Estimate lwr     upr
Médio - Fundamental == 0     3.9700   0.2715  7.6685
Superior - Fundamental == 0  8.9600   3.9979 13.9221
Superior - Médio == 0        4.9900   0.3117  9.6683

> old.oma <- par(oma=c(0,5,0,0))
> plot(confint(.Pairs))
> par(old.oma)
> remove(.Pairs)
```

Primeiramente, temos a tabela da `ANOVA`. Esta tabela contém os graus de liberdade, a soma dos quadrados, a média de quadrados ($\frac{SQ}{df}$) e a estatística F para o modelo. Ao lado da estatística F temos o p-valor. As estrelas ($*$) são os níveis de significância. Logo abaixo temos o sumário numérico para os dados. Note que ligamos o botão `Pairwise comparison of means`, que serve para que, primeiro, ele compare as médias nos grupos e, depois,para que ele exiba gráfico para a diferença. O gráfico para o exemplo segue abaixo:

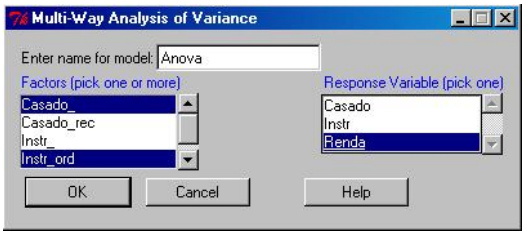


Note que, como a `ANOVA` é um modelo de regressão, ela fica salva como modelo ativo no R Commander:

Como todo modelo de regressão, vários testes podem ser feitos. Chegaremos a eles logo mais.

O ultimo comando deste sub-menu é o `Multi-Way ANOVA`. Usamos para fazer analise de variância com mais de uma variável explicativa. O modelo também computa eventuais efeitos de interação entre as variáveis. No caso, vamos estudar se há variação na Renda quando controlada pelo estado civil e pela escolaridade:

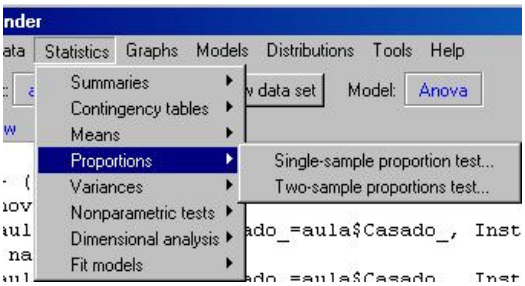
O resultado do teste é:



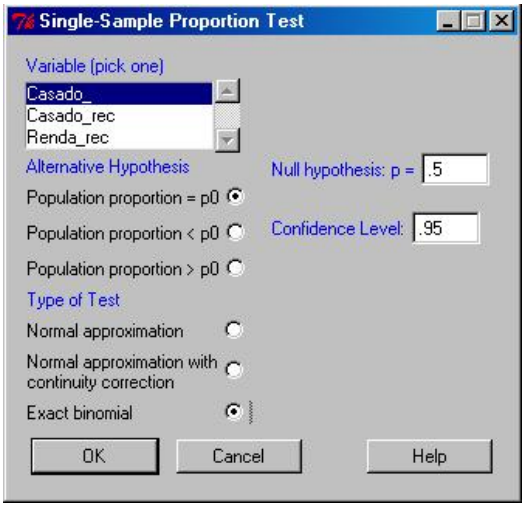

```
> Anova <- (lm(Renda ~ Casado_*Instr_ord, data=aula))
> Anova(Anova)
Anova Table (Type II tests)

Response: Renda
                  Sum Sq Df F value    Pr(>F)
Casado_            35.57  1  2.3611 0.1348765
Instr_ord         309.67  2 10.2770 0.0003985 ***
Casado_:Instr_ord  55.91  2  1.8555 0.1738848
Residuals         451.99 30
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> tapply(aula$Renda, list(Casado_=aula$Casado_, Instr_ord=aula$Instr_ord),
+   mean, na.rm=TRUE) # means
          Instr_ord
Casado_    Fundamental     Médio Superior
  Solteiro    8.635714  9.158333 15.59000
  Casado      7.226000 13.448333 18.42667

> tapply(aula$Renda, list(Casado_=aula$Casado_, Instr_ord=aula$Instr_ord), sd,
+    na.rm=TRUE) # std. deviations
          Instr_ord
Casado_    Fundamental    Médio Superior
  Solteiro    3.781141 2.315819 4.409388
  Casado      2.543330 4.510329 5.042780

> tapply(aula$Renda, list(Casado_=aula$Casado_, Instr_ord=aula$Instr_ord),
+   function(x) sum(!is.na(x))) # counts
          Instr_ord
Casado_    Fundamental Médio Superior
  Solteiro           7     6        3
  Casado             5    12        3
```

Ou seja, no modelo, a influência principal é entre Renda e Grau de Instrução. Não há relação significativa nem com relação ao estado civil, nem há interação entre este e o grau de instrução.

Nosso próximo menu diz respeito a testes de hipóteses para proporções, `Proportions`. Temos, semelhante as duas primeiras opções sobre médias, testes semelhantes mas agora para proporções. O sub-menu é o seguinte:



O primeiro comando que vamos escolher é então o `Single-sample proportion test`. Testaremos se na população que amostramos há a mesma quantidade de 'casados' e 'não casados'. Assim,



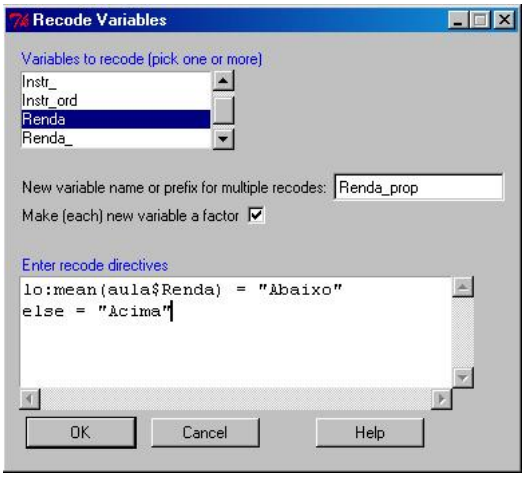
No primeiro boz nos é pedido para selecionarmos a variável que queremos testar as proporções. No box abaixo devemos definir a hipótese alternativa do teste. À direita, o intervalo de confiança e o valor do parâmetro sob a hipótese

nula. Mais abaixo, o tipo do teste. Como temos apenas 36 observações, vamos utilizar o teste binomial exato. O resultado segue abaixo.

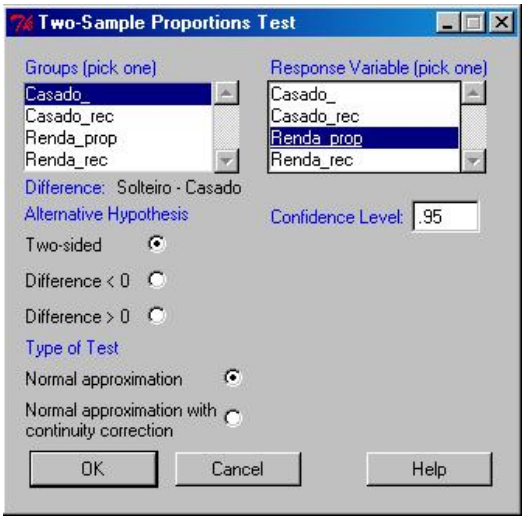
```
> .Table <- xtabs(~ Casado_ , data= aula )
> .Table
Casado_
Solteiro   Casado
      16       20

> binom.test(rbind(.Table), alternative='two.sided', p=.5, conf.level=.95)

Exact binomial test

data:  rbind(.Table)
number of successes = 16, number of trials = 36, p-value = 0.6177
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.2793542 0.6190232
sample estimates:
probability of success
             0.4444444
```

Ou seja, não há evidencias para supor que a proporção de 'casados' difira significativamente da de 'não-casados'.

Para o próximo teste vamos precisar criar uma variável que nos auxilie. Vamos segmentar a variável renda em dois grupos, até a média e acima da média. Assim, `Data` → `Manage variables in active data set` → `Recode variables` e daí:



Passando então para o teste, vamos tentar ver se há diferença na proporção de pessoas que ganham acima ou abaixo da média com relação ao estado civil. O teste:



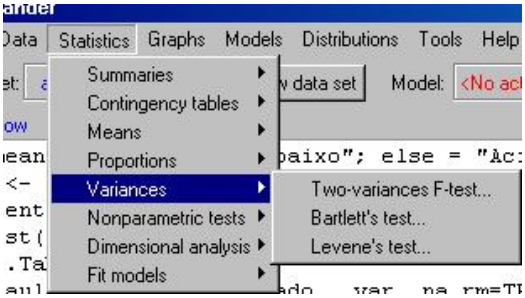
E o resultado final:

```
> .Table <- xtabs(~Casado_+Renda_prop, data=aula)
> rowPercents(.Table)
          Renda_prop
Casado_    Abaixo Acima Total Count
  Solteiro   68.8  31.2   100    16
  Casado     45.0  55.0   100    20
```

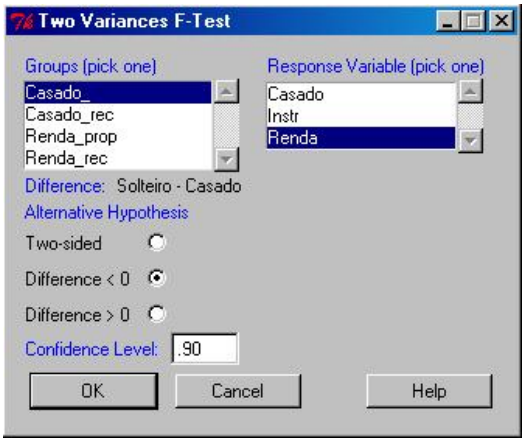
```
> prop.test(.Table, alternative='two.sided', conf.level=.95, correct=FALSE)

2-sample test for equality of proportions without continuity
correction

data:  .Table
X-squared = 2.0306, df = 1, p-value = 0.1542
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.07733351  0.55233351
sample estimates:
prop 1 prop 2
0.6875 0.4500

> remove(.Table)
```

Ou seja, não temos base para supor que a diferença seja significativa à $\alpha = 0.05$.

Vamos então para o próximo menu, `Variances`.



Temos então estes três testes a nossa disposição. O primeiro, que vamos tratar agora, serve para testarmos se há diferença de variâncias entre grupos que segmentamos nossa variável. Vejamos um exemplo:

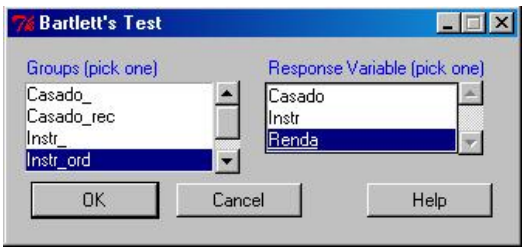


Ou seja, informamos os grupos e a variável. Em baixo, a hipótese alternativa e o intervalo de confiança. Vamos neste teste então supor que a variância na renda de pessoas casadas é maior que a variância na renda de pessoas solteiras, fixemos $\alpha = 0.10$ neste teste. O resultado:

```
> tapply(aula$Renda, aula$Casado_,  var, na.rm=TRUE)
Solteiro   Casado
17.48080 29.22955

> var.test(Renda ~ Casado_, alternative='less', conf.level=.90, data=aula)

F test to compare two variances

data:  Renda by Casado_
F = 0.5981, num df = 15, denom df = 19, p-value = 0.1581
alternative hypothesis: true ratio of variances is less than 1
90 percent confidence interval:
 0.000000 1.155498
sample estimates:
ratio of variances
         0.5980523
```

Ou seja, como o p-valor foi de aproximadamente 0.158, não rejeitamos a hipótese nula.

O teste acima impõe que o fator que agrupará a variável cuja variância queremos estudar seja binária. Para que não haja esta restrição usamos o Teste de Bartlett. Vejamos se a variância da renda em cada casela do grau de instrução é significativamente diferente.



O resultado do teste:

```
> tapply(aula$Renda, aula$Instr_ord, var, na.rm=TRUE)
Fundamental       Médio    Superior
   10.67751    19.07090    20.36294

> bartlett.test(Renda ~ Instr_ord, data=aula)

Bartlett test of homogeneity of variances

data:  Renda by Instr_ord
Bartlett's K-squared = 1.1504, df = 2, p-value = 0.5626
```

A hipótese nula é de que a variância é homogênea em cada grupo e pelo teste, a hipótese nula não foi rejeitada. Um teste com o mesmo espírito do Bartlett é o `Levene's test`. Resultado do teste para a mesma situação testada acima é:

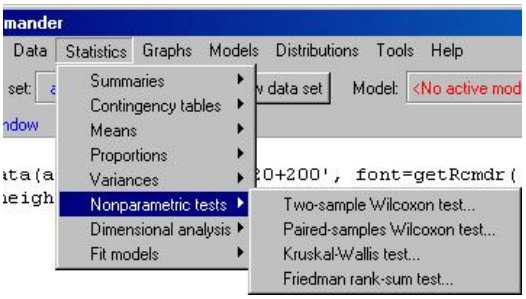
```
> tapply(aula$Renda, aula$Instr_ord, var, na.rm=TRUE)
Fundamental       Médio    Superior
   10.67751    19.07090    20.36294

> levene.test(aula$Renda, aula$Instr_ord)
Levene's Test for Homogeneity of Variance
      Df F value Pr(>F)
```

```
group  2  0.5537 0.5801
      33
```

Ou seja, algo bem semelhante. Enfim, estes são os testes básicos[14].

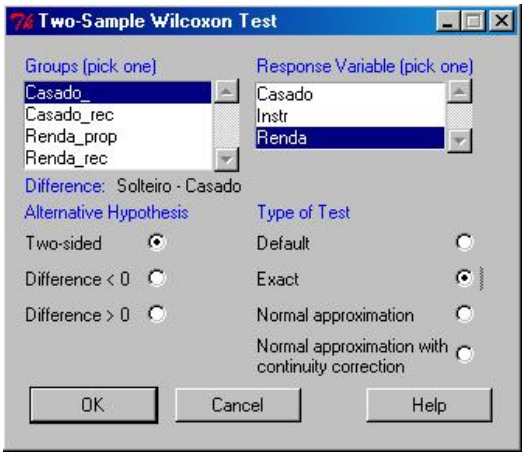
6 Estatísticas não-paramétricas no R Commander

Temos condensados no R Commander, em um menu, os principais modelos não-paramétricos. Vejamos:



Vamos então começar apresentando o primeiro teste, comparação de medianas por postos de Wilcoxon, o comando, `Two-sample Wilcoxon test...`. Assim, vamos comparar as diferenças de medianas da renda entre solteiros e casados:

[14] Caso o seu problema agora seja com as estatísticas calculadas, sugerimos dois textos bem interessantes. O primeiro, é o de Jack Levin, *Estatística Aplicada às Ciências Humanas*. O segundo, o livro dos profs. Bussab e Morettin, *Estatística Básica*. O primeiro é mais simples mas também é mais inconsistente. O segundo é o contrário...



O resultado...

```
> tapply(aula$Renda, aula$Casado_, median, na.rm=TRUE)
Solteiro   Casado
    9.36    12.41

> wilcox.test(Renda ~ Casado_, alternative='two.sided', exact=TRUE,
+   correct=FALSE, data=aula)

Wilcoxon rank sum test

data:  Renda by Casado_
W = 115, p-value = 0.1519
alternative hypothesis: true location shift is not equal to 0
```

Ou seja, não há diferença estatísticamente significativa.

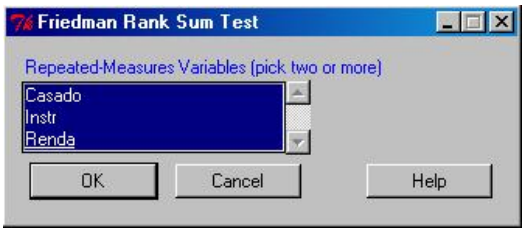
Em seguida, temos os testes pareados. Não faremos este tipo de teste pelas razões que já explicitamos mas ele é bem simples de se realizar (note a hipótese de dependência entre as amostras tomadas).

Na sequencia, vamos estudar a versão não-paramétrica da análise de variância (ANOVA) o teste de Kruskal-Wallis. Assim:

Novamente estamos lidando com medianas ao invés de médias. O resultado do teste...


```
> tapply(aula$Renda, aula$Instr_ord, median, na.rm=TRUE)
Fundamental       Médio    Superior
      7.125      11.325      17.680

> kruskal.test(Renda ~ Instr_ord, data=aula)

Kruskal-Wallis rank sum test

data:  Renda by Instr_ord
Kruskal-Wallis chi-squared = 13.8281, df = 2, p-value = 0.0009937
```

Por fim, a MANOVA não paramétrica:



O resultado:

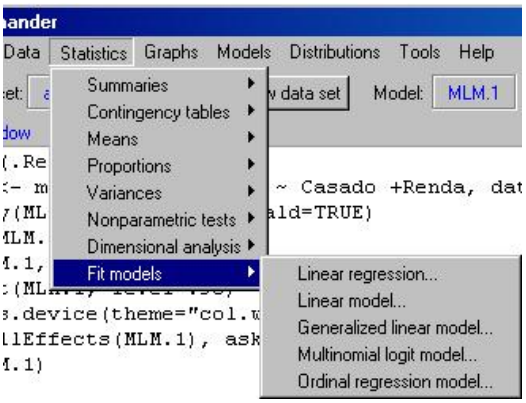
```
> .Responses <- na.omit(with(aula, cbind(Casado, Instr, Renda)))

> apply(.Responses, 2, median)
Casado  Instr  Renda
 1.000  1.000 10.645

> friedman.test(.Responses)

Friedman rank sum test

data:  .Responses
```

```
Friedman chi-squared = 62.992, df = 2, p-value = 2.096e-14

> remove(.Responses)
```

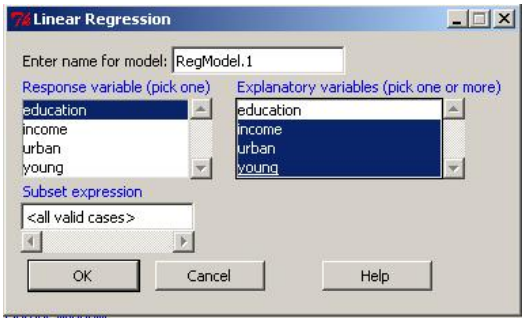
Na prática, isso já ajuda bem... Veremos mais a frente, alguns *plugins* (que não serão ensinados) que contém uma série de outros testes.

7 Modelos de regressão usando o R Commander

O primeiro passo da análise de regressão é formularmos um modelo. Nosso problema aqui não será apresentarmos nada teórico sobre o assunto e o leitor que estiver interessado pode procurar Gujarati (2006) para mais informações. Para regressão no R Commander, ajustamos o modelo com os seguintes comandos:



Vamos mostrar neste exemplo como fazer uma regressão linear multipla. O espírito pode ser aplicado igualmente para outros tipos de modelos. Carregue então o banco `Anscombe`, do pacote '`car`'. Vamos fazer um modelo de regressão em que vamos estimar o gasto com educação em função da renda, da proporção de jovens e da proporção urbana. O modelo então é feito da seguinte forma:

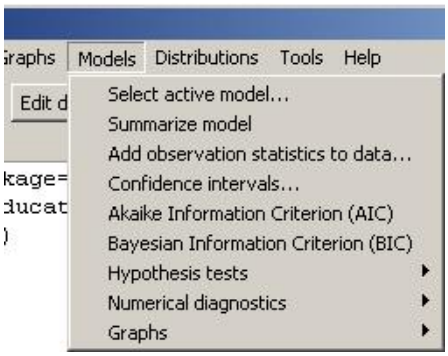


O modelo é salvo automaticamente com um nome do tipo `RegModel.n`. Por este nome podemos encontrar o modelo depois, se for o caso de usarmos ele novamente depois da análise. No primeiro box, escolhemos a variável explicada (só uma) e no seguinte, as explicativas, que pode ser mais de uma. Note em baixo que temos a opção de segmentar os dados de nossa regressão. Note ainda que no canto superior do seu `R Commander` vai aparecer o box `Model`, agora preenchido com o nome do modelo que está válido, que é o que acabamos de criar. Se você tiver feito mais de um modelo, assim como no box do banco de dados, você pode encontra-lo facilmente clicando no conteúdo em frente ao box `Model`. O resultado para o nosso modelo é apresentado pelo R logo após a confirmação do comando.

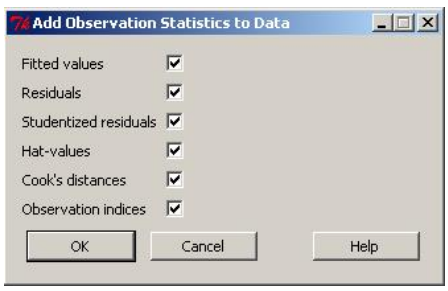
```
> RegModel.1 <- lm(education~income+urban+young, data=Anscombe)

> summary(RegModel.1)

Call:
lm(formula = education ~ income + urban + young, data = Anscombe)

Residuals:
    Min      1Q  Median      3Q     Max
-60.240 -15.738  -1.156  15.883  51.380

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.868e+02  6.492e+01  -4.418 5.82e-05 ***
income       8.065e-02  9.299e-03   8.674 2.56e-11 ***
```

```
urban       -1.058e-01  3.428e-02  -3.086  0.00339 **
young        8.173e-01  1.598e-01   5.115 5.69e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 26.69 on 47 degrees of freedom
Multiple R-squared: 0.6896,Adjusted R-squared: 0.6698
F-statistic: 34.81 on 3 and 47 DF,  p-value: 5.337e-12
```

Passemos então aos diagnósticos do ajuste. Nos concentraremos, depois da montagem do modelo, no menu `Models`:



O primeiro comando, `Select active model...` serve para selecionarmos o modelo em que as estatísticas serão aplicadas. O segundo comando, apresenta as estatísticas que são apresentadas quando pedimos para o sistema calcular a regressão. Já o terceiro comando, `Add observation statistics to data`, serve para que salvemos os diversos subprodutos da regressão no modelo. Podemos salvar os seguintes resultados:



Ou seja, podemos salvar os valores ajustados, os resíduos, os resíduos studentizados, os hatvalues, as distâncias de Cook e os índices. Você pode adicionar ao banco de dados os valores que você achar necessários. Não vamos fazer isso neste banco.

No menu seguinte, `Confidence Intervals...`, podemos pedir para o `R Commander` calcular os intervalos de confiança das estimativas, para um dado γ escolhido pelo usuário. Para $\gamma = 0.98$ temos:

E a resposta para o nível de confiança acima é:

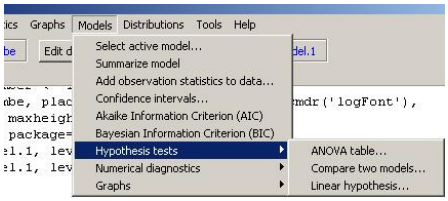
```
> Confint(RegModel.1, level=.98)
                       1 %          99 %
(Intercept) -443.18835849 -130.48916697
income         0.05825917    0.10304734
urban         -0.18836953   -0.02324293
young          0.43250901    1.20216647
```

Abaixo temos o AIC (*Akaike Information Criteria*) e o BIC (*Bayesian Information Criteria*), dois critérios que assim como o R^2 *Ajustado* criam barreiras contra a tentativa de *data mining*. Diferente do R^2 *Ajustado*, estes critérios devem ser minimizados e só fazem sentido em comparação com outro deles no mesmo modelo. Para o nosso ajuste:

```
> AIC(RegModel.1)
[1] 485.5767

> AIC(RegModel.1, k = log(nobs(RegModel.1))) # BIC
[1] 495.2359
```

Note que o valor do BIC é sempre maior que o do AIC independente da transformação. Isso denota que ele é ligeiramente mais rigoroso[15].

No próximo menu, temos alguns testes de hipóteses e comparações entre modelos. O menu tem as seguintes opções:



Primeira coisa que vamos pedir é uma tabela de ANOVA para o modelo. Enquanto no sumário temos os testes `t` para cada uma das hipóteses, na Anova temos o teste de F, que podemos fazer para todo o modelo, com $H_0 = b_i = 0, \forall\, i = 1, ..., n$. Se alguma variável tiver coeficiente de inclinação diferente de zero, então a ANOVA fica significativa, independente de qual variável estamos nos referindo. No R Commander, ele nos adianta qual a variável. Para nosso modelo:

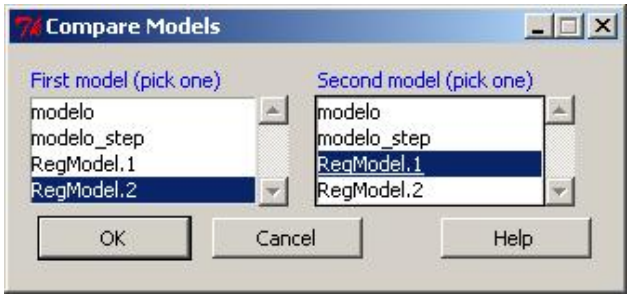
```
> Anova(RegModel.1, type="II")
Anova Table (Type II tests)

Response: education
          Sum Sq Df F value    Pr(>F)
income     53607  1 75.2341 2.564e-11 ***
urban       6787  1  9.5255  0.003393 **
young      18643  1 26.1641 5.695e-06 ***
Residuals  33489 47
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note que temos três tipos de Anova e, com alguma alteração, podemos fazer testes de Wald e de Qui-Quadrado da razão de maxima verossimiliança (*maximum likelihood*)[16]. O resultado é para o segundo tipo.

[15]Para encontrar a formula basta dar um `help(AIC)` no console do R.

[16]Mais informações, basta dar `help(Anova)` no console do R.

O próximo teste nos permite comparar dois ajustes diferentes. Vamos fazer um modelo, `RegModel.2`, onde aplicamos o mesmo modelo mas excluindo a variável `income`. Como o sistema mostrou que esta variável é importante, o teste nos deverá dizer que o modelo fica mais completo quando a incluimos. Assim,



A hipótese nula do teste é a de que o modelo sem a variável, o `RegModel.2`, já está suficiente (por isso escolhemos ele primeiro). A hipótese alternativa nos diz que o modelo `RegModel.1` está mais completo.

```
> anova(RegModel.2,RegModel.1)
Analysis of Variance Table

Model 1: education ~ urban + young
Model 2: education ~ income + urban + young
  Res.Df   RSS Df Sum of Sq      F    Pr(>F)
1     48 87097
2     47 33489  1     53607 75.234 2.564e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

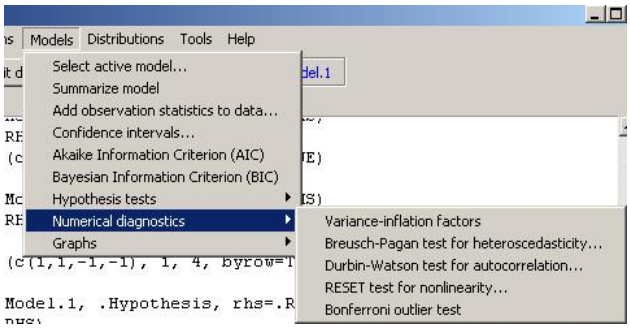
Como o teste foi significante, rejeitamos a hipótese nula e, portanto, o modelo fica mais completo com `income`. Note que os teste AIC e BIC são também melhores (menores) para o `RegModel.1` que para o `RegModel.2`[17].

```
> AIC(RegModel.1)
```

[17] Não se esqueça de selecionar novamente o `RegModel.1`.

```
[1] 485.5767

> AIC(RegModel.1, k = log(nobs(RegModel.1))) # BIC
[1] 495.2359

> AIC(RegModel.2)
[1] 532.322

> AIC(RegModel.2, k = log(nobs(RegModel.2))) # BIC
[1] 540.0493
```

Quanto ao teste de hipotese linear, não vamos tratar aqui. Passemos ao menu `Numerical Diagnostics...`.



Começando com o VIF, *variance inflating factor*, este diagnostico nos dá uma noção de quanto de multicolinearidade temos em nosso modelo. Não existe uma faixa de corte precisa, mas um dos efeitos de um valor muito grande do VIF em seu modelo seria de que a ANOVA mostraria o modelo como significante enquanto os testes t não seriam significantes para as variáveis individualmente. Para este modelo:

```
> vif(RegModel.1)
  income    urban    young
1.902860 1.889004 1.028564
```

Em seguida, temos o teste de Breusch-Pagan para heterocedasticidade. A hipótese nula deste teste é a de que há homocedasticidade. No caso de não

haver (haver heterocedasticidade), não podemos garantir que a estimação do modelo é BLUE (*Best Linear Unbiased Estimator*). O resultado do teste para o nosso modelo é:

```
> bptest(education ~ income + urban + young, varformula = ~
+   fitted.values(RegModel.1), studentize=FALSE, data=Anscombe)

Breusch-Pagan test

data:  education ~ income + urban + young
BP = 2.6384, df = 1, p-value = 0.1043
```

Note que testamos para os valores ajustados da regressão. Podemos testar para outras variáveis (explicativas) ou mesmo para outras especificações. Podemos usar ainda os valores estudentizados, o que torna o teste mais sensível (e preciso).

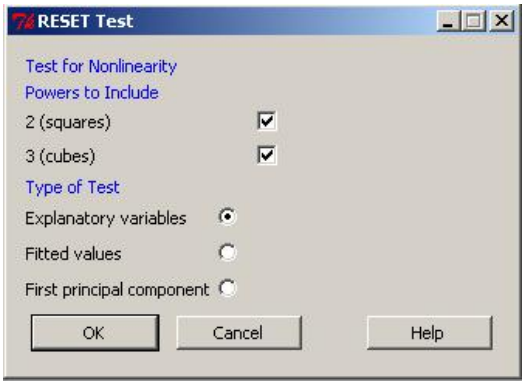
O teste seguinte, Durbin-Watson, serve para testarmos autocorrelação. Como não tratamos de séries de tempo nesta apostila, deixemos este teste de lado[18].

O teste seguinte diz respeito à especificação do modelo. Testamos basicamente se o modelo é do tipo $Y_i = a + \sum_{i=1}^{n} b_i X_i + \epsilon_i$ ou de outra forma funcional. O menu que aparece tem algumas alternativas para formatar o teste:

O resultado para o nosso modelo fica:

```
> resettest(education ~ income + urban + young, power=2:3, type="regressor",
+   data=Anscombe)

RESET test
```

[18]Você pode escolher qual a hipótese alternativa neste teste (aparece em um menu). Apesar de não tratarmos aqui, este teste é muito importante para que lida com séries temporais



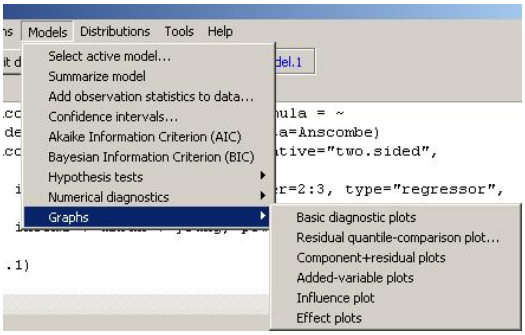

```
data:  education ~ income + urban + young
RESET = 2.1998, df1 = 6, df2 = 41, p-value = 0.06243
```

A hipótese nula é de que o modelo está bem especificado. No caso, a julgar pelo p-valor, o modelo está.

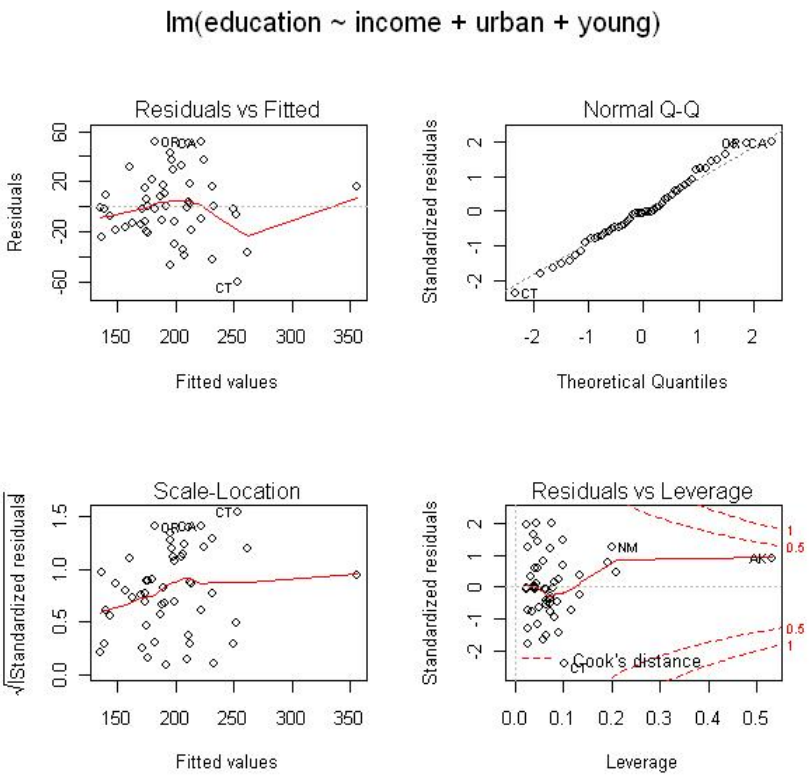
O ultimo diagnostico seria de Bonferoni, para *outliers*. A hipótese nula deste teste é de que não existem, supondo normalidade, dados muito discrepantes no modelo. Para nosso ajuste, o resultado deste teste é:

```
> outlier.test(RegModel.1)

max|rstudent| = 2.513899, degrees of freedom = 46,
unadjusted p = 0.01550359, Bonferroni p = 0.7906829

Observation: CT
```

Por fim, temos o menu `Graphs`. Nele podemos tirar alguns diagnosticos gráficos de nosso modelo:



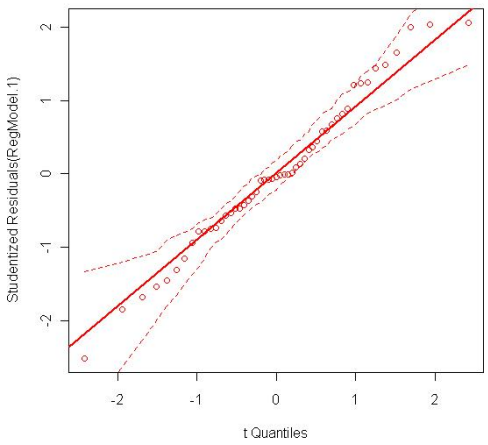
Começemos então com o primeiro menu, `Basic diagnostic plots...`. O resultado para esta opção é:



No primeiro quadro temos o resíduo contra o ajustado. Nos dá uma idéia de quanto bom foi nosso ajuste. No segundo, um QQ-Plot dos quantis dos resíduos padronizados contra os de uma normal teórica. Para afirmarmos

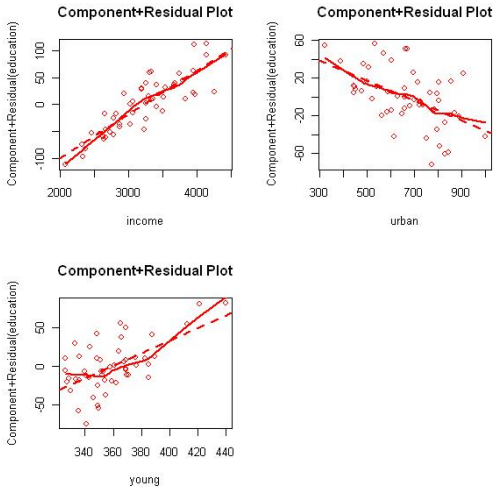
normalidade, eles devem estar o mais próximo possível da reta tracejada. No terceiro, o resíduo contra a raiz dos valores absolutos dos resíduos contra o ajustado. Nos dá uma idéia de quão discrepantes (ou não) são os dados. No ultimo, o resíduo contra o Leverage. Nos dá, assim como no terceiro gráfico, uma noção de quais pontos seriam os candidados a pontos de alavancagem do modelo.

O próximo menu, `Residual Quantile-Comparison plot...`, nos plota o gráfico QQ-Plot[19], com a vantagem de vir com os intervalos de confiança e de podermos nomear, com o mouse, os pontos que acharmos interessante. Um exemplo, para nosso ajuste:

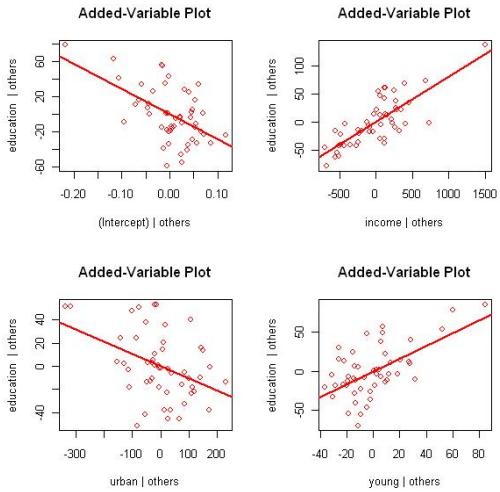


No gráfico seguinte, podemos ter a noção de quanto cada variável contribui para o ajuste do modelo individualmente. Segue o resultado para o nosso ajuste:

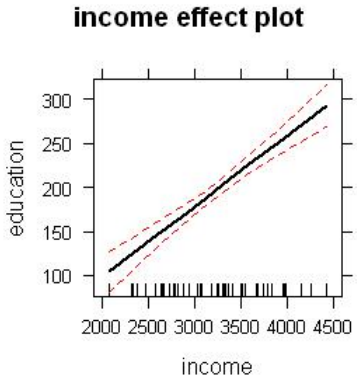
[19]Note que aqui não comparamos com uma normal e sim com uma `t`.

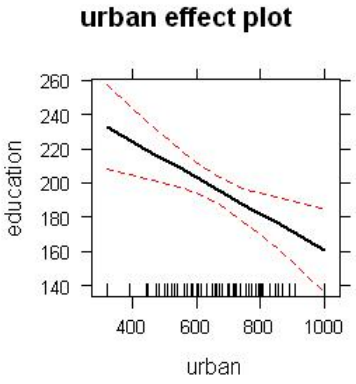


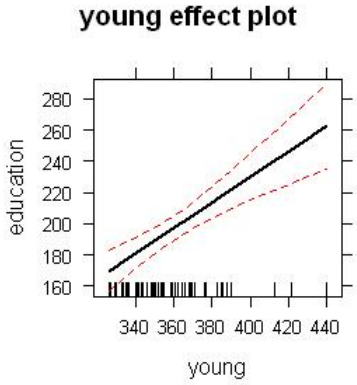
O próximo gráfico segue o mesmo princípio do anterior. Ambos chamamos de regressões parciais, pois seriam a contribuição de cada variável para o ajuste final do modelo.



Por fim, passemos para o ultimo menu, `Effect Plots`, em que plotamos as retas parciais juntamente com seus intervalos de confiança. O gráfico fica bem interessante a adiciona aos gráficos anteriores, o intervalo de confiança para a média dos valores. Para o nosso ajuste:


income effect plot
education
300
250
200
150
100
2000 2500 3000 3500 4000 4500
income


urban effect plot
education
260
240
220
200
180
160
140
400 600 800 1000
urban


young effect plot
education
280
260
240
220
200
180
160
340 360 380 400 420 440
young

8 Anexo

Tabela usada nos Exemplos

	Casado	Instr	Renda		Casado	Instr	Renda
1	0.00	0.00	4.00	19	0.00	2.00	10.76
2	1.00	0.00	4.56	20	0.00	1.00	11.06
3	1.00	0.00	5.25	21	1.00	1.00	11.59
4	0.00	1.00	5.73	22	0.00	1.00	12.00
5	0.00	0.00	6.26	23	0.00	0.00	12.79
6	1.00	0.00	6.66	24	1.00	2.00	13.23
7	0.00	0.00	6.86	25	1.00	1.00	13.60
8	0.00	0.00	7.39	26	1.00	1.00	13.85
9	1.00	1.00	7.59	27	0.00	0.00	14.69
10	0.00	1.00	7.44	28	1.00	1.00	14.71
11	1.00	1.00	8.12	29	1.00	1.00	15.99
12	0.00	0.00	8.46	30	1.00	1.00	16.22
13	0.00	1.00	8.95	31	0.00	2.00	16.61
14	1.00	0.00	9.13	32	1.00	1.00	17.26
15	1.00	1.00	9.35	33	1.00	2.00	18.75
16	0.00	1.00	9.77	34	0.00	2.00	19.40
17	1.00	1.00	9.80	35	1.00	1.00	23.30
18	1.00	0.00	10.53	36	1.00	2.00	23.30