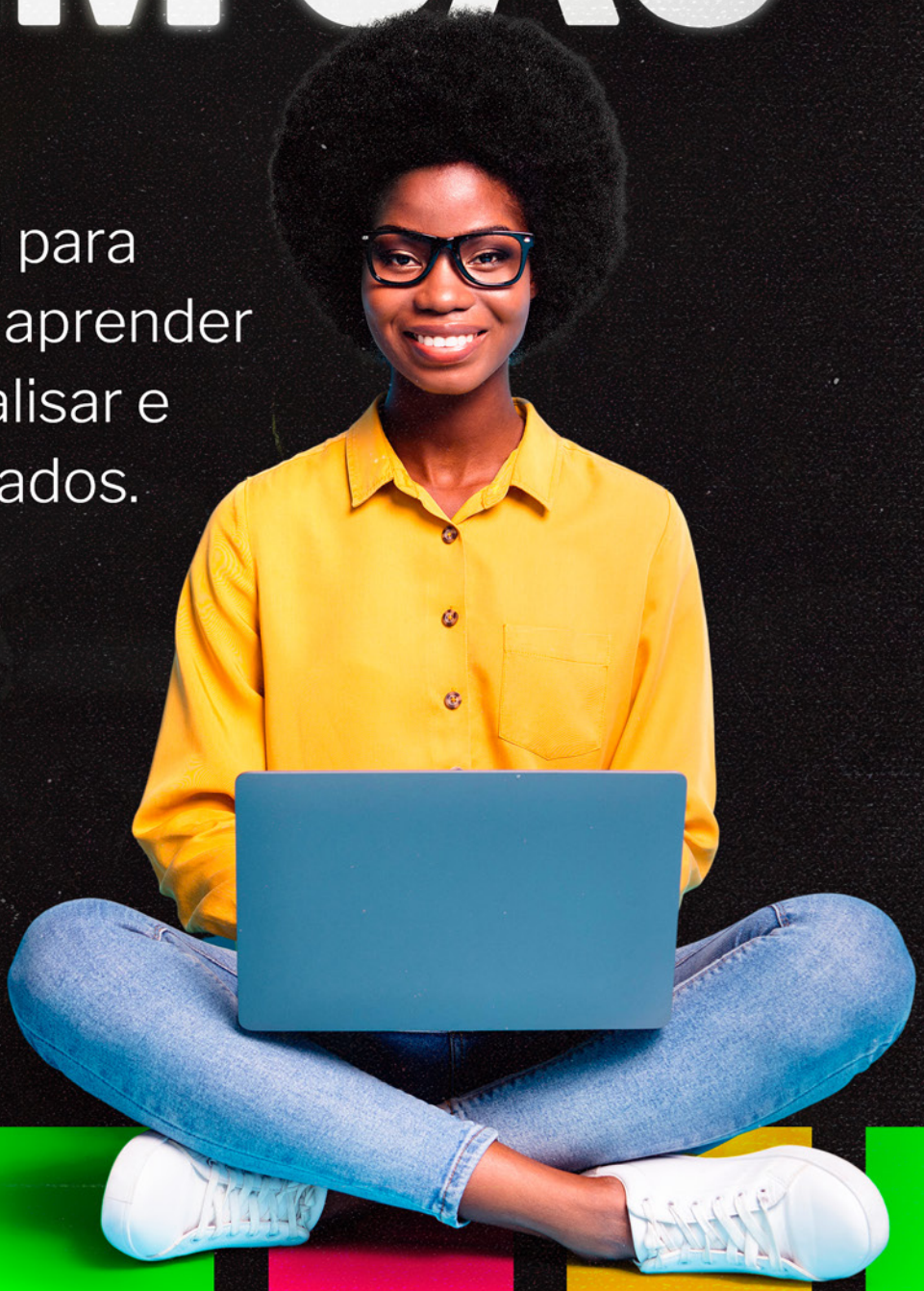
DADOS SEM CAÔ

Um manual para quem quer aprender a tratar, analisar e visualizar dados.



data_ labe

ÍNDICE

data
labe

Olha quem chegou! 1
Google, Pesquisar 2
Buscas além do Google 4
O que é base de dados e microdados? 5
Mas, afinal, quem é CAGED? 9
O tal do R: Software para tratar os dados 10
Trazendo a base de dados para o R 12
Reconhecendo a base 17
Mudando o tipo de dado 19
Um pacote para gerar tabelas 20
Juntando duas bases de dados 23
Tirando pedaços de dados 24
Trocando os valores pelo dicionário 26
Reduzindo a base 30
Exportando a base que você trabalhou 31
Salvando o script 32
Retirando informações da sua base de dados 33
Quando fazemos um gráfico é preciso entender alguns pontos chaves 36
Médias e medianas 39
Categorizando variável numérica 41
Análises específicas apenas dos trabalhadores com deficiência 43
Gerando mapas 44
Criando um gráfico interativo 50
Conclusão 51
Notas finais 52
Agradecimentos 53

Olha quem chegou!

Você já quis falar de um assunto específico e ficou frustrado com a ausência de informações novas e relevantes?

O data_labe surgiu em 2016 para buscar e fornecer essas informações.

> **Se a gente tivesse um dado que comprovasse que tem esgoto a céu aberto nas localidades da Maré, seria tudo!**

Na maioria das vezes, o dado existe. Ele só tá disponível de um jeito que pouca gente tem acesso ou que quase ninguém sabe como acessar.

Mas aqui nesta cartilha a gente vai te ensinar a buscar todas as informações possíveis em uma base de dados que possam contribuir para suas reportagens, conteúdos ou onde você quiser usá-las.

Vamos partir da experiência na nossa última formação de jornalismo, na qual produzimos o relatório de dados "Pessoas com Deficiência e Empregabilidade". Com base nesse passo a passo, acreditamos que mesmo quem nunca trabalhou com dados poderá reproduzir o processo nas diversas bases de dados disponíveis no Brasil.

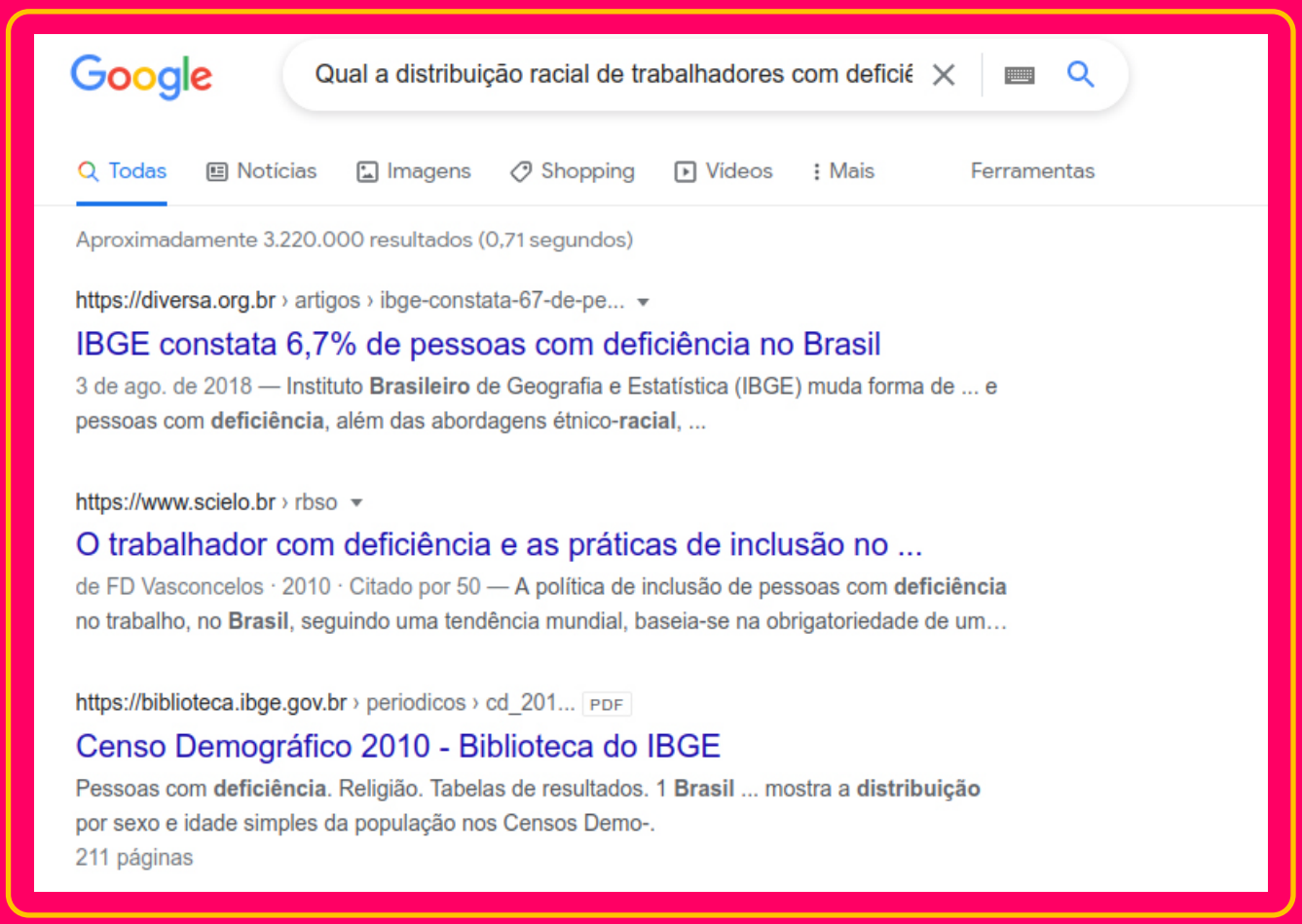
Tem muita gente braba nesse rolé de dados e jornalismo[1]. A gente tá aqui pra somar nessa conversa, compartilhando as ferramentas que usamos no nosso dia a dia. Então, pega o computador, os caderninhos, põe uma folha de arruda na orelha direita (para dar sorte) e bora aprender como TRATAR, ANALISAR e VISUALIZAR dados.

[1] Aqui você encontra um livro muito bom da Escola de Dados com mais detalhes sobre o trabalho de dados em geral.

Google, Pesquisar

Para a análise sobre empregabilidade e pessoas com deficiência (PCD's) nós começamos a pesquisar o tema no Google e digitamos palavras-chave ou perguntas como esta a seguir:

"Qual a distribuição racial de trabalhadores com deficiência no Brasil?"



Veja que, para essa pergunta muito direta sobre raça e PCD's, o Google retorna mais de 3 milhões de links. Já nas 3 respostas do topo, que deveriam ser as que mais se encaixam com as palavras que colocamos, percebemos que, com exceção do primeiro, os links não retornam a palavra **raça**.

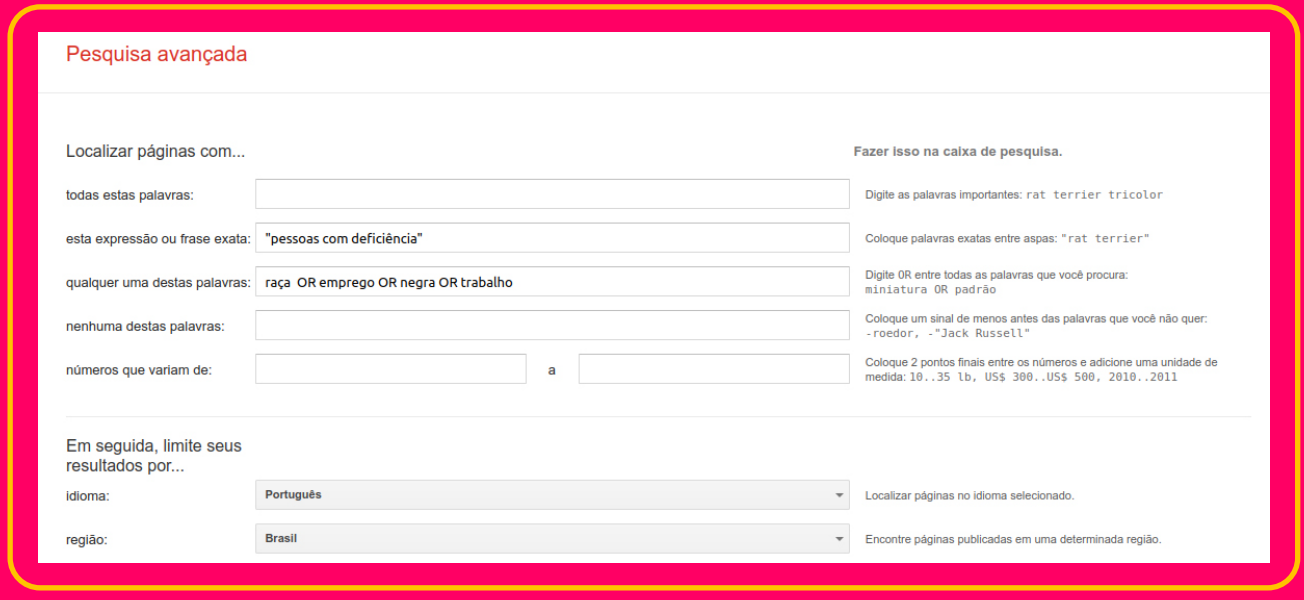
Para checar se os link's possuem a informação desejada, você precisaria entrar em cada link manualmente e ler todos eles. Porém, uma ferramenta torna o processo mais prático: a pesquisa avançada do Google.

ACESSE O LINK
https://www.google.com/advanced_search
para entender quais campos vão te ajudar a buscar o que precisa. A própria página já te indica como preencher cada campo.

No nosso caso, deixamos o campo "todas essas palavras" **vazio**, porque o que queríamos era restringir mais nossa pesquisa. Preenchemos, então: "esta expressão ou frase exata" com "**pessoas com deficiência**", porque obrigatoriamente só queremos conteúdos que tratem deste público. Como estamos buscando um conjunto de palavras, é muito importante que elas estejam entre aspas duplas ("); caso contrário, a busca procuraria qualquer uma das palavras.

Em seguida, no campo "quaisquer uma destas palavras" incluímos palavras relacionadas a nossa busca sempre separando pelo termo "OR", que em inglês significa "ou": "**raça OR emprego OR negra OR trabalho**"

Por fim, selecionamos no campo "Idioma" **Português** e, no campo "região", **Brasil**. Ainda é possível restringir sua busca por tipos de documentos em sites específicos como .gov ou .edu. Essas buscas ajudam a restringir a busca, assim a chance de você encontrar o que procura é maior. A partir daí, é colocar seus talentos de comunicador pra jogo para ler os arquivos, selecionar as fontes que confia e registrar tudo o que lhe interessa.



Buscas além do Google

Nem só de Google vive a pesquisa de dados. Caso você siga na investigação do jeito que começamos, sentirá falta de informações mais detalhadas.
Por exemplo, um ponto central para nossas análises é **verificar se existem desigualdades de raça ou gênero**.
Isso pode ser escrito dessa forma:

> **“Trabalhadores negros com deficiência possuem as mesmas condições de emprego do que trabalhadores brancos com deficiência? Ou ainda, qual o grupo social que ganha os maiores salários entre as pessoas com deficiência?”**

Essas perguntas possuem muitos elementos que, em geral, a gente aqui do data_labe só consegue achar quando temos acesso à **base de dados oficial** com seus **microdados**. Na maioria das vezes, as matérias são feitas baseadas em **relatórios**, ou seja, uma pessoa ou um grupo de pessoas já analisou a base de dados previamente e publicou os dados que considerou mais importantes.

Esta escolha pode fazer com que dados importantes fiquem de fora da publicação final.

Mais do que uma preocupação com o conteúdo, ampliar o conhecimento sobre análise de dados é um compromisso com a democracia.
É fundamental que mais pessoas que sempre foram excluídas do debate público ou consideradas minorias sociais entendam de dados. Nós que somos negros, da periferia, das comunidades mais vulneráveis socialmente, conseguimos olhar os dados a partir da nossa realidade e, assim, propor outras narrativas.
Então, bora conhecer mais dois pontos importantes.

O que é base de dados e microdados?

Você lembra das muitas histórias que seu avô ou uma tia de mais idade contavam? Tinham uma riqueza de detalhes e informações:

> **Eu tinha uns 20 anos (idade)...**
>
> **Eu ainda morava no interior do Amazonas (localização)...**
>
> **Eu nem conhecia sua avó (status civil)...**
>
> **Na época, eu trabalhava numa fazenda e ganhava uns 200 cruzados (profissão e renda).**

Veja quantos dados a gente pode extrair de apenas uma história!

A idade pode variar de 0 anos até o limite máximo de anos de vida.

A localização pode ser transformada em pontos geográficos para marcar num mapa.

O status civil pode ser registrado das mais diversas categorias que nossa legislação permite, além daquelas que a gente inventa: solteiro, casado, viúvo, enrolado, enrolando, etc.

Profissão é um outro dado que permite diversas categorias de preenchimento. E por último renda, uma informação que, para pelo menos 50% dos brasileiros, vai variar entre R$0,00 a R$995,00, atualmente.
[FONTE: BBC.COM]

E com certeza, a maioria dos avôs que contam histórias tem uma infinidade de casos e causos. O que de certa forma pode transformar seu avô numa BASE DE DADOS!

Se a gente considerar que cada história pudesse ser contada em uma linha, e que cada parte dessa linha pudesse ser resumida em um item de uma coluna, a gente teria algo mais ou menos assim:

Número da história	Data que ele contou	Idade na época	Localização	Status civil	Profissão	Renda
1	01/07/1990	20	Parintins/AM	Solteiro	Ajudante de fazenda	Cr$ 200,00

O acúmulo das informações de forma sistematizada e organizada pode ser apresentado em formato de **planilha**, ou como costumamos falar, em uma **base de dados**. Geralmente vai ter uma primeira linha (a que está marcada em negrito) que serve como um **cabeçalho**, e todas as demais linhas (que seriam as histórias) são os valores ou os dados dessa base. Cada diferente tipo desses **valores** pode ser considerado uma **variável**.

Então calma, cada história é uma linha, cada linha tem vários valores (variável), a reunião de todas as linhas, com um cabeçalho pode ser considerado uma base de dados.

Pronto! Agora você entendeu de forma prática como uma base de dados é composta com os microdados que são todas essas histórias que juntamos com suas devidas informações. Porém, conforme a gente for reunindo mais informações, você terá mais microdados para apresentar e acaba sendo mais rápido se você mostrar o dado de forma **agregada**.

Vamos voltar ao nosso vovô. A base de dados de histórias dele tem 1.000 histórias, a renda média dele por todo o período foi de Cr$1.200, variando de no mínimo Cr$50 a no máximo Cr$2.300. Veja como nós perdemos todos os detalhes. Sobre a renda, por exemplo, temos o valor médio, o valor mínimo e o máximo, mas jamais saberíamos que alguma vez ele ganhou exatamente Cr$200,00 ou por onde ele estava quando ganhou cada quantia dessa.

Percebemos a mesma perda de detalhes no caso dos dados de empregabilidade e PCD's. A maioria dos relatórios retorna a proporção de trabalhadores registrados que possuem deficiência a cada ano, mas não conseguimos achar quantos são mulheres, dados de raça/cor negra, etc. Contudo, seria possível obter essas informações se a gente tivesse acesso à **base de dados oficial** com todas as informações e os microdados.

O Brasil possui uma longa trajetória na disponibilização de dados oficiais com os microdados. Isso ajuda a sociedade civil e pesquisadores a monitorarem políticas públicas e acompanharem o debate de forma transparente e ética. Existem várias organizações e ativistas que dedicam muito do seu tempo e energia a ações sobre essa causa, uma das mais gigantes é o projeto **Base dos Dados**.

Essa plataforma é como se fosse um pote de ouro no final do arco-íris. Ela possui muitas das bases de dados disponíveis para download. Quem já possui alguma habilidade ou experiência com programação ainda consegue acessar direto da plataforma. Fica nosso agradecimento à galera que consegue, sem acesso à base de dados e seus microdados, montar narrativas completas, e que nos permitiu saber que existe uma base de dados como o CAGED, com todas as informações que a gente poderia usar na nossa produção sobre pessoas com deficiência e empregabilidade.

CAGEG é a sigla para Cadastro Geral de Empregados e Desempregados (CAGED), cujas informações estão disponíveis na Base dos Dados. Nós digitamos CAGED na barra de buscas dessa plataforma incrível e encontramos 4 bases de dados disponíveis para download, uma descrição de cada uma delas e mais informações. Para saber que informações são essas, basta acessar o dicionário de dados do CAGED.

Cada base oficial precisa vir com o seu dicionário, porque muitas informações não são disponibilizadas no formato de análise. Por exemplo, é comum que os campos de gênero (comumente chamado de sexo) venham com os códigos 1, 2 e 9. Só é possível saber o que significa cada um dos códigos com o seu dicionário. É igual um manual de instruções de uma máquina nova. Você até conseguiria botar pra funcionar sem ler o manual, mas é bem capaz de dar algum caô. Para ter certeza de que você não vai colocar nenhum código errado nem trocar os valores de “com deficiência” para “sem deficiência” só porque você já tinha experiência com outra base parecida, sempre busque pelo **dicionário de dados**.

Aliás, fica a dica: O trabalho com dados é muito incrível, mas ele exige uma certa rigidez na hora da análise e tratamento. Por isso, antes de qualquer processo vamos questionar essa base.

Mas, afinal, quem é CAGED?

É uma base de dados que compila informações de três outros sistemas eSocial, Caged e Empregador Web, que em 2020 sofre uma mudança na forma do registro ou da compilação da base de dados, isso é fundamental se nossa análise for fazer um levantamento de longo tempo. Porém as informações existem desde 1965, mas só a partir de 2001 é que há a extinção do registro feito de forma manual e impressa. A forma de coleta e envio de informações foi trocada nesse tempo, isso necessariamente impacta na qualidade dos dados, na própria forma de registro e outros impactos. Saber disso é fundamental para o analista de dados poder explicar alguma mudança muito brusca na tendência dos dados.

As informações são compiladas pela SEPRT (Secretaria Especial de Previdência e Trabalho), o que garante uma maior confiabilidade e coerência das informações. Não é a informação bruta prestada pelas empresas que estão disponíveis na base de dados, mas sim informações que foram enviadas e depois tratadas e consolidadas pela SEPRT. Os dados disponibilizados no CAGED e já compilados pela Base dos Dados são os seguintes:

Coluna

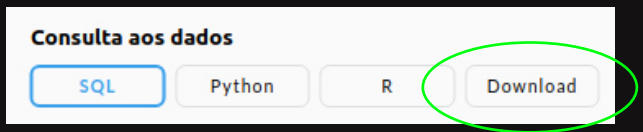
nome	descrição
ano	Ano
mes	Mês
sigla_uf	Sigla da Unidade da Federação
id_municipio	ID Município - IBGE 7 Dígitos
id_municipio_6	ID Município - IBGE 6 Dígitos
cnae_2	Classificação Nacional de Atividades Econômicas (CNAE) 2.0
cnae_2_subclasse	Classificação Nacional de Atividades Econômicas (CNAE) 2.0 Subclasse
cnae_2_secao	Classificação Nacional de Atividades Econômicas (CNAE) 2.0 Seção
cbo_2002	Classificação Brasileira de Ocupações (CBO) 2002
saldo_movimentacao	Saldo de movimentação (1 para admissão e -1 para desligamento)
categoria	Categoria de trabalhador
grau_instrucao	Grau de instrução ou escolaridade
idade	Idade do trabalhador (quando acumulada representa a soma das idades)
horas_contratuais	Horas contratuais
raca_cor	Raça e Cor do Trabalhador
sexo	Sexo (1-Homem, 3-Mulher e 9-Não Identificado)
tipo_empregador	Tipo de empregador (0-CNPJ RAIZ, 2-CPF e 9-Não Identificado)
tipo_estabelecimento	Tipo de estabelecimento (1-CNPJ, 3-CAEPF(Cadastro de Atividade Econômica de Pessoa Física), 4-CNO(Cadastro Nacional de Obra), 5-CEI(CAGED), 9-Não Identificado)
tipo_movimentacao	Tipo de movimentação
tipo_deficiencia	Tipo de deficiência/Beneficiário habilitado
indicador_trabalho_intermitente	Indicador de trabalhador intermitente (0-Não, 1-Sim, 9-Não Identificado)
indicador_trabalho_parcial	Indicador de movimentação referente a contrato parcial (0-Não, 1-Sim, 9-Não Identificado)
salario_mensal	Salário mensal em moeda corrente (R$)
tamanho_estabelecimento_janeiro	Faixa de tamanho do estabelecimento no início do ano
indicador_aprendiz	Indicador de trabalhador aprendiz (0-Não, 1-Sim, 9-Não Identificado)
fonte	Fonte da movimentação

Observe que, apesar de termos dados diretos como o ano e o mês, não encontraremos nomes dos municípios nem das unidades federativas, e sim os códigos sigla_uf e o id_município. Isso acontece com outros dados também. Para essas variáveis que possuem apenas um código e não o conteúdo referido será preciso fazer um tratamento de dados, para que ao invés do código 330455 a gente tenha “Rio de Janeiro”.

Você deve ter sacado também que o CAGED oferece até mesmo dados de outros sistemas de informação. O IBGE possui uma lista de códigos e municípios que será necessária para realizarmos um tratamento e visualização de dados. Todas essas etapas de padronização e junção com outras informações também serão cobertas aqui.

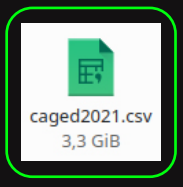
O tal do R: Software para tratar os dados

Para nossa sorte, a Base de Dados possui um botão para download:



Ao clicarmos no botão “Download”, automaticamente a base de dados será baixada para nosso computador. Em geral todas as linguagens de programação colocarão esse arquivo dentro da pasta chamada “Downloads”, mas isso é reprogramável.

É sempre importante você checar o tamanho da base baixada para poder saber melhor o desafio que irá enfrentar. Essa base, por exemplo, possui MAIS DE 3GB! [emoji SOS ou aquele suando]. Isso acontece porque essa base de dados possui 2.396.011 de linhas, com 26 colunas.



É aqui que chegamos a um momento importante da nossa jornada.

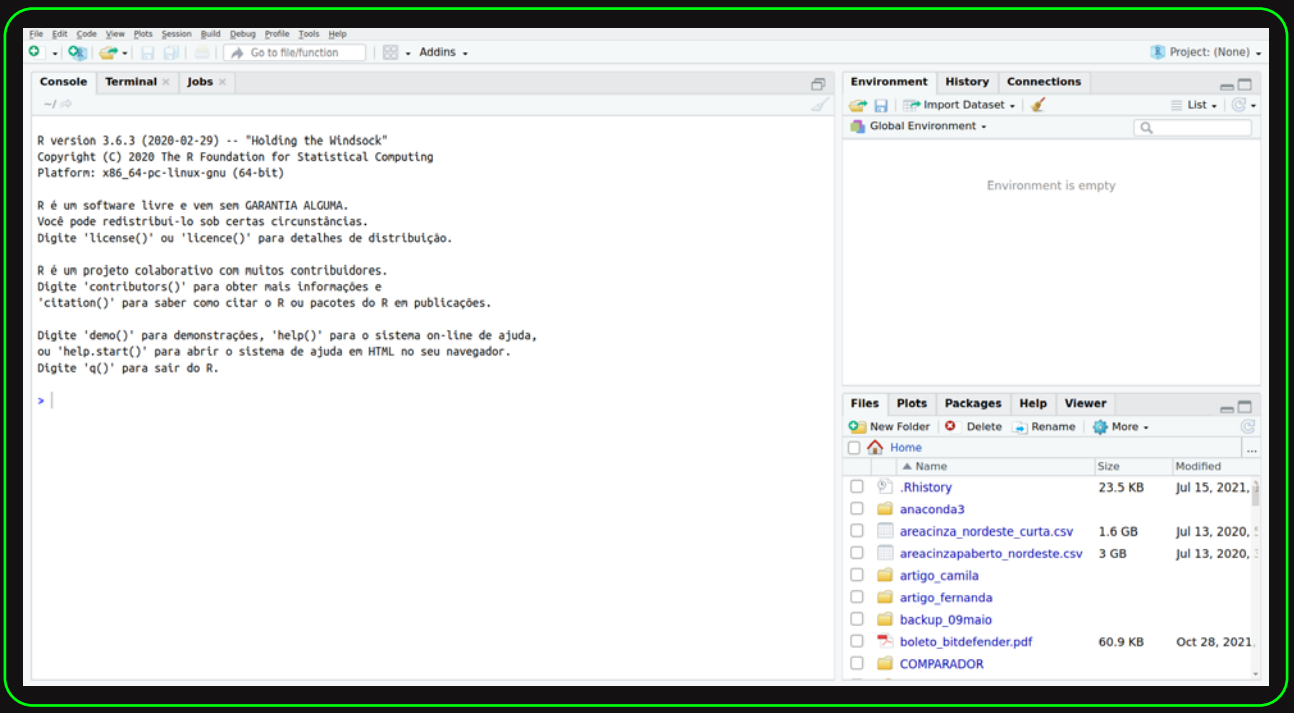
Programas tabuladores comuns, como o excel da Microsoft ou o LibreCalc, não conseguirão sequer abrir um arquivo tão grande assim.

A boa notícia é que existem várias linguagens de programação específicas para manipular e analisar bases de dados como essas. E nessa cartilha vamos te ensinar o passo a passo para analisar a base do CAGED usando o R e R studio.

Nós escolhemos esse programa, pois ele já é extremamente utilizado por estatísticos e possui muitos pacotes que auxiliam nas análises. Além disso, existem muitas organizações que ensinam de forma gratuita a utilizar o software como a RLadies. Para baixar o programa de forma correta basta seguir o passo a passo presente nesse link.

Mais um salve a todo mundo envolvido em comunidades de software livre e aberto, que possibilita o acesso a tecnologias e programas de análise de forma gratuita e possível para todes. <3

Após a instalação, clique sobre o ícone do R studio na sua área de trabalho.
O programa abrirá dessa forma:



O lado esquerdo, onde está escrito “Console”, “Terminal”, “Jobs” é a parte que você mais irá manipular. Nessa seção os nossos comandos (scripts) serão escritos e é nela também onde vamos incluir comentários e observações.

No lado direito nós temos duas janelas. A que fica na parte de cima - com as abas “Environment”, “History” e “Connections” - vai alocar todos os objetos que criarmos. As bases de dados que trouxermos ficarão visíveis por um ícone nessa área.

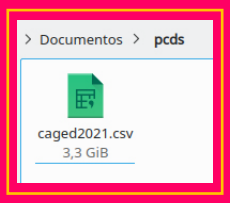
Na janela mais embaixo, ficam as abas “Files”, “Plots”, “Packages”, “Help” e “Viewer”. Nela será exibida qual o diretório que estamos trabalhando (Files), que é a pasta com nossos arquivos; os gráficos que criarmos (Plots); as bibliotecas (packages) que estamos utilizando; uma aba de ajuda (help) e uma sessão de visualização (Viewer).

O R é um programa de análise e manipulação de base de dados, contudo podemos fazer painéis, enviar email em massa, criar textos, relatórios e uma infinidade de atividades. Aqui é para o infinito e além.

Trazendo a base de dados para o R

Depois que baixamos a base precisamos trazê-la para ‘dentro’ do R. Para isso, a gente sempre recomenda que você crie uma pasta nova em seu computador com um nome que seja fácil para você organizar, manter todos os scripts e gráficos que gerar. No nosso exemplo, dentro da pasta Documentos, nós criamos uma pasta chama “pcds” e colamos o arquivo que baixamos dentro dela.

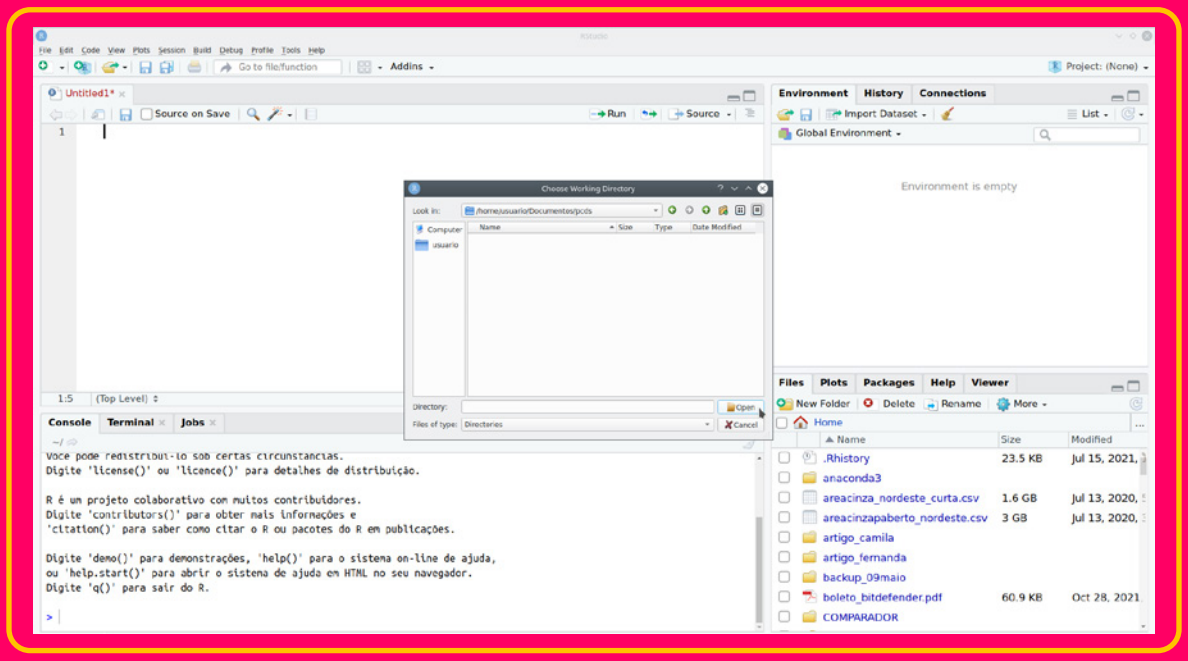
Dica: Evite colocar nomes com espaços e acentos, pois isso dificulta na escrita dos scripts.



Um dos primeiros passos é indicar para o R o caminho para nossa base de dados. Nós faremos isso manualmente, clicando no mouse em:

Ctrl+Shift+H

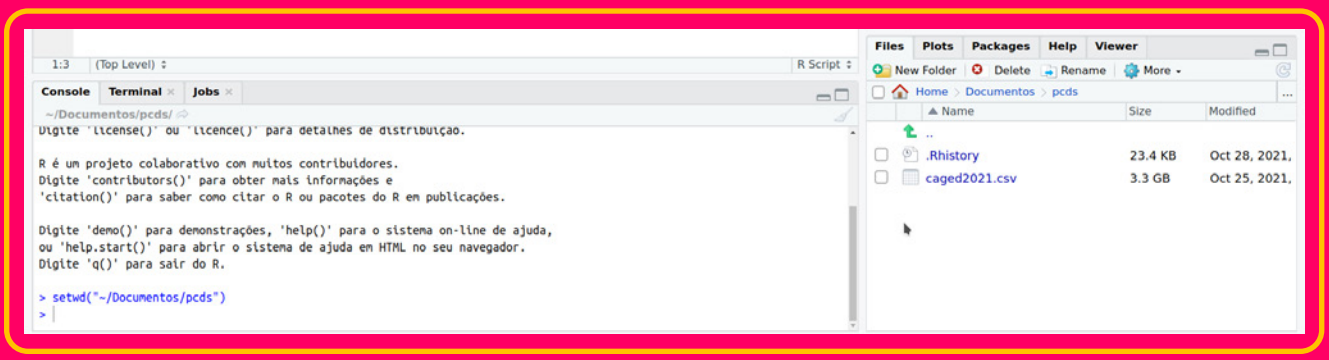
Automaticamente se abrirá uma janela no diretório do seu computador para que você indique onde você quer que o R se conecte. No nosso caso basta indicar a pasta Documentos, clicar na pasta pcds que criamos e clicar no Open.



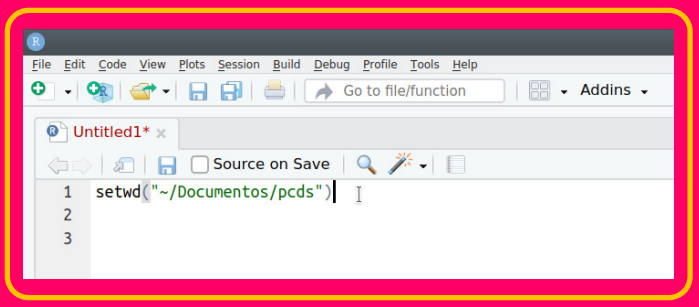
Agora, no lado direito do seu R studio aparecerá exatamente o seu arquivo **“caged2021.csv”**.

Uhul, primeira operação realizada com sucesso! No canto esquerdo inferior um primeiro comando foi executado. O R escreveu essa operação que você realizou através do mouse como:

> setwd(“~/Documentos/pcds”)



Agora copie esse comando que o próprio R escreveu e cole na janela superior da esquerda, para que esse comando fique registrado e, na próxima vez, seja mais rápido.



Na linha de baixo nós conseguiremos trazer o documento **caged2021.csv** para dentro do R.

Como ele é um arquivo csv, escreveremos o comando base **<- read.csv("caged2021.csv", sep=";", colClasses = "character")**

DICA: o comando Alt- é escrito para ALOCAÇÃO de objetos em R.

Quando você clicar no teclado sobre as teclas Alt- o R escreverá para você o símbolo <-

O que estamos escrevendo é que queremos que o arquivo **caged2021.csv** fique dentro do R com o nome base. Nós faremos muito isso, porque é mais rápido se você executar esse comando pelo teclado com Alt- do que ficar escrevendo <-

Entendendo o comando

read.csv quer dizer que você está lendo um arquivo csv. Arquivos csv são feitos de linhas e colunas. Logo que você digitar read.csv o próprio completará com os parênteses **( )** e dentro as aspas duplas **“ ”**.

Dessa forma, a primeira coisa logo depois do primeiro parênteses é o nome do arquivo que você está importando, de forma idêntica.

sep=”;” quer dizer que o separador do arquivo é o ponto vírgula. Arquivos csv podem ser separados por diferentes símbolos como o pipe **(“|”)**, apenas vírgula **(“,”)** ou até mesmo o tab **(“\t”)**. No dicionário de dados, existe a especificação do separador do arquivo.

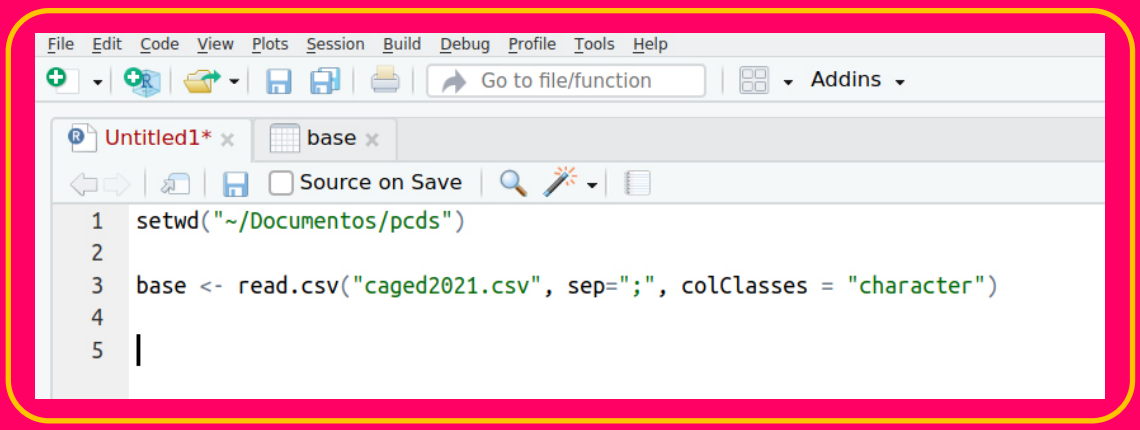
colClasses=”character” quer dizer que queremos que todos os dados sejam tratados como texto na íntegra, ou seja, do tipo string. Isso é importante porque muitas vezes campos de datas são escritos da forma **02071986**, que seria 02 de Julho de 1986. Como o R está tratando um dado que é composto de número isso acabaria apagando o primeiro 0 (zero) à esquerda, pois para números, zeros à esquerda são inválidos. Bem como, setar/configurar que todos os dados sejam lidos como character impede que o R exponencie números muito extremos. Por exemplo, o número **13.500.662.189** seria treze bilhões quinhentos milhões seiscentos e e sessenta e dois e cento e oitenta e nove, mas o R o colocaria como **13e9**, o que dificultaria o entendimento.

IMPORTANTE!

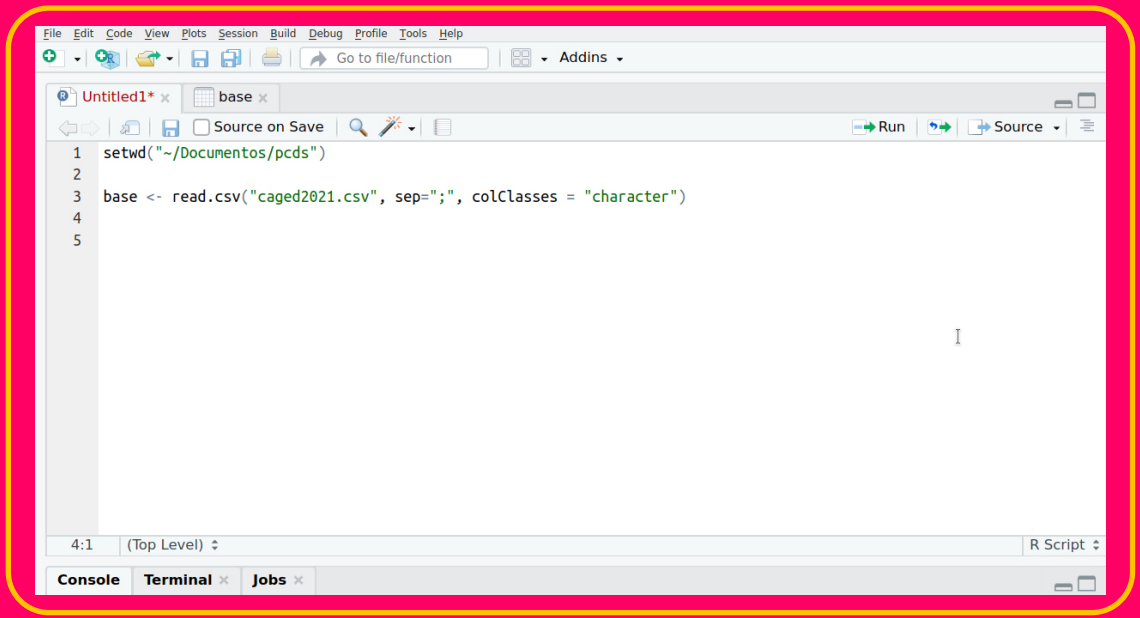
[character : caracter, entendido como dado do tipo texto, podendo ser apenas letras, e quando for números, o dado será lido como um texto.numeric: número, que varia de menos infinito (valores negativos) até mais infinito (valores positivos). Para realizar operações matemáticas, os dados precisam estar em formato numérico. factor: categórico, que possui diversos fatores. É possível que um número ou texto sejam categorias. Por isso é importante entender e ler o dicionário de dados, pois uma variável pode conter as categorias 1, 2, 3 e 4. Mas na verdade se tratam de categorias, onde 1 corresponde a uma categoria determinada, como ser do gênero masculino e, assim sucessivamente. Existem ainda outras formas de dados, porém para a nossa cartilha de dados, vamos precisar entender esses três formatos.]

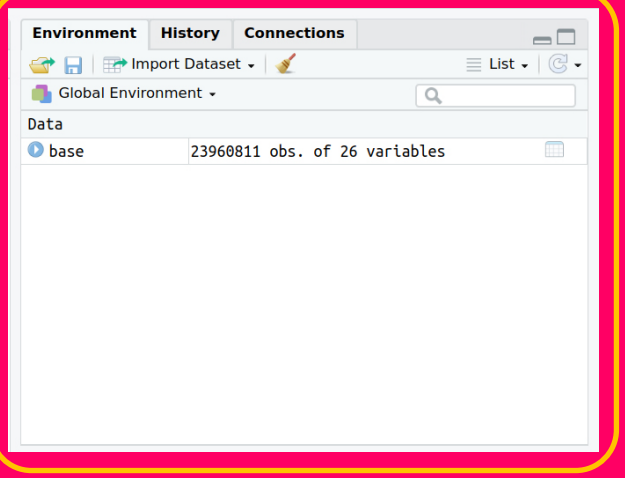
Quem é de humanas a essa hora já tá sentindo o coração palpitar, então vamos revisar o que vimos até aqui. Para trazer uma base de dados para o R é necessário:

nome do objeto seguido de **<- read.csv ("nome do seu arquivo", sep="o tipo de separador do arquivo", colClasses="Character")**



Certifique-se que seu comando possui todas as vírgulas e as aspas necessárias. Depois disso, com o cursor do mouse na linha do código, digite no teclado Ctrl+Enter. Espere o R executar o comando, e um ícone vermelho de STOP irá aparecer na parte de baixo. Após o software processar o comando, na janela ao lado você verá um objeto chamado base com o número de linhas e colunas. Seja paciente que esse processo demora tanto quanto o tamanho da sua base.





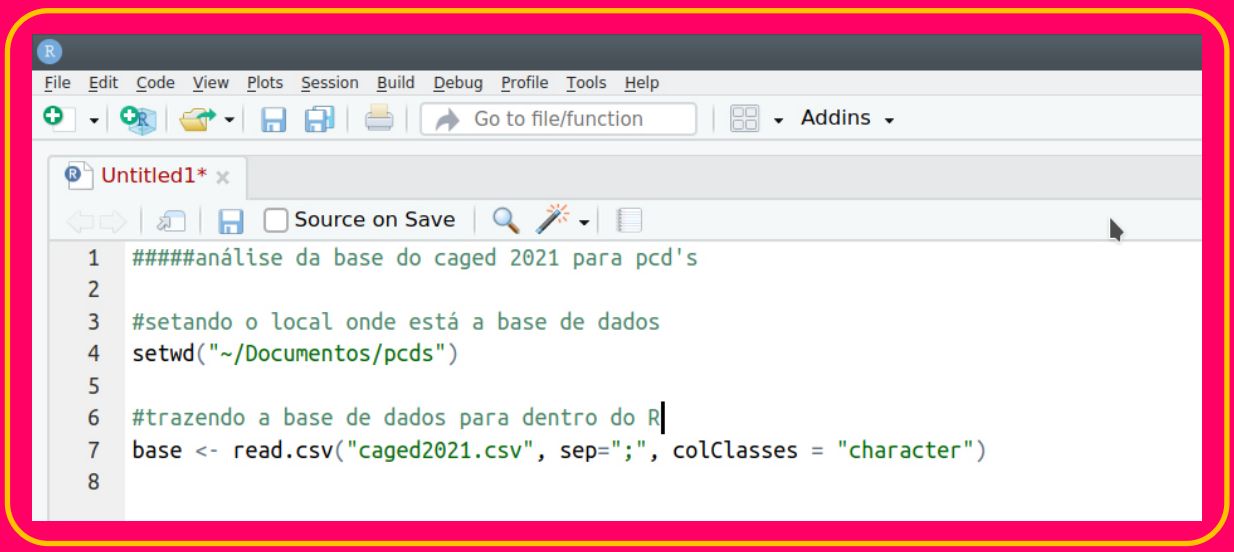
Quando o sinal de STOP sair, aparecerá o seguinte ao lado da sua janela:

Veja que temos 2.396.011 de linhas, com 26 variáveis. Caso você tenha dúvidas de como esse processo é feito, acesse o passo a passo nesse vídeo: link

ATENÇÃO!

É muito importante que você comente e documente tudo o que você faz! Acredite, será muito fácil de você se perder se não for comentando tudo o que faz. A forma de fazer isso é adicionando # (hashtag).

Por exemplo:



Basta você adicionar uma hashtag # e colocar o texto descrevendo o que está executando abaixo. Sempre fique atento pois o R entenderá que tudo que está na mesma linha da hashtag não é um código e sim um comentário

Reconhecendo a base

Uma forma rápida de analisarmos o conteúdo da base é através de um resumo geral. O comando utilizado para isso é através de summary.

Aqui já vale te ensinar que no R alguns sinais e letras são fundamentais para aprender a programar. Em geral depois de cada função ou comando a gente escreve entre parênteses onde queremos que isso aconteça, no caso do comando summary será: **summary(base)**

Porém, como setamos para que todos os dados fossem lidos como characters você apenas terá um resumo da quantidade de variáveis.

Uma forma mais fácil de saber apenas os nomes das variáveis contidas nessa base de dados é: **names(base)**

Na região do console você verá o nome de cada uma das 26 variáveis contidas na sua base de dados.

A primeira forma de saber o resumo das variáveis é apresentá-las em formato de tabela.

Em R, quando a gente quer especificar que um comando aconteça em apenas uma variável da base de dados, nós informamos o nome da base seguido do símbolo de dólar ($) e o nome da variável, ou seja **base$variável**

Para gerar tabela o comando é chamado table. Portanto para ter uma tabela da variável ano basta fazermos:
table(base$ano)

Na área do console você terá uma tabela contendo todos os registros existentes na coluna ano. Como estamos trabalhando apenas com o ano de 2021, o R retornou que o valor 2021 possui 23.960.811 entradas

Porém ao repetirmos o mesmo comando para a variável mês da seguinte forma
table(base$mes)

Teremos o seguinte retorno:

```
      1       2       3       4       5       6       7       8
2982960 3133975 3138581 2672610 2841785 2916006 3026413 3248481
>
```

Veja que aqui precisaremos fazer um primeiro **tratamento** nos dados. Pois apesar de sabermos que mês 1 é Janeiro, o dado está escrito apenas como 1 ou 2 ou 3 etc.

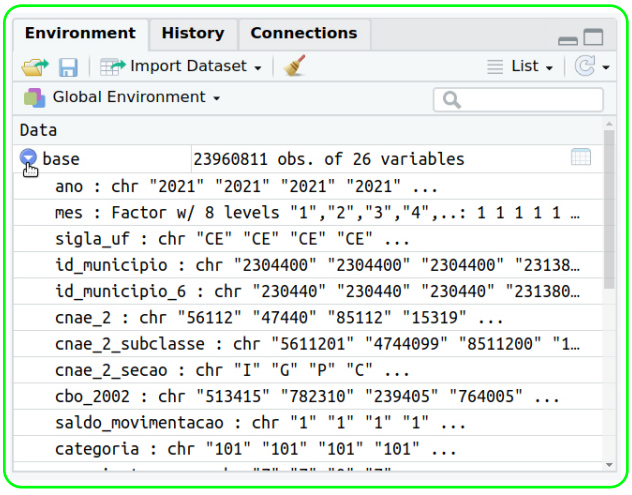
Essa é uma decisão sua, para fins de compreensão de mais conteúdos.

Um dado como esse, que assuma uma categoria ou outra, é chamado de categórico, que pro R é chamado factor. Então, a primeira transformação será mudar o tipo de dado da variável mês, de *character* para factor.

Mudando o tipo de dado

base$mes <- as.factor(base$mes)
Nesse script estamos pedindo que a variável mês da base receba ela mesma como factor.

Você não verá nenhuma modificação aparente, mas se clicar na setinha do ícone da base no lado direito do R, verá que essa variável está de outra forma.



Veja que ela é a única que está como factor. Isso vai tornar mais fácil a troca do valor númerico pelo nome do mês. Primeiro, vamos exibir as categorias dentro da variável mês:
levels(base$mes)

Agora, nós queremos que onde está escrito 1 seja mudado para janeiro, onde está escrito 2 seja mudado para fevereiro, e assim sucessivamente.

Toda vez que você quiser mudar os rótulos de uma variável categórica basta pedir que os levels recebam os nomes que você quer. No nosso caso, é uma lista de nomes de meses:
levels(base$mes) <- c(“Janeiro”, “Fevereiro”, “Março”, “Abril”, “Maio”, “Junho”, “Julho”, “Agosto”)

Neste script nós estamos dizendo que cada levels da variável mês receberá esse novo nome, na ordem escrita. Perceba que colocamos um c antes dos parênteses. Essa letra c significa um conjunto de valores, porque o primeiro rótulo receberá o que escrevemos entre as primeiras aspas, o segundo valor o segundo valor, e assim por diante. **Não pode esquecer de colocar cada valor entre aspas e separado por vírgula.**

Mais uma vez essa mudança não fará nada aparente, porém se você repetir o comando pedindo uma tabela terá o seguinte resultado:
table(base$mes)

```
  Janeiro Fevereiro     Março     Abril      Maio     Junho     Julho    Agosto
  2982960   3133975   3138581   2672610   2841785   2916006   3026413   3248481
> 
```

Parabéns, essa foi sua primeira manipulação de dados. Lembre-se que se trata de um comando padrão, isto é, toda vez que quiser mudar os rótulos de uma variável categórica, primeiro transforme para tipo factor, peça os levels e atribua os nomes novos

Um pacote para gerar

tabelas

A maioria das linguagens de programação possui bibliotecas ou pacotes desenvolvidos ao longo dos anos e que ajudam muito a vida do programador. Em R nós chamamos de package. Eles são desenvolvidos por todos os usuários e encontrados em um repositório. Tem pacote pra tudo, pra tabela, pra gráfico, pra teste estatístico, etc. Sempre é necessário instalar o pacote na primeira vez que for usar, depois basta ativá-lo quando quiser.

Para tabelas e gráficos simples existe um pacote chamado epiDisplay, para instalá-lo basta dar o comando **install.packages("epiDisplay")**

Assim que você clicar em Ctrl+Enter o R irá acessar automaticamente o repositório de pacotes e instalar automaticamente em seu R esse pacote.

Para usá-lo é preciso ativá-lo, então sempre que você for começar uma análise e for executar um pacote use o comando library seguido de parênteses e o nome do pacote: **library(epiDisplay)**

O R vai carregar seu pacote e liberar suas funcionalidades.

Para gerar tabela de uma variável o comando é **tab1(base$mes)**

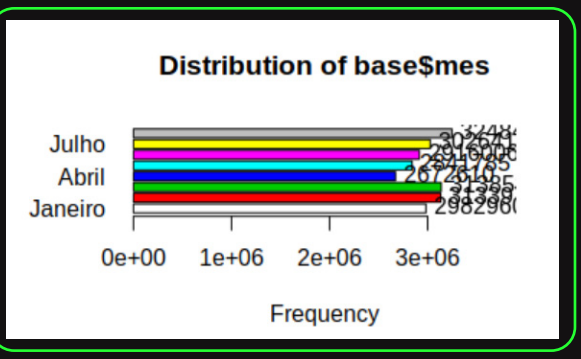
O resultado será diferente do comando table anterior:

```
> tab1(base$mes)
base$mes :
           Frequency Percent Cum. percent
Janeiro      2982960    12.4         12.4
Fevereiro    3133975    13.1         25.5
Março        3138581    13.1         38.6
Abril        2672610    11.2         49.8
Maio         2841785    11.9         61.6
Junho        2916006    12.2         73.8
Julho        3026413    12.6         86.4
Agosto       3248481    13.6        100.0
  Total     23960811   100.0        100.0
>
```

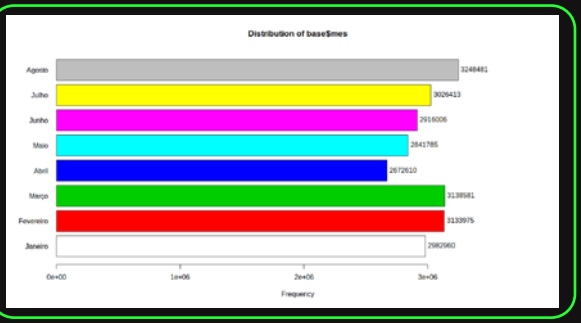
Veja que você já tem a frequência, que é a soma de todas as vezes que o mês de janeiro aparece; a porcentagem, que é a soma de um mês dividido pelo total; e a porcentagem cumulativa, que vai somando cada porcentagem.

Com isso podemos ver que o mês com a menor quantidade de registro foi maio, com 2.841.785 empregos, representando 11,9%. E o mês com maior quantidade de registro foi agosto com 3.248.481, representando 13,6%.

O mais bacana desse pacote é que, se tudo ocorrer bem, ele exibirá um gráfico de barra automaticamente. Basta você clicar na janela do R direita na parte inferior em *plots*:



Clique no zoom para ter o gráfico maximizado e ler melhor:



É possível já fazer algumas inferências. Veja que de março para abril tivemos uma grande queda no número de registrados e que a partir de maio existe um crescimento no número de registrados no CAGED, alcançando números superiores ao início do ano.

Da mesma forma que fizemos uma tabela para o mês, podemos repetir a mesma coisa para a variável sigla_uf, basta usar o comando tab1 novamente **tab1(base$sigla_uf)**

Você terá uma tabela bem maior que a anterior, pois agora temos 27 entradas, uma para cada unidade federativa. A melhor forma de apresentar uma tabela e gráfico é ordenando pela quantidade de registros. Para isso basta acrescentar mais um parâmetro ao comando tab1 tab1(base$sigla_uf, **sort.group="decreasing")**

```
> tab1(base$sigla_uf, sort.group = "decreasing")
base$sigla_uf :
        Frequency Percent Cum. percent
SP        7978758    33.3         33.3
MG        2562072    10.7         44.0
PR        1879548     7.8         51.8
SC        1750656     7.3         59.1
RS        1585436     6.6         65.8
RJ        1497330     6.2         72.0
BA         843518     3.5         75.5
GO         838434     3.5         79.0
MT         565619     2.4         81.4
PE         564335     2.4         83.7
CE         563356     2.4         86.1
ES         481611     2.0         88.1
PA         447493     1.9         90.0
DF         391191     1.6         91.6
MS         327132     1.4         93.0
MA         239370     1.0         94.0
AM         236034     1.0         95.0
RN         219849     0.9         95.9
PB         198414     0.8         96.7
AL         169127     0.7         97.4
RO         151799     0.6         98.0
PI         136148     0.6         98.6
SE         113732     0.5         99.1
TO         105283     0.4         99.5
AC          43679     0.2         99.7
RR          39261     0.2         99.9
AP          31626     0.1        100.0
  Total  23960811   100.0        100.0
>
```

Perceba que, por termos ordenado, fica mais fácil tirarmos uma informação quando temos muitos registros.

Conseguimos saber que o estado com maior número de registros no CAGED de janeiro a agosto de 2021 é o estado de São Paulo **(n=7.978.758; 33%)**, enquanto o Amapá é o estado com menor quantidade de registros **(n=31.626; 0,1%)**.

Você também tem um gráfico ordenado bem como a tabela.

A próxima variável é a id_municipio. Pelo dicionário de dados sabemos que é o código do IBGE referente a cada município do Brasil. Entretanto, existe uma forma de trocarmos todos os valores de uma vez se tivermos uma tabela com todos os códigos e os nomes ao lado. É aí que o data-labe te fortalece. Deixamos isso prontinho para download aqui nesse link: base_municipios

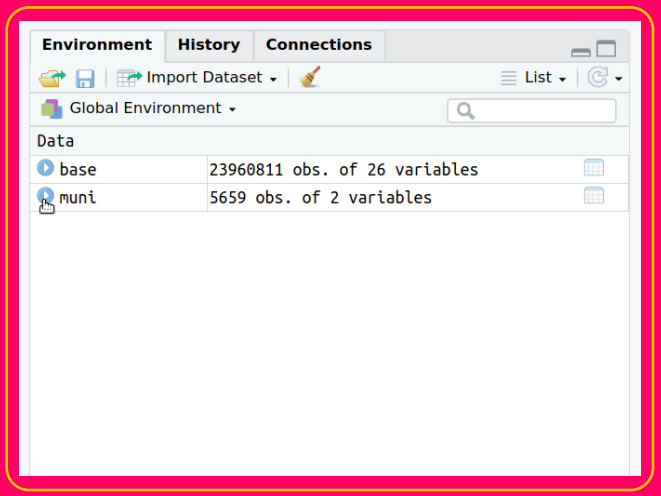
Juntando duas bases de dados

Ao clicar no link acima, baixe o arquivo e traga para a mesma pasta onde você está trabalhando. Assim, não será necessário setar outro diretório.

Agora que você já sabe o comando para importar bases, fique atento na diferença deste novo código:
muni <- read.csv("municipios_ibge.csv", sep = ",", colClasses = "character")

Veja que, desta vez, o separador é a vírgula **“,”**. É comum que diferentes bases tenham diferentes separadores. Em geral, se você não se atentar a este detalhe o R trará todas as informações em uma mesma coluna, e você conseguirá perceber visualmente qual o separador correto e incluir em seu script.

Após executar esse comando você ficará com duas bases. Nós demos o nome de “muni” apenas para não termos que digitar palavras muito grandes e acelerar a programação.



A segunda base possui apenas duas variáveis com o comando “names”. Você já aprendeu que uma coluna é o código e o outra é o nome por extenso do município.

Ou seja, nós temos uma variável em comum nas duas bases, o código do município. Isso nos permite juntar as duas bases (merge). É necessário que essas duas variáveis sejam IDÊNTICAS, qualquer mudança será levada em consideração. Por isso, sempre verifique se não está faltando nenhum zero, ou um hífen para consertar.

Para verificar como a base está com suas variáveis, você pode clicar sobre o nome da base, ou executar o comando: **View(muni)**

Veja que o código possui 6 dígitos e na descrição nós temos a sigla do estado, seguido de um traço e aí sim a informação que queremos. Quando queremos apenas uma parte de um dado escrito, nós executamos uma *substring*. Neste comando você precisa sempre indicar em que coluna estará operando o script, de que posição quer iniciar o ‘corte’ e onde quer parar.

Tirando pedaços de dados

Aqui temos um desafio, pois queremos começar sempre do quarto dígito, já que os dois primeiros serão a sigla da UF e o terceiro o hífen. E nós queremos parar no último dígito do dado, quantidade difícil de definir já que alguns municípios têm nomes mais longos e outros mais curtos. Vamos resolver isso de uma forma lógica.

Saca o script abaixo:

substr(muni$Descrição, 4, nchar(muni$Descrição))
substr : o nome da função
muni$Descrição : a coluna que queremos operar o comando

, 4 : onde começa a pegar o dado

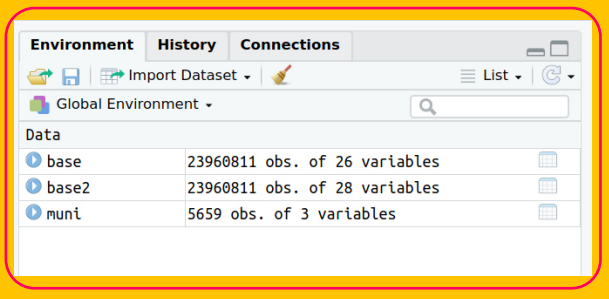
, nchar(muni$Descrição) : nchar quer dizer a quantidade de caracteres de um dado, portanto estamos dizendo que irá parar no máximo possível

Veja do lado direito do seu console que a base continua com apenas duas variáveis. Isso porque o R executou o comando conforme exibido na área do console, mas nós não atribuímos isso em nenhum objeto.

Ouça o bom conselho: Quando você for mudar o conteúdo original de uma base de dados, crie uma outra variável e nela aloque os dados manipulados. Isso ajuda na transparência dos dados e permite que você nunca perca o dado original caso algo errado aconteça.

Portanto chamaremos a variável nova de "**desc2**", ficando o script:
muni$desc2 <- substr(muni$Descrição, 4, nchar(muni$Descrição))

Esse comando fará com que você possua uma variável original intacta, e outra apenas com a substring que você executou. Agora vamos juntar as bases com o script abaixo:
base2 <- merge(base, muni, by.x = "id_municipio_6", by.y = "Código", all.x = T, all.y = F, incomparables = NA)



Boa! Você possui uma outra base chamada base2 que tem 28 variáveis, ou seja a base original tinha 26, a de municípios 3, porém 1 variável de cada era a mesma.

Portanto o somatório delas, será o número de variáveis das duas juntas menos **1 ((26 + 3)-1))**

Entenda o script:

merge : função para juntar duas bases

base, : nome da primeira base

muni, : nome da segunda base

by.x="id_municipio_6", : o nome da variável que está na primeira base que é idêntica à segunda

by.y="Código", : o nome da variável que está na segunda base que é idêntica à primeira

all.x=T, : Explicita que você quer manter todos os registros da primeira base e apenas adicionar os dados que contenham na segunda base

all.y=F, : Não precisa manter registros que constem apenas na segunda base

incomparable=NA : todos os dados que não estiverem contidos devem receber NA.

Agora que você já possui uma base com mais informações poderá remover as duas bases anteriores para facilitar a memória utilizada, com o script abaixo:
rm(base)
rm(muni)

Trocando os valores pelo dicionário

A próxima variável que iremos manipular é de extrema importância: o grau de instrução. A escolaridade de um indivíduo nem sempre representa bem suas vulnerabilidades e condições. Contudo, em um país muito desigual como o Brasil, em termos populacionais, é possível afirmar que pessoas com o ensino médio têm condições diferentes das que possuem ensino superior.

Veja bem, não estamos falando das exceções e sim, do que é mais comum. Portanto, em geral pessoas com ensino superior ganham melhores salários do que pessoas que cursaram apenas o ensino médio. Para saber como isso se dá em diferentes situações é que entra a apuração jornalística, a entrevista, a opinião de especialista, entre outros recursos.

Para a mudança da escolaridade, utilizaremos essas informações do dicionário de dados:

CÓDIGO	DESCRIÇÃO
1	Analfabeto
2	Até 5º Incompleto
3	5º Completo Fundamental
4	6º a 9º Fundamental
5	Fundamental Completo
6	Médio Incompleto
7	Médio Completo
8	Superior Incompleto
9	Superior Completo
10	Mestrado
11	Doutorado
80	Pós-Graduação completa
99	Não Identificado

Primeiro, verifique se os códigos contidos na variável estão de acordo com o dicionário.
tab1(base2$grau_instrucao, graph = F)

Agora você pede para que onde está contido “1”, seja trocado por “Analfabeto” e todos os outros correspondentes:
base2$grau_instrucao[which(base2$grau_instrucao=="1")] <- "Analfabeto"

Entenda o script:

base2$grau_instrucao : o nome da variável que queremos operar a mudança

[which(base2$grau_instrucao==”1”)] : onde a variável grau de instrução seja idêntica a “1”

<- : recebe

“Analfabeto” : o novo rótulo

Eu sei que você já deve estar com ódio, pois terá que escrever isso 13 vezes, porém se você mantiver o cursor do R sobre seu script e clicar nas teclas do teclado

Alt + Shift + Seta para baixo o programa irá duplicar toda a linha escrita. Isso já facilita muito, pois daí será necessário só prestar muita atenção para trocar os códigos e novos rótulos. Muita atenção nessa hora!

Depois de ter digitado e verificado as linhas, você pode selecionar todas de uma vez e clicar para executar no teclado **Ctrl + Enter**

Assim, o R executará uma linha após a outra. Sempre que você executar uma mudança como essa, peça uma tabela novamente e compare com a anterior que você executou. Os números PRECISAM ser idênticos. Caso não sejam, verifique onde está o erro e faça tudo de novo. =)

Aqui já temos uma informação social que 60,1% dos cadastrados no CAGED no primeiro semestre possuem o ensino médio completo e que a segunda escolaridade mais predominante é a de pessoas que possuem ensino superior, sendo 95% do total.

Para a nossa análise, nós ainda iremos trocar os rótulos e conteúdos de acordo com o dicionário para as variáveis raça/cor, sexo, fonte e tipo de deficiência. Para cada uma delas, você encontra os scripts abaixo

PARA RAÇA/COR

```
base2$raca_cor[which(base2$raca_cor==1)] <- "Branca"
base2$raca_cor[which(base2$raca_cor==2)] <- "Preta"
base2$raca_cor[which(base2$raca_cor==3)] <- "Parda"
base2$raca_cor[which(base2$raca_cor==4)] <- "Amarela"
base2$raca_cor[which(base2$raca_cor==5)] <- "Indígena"
base2$raca_cor[which(base2$raca_cor==6)] <- "Ignorado"
base2$raca_cor[which(base2$raca_cor==9)] <- "Ignorado"
```

PARA GÊNERO

```
BASE2$SEXO[WHICH(BASE2$SEXO==1)] <- "MASCULINO"
BASE2$SEXO[WHICH(BASE2$SEXO==3)] <- "FEMININO"
BASE2$SEXO[WHICH(BASE2$SEXO==9)] <- "IGNORADO"
```

PARA TIPO DE FONTE

```
BASE2$FONTE[WHICH(BASE2$FONTE==1)] <- "NÃO DESLIGADO"
BASE2$FONTE[WHICH(BASE2$FONTE==2)] <- "DESLIGADO"
BASE2$FONTE[WHICH(BASE2$FONTE==3)] <- "DESLIGADO"
```

PARA TIPO DE DEFICIÊNCIA

```
BASE2$TIPO_DEFICIENCIA[WHICH(BASE2$TIPO_DEFICIENCIA==0)] <- "NÃO DEFICIENTE"
BASE2$TIPO_DEFICIENCIA[WHICH(BASE2$TIPO_DEFICIENCIA==1)] <- "FÍSICA"
BASE2$TIPO_DEFICIENCIA[WHICH(BASE2$TIPO_DEFICIENCIA==2)] <- "AUDITIVA"
BASE2$TIPO_DEFICIENCIA[WHICH(BASE2$TIPO_DEFICIENCIA==3)] <- "VISUAL"
BASE2$TIPO_DEFICIENCIA[WHICH(BASE2$TIPO_DEFICIENCIA==4)] <- "INTELECTUAL"
BASE2$TIPO_DEFICIENCIA[WHICH(BASE2$TIPO_DEFICIENCIA==5)] <- "MÚLTIPLA"
BASE2$TIPO_DEFICIENCIA[WHICH(BASE2$TIPO_DEFICIENCIA==6)] <- "REABILITADO"
BASE2$TIPO_DEFICIENCIA[WHICH(BASE2$TIPO_DEFICIENCIA==9)] <- "NÃO IDENTIFICADO"
```

Veja que muitos rótulos possuem a informação de “ignorado” ou “não identificado”. Em geral, para fazermos análises e modelos matemáticos é preciso retirar/excluir variáveis sem informação. No entanto, no data_labe nós acreditamos que a falta de informação também gera conhecimento. É comum que grupos mais vulneráveis sejam mais negligenciados na hora do preenchimento do dado. Por isso, por enquanto, não vamos deletar nenhum dado.

A última variável que iremos manipular pelo dicionário é chamada “seção”. Nela existe a informação de que setor aquele emprego pertence. A informação do dicionário de dados do CAGED nos diz que:

CÓDIGO	DESCRIÇÃO
A	Agricultura, Pecuária, Produção Florestal, Pesca e Aquicultura
B	Indústrias Extrativas
C	Indústrias de Transformação
D	Eletricidade e Gás
E	Água, Esgoto, Atividades de Gestão de Resíduos e Descontaminação
F	Construção
G	Comércio, Reparação de Veículos Automotores e Motocicletas
H	Transporte, Armazenagem e Correio
I	Alojamento e Alimentação
J	Informação e Comunicação
K	Atividades Financeiras, de Seguros e Serviços Relacionados
L	Atividades Imobiliárias
M	Atividades Profissionais, Científicas e Técnicas
N	Atividades Administrativas e Serviços Complementares
O	Administração Pública, Defesa e Seguridade Social
P	Educação
Q	Saúde Humana e Serviços Sociais
R	Artes, Cultura, Esporte e Recreação
S	Outras Atividades de Serviços
T	Serviços Domésticos
U	Organismos Internacionais e Outras Instituições Extraterritoriais
Z	Não identificado

Como são muitas linhas para mudarmos manualmente, vamos trocar os rótulos pelo “merge”.
A base encontra-se disponível **aqui**.
Basta baixar, pôr na mesma pasta que você está trabalhando e executar o merge como você já aprendeu.

O script está abaixo:

Para importar a base
secao <- read.csv("secao.csv", colClasses = "character")

Para fazer o novo merge
base3 <- merge(base2, secao, by.x = "cnae_2_secao", by.y = "Código", all.x = T, all.y = F, incomparables = NA)

Para excluir as bases antigas
rm(base2)
rm(secao)

Agora que as variáveis que queremos estão editadas/estruturadas do jeito que precisamos podemos retirar as variáveis que não iremos usar.

Reduzindo a base

Primeiro, peça o nome das variáveis para se certificar em que posições as variáveis que você deseja estão: **names(base3)**

Veja que na posição 1 temos a variável sobre seção e na última linha temos uma variável "**Descrição.y**" que representa seu significado, então podemos ficar apenas com a descrição.

Neste caso, a lista de variáveis que nos interessa seria a seguinte:
ano: posição 3
mês: posição 4
sigla_uf: posição 5
grau_instrução: posição 12
idade: posição 13
raça_cor: posição 15
sexo: posição 16
tipo_deficiência: posição 20
salario_mensal: posição 23
fonte: posição 26
Descrição.x: posição 27
desc2: posição 28
Descrição.y: posição 29

O script para essa operação é:
base3 <-
base3[,c(3,4,5,12,13,15,16,20,23,26,27,28,29)]

Entenda o script
base3 <-: A base 3 recebe
base3[,c: A base 3 com todas as linhas e a lista das variáveis entre os parênteses
(3,4,5,12,13,15,16,20,23,26,27,28,29)]

Toda vez que queremos localizar apenas partes específicas de uma base de dados em R, podemos utilizar os colchetes [ ].

Nesse caso é sempre o nome da base seguido de [] dentro do colchete, a primeira parte são as linhas. Como queremos todas as linhas, deixamos em branco.

Depois da linha é a coluna. Visto que queremos uma lista de colunas, introduzimos o c (concatena).

Exportando a base que você trabalhou

Como você terminou uma etapa muito importante, é aconselhável salvar/exportar essa base de dados. Para não ter que repetir tudo de novo caso precise parar.

Logo, nós exportaremos essa base que estamos trabalhando com o nome de caged_tratado, da seguinte forma:

```
write.csv2(base3,"caged_tratado.csv", row.names = F)
```

Entenda o script

write.csv2: exportar o objeto que você criou em um arquivo do tipo csv, como está setado csv2, o separador desse arquivo será o ponto e vírgula “;” .

(base3, “caged_tratado.csv”,: o nome que esse arquivo terá será caged_tratado.csv

,row.names=F: não precisa contar a quantidade de linhas, se você deixar sem esse parâmetro o R irá incluir um número para cada linha do 1 até o limite máximo de linhas.

Salvando o script

Da mesma forma que você salvou o objeto e exportou o csv, vale a pena também salvar o script. Pois apesar de ser muito raro do R travar, o seguro morreu de velho, então certifique-se de sempre a cada etapa exportar a sua base e salvar seu script para não perder nenhuma etapa do seu trabalho.

Basta você digitar no teclado Ctrl+S (tal qual salvar um documento qualquer), uma janela do tipo de codificação que você deseja, pode deixar o tipo UTF-8, clique em salvar, escolha um nome para o seu script e salve. Veja que no mesmo diretório que você setou o início do seu trabalho também estarão o arquivo caged_tratado.csv e o seu script.

Retirando informações da sua base de dados

Existem muitas formas de começar a contar os resultados de suas bases de dados. Para trabalhos acadêmicos e científicos a gente sempre começa do mais geral para o mais específico. Porém para matérias, reportagens e outros produtos, a decisão narrativa pode ser pensada de forma mais flexível. Para este exemplo, vamos seguir do maior resultado para o menor.

Quantos cadastrados na base do CAGED de 2021, entre os meses de Janeiro a Agosto, possuem deficiência?

Uma das formas de você exibir contagens, que você já aprendeu, é através de tabelas. Nesse caso, como os trabalhadores podem se repetir mês a mês, precisamos contar a cada mês essa quantidade de trabalhadores com ou sem deficiência. Um dos pacotes que facilita a criação de tabelas é chamado dplyr. Portanto, se essa é sua primeira vez usando, primeiro instale o pacote através do comando abaixo:
install.packages("dplyr")

E depois, ative o pacote com o comando abaixo:
library(dplyr)

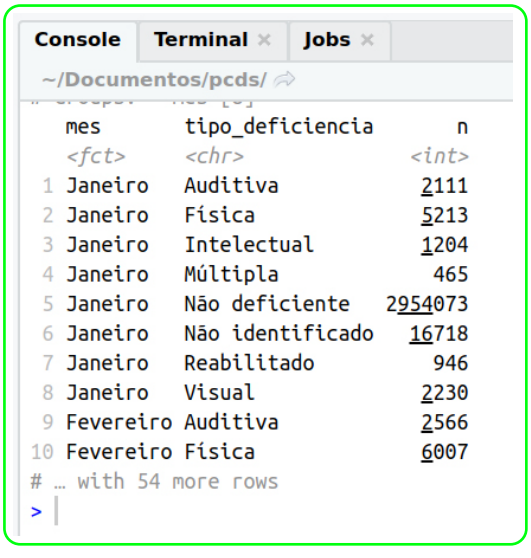
Após isso, iremos construir uma tabela que, a cada mês tenha o total de trabalhadores com ou sem deficiência através do script abaixo: **base3 %>% group_by(mes) %>% count(tipo_deficiencia)**

Entenda o script:

base3 %>% :O nome do objeto que iremos construir, a cada etapa de comando utilizamos o operador "%>%" que de forma bem coloquial significaria "depois disso".

group_by(mes) :Agrupe mês a mês desse objeto

count(tipo_deficiencia) : Conte quantas categorias tem na variável tipo_deficiencia o resultado será:



Veja que o próprio R studio falou para você que tem mais 54 linhas nessa tabela, para visualizá-la na íntegra, coloque esse comando em um objeto e visualize através dos comandos abaixo:
tab1 <- base3 %>% group_by(mes) %>% count(tipo_deficiencia)
View(tab1)

O R abrirá uma tabela contendo, a cada mês, o total de trabalhadores por tipo de deficiência. Porém isso é quase o que queremos, como precisamos contar primeiro quem é deficiente e quem não, podemos criar uma outra variável, que chamaremos de defic_der apenas informando se o trabalhador possui deficiência ou não, sem descriminar o tipo, para isso faremos o comando a seguir:

base3$defic_der <- ""
base3$defic_der[which(base3$tipo_deficiencia=="Não deficiente")] <- "Sem deficiência"
base3$defic_der[which(base3$tipo_deficiencia=="Não identificado")] <- "Sem deficiência"

Entenda o script:

Linha 1, estamos apenas gerando uma variável nova com conteúdo vazio
Linha 2:
base3$defic_der[which : na variável defic_der da base3 quando

(base3$tipo_deficiencia==”Não deficiente”)] : a variável tipo_deficiencia da base3 for idêntica a Não deficiente

<- “Sem deficiência” : Recebe o termo Sem deficiência.

Linha 3, fizemos o mesmo processo da linha, porém para o termo Não identificado. Após isso, você pode testar se está indo tudo bem, e ao final de tudo, estabelecer que tudo o que está vazio recebe termo “Com deficiência”, através do script abaixo:
base3$defic_der[which(base3$defic_der=="")] <- "Com deficiência"

Ao repetir o comando da tabela, atente-se para pôr o nome da v ariável nova “defic_der”, atraveś dos comandos:
tab1 <- base3 %>% group_by(mes) %>% count(defic_der)
View(tab1)

A tabela que você irá gerar, será a seguinte:

	mes	defic_der	n
1	Janeiro	Com deficiência	12169
2	Janeiro	Sem deficiência	2970791
3	Fevereiro	Com deficiência	14143
4	Fevereiro	Sem deficiência	3119832
5	Março	Com deficiência	14912
6	Março	Sem deficiência	3123669
7	Abril	Com deficiência	13176
8	Abril	Sem deficiência	2659434
9	Maio	Com deficiência	14523
10	Maio	Sem deficiência	2827262
11	Junho	Com deficiência	14788
12	Junho	Sem deficiência	2901218
13	Julho	Com deficiência	15616
14	Julho	Sem deficiência	3010797
15	Agosto	Com deficiência	17048
16	Agosto	Sem deficiência	3231433

Veja que, sem grandes surpresas, a quantidade de trabalhadores sem deficiência é muito maior que trabalhadores com deficiência, independente do mês. Porém para fins de comunicação escrever a quantidade de todos esses meses seria exaustivo e não comunicativo. Para dados que exigem muitas linhas e poucas comparações, uma ótima forma de realizar é através de gráficos. Para isso vamos usar o pacote "ggplot2" que facilita muito as visualizações.

Nossa intenção é mostrar mês a mês a quantidade de trabalhadores com deficiência versus os trabalhadores sem deficiência. Isso de forma básica e simples é feita através do seguinte comando:

ggplot(base3, aes(mes, fill=defic_der))+
geom_bar()

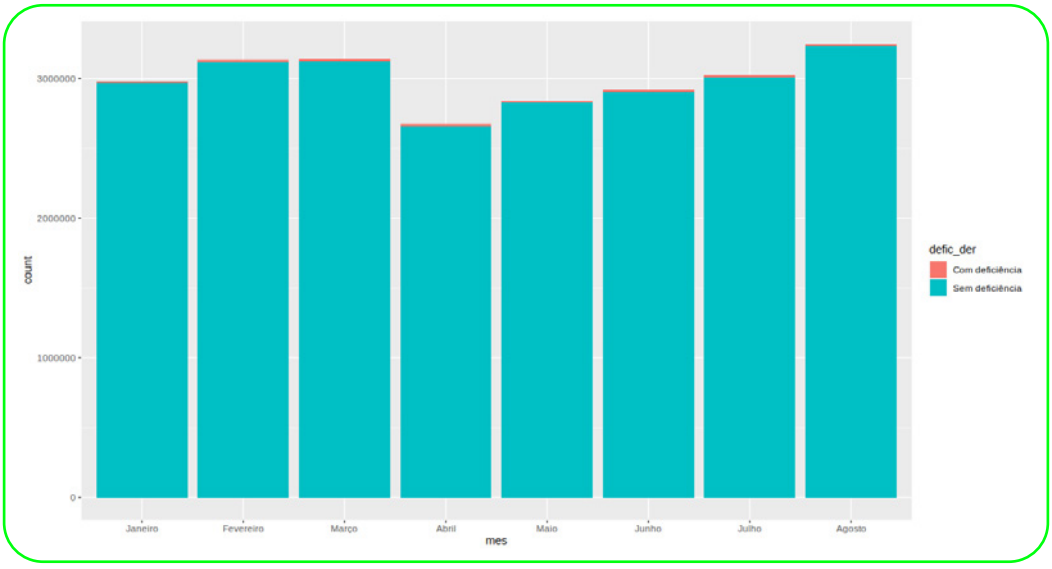
Entenda o comando:

ggplot(base3, : Um gráfico usando o pacote ggplot, aonde iremos plotar dados da base 3

aes(mes, : A variável que iremos trabalhar, aes significa aesthetic, é um parâmetro para indicar ao ggplot aonde ele deve olhar, no caso é na variável mês

fill=defic_der))+ : preencha a variável mês com a variável defic_der. O sinal de mais indica que terá outro comando, no caso:

geom_bar() : Opere todas essas operações através de um gráfico de barras.
O resultado será o seguinte:



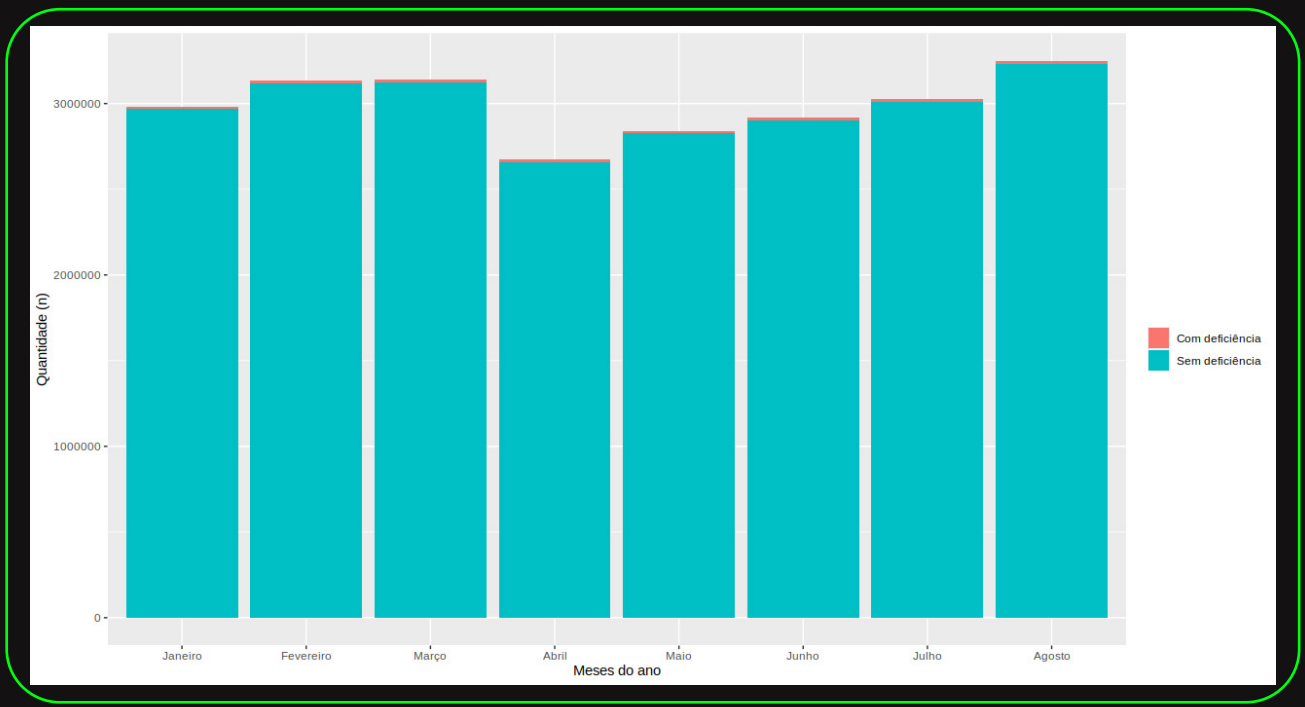
Veja que o design desse gráfico é melhor pensado do que os outros que fizemos anteriormente. E que, automaticamente o nome das variáveis já estão na parte de baixo e na lateral, incluído uma legenda para a variável que está preenchendo.

Quando fazemos um gráfico é preciso entender alguns pontos chaves.

Os eixos, são por onde os gráficos irão se guiar para traçar os pontos e linhas. Neste exemplo, no **eixo x** temos a variável mês. O eixo x sempre é o de baixo, podendo ser alterado apenas se você quiser. E o **eixo y** é sempre o vertical, nesse exemplo é a contagem dos registros, que varia até mais de três milhões. E a **legenda** do gráfico está na lateral direita. Nós podemos mudar manualmente o rótulo desses eixos adicionado uma linha de comando:

```
ggplot(base3, aes(mes, fill=defic_der))+
geom_bar()+
labs(x="Meses do ano", y="Quantidade (n)", fill="")
```

Nessa última linha que incluímos, estamos indicando através do comando labs, como queremos que se chame cada rótulo, o eixo será meses do ano, o eixo y será quantidade (n), e a legenda não terá título. O gráfico será dessa forma:



Escolhemos deixar a legenda sem título, pois ela é auto explicativa e, resumir variáveis é importante em um gráfico.

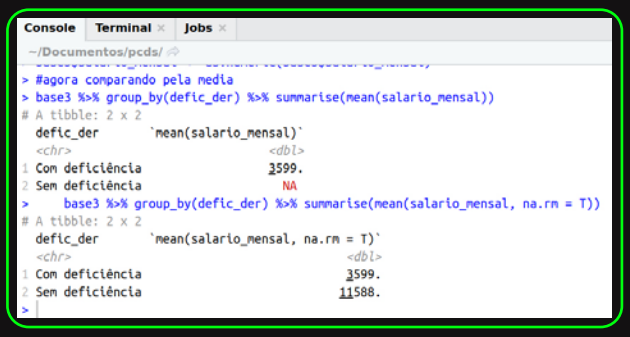
Agora precisamos adicionar um título, através do próprio comando labs, você pode adicionar um parâmetro de:
labs(x="Meses do ano", y="Quantidade (n)", fill="", title="Quantidade de trabalhadores com ou sem deficiência pelos meses de 2021")

Com esse gráfico e a tabela, você poderia resumir:
Entre os meses de Janeiro a Agosto o Brasil possuía aproximadamente 3 milhões de empregados sem deficiência e menos de 15 mil trabalhadores sem deficiência, de acordo com os dados do CAGED.

Essa primeira informação pode ainda levar em consideração os meses com maior e menor volume, e é nesta hora que a comunicação precisa entender o que gostaria de comunicar. A última comparação que faremos entre trabalhadores com ou sem deficiência é pela **renda**. Gostaríamos de responder a pergunta se trabalhadores com ou sem deficiência recebem de forma parecida. Como a variável renda salário mensal está em formato character precisamos modificá-la para formato numérico, afim de fazermos operações matemáticas:
base3$salario_mensal <- as.numeric(base3$salario_mensal)

Agora podemos comparar as médias, através do comando:
base3 %>% group_by(defic_der) %>% summarise(mean(salario_mensal))

Veja que o resultado exibido foi que a variável de salário de pessoas sem deficiência está vazio, nesse caso através do indicativo de NA. Isso ocorre porque, existem linhas sem esses registros, e podemos apenas pedir para que a média não leve em consideração registros ausentes, com o comando:
base3 %>% group_by(defic_der) %>% summarize (mean(salario_mensal, na.rm=T))



Através desta tabela, podemos afirmar que em média trabalhadores com deficiência ganham 31% a menos do que trabalhadores sem deficiência.
Ué mas de onde saiu esses 31%?

Bom, se o valor maior dos grupos é de 11588 basta você dividir 3590 por esse valor e, multiplicar por 100.

Vejam como conseguimos deflagrar desigualdades entre trabalhadores com ou sem deficiência. Mas você pode se perguntar, quem no Brasil ganha R$11.588 !?

Isso acontece porque a distribuição de salários é muito desigual em nosso país. Nós teremos pessoas que ganham menos que um salário mínimo e outras que ganham mais de R$200.000 por mês.

A **média é calculada através da soma de todos os registros, divididos pela quantidade de elementos**, ou seja, a gente vai somar o salário de todo mundo e dividir pela quantidade de pessoas, assim, se uma pessoa ganha R$10,00 e outra ganha R$30.000, elas duas em média ganham R$15.005.

Então veja que apesar da média ser uma medida de resumo que pode servir para muitos cenários, quando os dados são muito **heterogêneos**, ela pode representar mal os valores.

Para isso existem outras medidas de concentração dos dados, como a **mediana**.

Para saber a mediana, basta acrescentarmos mais um parâmetro no script, dessa forma:
base3 %>% group_by(defic_der) %>% summarise(mean(salario_mensal, na.rm = T), median(salario_mensal, na.rm = T))

O resultado será:

```
# A tibble: 2 x 3
  defic_der       `mean(salario_mensal, na.rm = T)` `median(salario_mensal, na.rm = T)`
  <chr>                                       <dbl>                               <dbl>
1 Com deficiência                             3599.                               1330.
2 Sem deficiência                            11588.                               1373.
> 
```

Veja que agora o valor da mediana informa que apesar de ser diferente, não é tão distante assim. **A mediana é o valor central dos registros**, ou seja nós organizamos do menor para o maior e pegamos exatamente o do meio (a mediana), ao fazer isso nós estamos dividindo os valores ao meio, e podemos afirmar:

50% dos trabalhadores com deficiência ganha até 1330, enquanto 50% dos trabalhadores sem deficiência ganham até 1373.

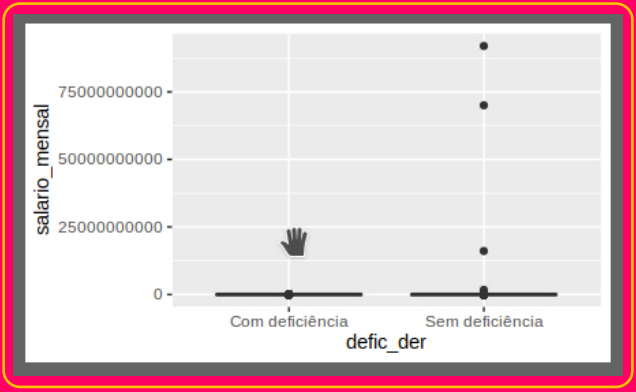
Perceba como é desafiador, quando temos muitos dados e precisamos reportar. Porque enquanto o valor da média é extremamente diferente, o valor mediano não é.

Mas e o restante dos valores?

Médias e medianas

Uma boa forma de representar isso é através de um boxplot, ou gráfico de caixinha como gostamos de falar. Ele irá trazer mais informações para além da mediana, e com o ggplot basta fazermos pequenas alterações para termos esse gráfico, com o script a seguir:

```
ggplot(base3, aes(defic_der, salario_mensal))+
geom_boxplot()
```



O gráfico exibido deflagrou que entre trabalhadores sem deficiência temos registros extremamente altos, passando de 75 **bilhões**. O que é representado por essas bolinhas no gráfico. E portanto, os valores são tão, tão, altos que nem mesmo uma caixinha foi possível de plotar, transformando o gráfico apenas em uma linha.

Para isso, podemos incluir um comando que **limite** a amplitude do eixo y. Para não fazermos de forma completamente arbitrária, podemos pedir primeiro que se exiba um resumo geral da variável salário, com o comando:

```
summary(base3$salario_mensal)
```

Teremos os seguintes valores:

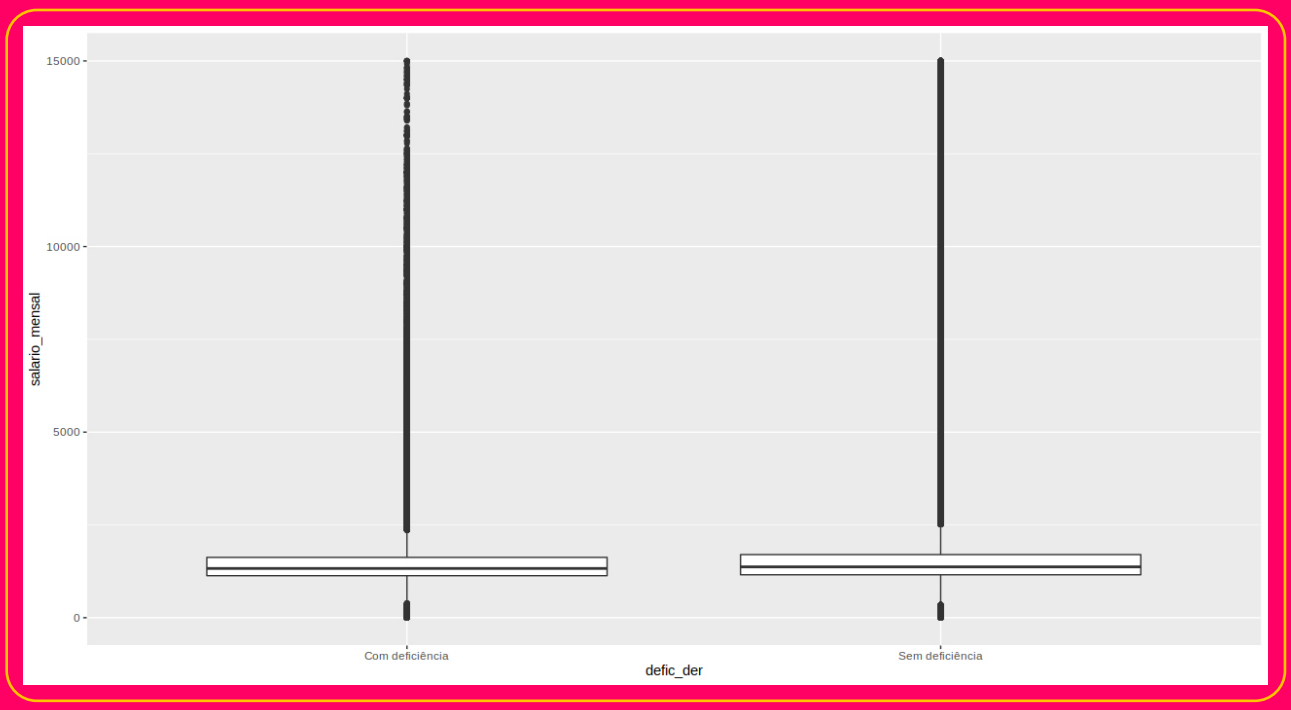
```
> #resumo da variável salário
>     summary(base3$salario_mensal)
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.       NA's
         0       1158       1373      11549       1711 91985969152         54
>
```

Que o valor **mínimo** é de 0,00, o **primeiro quartil (25%)** é de 1158, o valor mediano é de 1373, a média de 11549, o **terceiro quartil (75%)** possuem valores de até 1711, enquanto temos registro **máximo** de 91.958.969.152. E ainda sabemos que esse valor pertence a algum trabalhador sem deficiência, de acordo com o gráfico. Não temos como saber se esse valor é correto, ou se trata de um erro de digitação. Mas pelo menos podemos com segurança colocarmos limites no **eixo y** que incluam a maioria dos registros. Podemos incluir que o limite a ser plotado seja até de 15000, através do comando:

```
ggplot(base3, aes(defic_der, salario_mensal))+
geom_boxplot()+
ylim(0,15000)
```

A única linha que incluímos foi que queremos que o eixo y vá de 0 a 15000.

O resultado será o gráfico abaixo



A primeira linha da caixinha é justamente os 25% (ou primeiro quartil) dos registros de salário. A linha do meio é exatamente a mediana, a linha de cima da caixinha é o 75% (terceiro quartil) e a partir da linha mais grossa são todos os **valores extremos**. Então veja que, por mais que a gente tente reduzir a área do gráfico, por termos valores muito extremos, trabalhar com a renda de forma numérica **não funcionou tão bem**. Uma das soluções para esse problema é transformar a variável em **categorias** (as.factor) e, como estamos falando de salário mensal, podemos dividir em limites de salários mínimos, assim podemos ter outro tipo de informação, que é quantos por centos dos trabalhadores com e sem deficiência ganham 1 ou 2, ou mais salários mínimos. Para isso utilizaremos a função cut.

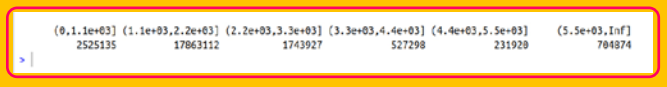
Categorizando variável numérica

Primeiro, crie uma variável nova:
base3$salario_cat <- ""

Depois coloque nessa variável nova, os valores "cortados" da variável do salário mensal, sabendo que a cada 1100 é um salário mínimo:
base3$salario_cat <- cut(base3$salario_mensal, breaks = c (0, 1099, 2199, 3299, 4399, 5499, Inf))

Veja como ficou, pedindo uma tabela:
table(base3$salario_cat)

O resultado será esse:



A informação que você está lendo é que existem 252.135 que ganham até um salário mínimo, 17.863.112 que ganham mais que um salário mínimo até dois salários mínimos, e assim sucessivamente, até a última categoria que varia de no mínimo 5 salários mínimos até o infinito. Agora podemos renomear os labels dessas variáveis para ficar mais fácil de entender dessa forma:

Primeiro verifique se já está como fator:
is.factor(base3$salario_cat)
Depois como estão os rótulos:
levels(base3$salario_cat)
E por fim atribua novos nomes:
levels(base3$salario_cat) <- c("< 1SM", "1SM < 2SM", "2SM < 3SM", "3SM < 4SM", "4SM < 5SM", "5SM OU +")

Agora podemos pedir uma tabela da variável de salário categorizada comparando entre trabalhadores com ou sem deficiência:
tabpct(base3$defic_der, base3$salario_cat, graph = F, percent = "row")

Entenda

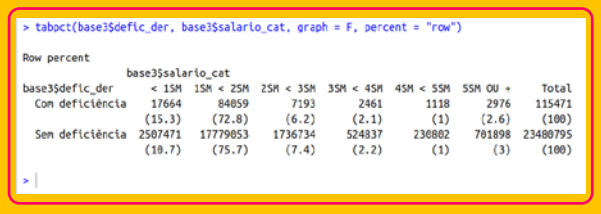
tabpct(: é uma tabela do pacote epidisplay para duas variáveis

base3$defic_der, base3$salario_cat, : a variável deficiência pelo salário categorizado

graph = F, : sem gráfico

percent = "row") : as porcentagens serão pela linha da tabela

O resultado que temos é esse:



Onde temos a informação que entre os trabalhadores com deficiência 15,3% ganham até um salário mínimo, enquanto entre os trabalhadores sem deficiência essa proporção é menor de apenas 10,7%.

E conforme você vai avaliando as outras colunas, percebemos que trabalhadores sem deficiência possuem maiores proporções para salários maiores.

Como essa desigualdade de recebimentos é esperada, escolhemos continuar a análise apenas entre os trabalhadores com deficiência.

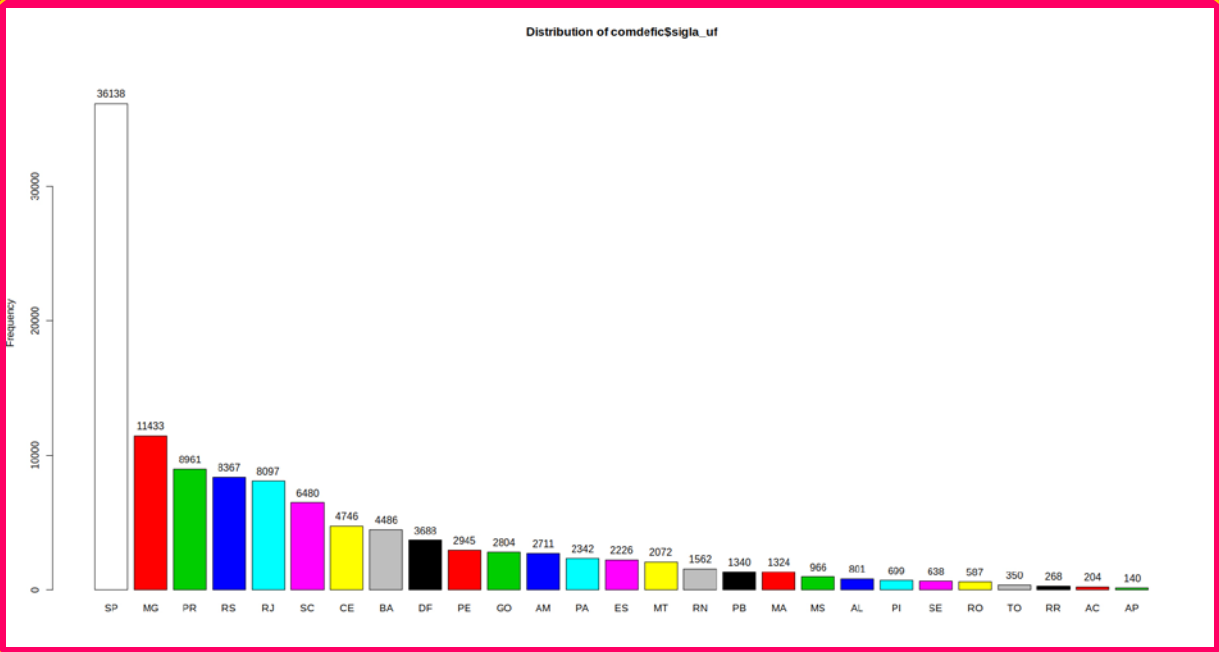
Lembra que decidimos partir de análises mais gerais para mais específicas?

Então as comparações regionais, de gênero, escolaridade e raça serão feitas apenas paranos trabalhadores com deficiência, para isso faremos um subconjunto apenas desses casos, através do comando:
comdefic <- subset(base3, defic_der=="Com deficiência")

Análises específicas apenas dos trabalhadores com deficiência

Qual a diferença por estados?
tab1(comdefic$sigla_uf, sort.group="decreasing")

Resultado em gráfico:



Gerando mapas

Como você pode perceber, a distribuição segue um contingente populacional, ou seja, estados com maior população também possuem maior volume de trabalhadores com deficiência. Como estamos trabalhando com dados por área geográfica, uma melhor visualização seria através de um mapa. Para isso, precisamos de um outro pacote que trabalha com mapas:

```
library(geobr)
library(sf)
```

Com esses pacotes ativados, podemos carregar o arquivo que possui os dados geográficos dos estados brasileiros:

```
estados <- read_state(year=2019)
```

Então agora temos dentro do R, um objeto que será a base de nosso mapa dos estados brasileiros. Depois disso precisamos da tabela de contagem para cada estado com a quantidade de trabalhadores com deficiência, para isso primeiro colocaremos dentro de um objeto qtde a tabela dessa contagem:

```
qtde <- table(comdefic$sigla_uf)
```

Agora transformaremos essa lista para que em uma coluna esteja a sigla da UF, e na outra coluna a quantidade informada através de um cbind. Que nada mais é do que colar lado a lado um objeto:

```
qtde <- cbind(Estados=row.names(qtde), qtde)
```

Para ver se está tudo certo, apenas chame esse objeto que você acabou de criar:

```
qtde
```

Agora transforme essa tabela em uma base de dados, ou seja um dataframe. Que possui nomes e números:

```
qtde <- as.data.frame.matrix(qtde)
```

Pelas operações que fizemos, a variável qtde está como tipo factor, para transformá-la em número, vamos primeiro deixá-la como caracter e depois em número:

```
qtde$qtde <- as.character(qtde$qtde)
qtde$qtde <- as.numeric(qtde$qtde)
```

Estamos quase lá, precisamos juntar o objeto com as informações geográficas do Brasil (estados) com esse dataframe que acabemos de criar, para isso faremos uma junção, tipo merge, mas pelo pacote dplyr:

```
estados2 <- dplyr::left_join(estados,
qtde, by = c("abbrev_state" =
"Estados"))
```

Só para dar um glam melhor no nosso mapa, vamos criar um objeto que informa que não queremos nenhum eixo plotado em nosso mapa:

```
no_axis <-
theme(axis.title=element_blank(),
      axis.text=element_blank(),
      axis.ticks=element_blank())
```

E por fim, vamos carregar um pacote auxiliar para gráficos, que entende muito sobre como preencher informações de forma gradiente. Você vai ver como vai ficar legal no produto final:

```
library(RColorBrewer)
```

E finalmente, plotaremos o mapa:

```
mapa <- ggplot() +
geom_sf(data=estados2, aes(fill=qtde),
size=.15) +
labs(subtitle="Distribuiçao de
trabalhadores com deficiência por
estados do BR", size=8) +
scale_fill_distiller(palette = "YlOrRd",
direction = 1, name="Quantidade", limits
= c(min(estados2$qtde),
max(estados2$qtde))) +
theme_bw()+
no_axis
```

MAPA

Entenda:

ggplot() + : Você já sabe que é para criar gráficos com o ggplotw

geom_sf(: Informa que o gráfico se trata de um mapa

data=estados2, : A base é a estados2 que criamos com o join

aes(fill=qtde), : O preenchimento de cada unidade do gráfico será de acordo com a quantidade

size=.15)+ : é a espessura da linha entre cada UF

labs : Será todos os rótulos exibidos, nesse caso estamos pondo apenas um subtítulo, e esse rótulo terá o tamanho 8

scale_fill_distiler(: O preenchimento das UFs será de forma gradual com a quantidade

palette="YlOrRd", : a paleta de cores que trabalha com tons de amarelo e vermelho

direction=1, : Os tons mais fortes serão de acordo com maior quantidade

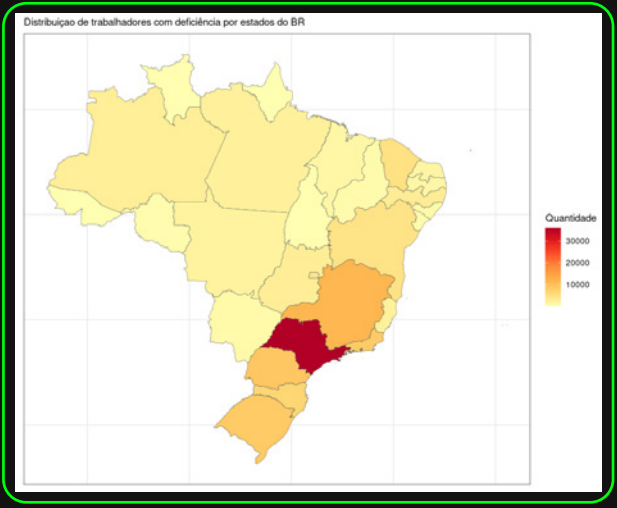
name="Quantidade", : Esse preenchimento funciona como uma legenda e exibirá o rótulo de Quantidade

limits = c(min(estados2$qtde), max(estados2$qtde))) + : A cor mais clara será de acordo com o valor mínimo da variável qtde, bem como a cor mais forte será de acordo com o valor máximo dessa mesma variável

theme_bw()+ : Um tema simplista, sem muitos detalhes do ggplot

no_axis : o objeto que criamos acima, retirando todos os eixos do gráfico final.

UFA! Esse script é longo, mas o resultado é ótimo:



QUAL A DIFERENÇA ENTRE GÊNEROS DOS TRABALHADORES COM DEFICIÊNCIA?

tab1(comdefic$sexo, sort.group="decreasing", graph=F)

```
comdefic$sexo :
          Frequency Percent Cum. percent
Masculino     73208    62.9         62.9
Feminino      43167    37.1        100.0
  Total      116375   100.0        100.0
```

A maioria dos trabalhadores com deficiência são de gênero masculino sendo 62,9% do total.

É importante frisar que quando nós temos apenas duas categorias, como nessa variável do gênero, basta informar uma categoria, pois a segunda é sua complementar. Mas se quiséssemos dar ênfase na minoria feminina, poderíamos informar ao contrário, que mulheres representam apenas 37,1% dos cadastrados no CAGED.

QUAL A DIFERENÇA RACIAL ENTRE OS TRABALHADORES COM DEFICIÊNCIA?

tab1(comdefic$raca_cor, sort.group="decreasing", graph=F)

```
comdefic$raca_cor :
          Frequency Percent Cum. percent
Branca        46941    40.3         40.3
Parda         44499    38.2         78.6
Ignorado      15354    13.2         91.8
Preta          8121     7.0         98.7
Amarela        1255     1.1         99.8
Indígena        205     0.2        100.0
  Total      116375   100.0        100.0
```

A raça mais predominante entre os cadastrados no caged são de trabalhadores declarados brancos, representando 40,3% do total, seguidos por trabalhadores da cor parda.

Aqui no data_labe nós entendemos a importância do Movimento Negro em agrupar as raças/cores preta e parda como negra, portanto , podemos criar uma outra variável para raça, apenas mudando os labels Preta e Parda para Negra:

Criando uma variável nova que recebe todas as informações da variável raca_cor

comdefic$raca2 <- comdefic$raca_cor

Pedindo para visualizar a ordem dos labels da variável:

levels(comdefic$raca2)

Mudando apenas a Parda e Preta para Negra

levels(comdefic$raca2) <- c("Amarela", "Branca", "Ignorado", "Indígena","Negra","Negra")

Agora podemos pedir novamente a tabela com essa nova variável:

tab1(comdefic$raca2, sort.group="decreasing", graph = F)

O resultado será:

```
comdefic$raca2 :
          Frequency Percent Cum. percent
Negra         52620    45.2         45.2
Branca        46941    40.3         85.6
Ignorado      15354    13.2         98.7
Amarela        1255     1.1         99.8
Indígena        205     0.2        100.0
  Total      116375   100.0        100.0
```

Veja que os trabalhadores negros com deficiência apesar de serem mais predominantes, não são a maioria, representam apenas 45,2% do total. Enquanto a população negra brasileira é aproximadamente 54% (Fonte: Jornal da USP com dados do IBGE). Além disso 13,2% dos trabalhadores não possuem essa informação. A raça é um ponto muito marcante de desigualdade social no Brasil e esse volume de desinformação é fundamental de ser relatado.

GRAU DE INSTRUÇÃO
tab1(comdefic$grau_instrucao, sort.group="decreasing")

comdefic$grau_instrucao :

	Frequency	Percent	Cum. percent
Médio Completo	65139	56.0	56.0
Superior Completo	12578	10.8	66.8
Médio Incompleto	9043	7.8	74.6
Fundamental Completo	8626	7.4	82.0
6ª a 9ª Fundamental	7067	6.1	88.0
Superior Incompleto	6396	5.5	93.5
Até 5ª Incompleto	3024	2.6	96.1
5ª Completo	2040	1.8	97.9
Pós Grad. Completa	1460	1.3	99.1
Analfabeto	737	0.6	99.8
Mestrado	192	0.2	99.9
Doutorado	73	0.1	100.0
Total	116375	100.0	100.0

A maioria dos trabalhadores possuem o ensino médio completo, representando 56% do total. Trabalhadores com ensino superior completo são o segundo mais predominante, representando apenas 10,8%.

POR TIPO DE DEFICIÊNCIA
tab1(comdefic$tipo_deficiencia, sort.group="decreasing", graph=F)

comdefic$tipo_deficiencia :

	Frequency	Percent	Cum. percent
Física	50260	43.2	43.2
Visual	21703	18.6	61.8
Auditiva	21264	18.3	80.1
Intelectual	11179	9.6	89.7
Reabilitado	7626	6.6	96.3
Múltipla	4343	3.7	100.0
Total	116375	100.0	100.0

Os trabalhadores com deficiência física são os mais predominantes, sendo 43,2%. Aqui vamos trazer uma informação qualitativa de nossa residência com pessoas com deficiência, os participantes relataram que a maioria das empresas prefere contratar pessoas com deficiências menos perceptíveis ou de aparente menor comprometimento, e que na maioria das vezes são deficiências físicas. O que isso quer dizer sobre a forma que as empresas selecionam trabalhadores com deficiência?

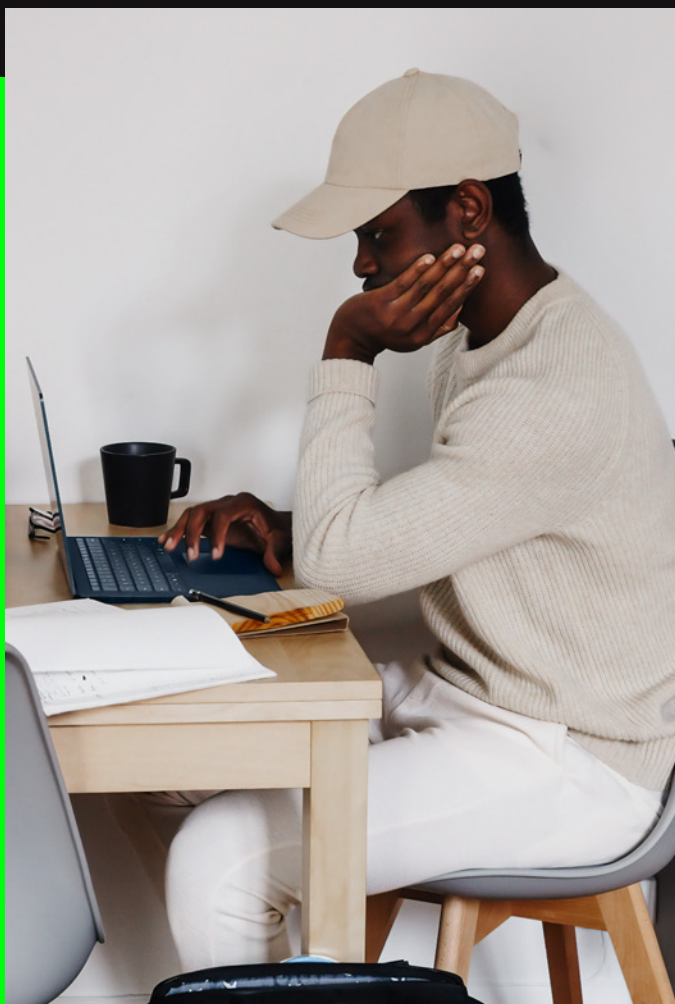
POR SETOR

tab1(comdefic$Descrição.y,graph=F ,sort.group="decreasing")

	Frequency	Percent	Cum. percent
Comércio, Reparação de Veículos Automotores e Motocicletas	28441	24.4	24.4
Indústrias de Transformação	25133	21.6	46.0
Atividades Administrativas e Serviços Complementares	13617	11.7	57.7
Saúde Humana e Serviços Sociais	8781	7.5	65.3
Construção	8581	7.4	72.7
Transporte, Armazenagem e Correio	6692	5.8	78.4
Educação	4325	3.7	82.1
Alojamento e Alimentação	3744	3.2	85.3
Informação e Comunicação	3175	2.7	88.1
Atividades Financeiras, de Seguros e Serviços Relacionados	3069	2.6	90.7
Atividades Profissionais, Científicas e Técnicas	2885	2.5	93.2
Outras Atividades de Serviços	2785	2.4	95.6
Agricultura, Pecuária, Produção Florestal, Pesca e AqÜIcultura	2391	2.1	97.6
Água, Esgoto, Atividades de Gestão de Resíduos e Descontaminação	977	0.8	98.5
Artes, Cultura, Esporte e Recreação	474	0.4	98.9
Indústrias Extrativas	376	0.3	99.2
Eletricidade e Gás	339	0.3	99.5
Administração Pública, Defesa e Seguridade Social	335	0.3	99.8
Atividades Imobiliárias	249	0.2	100.0
Serviços Domésticos	6	0.0	100.0
Total	116375	100.0	100.0

Se juntarmos as três categorias mais predominantes podemos afirmar que mais da metade dos trabalhadores com deficiência trabalham no comércio, reparação de veículos automotores ou em indústrias de transformação ou em atividades administrativas. E que o setor com menor participação de trabalhadores com deficiência é no de serviços domésticos, contando com apenas 6 pessoas em todo o período de análise.

Olha quanta informação nova nós conseguimos extrair graças ao trabalho que tivemos de manipular, extrair, tratar o dado original. Por isso é muito importante a gente democratizar esse conhecimento em dados, para que mais pessoas possam produzir e consumir informação.



Para sabermos se existem diferenças da renda entre essas diversas categorias poderíamos visualizar tanto tabelas como gráficos. Uma para cada um dariam noção de certas diferenças. No data_labe, grandes discussões são feitas sobre o que reportar de um relatório de dados como esse.

Um objetivo da equipe de dados é justamente tentar extrair o máximo de dados das bases e reportar à equipe de jornalismo tudo isso. Posteriormente, a jornalista com a edição de reportagens escolhe qual caminho seguir.

Para facilitar nossa vida, podemos tentar resumir o máximo de variáveis em um gráfico. Como em muitos casos, mulheres negras são mais vulnerabilizadas pelas interseccionalidades de gênero e raça, podemos fazer um gráfico que resuma as variáveis de gênero e raça pela variável do salário categorizado:

```
ggplot(comdefic, aes(raca2,
fill=salario_cat))+
geom_bar(position = 'fill')+
facet_wrap(~sexo)+
labs(x="Raça/cor", y="Porcentagem",
fill="Faixas salarias")
```

Entenda:

ggplot(comdefic, : Um gráfico pelo ggplot usando a base comdefic

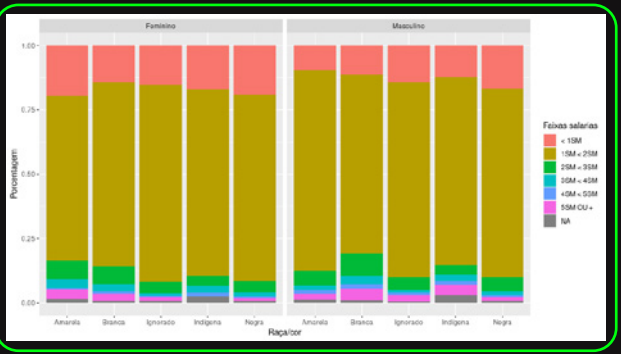
aes(raca2, fill=salario_cat))+ : Usando a variável nova da raça, preenchendo pelas faixas salariais e

geom_bar(position = 'fill')+ : A forma de preenchimento será total, ou seja preenchida percentualmente

facet_wrap(~sexo)+ : Divida o gráfico pelas categorias de gênero

labs(x="Raça/cor", y="Porcentagem", fill="Faixas salarias") : As informações dispostas nos rótulos

O resultado será:



Nesse gráfico, nós temos MUITAS informações, e fica difícil ao olho não treinado entender tudo o que está acontecendo.

Em primeiro, olhe a parte superior do gráfico, está dividido entre feminino e masculino. Ou seja, podemos comparar de forma geral homens de mulheres.

As barras estão preenchidas com a representação proporcional da quantidade de pessoas que recebem de acordo com as faixas salariais informadas.

Se olharmos apenas a parte de cima das barras na cor rosa, vejam que em geral o lado “feminino” possui maior proporção de pessoas que ganham até um salário mínimo, indicando que no geral mais mulheres com deficiência ganham até um salário mínimo quando comparadas aos homens.

Deslocando agora os olhos para baixo, com as barras nas cores de maiores salários (verde, azul e roxo) as mulheres negras possuem proporções menores comparadas às mulheres de outras raça, e também comparado a qualquer grupo racial entre os homens.

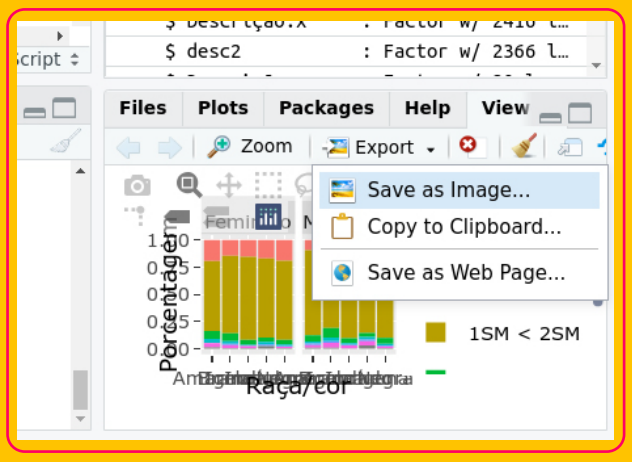
Criando um gráfico interativo

Por fim, podemos ainda criar um gráfico que exiba essas porcentagens ao passar o mouse através de um objeto em html pelo pacote plotly.
Então, primeiro instale o pacote e depois ative-o:
install.packages("plotly")
library(plotly)

Repita o mesmo script do gráfico anterior guardando em um objeto plotly e veja:

```
ggplotly(ggplot(comdefic, aes(raca2,
fill=salario_cat))+
        geom_bar(position = 'fill')+
        facet_wrap(~sexo)+
        labs(x="Raça/cor",
y="Porcentagem", fill="Faixas salarias"))
```

Você pode exportar o gráfico para uma página tipo web, através do botão "export" do lado direito inferior do R studio.



Esse arquivo html pode ser visualizado através do download deste link abaixo:
Clique para abrir em seu navegador

Passando o mouse sobre cada casela das barras, você pode reportar as diferenças da renda mensal de cada grupo de gênero e raça. Faremos essa comparação entre homens brancos e mulheres negras:

19% das trabalhadoras negras com deficiência ganham até um salário mínimo. Entre trabalhadores brancos com deficiência, apenas 11% recebem o mesmo valor.
No outro extremo, apenas 1% de trabalhadoras negras com deficiência ganham mais que cinco salários mínimos, enquanto 5% de homens branco ganham a mesma quantia.

Por fim, existem outras análises que você pode continuar fazendo com as variáveis que tratamos, como a diferença entre as faixas escolares, entre os empregos por seção, etc. Use e abuse das tabelas e gráficos que você aprendeu. Para mais cruzamentos use as funções de group_by do dplyr e, a função de facet_wrap do ggplot.

CONCLUSÃO

A participação de pessoas com deficiência no mercado de trabalho ainda é muito pequena. Essas desigualdades são agravadas fora das regiões sul e sudeste do Brasil. As remunerações de pessoas com e sem deficiência são discrepantes, principalmente nos rendimentos mais altos para pessoas sem deficiência. Há uma maior participação de homens com deficiência no mercado de trabalho, mas eles estão concentrados em poucos setores do mercado.

Por fim, desigualdades salariais podem ser vistas entre mulheres e homens, e entre raças, sendo o grupo de mulheres negras continuamente o grupo que sofre maiores exclusões e desvalorização.

NOTAS FINAIS

Nós acreditamos que um bom prato se faz com muito tempero, dessa forma, a mistura dos dados, reportagem, visualização e edição podem ser muito saborosas na disseminação do conhecimento e na notificação de realidades que nós vivemos nas periferias.

Incentivamos a formação de todas as pessoas que queiram aprender essas ferramentas e compartilhar narrativas através do conhecimento de dados. Se você quiser continuar trocando ideia sobre isso ou se tiver alguma dúvida, fale com a gente pelo nosso email institucional: contato@datalabe.org

Queremos ainda lhe convidar para continuar acompanhando o nosso trabalho em nosso site datalabe.org ou através de nossas redes sociais pelo @data_labe.

O script completo das análises realizadas nesta cartilha encontra-se disponível para download neste repositório.

AGRADECIMENTOS

Agradecemos à Fundação Ford por acreditar em nosso potencial como produtores e formadores de conhecimento, através do financiamento para este projeto.

Agradecemos ainda à toda a equipe do data_labe, principalmente a quem participou diretamente na elaboração e produção desta cartilha, em especial a: Elena Wesley, Dhannyd Quaresma, Priscilla Souza, Gilberto Vieira e Nicolas Noel.

Agradecemos imensamente a Tainara Cardoso e a Gabriely Dutra, que foram as facilitadoras de nossa residência para jovens com deficiência. Sem elas teria sido muito mais difícil produzir este material.

E agradecemos a você que chegou até aqui.
Obrigado!

Polinho (Paulo Mota Medeiros Júnior)
Coordenador de dados do data_labe.

data_
labe