GOVERNO FEDERAL

BRASIL

UNIÃO E RECONSTRUÇÃO

Ministério da Previdência Social
Secretaria do Regime Geral da Previdência Social
Departamento de Perícia Médica Federal
Coordenação-Geral de Assuntos Corporativos e Disseminação de Conhecimento

Gustavo Magalhães Mendes de Tarso

RELATÓRIO INSTITUCIONAL – SISTEMA atestmed-defender

Sistema: **atestmed-defender**

Gerado em: 04/09/2025 19:01

Introdução

O projeto atestmed-defender é uma iniciativa que visa otimizar a geração de relatórios institucionais e a análise de dados de produtividade por meio de scripts automatizados. Utilizando tecnologias avançadas como inteligência artificial e automação de navegadores, o projeto oferece ferramentas robustas para a criação de documentos em formatos variados, como Org-mode, HTML e PDF. Os scripts em Python e R são projetados para facilitar a manipulação de dados e a geração de conteúdo, permitindo que os usuários personalizem e ajustem os relatórios conforme suas necessidades específicas.

Os scripts em Python, como `gerar_documentacao_projeto.py` e `gerar_relatorio_html_pdf_IA.py`, são fundamentais para a criação de relatórios detalhados e personalizados. Eles utilizam a API da OpenAI para gerar conteúdo de forma fragmentada, garantindo um tom consistente através de um pós-polimento opcional. Além disso, a integração com Playwright/Chromium permite a exportação de relatórios em formatos visuais atraentes, como HTML e PDF, com opções de personalização de capa e layout.

Paralelamente, os scripts em R, como `r_checks/03_productivity_check.R` e `r_checks/02_le15s_check.R`, são essenciais para a análise de dados de produtividade. Eles utilizam pacotes como `DBI`, `dplyr` e `ggplot2` para realizar consultas em bancos de dados SQLite e gerar visualizações gráficas. Esses scripts são configurados para aceitar argumentos de linha de comando, permitindo uma análise flexível e adaptável às necessidades dos usuários. A combinação dessas ferramentas proporciona uma solução abrangente para a geração de relatórios e análise de dados, promovendo eficiência e precisão nas análises de tarefas.

Contexto do Problema

O sistema atestmed-defender enfrenta o desafio de integrar e otimizar a geração de relatórios institucionais e análises de produtividade de especialistas, utilizando uma combinação de scripts em Python e R. Os scripts em Python, como `gerar_documentacao_projeto.py` e `gerar_relatorio_html_pdf_IA.py`, são responsáveis por criar relatórios em diferentes formatos, aproveitando a inteligência artificial para gerar conteúdo e personalizar a apresentação. Esses scripts dependem de variáveis de ambiente para configurar o modelo de IA, a exportação de documentos e o pós-polimento, além de interagir com a API da OpenAI para obter conteúdo gerado automaticamente.

Paralelamente, os scripts em R, como `r_checks/03_productivity_check.R` e `r_checks/02_le15s_check.R`, realizam análises detalhadas de produtividade e outras métricas relacionadas a dados de tarefas, utilizando bancos de dados SQLite. Esses scripts empregam pacotes como `DBI`, `dplyr` e `ggplot2` para manipulação e visualização de dados, permitindo uma análise aprofundada e a geração de gráficos e relatórios em formatos de imagem e texto. A integração dessas ferramentas visa proporcionar uma visão abrangente e detalhada do desempenho e das métricas de produtividade, facilitando a tomada de decisões informadas.

Os arquivos alvo listados, como `./reports/make_impact_report.py` e `graphs_and_tables/compare_productivity.py`, são componentes críticos desse sistema, cada um contribuindo com funcionalidades específicas para a geração de relatórios e análises. A coordenação entre esses scripts é essencial para garantir que os dados sejam processados de maneira eficiente e que os relatórios finais reflitam com precisão as análises realizadas. O desafio reside em manter a coerência e a precisão dos dados ao longo de todo o processo, desde a coleta até a apresentação final.

Soluções Desenvolvidas

O sistema atestmed-defender desenvolveu uma série de soluções inovadoras para otimizar a geração e análise de relatórios institucionais. O script `gerar_documentacao_projeto.py` utiliza inteligência artificial para criar relatórios em formato Org-mode, permitindo um processo fragmentado que facilita a integração de diferentes seções e um pós-polimento opcional para uniformizar o tom do documento. Este script é altamente configurável através de variáveis de ambiente que controlam aspectos como o modelo de IA utilizado e a exportação para PDF, garantindo flexibilidade e personalização conforme as necessidades do usuário.

Além disso, o script `gerar_relatorio_html_pdf_IA.py` expande as capacidades de geração de relatórios ao produzir documentos em HTML e PDF com o auxílio do Playwright/Chromium. Ele oferece funcionalidades avançadas, como a personalização de capas e ajustes de layout, enquanto mantém a interação com a API da OpenAI para a criação de conteúdo. As variáveis de ambiente permitem uma configuração detalhada, assegurando que os relatórios atendam aos padrões institucionais desejados.

Para análises de produtividade e outras métricas, os scripts em R, como `r_checks/03_productivity_check.R` e `r_checks/02_le15s_check.R`, oferecem ferramentas robustas para a análise de dados armazenados em bancos de dados SQLite. Utilizando pacotes como `DBI`, `dplyr`, e `ggplot2`, esses scripts permitem a manipulação eficiente de dados e a geração de visualizações gráficas, facilitando a interpretação dos resultados. A capacidade de aceitar argumentos de linha de comando para especificar parâmetros de análise torna esses scripts altamente adaptáveis a diferentes contextos e necessidades analíticas.

Indicadores Estratégicos

Os indicadores estratégicos do sistema atestmed-defender são fundamentais para avaliar o desempenho e a eficácia das análises de tarefas realizadas. O script `r_checks/03_productivity_check.R`, por exemplo, desempenha um papel crucial ao calcular a produtividade dos especialistas, utilizando dados de um banco de dados SQLite. Ele não apenas analisa a eficiência individual, mas também gera visualizações gráficas que facilitam a interpretação dos resultados, permitindo ajustes estratégicos baseados em evidências concretas.

Além disso, o script `r_checks/02_le15s_check.R` complementa essa análise ao focar em métricas específicas relacionadas ao tempo de execução das tarefas, oferecendo uma visão detalhada sobre a eficiência temporal das operações. A combinação desses scripts com os relatórios gerados por `make_kpi_report.py` e `make_impact_report.py` proporciona uma visão abrangente dos indicadores de desempenho, permitindo que a organização identifique áreas de melhoria e potencialize suas estratégias operacionais.

Os scripts de comparação, como `graphs_and_tables/compare_productivity.py` e `graphs_and_tables/compare_nc_rate.py`, oferecem insights adicionais ao contrastar diferentes métricas de desempenho. Esses comparativos são essenciais para entender variações e tendências ao longo do tempo, ajudando a alinhar as práticas operacionais com os objetivos estratégicos da organização. Em conjunto, esses indicadores formam uma base sólida para a tomada de decisões informadas, garantindo que o sistema atestmed-defender continue a evoluir e a atender às necessidades dos seus usuários de maneira eficaz.

Importância para a Gestão Estratégica

A gestão estratégica se beneficia significativamente da implementação de ferramentas como o `atestmed-defender`, que integra scripts em Python e R para a geração de relatórios e análises de dados. Esses scripts automatizam a coleta e análise de dados, permitindo que gestores tomem decisões informadas com base em informações precisas e atualizadas. Por exemplo, o script `gerar_documentacao_projeto.py` utiliza inteligência artificial para criar relatórios detalhados, enquanto o `gerar_relatorio_html_pdf_IA.py` oferece flexibilidade na apresentação dos dados, gerando documentos em formatos HTML e PDF. Essa automação não apenas economiza tempo, mas também garante consistência e precisão nos relatórios, elementos cruciais para a formulação de estratégias eficazes.

Além disso, os scripts em R, como o `r_checks/03_productivity_check.R`, fornecem insights valiosos sobre a produtividade dos especialistas, permitindo que a gestão identifique áreas de melhoria e aloque recursos de maneira mais eficiente. A capacidade de analisar dados complexos e gerar visualizações claras e compreensíveis facilita a comunicação dos resultados para diferentes partes interessadas, promovendo uma cultura de transparência e responsabilidade. A integração de análises quantitativas e qualitativas através desses scripts fortalece a capacidade da organização de responder rapidamente a mudanças no ambiente operacional, ajustando suas estratégias conforme necessário.

Por fim, a utilização de scripts como `make_impact_report.py` e `make_kpi_report.py` para a criação de relatórios de impacto e indicadores-chave de desempenho (KPIs) é essencial para monitorar o progresso em direção aos objetivos estratégicos. Esses relatórios fornecem uma visão abrangente do desempenho organizacional, permitindo que a gestão identifique tendências, avalie a eficácia das iniciativas em andamento e ajuste suas abordagens para maximizar o impacto. A capacidade de gerar relatórios detalhados e precisos é um componente vital da gestão estratégica moderna, garantindo que as decisões sejam baseadas em dados concretos e análises robustas.

Importância do Uso de Inteligência Artificial

A utilização de inteligência artificial (IA) no desenvolvimento de scripts como `gerar_documentacao_projeto.py` e `gerar_relatorio_html_pdf_IA.py` é crucial para otimizar a geração de relatórios institucionais. Esses scripts automatizam a criação de documentos complexos, permitindo que as organizações produzam relatórios detalhados e personalizados com eficiência e precisão. A IA facilita a divisão do trabalho em tarefas menores e gerenciáveis, garantindo que cada seção do relatório seja elaborada com um tom consistente e profissional. Além disso, a capacidade de interagir com APIs, como a da OpenAI, amplia as possibilidades de personalização e refinamento dos conteúdos gerados, assegurando que os relatórios atendam às necessidades específicas de cada projeto.

Nos scripts em R, como `r_checks/03_productivity_check.R` e `r_checks/02_le15s_check.R`, a inteligência artificial desempenha um papel fundamental na análise de dados complexos. Através do uso de pacotes avançados para manipulação e visualização de dados, esses scripts conseguem processar grandes volumes de informações de maneira eficiente, gerando insights valiosos sobre a produtividade e outros indicadores de desempenho. A IA permite que essas análises sejam realizadas de forma mais rápida e precisa, reduzindo o tempo necessário para a interpretação dos dados e aumentando a capacidade de resposta das organizações às mudanças nos padrões de desempenho.

A integração da inteligência artificial em processos de geração de relatórios e análise de dados não apenas melhora a eficiência operacional, mas também eleva a qualidade das informações apresentadas. Isso é especialmente importante em contextos onde decisões críticas dependem da precisão e clareza dos dados analisados. Ao automatizar tarefas repetitivas e complexas, a IA libera recursos humanos para se concentrarem em atividades estratégicas, promovendo inovação e melhorando a competitividade das organizações no mercado.

Impactos Institucionais

Os impactos institucionais do sistema atestmed-defender são significativos, especialmente no que diz respeito à automação e eficiência na geração de relatórios e análises de dados. O uso dos scripts em Python e R, como `gerar_documentacao_projeto.py` e `gerar_relatorio_html_pdf_IA.py`, permite a criação de relatórios detalhados e personalizados, otimizando o tempo e os recursos necessários para a elaboração de documentos institucionais. A integração com a API da OpenAI para a geração de conteúdo e a utilização de Playwright/Chromium para a formatação em HTML e PDF garantem que os relatórios sejam não apenas informativos, mas também visualmente atraentes e de fácil leitura.

Além disso, os scripts em R, como `r_checks/03_productivity_check.R` e `r_checks/02_le15s_check.R`, proporcionam uma análise aprofundada da produtividade e eficiência das análises de tarefas, utilizando dados armazenados em bancos de dados SQLite. A capacidade de gerar gráficos e relatórios de produtividade, bem como de realizar comparações detalhadas através de scripts como `compare_productivity.py` e `compare_nc_rate.py`, oferece uma visão abrangente do desempenho institucional. Isso facilita a identificação de áreas de melhoria e a tomada de decisões estratégicas baseadas em dados concretos.

A implementação desses scripts e a estruturação dos dados em relatórios claros e concisos têm o potencial de transformar a maneira como as instituições gerenciam e avaliam suas operações. Ao automatizar processos complexos e fornecer insights acionáveis, o sistema atestmed-defender não apenas melhora a eficiência operacional, mas também fortalece a capacidade das instituições de responder rapidamente a desafios e oportunidades emergentes.

Conclusão

Em conclusão, o conjunto de scripts descritos demonstra uma abordagem robusta e integrada para a geração de relatórios institucionais e a análise de dados de produtividade. Os scripts em Python, como `gerar_documentacao_projeto.py` e `gerar_relatorio_html_pdf_IA.py`, utilizam inteligência artificial para automatizar a criação de documentos, permitindo personalização e eficiência no processo de geração de relatórios em diferentes formatos. A capacidade de dividir o trabalho em chamadas menores e aplicar um pós-polimento garante que o conteúdo final seja coeso e de alta qualidade.

Paralelamente, os scripts em R, como `r_checks/03_productivity_check.R` e `r_checks/02_le15s_check.R`, oferecem uma análise detalhada de dados de produtividade, utilizando pacotes poderosos para manipulação e visualização de dados. A estrutura modular desses scripts permite uma análise flexível e precisa, com a possibilidade de exportar resultados em formatos visuais e textuais. A integração com bancos de dados SQLite e a gestão cuidadosa das conexões garantem a eficiência e a integridade dos dados analisados.

No geral, a combinação dessas ferramentas proporciona uma solução abrangente para a geração de relatórios e análise de dados, atendendo às necessidades institucionais de maneira eficaz. A utilização de tecnologias avançadas, como a API da OpenAI e pacotes de análise de dados em R, destaca o compromisso com a inovação e a melhoria contínua dos processos de documentação e análise.

Arquivos e Saídas Utilizados pelo Relatório

`./reports/make_impact_report.py` gera um relatório em PDF e PNGs sobre o impacto na fila, utilizando bibliotecas como ReportLab e Matplotlib. O script organiza dados em tabelas, removendo colunas específicas e ordenando de acordo com critérios predefinidos, como IV e Excesso. As saídas incluem gráficos de alta resolução e um PDF que não depende de LaTeX. O script também formata métricas com duas casas decimais e ajusta nomes de peritos para Title Case. Ele integra-se com outro script para obter os 10 piores peritos, caso necessário.

`./reports/make_kpi_report.py` cria relatórios ATESTMED, que podem ser individuais ou focados nos 10 piores peritos. O script combina gráficos gerados em Python, análises em R e comentários, resultando em relatórios em formatos Org e PDF. Ele também analisa padrões de dias da semana versus fins de semana e inclui apêndices. O script organiza diretórios para exportação e saída, e carrega variáveis de ambiente de um arquivo .env, se disponível.

`graphs_and_tables/compare_fifteen_seconds.py` compara a porcentagem de perícias concluídas em até um determinado tempo (THRESHOLD) entre um perito específico ou os 10 piores peritos e o restante do Brasil. O script normaliza durações, calcula métricas específicas e oferece várias opções de exportação, incluindo PNG, Org, e comentários automáticos via API. Ele também pode gerar gráficos ASCII no terminal e incorpora comentários no arquivo Org principal.

`graphs_and_tables/compare_indicadores_composto.py` analisa indicadores compostos de um perito ou dos 10 piores peritos em comparação com o Brasil, excluindo o grupo analisado. Os indicadores incluem porcentagem de não conformidade, produtividade, tarefas concluídas em até 15 segundos e sobreposição de análises. O script permite definir cortes para esses indicadores e exporta resultados em PNG, Org, e comentários automáticos. Ele também pode gerar gráficos ASCII e integrar-se com uma API para comentários.

`graphs_and_tables/compare_motivos_perito_vs_brasil.py` compara os motivos de não conformidade entre um perito específico ou os 10 piores peritos e o restante do Brasil. O script utiliza descrições textuais para os motivos e oferece várias opções de exportação, incluindo Org, MD, PNG, e comentários automáticos. Parâmetros visuais permitem ajustar o tamanho e a abreviação dos rótulos dos gráficos. O modo Top 10 agrupa os piores peritos com base em um critério de elegibilidade.

`graphs_and_tables/compare_nc_rate.py` compara a porcentagem de motivos de não conformidade entre um perito específico ou os 10 piores peritos e o restante do Brasil. O script utiliza descrições textuais para os motivos e oferece opções de exportação em Org, MD, PNG, e comentários automáticos. Parâmetros visuais permitem ajustar o tamanho e a abreviação dos rótulos dos gráficos. Ele também suporta a geração de gráficos ASCII e a integração com uma API para comentários.

`graphs_and_tables/compare_overlap.py` analisa a sobreposição de análises entre um perito específico ou os 10 piores peritos e o restante do Brasil. O script utiliza Matplotlib para gerar gráficos PNG em ambientes sem interface gráfica e oferece opções de exportação em Org e comentários automáticos. Ele também pode gerar gráficos ASCII e integrar-se com uma API para comentários, embora essa integração não seja mais obrigatória para gerar comentários.

`graphs_and_tables/compare_productivity.py` compara a produtividade de um perito específico ou dos 10 piores peritos com o restante do Brasil, excluindo o grupo analisado. O script calcula a produtividade em termos de análises por hora e oferece várias opções de exportação, incluindo PNG e comentários automáticos. Ele permite definir um limiar de produtividade e um número mínimo de análises para elegibilidade no Top 10. Gráficos ASCII também podem ser gerados para visualização no terminal.

`graphs_and_tables/g_weekday_to_weekend_table.py` gera uma tabela que detalha a quantidade de tarefas iniciadas em dias úteis e concluídas no fim de semana, categorizadas por perito, matrícula, CR e DR, dentro de um período especificado. As saídas incluem arquivos CSV, ORG, PDF e PNG, além

de uma lista de protocolos em ORG, dependendo das opções de exportação escolhidas. O script aceita flags como --start e --end para definir o intervalo de datas, e --export-csv, --export-org, --export-pdf, --export-png, e --export-protocols para especificar os formatos de saída desejados. A finalidade é fornecer uma análise detalhada do comportamento de trabalho dos peritos em relação ao início e término das tarefas.

`r_checks/01_nc_rate_check.R` é um script em R que verifica a taxa de não conformidade (NC) em um banco de dados SQLite, utilizando pacotes como DBI, RSQLite, dplyr e ggplot2 para manipulação de dados e visualização. Ele carrega funções comuns de um arquivo _common.R, se disponível, e define funções fallback para normalização de argumentos de linha de comando, conexão ao banco de dados e resolução do diretório de exportação. O script é projetado para ser executado via linha de comando, permitindo a análise de dados de conformidade de forma automatizada e flexível.

`r_checks/02_le15s_check.R` é um script em R que realiza verificações específicas em um banco de dados SQLite, utilizando pacotes como DBI, RSQLite, dplyr e ggplot2. Ele tenta carregar funções comuns de um arquivo _common.R e define funções fallback para normalização de argumentos de linha de comando, conexão ao banco de dados e resolução do diretório de exportação. O script é executado via linha de comando e é projetado para realizar análises automatizadas, focando em verificações que podem ser específicas para o contexto de conformidade ou desempenho.

`r_checks/03_productivity_check.R` é um script em R que realiza verificações de produtividade em um banco de dados SQLite, utilizando pacotes como DBI, RSQLite, dplyr e ggplot2. Ele tenta carregar funções comuns de um arquivo _common.R e define funções fallback para normalização de argumentos de linha de comando, conexão ao banco de dados e resolução do diretório de exportação. O script é executado via linha de comando e é projetado para realizar análises automatizadas de produtividade, permitindo uma avaliação detalhada do desempenho em um contexto específico.

`r_checks/04_overlap_check.R` é um script em R que verifica sobreposições em um banco de dados SQLite, utilizando pacotes como DBI, RSQLite, dplyr, ggplot2 e lubridate. Ele tenta carregar funções comuns de um arquivo _common.R e define funções fallback para normalização de argumentos de linha de comando, conexão ao banco de dados e resolução do diretório de exportação. O script é executado via linha de comando e é projetado para identificar e analisar sobreposições de dados, o que pode ser crucial para garantir a integridade e a precisão dos dados em análises subsequentes.

`r_checks/05_motivos_chisq.R` é um script em R que realiza uma análise estatística usando o teste qui-quadrado para comparar motivos de não conformidade entre um perito específico e o restante. As saídas incluem arquivos PNG e ORG, que contêm gráficos e comentários, e são nomeados de acordo com o perito analisado. O script utiliza pacotes como DBI, RSQLite, dplyr e ggplot2, e tenta carregar funções comuns de um arquivo _common.R. Ele é projetado para ser executado via linha de comando, permitindo uma análise estatística detalhada e automatizada das razões de não conformidade.

`r_checks/06_composite_robustness.R` é um script em R que realiza verificações de robustez composta em um banco de dados SQLite, utilizando pacotes como DBI, RSQLite, dplyr e ggplot2. Ele tenta carregar funções comuns de um arquivo _common.R e define funções fallback para normalização de argumentos de linha de comando, conexão ao banco de dados e resolução do diretório de exportação. O script é executado via linha de comando e é projetado para realizar análises automatizadas de robustez, permitindo uma avaliação abrangente da estabilidade e confiabilidade dos dados em um contexto específico.

`r_checks/07_kpi_icra_iatd_score.R` é um script em R que calcula KPIs como ICRA, IATD e ScoreFinal, comparando um perito específico com o restante do Brasil. As saídas incluem arquivos PNG, ORG e MD, que contêm gráficos e comentários, e são nomeados de acordo com o perito analisado. O script utiliza pacotes como DBI, RSQLite, dplyr e ggplot2, e tenta carregar funções comuns de um arquivo _common.R. Ele é projetado para ser executado via linha de comando, permitindo uma análise detalhada e

automatizada dos KPIs, facilitando a comparação de desempenho entre diferentes regiões ou indivíduos.

`r_checks/08_weighted_props.R` é um script em R que utiliza várias bibliotecas, incluindo DBI e RSQLite, para realizar consultas em um banco de dados SQLite. Ele define funções para detectar tabelas e colunas no banco de dados e inclui um prólogo específico para a execução de consultas robustas. O script parece ser parte de um conjunto maior de scripts, com funções de fallback para garantir que as operações de banco de dados sejam realizadas corretamente. As flags mencionadas, como --start, --end, --top10, e --perito, sugerem que o script pode ser configurado para executar análises específicas dentro de um intervalo de datas ou para um conjunto específico de dados, mas o trecho não fornece detalhes sobre as saídas geradas.

`r_checks/_ensure_deps.R` é um script em R que garante que todas as dependências necessárias para a execução de outros scripts estejam instaladas. Ele verifica se os pacotes R essenciais, como dplyr, ggplot2, e DBI, estão presentes e tenta instalá-los se estiverem ausentes. O script também ajusta o caminho da biblioteca do usuário e utiliza múltiplos núcleos para acelerar a instalação dos pacotes. Mensagens são exibidas para informar o usuário sobre o status das dependências, e um aviso é emitido se houver pacotes de sistema ausentes. O script termina exibindo informações da sessão R atual.

`r_checks/g01_top10_nc_rate_check.R` é um script em R que utiliza bibliotecas como DBI, RSQLite, e ggplot2 para realizar verificações de taxa de não conformidade (NC) no top 10 de registros. Ele inclui um prólogo que carrega um arquivo comum de funções, se disponível, e define funções de fallback para normalizar argumentos de linha de comando e abrir conexões de banco de dados. O script parece ser configurável via linha de comando, permitindo especificar parâmetros como o diretório de exportação. As saídas não são explicitamente mencionadas no trecho, mas o uso de ggplot2 sugere que gráficos podem ser gerados.

`r_checks/g02_top10_le15s_check.R` é um script em R que realiza verificações relacionadas ao top 10 de registros com duração menor ou igual a 15 segundos. Ele utiliza bibliotecas como DBI, RSQLite, e ggplot2, e inclui um

prólogo para carregar funções comuns e definir fallbacks para operações de linha de comando e banco de dados. O script é configurável via linha de comando, permitindo especificar parâmetros como o diretório de exportação. Embora o trecho não detalhe as saídas, o uso de ggplot2 sugere que gráficos podem ser gerados como parte do processo de verificação.

`r_checks/g03_top10_productivity_check.R` é um script em R que realiza verificações de produtividade no top 10 de registros, utilizando bibliotecas como DBI, RSQLite, e ggplot2. Ele inclui um prólogo para carregar funções comuns e definir fallbacks para normalizar argumentos de linha de comando e abrir conexões de banco de dados. O script é configurável via linha de comando, permitindo especificar parâmetros como o diretório de exportação. As saídas não são explicitamente mencionadas no trecho, mas o uso de ggplot2 sugere que gráficos podem ser gerados para visualizar os resultados das verificações de produtividade.

`r_checks/g04_top10_overlap_check.R` é um script em R que realiza verificações de sobreposição no top 10 de registros, utilizando bibliotecas como DBI, RSQLite, e ggplot2. Ele inclui um prólogo para carregar funções comuns e definir fallbacks para normalizar argumentos de linha de comando e abrir conexões de banco de dados. O script é configurável via linha de comando, permitindo especificar parâmetros como o diretório de exportação. Embora o trecho não detalhe as saídas, o uso de ggplot2 sugere que gráficos podem ser gerados para ilustrar os resultados das verificações de sobreposição.

`r_checks/g05_top10_motivos_chisq.R` é um script em R que realiza análises estatísticas usando o teste qui-quadrado para comparar os motivos de não conformidade (NC) do top 10 de registros com o restante do Brasil. Ele gera saídas em formato PNG e org, incluindo um gráfico e comentários. O script é configurável via linha de comando, com flags como --db para especificar o caminho do banco de dados, --start e --end para definir o intervalo de datas, e --min-analises para o número mínimo de análises. As saídas são salvas no diretório especificado por --out-dir.

`r_checks/g06_top10_composite_robustness.R` é um script em R que avalia a robustez do composto (z-score médio) para o top 10 de registros. Ele

utiliza bibliotecas como DBI, RSQLite, e ggplot2, e inclui um prólogo para carregar funções comuns e definir fallbacks para operações de linha de comando e banco de dados. O script gera saídas em formato PNG e org, incluindo gráficos e comentários sobre a robustez do composto. As saídas são salvas em um diretório de exportação, que pode ser especificado via linha de comando.

`r_checks/g07_top10_kpi_icra_iatd_score.R` é um script em R que gera visualizações e comentários sobre os principais KPIs (ICRA, IATD, ScoreFinal) comparando o Top 10 de um grupo específico com o restante do Brasil. As saídas incluem um gráfico em PNG, um arquivo .org com a imagem e comentários, e um arquivo .org apenas com comentários. O script utiliza bibliotecas como ggplot2 para visualização e dplyr para manipulação de dados. Ele também inclui um sistema de fallback para funções essenciais caso o arquivo _common.R não seja encontrado. As opções de linha de comando são processadas para permitir a personalização da execução.

`reports/make_impact_report.py` é um script em Python que gera um relatório de impacto na fila em formato PDF e imagens PNG de alta resolução usando bibliotecas como ReportLab e Matplotlib. O relatório inclui gráficos como Top, Curva do Cotovelo, Tornado, Estratos, Permutação e PSA. As tabelas são ordenadas do pior para o melhor com base em critérios específicos e formatadas com duas casas decimais. O script também remove referências e colunas 'uo' das tabelas e formata nomes de peritos em Title Case. Ele integra-se com o script make_kpi_report para obter os 10 piores peritos, caso necessário.

`reports/make_kpi_report.py` é um script em Python que gera relatórios ATESTMED, tanto individuais quanto para o Top 10, combinando gráficos gerados em Python, análises em R, comentários e relatórios em formatos Org e PDF. O script abrange análises de padrões de dias da semana versus fins de semana e inclui apêndices. Ele organiza diretórios para armazenar gráficos, tabelas e saídas, e carrega variáveis de ambiente de um arquivo .env, se disponível. O script utiliza bibliotecas como pandas para manipulação de dados e PyPDF2 para manipulação de PDFs, garantindo que todos os componentes necessários para a geração do relatório estejam disponíveis e organizados.